**TUGAS PROYEK PEMROGRAMAN BERORIENTASI OBJEK**

**APLIKASI PENGELOLA RESERVASI RESTORAN**

****

DISUSUN OLEH :

1. AHMAD FERDIANSYAH (232120)
2. MUH. SYAHRUL IRSAN (232094)
3. RIVALDY CRISTIANTO L. (232111)
4. MUHAMMAD ILHAM (232112)
5. DIMAZ SEBASTIAN DONI (232113)
6. ALWAN DELON PATIUNG (232114)
7. JOE RIZAL CAVIN TANDI (232119)

**UNIVERSITAS DIPA MAKASSAR**

**2024**

**ABSTRAK**

Sebuah aplikasi reservasi untuk restoran berbasis java yang memudahkan pelayan dengan penggunaan aplikasi yang sangat mudah di mengerti.

Dalam aplikasi ini terdapat beberapa fitur yakni seperti membuat reservasi, membatalkan reservasi, menambahkan meja, dan menampilkan meja.

Penggunaan aplikasi ini juga sangat mudah dan terhubung dengan database sehingga pesanan dapat langsung tampil di aplikasi Netbeans. Sistem informasi ini dikembangkan dengan menggunakan Netbeans Sebagai aplikasi editor dan menggunakan framework Javafx untuk tampilan interface .

Aplikasi Pengelolaan Reservasi Restoran ini dirancang untuk memudahkan proses reservasi meja di restoran secara efektif dan efisien.

Aplikasi memungkinkan pengguna untuk melakukan pemesanan meja secara online dengan memilih tanggal dan waktu yang diinginkan, serta mencatat informasi detail pemesan seperti nama, dan nomor telepon. Selain itu, aplikasi ini dilengkapi dengan fitur untuk menampilkan status meja secara real-time, memberikan informasi apakah meja yang diinginkan tersedia atau sudah terisi.

Dengan menggunakan teknologi terkini, aplikasi ini bertujuan untuk meningkatkan pengalaman pelanggan serta mengoptimalkan manajemen pemesanan meja di restoran, sehingga dapat meminimalisir kesalahan dan meningkatkan kepuasan pelanggan.

**PENDAHULUAN**

Dalam industri restoran yang semakin berkembang, manajemen pemesanan meja yang efisien sangat penting untuk menjaga kelancaran operasional dan kepuasan pelanggan. Pemesanan meja yang terorganisir dengan baik dapat meminimalkan waktu tunggu, menghindari tumpang tindih pemesanan, serta memastikan restoran beroperasi secara optimal. Seiring dengan kemajuan teknologi, aplikasi berbasis perangkat lunak kini menjadi solusi praktis untuk mengelola reservasi, menggantikan sistem manual yang rawan kesalahan.

Aplikasi pengelolaan reservasi restoran ini dikembangkan untuk mempermudah pelanggan dalam melakukan pemesanan meja secara online, serta memungkinkan restoran untuk mengecek status meja secara real-time. Fitur-fitur utama yang ditawarkan mencakup membuat reservasi, membatalkan reservasi dan pembaruan status meja, yang semuanya dapat diakses melalui antarmuka yang sederhana dan user-friendly menggunakan javaFx. Penerapan aplikasi ini diharapkan tidak hanya meningkatkan efisiensi dalam pengelolaan pemesanan, tetapi juga memperbaiki pengalaman pelanggan dalam melakukan reservasi. Selain itu, sistem ini memberikan restoran kemampuan untuk memonitor kapasitas meja secara real-time, yang dapat membantu dalam pengambilan keputusan yang lebih cepat dan tepat mengenai pengelolaan sumber daya.

Dengan latar belakang ini, penelitian dan pengembangan aplikasi pengelolaan reservasi restoran bertujuan untuk memberikan solusi yang lebih baik, lebih cepat, dan lebih efisien bagi kedua belah pihak, yaitu pengelola restoran dan pelanggan.

1. **LATAR BELAKANG**

Restoran merupakan salah satu sektor industri yang terus berkembang seiring dengan meningkatnya kebutuhan masyarakat akan tempat untuk bersantap dan bersosialisasi. Dalam operasional sehari-hari, manajemen reservasi meja menjadi salah satu aspek penting yang berpengaruh terhadap pengalaman pelanggan dan efisiensi layanan. Namun, banyak restoran masih menggunakan metode reservasi manual, seperti pencatatan di buku atau melalui panggilan telepon, yang sering kali tidak efisien dan rentan terhadap kesalahan, seperti tumpang tindih jadwal pemesanan, kehilangan data, atau ketidakjelasan informasi.

Perkembangan teknologi informasi membuka peluang untuk menciptakan solusi digital yang dapat menggantikan sistem reservasi manual dengan cara yang lebih modern dan efektif. Aplikasi berbasis perangkat lunak untuk mengelola reservasi meja memungkinkan restoran untuk mempermudah proses pemesanan, memonitor kapasitas meja secara real-time, dan memberikan pengalaman yang lebih baik bagi pelanggan. Dengan integrasi fitur seperti pencatatan data pelanggan, status ketersediaan meja, dan pengelolaan jadwal secara otomatis, aplikasi ini mampu mengatasi berbagai kendala yang sering terjadi pada sistem manual.

Selain itu, perubahan gaya hidup masyarakat yang semakin terhubung dengan teknologi juga mendorong kebutuhan akan layanan reservasi yang cepat, mudah, dan dapat diakses kapan saja. Pelanggan menginginkan kemudahan dalam melakukan pemesanan tanpa perlu datang langsung ke restoran atau menghabiskan waktu menunggu ketersediaan meja. Oleh karena itu, pengembangan aplikasi pengelolaan reservasi restoran menjadi solusi yang relevan untuk memenuhi kebutuhan pelanggan sekaligus mendukung peningkatan operasional restoran.

Dengan mempertimbangkan pentingnya efisiensi dalam pengelolaan reservasi dan kebutuhan masyarakat akan layanan berbasis teknologi, aplikasi ini diharapkan dapat menjadi inovasi yang membantu restoran dalam meningkatkan pelayanan, memaksimalkan kapasitas, dan memberikan pengalaman yang lebih baik bagi pelanggan.

1. **TUJUAN**

Mempermudah Proses Reservasi Meja yakni dengan mengembangkan aplikasi yang memungkinkan pelanggan untuk memesan meja secara online dengan mudah, cepat, dan praktis tanpa harus datang langsung ke restoran. Meningkatkan Efisiensi Pengelolaan Reservasi dengan Memberikan solusi digital untuk menggantikan sistem manual dalam mencatat dan mengelola pemesanan meja, sehingga meminimalkan kesalahan seperti tumpang tindih jadwal atau kehilangan data. Menyediakan fitur yang memungkinkan pelanggan dan pengelola restoran memantau ketersediaan meja secara langsung, sehingga membantu pengambilan keputusan yang lebih cepat dan akurat. Serta Membantu restoran dalam memaksimalkan kapasitas penggunaan meja dan meningkatkan efisiensi alur kerja melalui sistem yang terorganisir dan terintegrasi.

1. **MANFAAT**

Pengembangan aplikasi pengelolaan reservasi restoran memberikan manfaat utama bagi pelanggan, pengelola restoran, dan industri restoran secara umum. Pelanggan dapat dengan mudah memesan meja secara online, memperoleh informasi real-time terkait ketersediaan meja, dan menikmati pengalaman reservasi yang nyaman. Bagi pengelola restoran, aplikasi ini meningkatkan efisiensi operasional dengan mempermudah pencatatan dan pengelolaan reservasi, mengoptimalkan kapasitas meja, serta mengurangi risiko kesalahan. Selain itu, aplikasi ini mendorong transformasi digital di sektor restoran, meningkatkan daya saing dengan mengadopsi teknologi modern, serta menjadi model pengembangan aplikasi serupa di industri jasa lainnya.

**DASAR TEORI**

1. **DEFINISI JUDUL**

Judul ini merujuk pada pengembangan sebuah aplikasi berbasis java yang dirancang untuk mempermudah pengelolaan proses reservasi di restoran. Aplikasi menunjukkan bahwa proyek ini berbasis teknologi, memungkinkan pengguna mengaksesnya melalui perangkat elektronik seperti smartphone atau komputer. Pengelolaan Reservasi mengacu pada sistem yang dapat mencatat, memantau, dan mengatur pemesanan meja secara otomatis dan terstruktur. Sedangkan Restoran menunjukkan bahwa aplikasi ini dirancang khusus untuk kebutuhan bisnis di sektor kuliner, membantu pengelola dalam menyediakan layanan reservasi yang lebih efisien dan memberikan pengalaman lebih baik bagi pelanggan.

1. **PENJELASAN JAVA**

Apa itu Java? Java merupakan bahasa pemrograman yang digunakan secara luas untuk pengodean aplikasi web. Bahasa ini telah menjadi pilihan populer di antara developer selama lebih dari dua dekade, dengan jutaan aplikasi Java yang digunakan saat ini. Java merupakan bahasa multiplatform yang berorientasi pada objek dan berpusat pada jaringan yang dapat digunakan sebagai platform di dalamnya. Java merupakan bahasa pemrograman yang cepat, aman, dan andal untuk mengodekan segala sesuatu mulai dari aplikasi seluler dan perangkat lunak korporasi hingga aplikasi big data dan teknologi sisi server.

Apa kegunaan bahasa pemrograman Java?

Karena merupakan bahasa yang gratis dan serbaguna, Java membangun perangkat lunak yang dilokalkan dan terdistribusi. Beberapa penggunaan Java yang umum di antaranya adalah:

1. **Perkembangan Game**

Banyak game seluler, komputer, dan video populer dibangun di Java. Bahkan game modern yang mengintegrasikan teknologi canggih, seperti machine learning atau realitas virtual (virtual reality) dibangun dengan teknologi Java.

1. **Komputasi cloud**

Java sering disebut sebagai WORA – Write Once and Run Anywhere, yang membuatnya sempurna untuk aplikasi berbasis cloud yang terdesentralisasi. Penyedia cloud memilih bahasa Java untuk menjalankan program di berbagai platform yang mendasari.

1. **Big Data**

Java digunakan untuk mesin pemrosesan data yang dapat bekerja dengan set data yang kompleks dan data waktu nyata yang berjumlah besar.

1. **Kecerdasan Buatan**

Java merupakan pembangkit (powerhouse) bagi pustaka machine learning. Stabilitas dan kecepatan Java membuatnya sempurna untuk pengembangan aplikasi kecerdasan buatan, seperti pemrosesan bahasa alami dan deep learning.

1. **Internet of Things**

Java telah digunakan untuk memprogram sensor dan perangkat keras di perangkat edge yang dapat terhubung ke internet secara mandiri.

1. **PENJELASAN NETBEANS**

NetBeans adalah salah satu software yang sangat sering digunakan dalam dunia programmer atau digunakan oleh banyak developer. Jadi ini adalah aplikasi IDE atau Integrated Development Environment yang basis bahasanya itu Java dan berjalan di atas Swing.

Maksud dari Swing itu sendiri adalah teknologi yang memungkinkan pengembangan aplikasi desktop dan bisa berjalan pada berbagai platform sistem operasi. Sementara IDE atau Integrated Development Environment merupakan satu sistem pemrograman yang akan diintegrasikan ke dalam suatu perangkat lunak. Software ini bukan hanya mendukung bahasa Java saja, melainkan bisa dipakai membuat atau mengembangkan program dengan basis C atau C++.

Aplikasi ini merupakan salah satu aplikasi dengan jenis open source yang terbilang cukup sukses dan berhasil menarik banyak sekali pengguna. Komunitas dari pengguna aplikasi ini pun sudah semakin berkembang dan sudah memiliki sekitar 100 mitra bisnis dan mungkin akan terus berkembang. Sun Microsystems sebagai pihak sponsor utama sudah memulai dan mengembangkan sejak tahun 2000 dan sampai sekarang masih melanjutkan kerja samanya dengan NetBeans ini.

**KERANGKA PROGRAM**

**Algoritma Pseudocode**

1. **Menampilkan Daftar Meja (View Tables)**

FUNCTION viewTables()

CALL Restaurant.getAllTables()

DISPLAY list of tables (ID, Capacity, Status)

END FUNCTION

1. **Menambahkan Meja Baru (Add Table)**

FUNCTION addTable(id, capacity)

IF id IS EMPTY OR capacity IS NOT A NUMBER THEN

DISPLAY "Invalid input"

RETURN

END IF

CREATE new Table object with id and capacity

CALL Restaurant.addTable(new Table)

IF SUCCESS THEN

DISPLAY "Table added successfully"

ELSE

DISPLAY "Error adding table"

END IF

END FUNCTION

1. **Membuat Reservasi (Make Reservation)**

FUNCTION makeReservation(customerName, phoneNumber, tableId, reservationTime)

IF customerName IS EMPTY OR phoneNumber IS EMPTY OR tableId IS EMPTY OR reservationTime IS INVALID THEN

DISPLAY "Invalid input"

RETURN

END IF

// Add customer to database

customerId = CALL Restaurant.addCustomer(customerName, phoneNumber)

IF customerId IS NULL THEN

DISPLAY "Error adding customer"

RETURN

END IF

// Add reservation to database

reservationId = CALL Restaurant.addReservation(customerId, tableId, reservationTime)

IF reservationId IS NULL THEN

DISPLAY "Error making reservation"

RETURN

END IF

// Update table status

CALL Restaurant.updateTableStatus(tableId, true)

DISPLAY "Reservation made successfully"

END FUNCTION

1. **Membatalkan Reservasi (Cancel Reservation)**

FUNCTION cancelReservation(reservationId)

IF reservationId IS NOT A NUMBER THEN

DISPLAY "Invalid input"

RETURN

END IF

// Delete reservation from database

tableId = CALL Restaurant.deleteReservation(reservationId)

IF tableId IS NULL THEN

DISPLAY "Error cancelling reservation"

RETURN

END IF

// Update table status

CALL Restaurant.updateTableStatus(tableId, false)

DISPLAY "Reservation cancelled successfully"

END FUNCTION

1. **Menampilkan Daftar Reservasi (View Reservations)**

FUNCTION viewReservations()

CALL Restaurant.getAllReservations()

DISPLAY list of reservations (Reservation ID, Customer Name, Table ID, Reservation Time)

END FUNCTION

1. **Sinkronisasi Real-Time**

FUNCTION refreshView()

IF Table data changed THEN

CALL viewTables()

END IF

IF Reservation data changed THEN

CALL viewReservations()

END IF

END FUNCTION

1. **Alur Utama Sistem**

FUNCTION main()

DISPLAY "Welcome to Restaurant Reservation System"

DISPLAY "1. View Tables"

DISPLAY "2. Make Reservation"

DISPLAY "3. Cancel Reservation"

DISPLAY "4. Add Table"

choice = GET USER INPUT

SWITCH (choice)

CASE 1:

CALL viewTables()

CASE 2:

DISPLAY "Enter Customer Name, Phone, Table ID, Reservation Time"

CALL makeReservation(name, phone, tableId, time)

CASE 3:

DISPLAY "Enter Reservation ID"

CALL cancelReservation(reservationId)

CASE 4:

DISPLAY "Enter Table ID and Capacity"

CALL addTable(id, capacity)

DEFAULT:

DISPLAY "Invalid choice"

END SWITCH

END FUNCTION

**HASIL**

1. **SOURCE CODE**

* **Class Table.java**

public class Table {

private String tableId;

private int capacity;

private boolean isReserved;

public Table(String tableId, int capacity, boolean isReserved) {

this.tableId = tableId;

this.capacity = capacity;

this.isReserved = isReserved;

}

public String getTableId() {

return tableId;

}

public void setTableId(String tableId) {

this.tableId = tableId;

}

public int getCapacity() {

return capacity;

}

public void setCapacity(int capacity) {

this.capacity = capacity;

}

public boolean isReserved() {

return isReserved;

}

public void reserve() {

this.isReserved = true;

}

public void release() {

this.isReserved = false;

}

@Override

public String toString() {

return "Table ID: " + tableId + ", Capacity: " + capacity + ", Reserved: " + isReserved;

}

}

* **Class Costumer.java**

public class Customer {

private int customerId;

private String name;

private String phoneNumber;

public Customer(int customerId, String name, String phoneNumber) {

this.customerId = customerId;

this.name = name;

this.phoneNumber = phoneNumber;

}

public int getCustomerId() {

return customerId;

}

public void setCustomerId(int customerId) {

this.customerId = customerId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

@Override

public String toString() {

return "Customer ID: " + customerId + ", Name: " + name + ", Phone: " + phoneNumber;

}

}

* **Class Reservation.java**

import java.time.LocalDateTime;

public class Reservation {

private int reservationId;

private Customer customer;

private Table table;

private LocalDateTime reservationDate;

public Reservation(int reservationId, Customer customer, Table table, LocalDateTime reservationDate) {

this.reservationId = reservationId;

this.customer = customer;

this.table = table;

this.reservationDate = reservationDate;

}

public int getReservationId() {

return reservationId;

}

public void setReservationId(int reservationId) {

this.reservationId = reservationId;

}

public Customer getCustomer() {

return customer;

}

public void setCustomer(Customer customer) {

this.customer = customer;

}

public Table getTable() {

return table;

}

public void setTable(Table table) {

this.table = table;

}

public LocalDateTime getReservationDate() {

return reservationDate;

}

public void setReservationDate(LocalDateTime reservationDate) {

this.reservationDate = reservationDate;

}

@Override

public String toString() {

return "Reservation ID: " + reservationId +

", Customer: [" + customer.toString() +

"], Table: [" + table.toString() +

"], Reservation Date: " + reservationDate;

}

}

* **Class Restaurant.java**

import java.sql.\*;

import java.time.LocalDateTime;

import java.util.ArrayList;

import java.util.List;

public class Restaurant {

private Connection connection; // Koneksi ke database

private List<Table> tables; // Menyimpan daftar meja

private List<Customer> customers; // Menyimpan daftar pelanggan

private List<Reservation> reservations; // Menyimpan daftar reservasi

public Restaurant() {

this.connection = connection;

this.tables = new ArrayList<>(); // Inisialisasi list tables

this.customers = new ArrayList<>();

this.reservations = new ArrayList<>();

try {

// Koneksi ke database MySQL

String url = "jdbc:mysql://localhost:3306/restaurant";

String user = "root"; // Sesuaikan dengan user MySQL Anda

String password = ""; // Sesuaikan dengan password MySQL Anda

connection = DriverManager.getConnection(url, user, password);

} catch (SQLException e) {

e.printStackTrace();

}

}

public List<Table> getAllTables() throws SQLException {

String query = "SELECT \* FROM tables";

List<Table> tableList = new ArrayList<>();

try (PreparedStatement stmt = connection.prepareStatement(query);

ResultSet rs = stmt.executeQuery()) {

while (rs.next()) {

String tableId = rs.getString("tableId");

int capacity = rs.getInt("capacity");

boolean isReserved = rs.getBoolean("isReserved");

tableList.add(new Table(tableId, capacity, isReserved));

}

}

this.tables = tableList; // Sinkronisasi dengan daftar lokal

return tableList;

}

public void createReservation(String customerName, String phoneNumber, String tableId, LocalDateTime reservationDate) {

try {

// Tambahkan pelanggan

PreparedStatement customerStmt = connection.prepareStatement(

"INSERT INTO customers (name, phoneNumber) VALUES (?, ?)",

Statement.RETURN\_GENERATED\_KEYS

);

customerStmt.setString(1, customerName);

customerStmt.setString(2, phoneNumber);

customerStmt.executeUpdate();

ResultSet rs = customerStmt.getGeneratedKeys();

int customerId = 0;

if (rs.next()) {

customerId = rs.getInt(1);

}

// Tambahkan reservasi

PreparedStatement reservationStmt = connection.prepareStatement(

"INSERT INTO reservations (customerId, tableId, reservationDate) VALUES (?, ?, ?)"

);

reservationStmt.setInt(1, customerId);

reservationStmt.setString(2, tableId);

reservationStmt.setTimestamp(3, Timestamp.valueOf(reservationDate));

reservationStmt.executeUpdate();

// Update status meja

PreparedStatement updateTableStmt = connection.prepareStatement(

"UPDATE tables SET isReserved = TRUE WHERE tableId = ?"

);

updateTableStmt.setString(1, tableId);

updateTableStmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void cancelReservation(int reservationId) {

try {

// Hapus reservasi

PreparedStatement stmt = connection.prepareStatement("DELETE FROM reservations WHERE reservationId = ?");

stmt.setInt(1, reservationId);

stmt.executeUpdate();

// Update status meja

PreparedStatement updateTableStmt = connection.prepareStatement(

"UPDATE tables SET isReserved = FALSE WHERE tableId = " +

"(SELECT tableId FROM reservations WHERE reservationId = ?)"

);

updateTableStmt.setInt(1, reservationId);

updateTableStmt.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

}

}

public void addTable(Table table) throws SQLException {

String query = "INSERT INTO tables (tableId, capacity, isReserved) VALUES (?, ?, ?)";

try (PreparedStatement stmt = connection.prepareStatement(query)) {

stmt.setString(1, table.getTableId());

stmt.setInt(2, table.getCapacity());

stmt.setBoolean(3, table.isReserved());

stmt.executeUpdate();

}

tables.add(table); // Tambahkan ke daftar lokal juga

}

public List<Reservation> getAllReservations() throws SQLException {

List<Reservation> reservations = new ArrayList<>();

Statement stmt = connection.createStatement();

ResultSet rs = stmt.executeQuery(

"SELECT r.reservationId, c.name, c.phoneNumber, t.tableId, t.capacity, r.reservationDate " +

"FROM reservations r " +

"JOIN customers c ON r.customerId = c.customerId " +

"JOIN tables t ON r.tableId = t.tableId"

);

while (rs.next()) {

Customer customer = new Customer(0, rs.getString("name"), rs.getString("phoneNumber"));

Table table = new Table(rs.getString("tableId"), rs.getInt("capacity"), true);

Reservation reservation = new Reservation(

rs.getInt("reservationId"),

customer,

table,

rs.getTimestamp("reservationDate").toLocalDateTime()

);

reservations.add(reservation);

}

return reservations;

}

}

* **Class Main.java**

import javafx.application.Application;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

public class Main extends Application {

private Restaurant restaurant = new Restaurant();

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("Restaurant Reservation System");

// Tombol untuk fitur utama

Button viewTablesButton = new Button("View Tables");

Button makeReservationButton = new Button("Make Reservation");

Button cancelReservationButton = new Button("Cancel Reservation");

Button addTableButton = new Button("Add Table");

// Aksi tombol

viewTablesButton.setOnAction(e -> ViewTables.display(restaurant));

makeReservationButton.setOnAction(e -> MakeReservation.display(restaurant));

cancelReservationButton.setOnAction(e -> CancelReservation.display(restaurant));

addTableButton.setOnAction(e -> AddTable.display(restaurant));

// Layout menu utama

VBox layout = new VBox(10, viewTablesButton, makeReservationButton, cancelReservationButton, addTableButton);

layout.setStyle("-fx-padding: 20; -fx-alignment: center;");

Scene scene = new Scene(layout, 400, 300);

primaryStage.setScene(scene);

primaryStage.show();

}

public static void main(String[] args) {

launch(args);

}

}

* **Class MakeReservation.java**

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class MakeReservation {

public static void display(Restaurant restaurant) {

Stage window = new Stage();

window.setTitle("Make Reservation");

TextField customerNameInput = new TextField();

customerNameInput.setPromptText("Customer Name");

TextField phoneInput = new TextField();

phoneInput.setPromptText("Phone Number");

TextField tableIdInput = new TextField();

tableIdInput.setPromptText("Table ID");

TextField dateInput = new TextField();

dateInput.setPromptText("Reservation Date (yyyy-MM-dd HH:mm)");

Button reserveButton = new Button("Reserve");

Label messageLabel = new Label();

reserveButton.setOnAction(e -> {

try {

String customerName = customerNameInput.getText();

String phone = phoneInput.getText();

String tableId = tableIdInput.getText();

LocalDateTime reservationDate = LocalDateTime.parse(

dateInput.getText(),

DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm")

);

restaurant.createReservation(customerName, phone, tableId, reservationDate);

messageLabel.setText("Reservation created successfully!");

} catch (Exception ex) {

messageLabel.setText("Error: " + ex.getMessage());

}

});

VBox layout = new VBox(10, customerNameInput, phoneInput, tableIdInput, dateInput, reserveButton, messageLabel);

layout.setStyle("-fx-padding: 20;");

layout.setAlignment(Pos.CENTER);

Scene scene = new Scene(layout, 400, 300);

window.setScene(scene);

window.showAndWait();

}

}

* **Class CancelReservation.java**

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

public class CancelReservation {

public static void display(Restaurant restaurant) {

Stage window = new Stage();

window.setTitle("Cancel Reservation");

TextField reservationIdInput = new TextField();

reservationIdInput.setPromptText("Reservation ID");

Button cancelButton = new Button("Cancel Reservation");

Label messageLabel = new Label();

cancelButton.setOnAction(e -> {

try {

int reservationId = Integer.parseInt(reservationIdInput.getText());

restaurant.cancelReservation(reservationId);

messageLabel.setText("Reservation canceled successfully!");

} catch (Exception ex) {

messageLabel.setText("Error: " + ex.getMessage());

}

});

VBox layout = new VBox(10, reservationIdInput, cancelButton, messageLabel);

layout.setStyle("-fx-padding: 20;");

layout.setAlignment(Pos.CENTER);

Scene scene = new Scene(layout, 400, 200);

window.setScene(scene);

window.showAndWait();

}

}

* **Class AddTable.java**

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

public class AddTable {

public static void display(Restaurant restaurant) {

Stage window = new Stage();

window.setTitle("Add Table");

// Input fields

TextField tableIdInput = new TextField();

tableIdInput.setPromptText("Table ID");

TextField capacityInput = new TextField();

capacityInput.setPromptText("Capacity (Number of Seats)");

Button addButton = new Button("Add Table");

Label messageLabel = new Label();

addButton.setOnAction(e -> {

try {

// Validasi input kosong

if (tableIdInput.getText().isEmpty() || capacityInput.getText().isEmpty()) {

messageLabel.setText("All fields must be filled!");

return;

}

// Parse capacity to integer

String tableId = tableIdInput.getText();

int capacity = Integer.parseInt(capacityInput.getText());

// Tambahkan meja ke database

Table newTable = new Table(tableId, capacity, false);

restaurant.addTable(newTable);

messageLabel.setText("Table added successfully!");

tableIdInput.clear();

capacityInput.clear();

} catch (NumberFormatException ex) {

messageLabel.setText("Error: Capacity must be a number!");

} catch (Exception ex) {

messageLabel.setText("Error: " + ex.getMessage());

}

});

VBox layout = new VBox(10, tableIdInput, capacityInput, addButton, messageLabel);

layout.setStyle("-fx-padding: 20;");

layout.setAlignment(Pos.CENTER);

Scene scene = new Scene(layout, 400, 250);

window.setScene(scene);

window.showAndWait();

}

}

* **Class ViewTables.java**

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ListView;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import java.sql.SQLException;

public class ViewTables {

public static void display(Restaurant restaurant) {

Stage window = new Stage();

window.setTitle("View Tables");

ObservableList<String> tableItems = FXCollections.observableArrayList();

ListView<String> tableList = new ListView<>(tableItems);

try {

restaurant.getAllTables().forEach(table -> tableItems.add(table.toString()));

} catch (SQLException e) {

tableItems.add("Error loading tables: " + e.getMessage());

}

Button closeButton = new Button("Close");

closeButton.setOnAction(e -> window.close());

VBox layout = new VBox(10, tableList, closeButton);

layout.setStyle("-fx-padding: 20;");

layout.setAlignment(Pos.CENTER);

Scene scene = new Scene(layout, 500, 400);

window.setScene(scene);

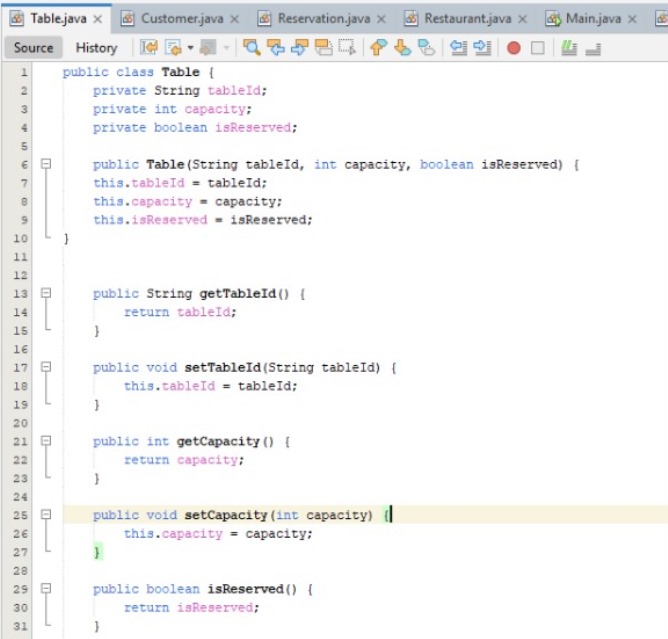
window.showAndWait();

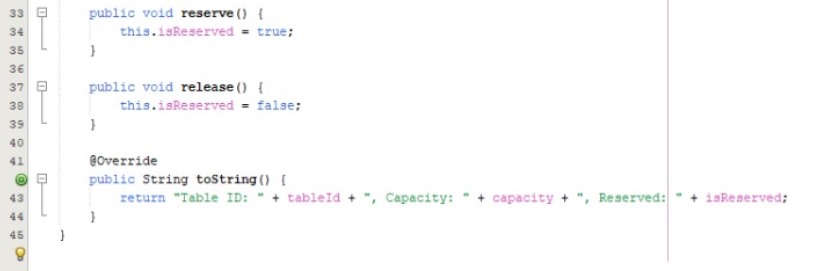
}

}

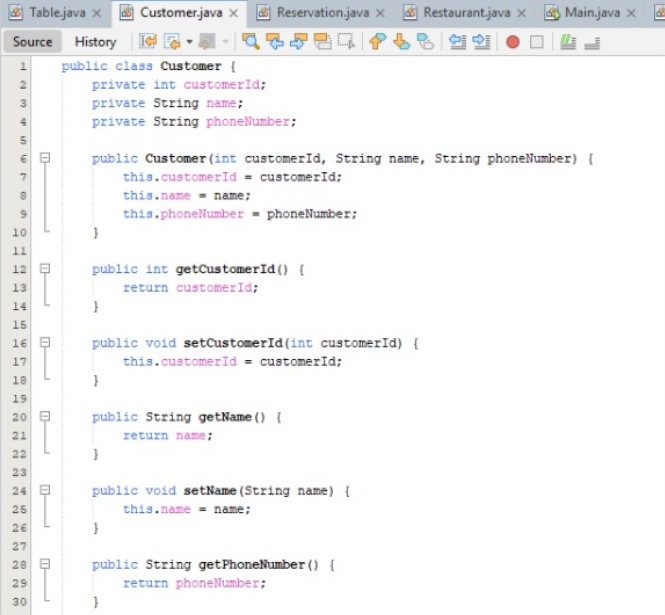
1. **PRINT SCREEN**

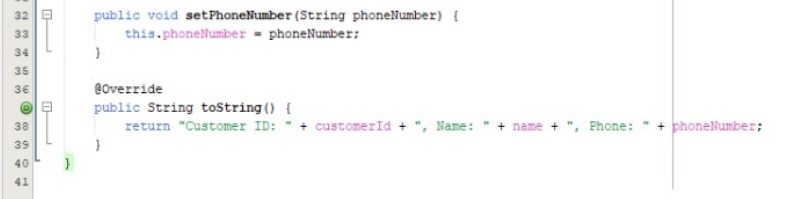
* Table.java



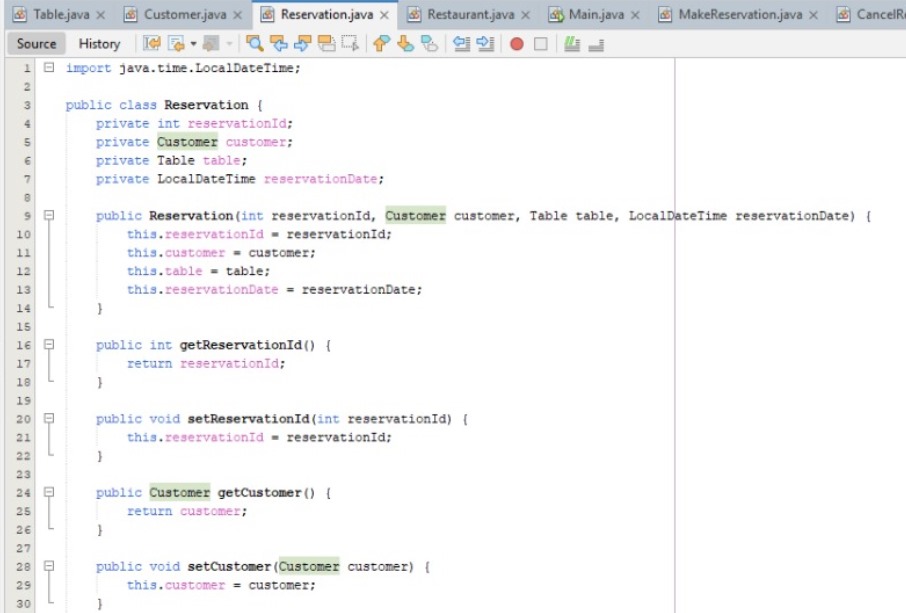


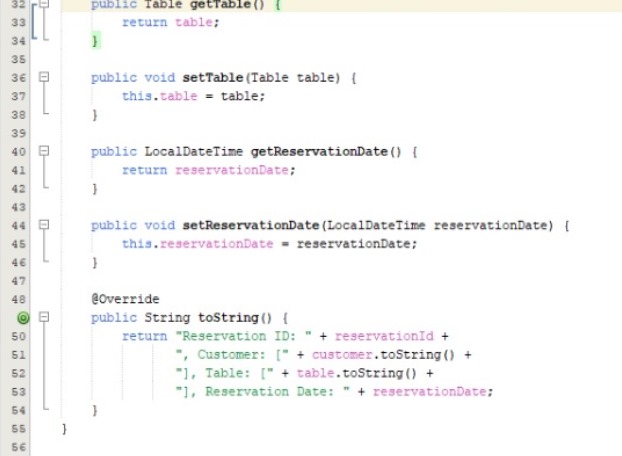
* Customer.java



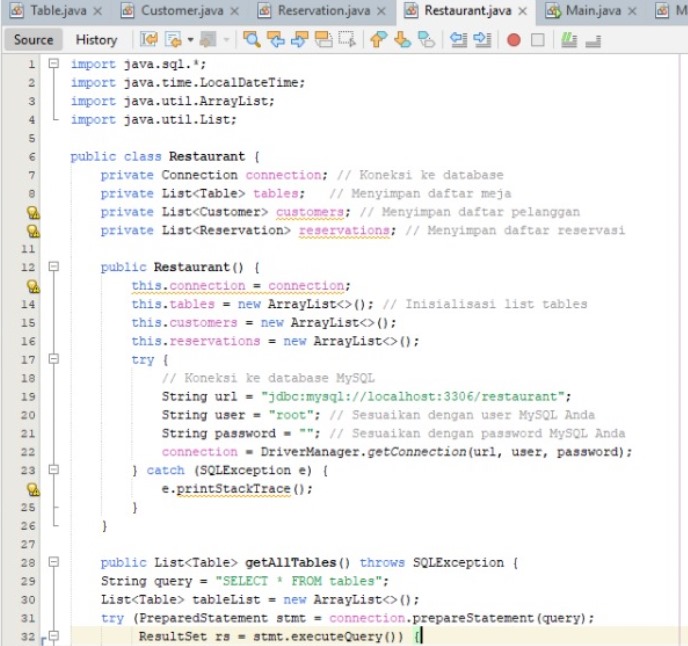


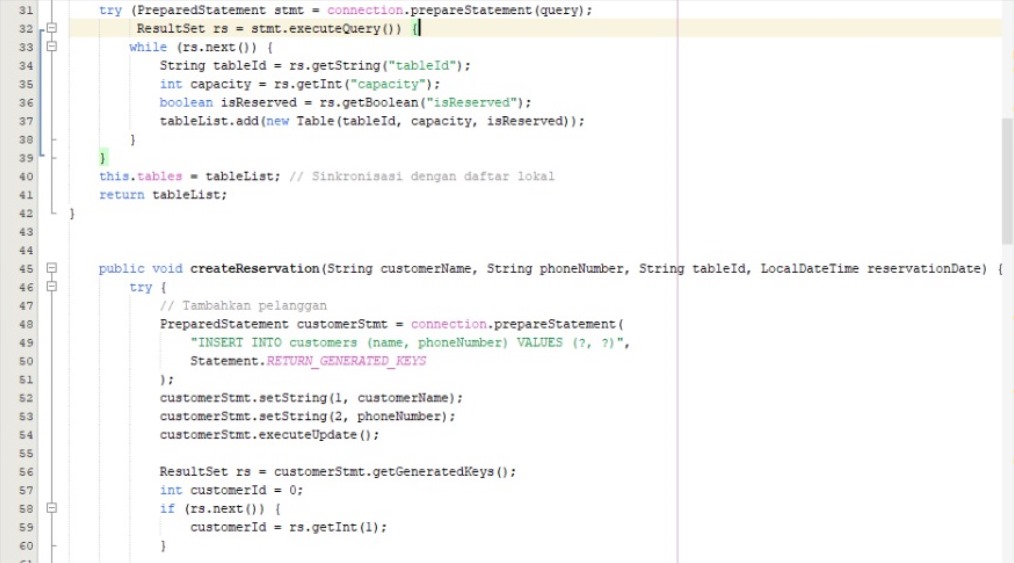
* Reservation.java



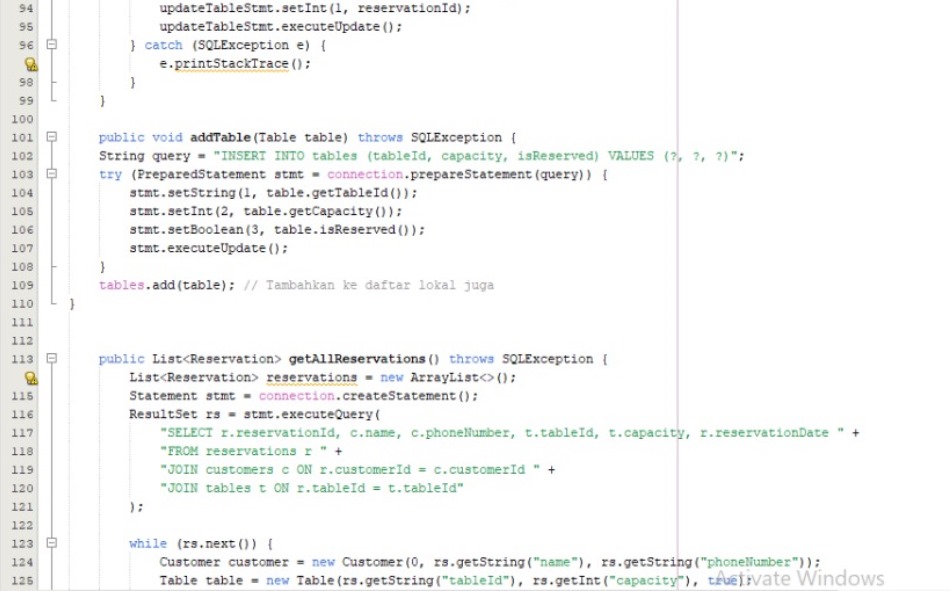


* Restaurant.java



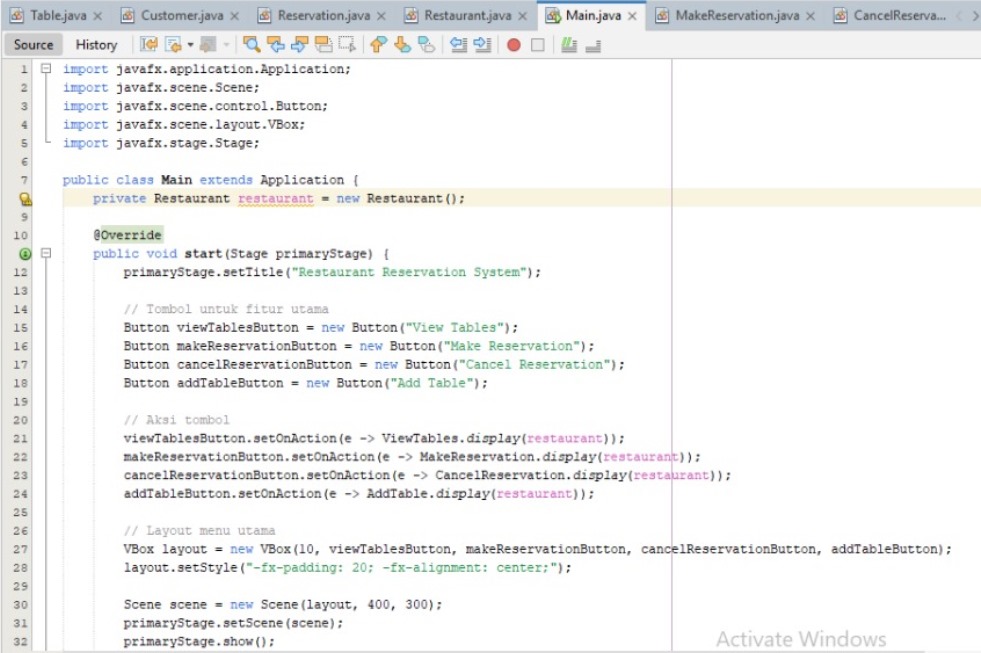


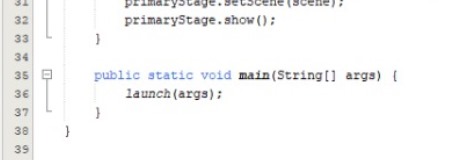




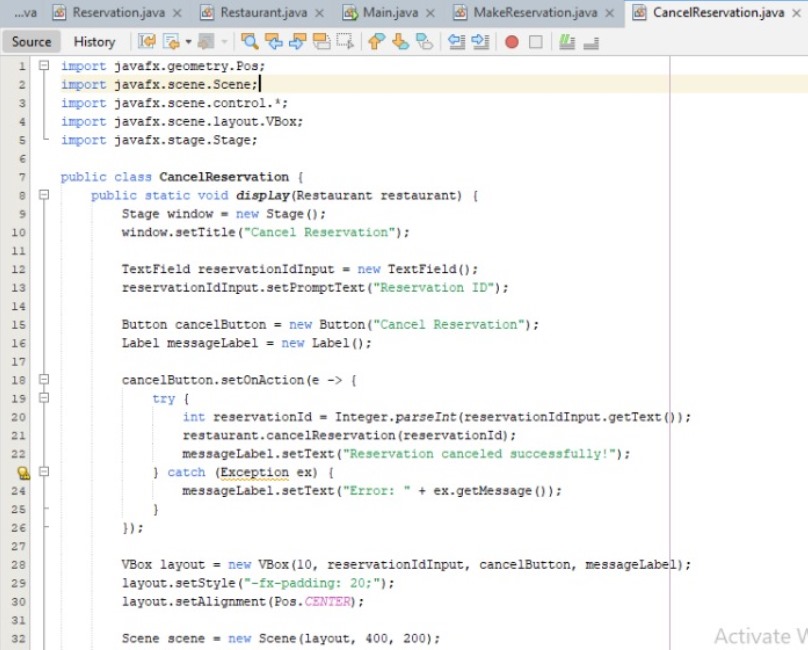


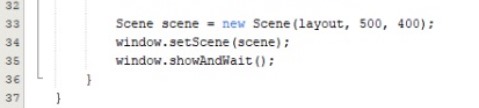
* Main.java



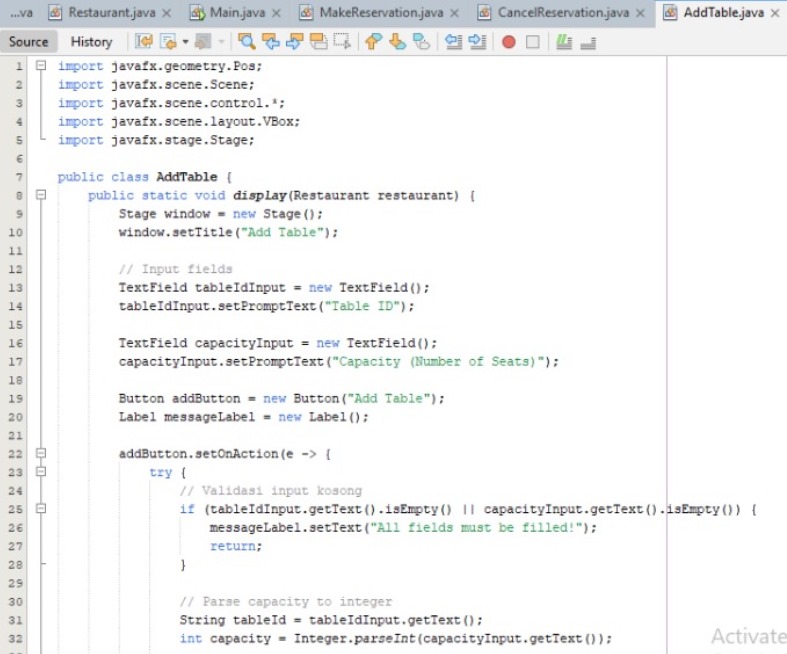


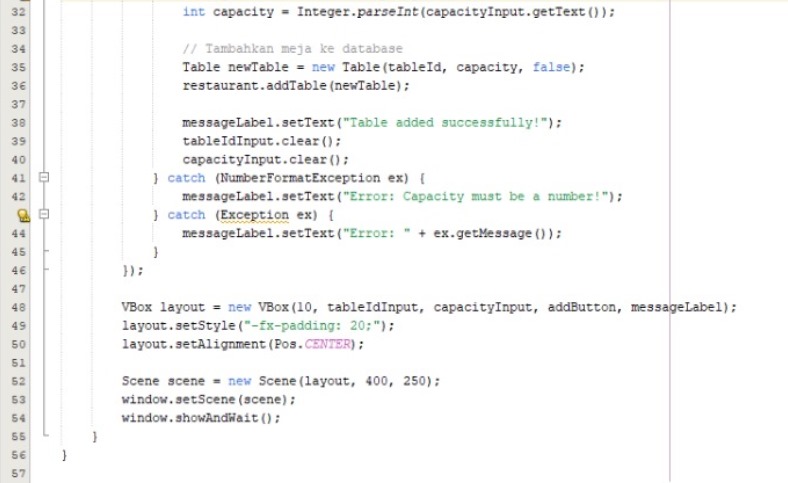
* CancelReservation.java



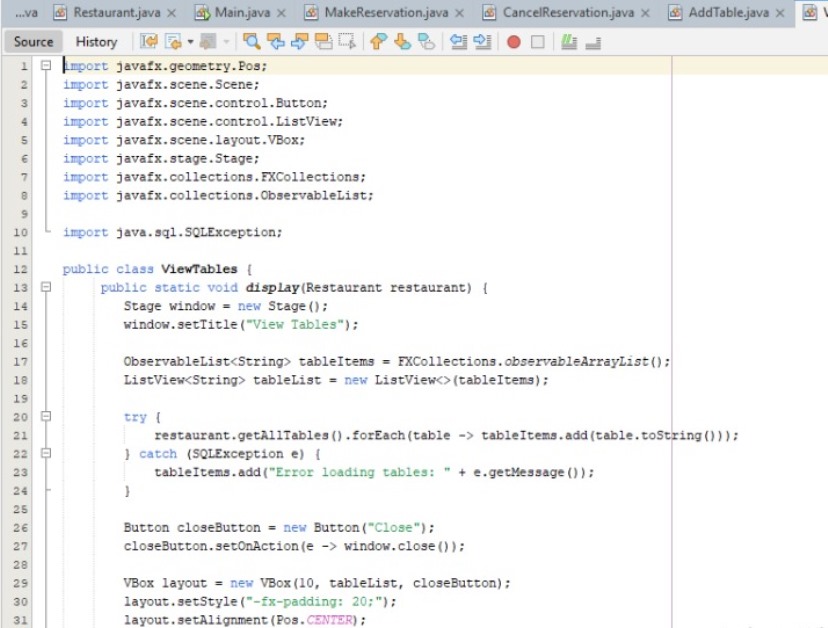


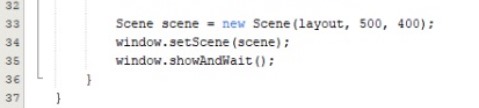
* AddTable.java



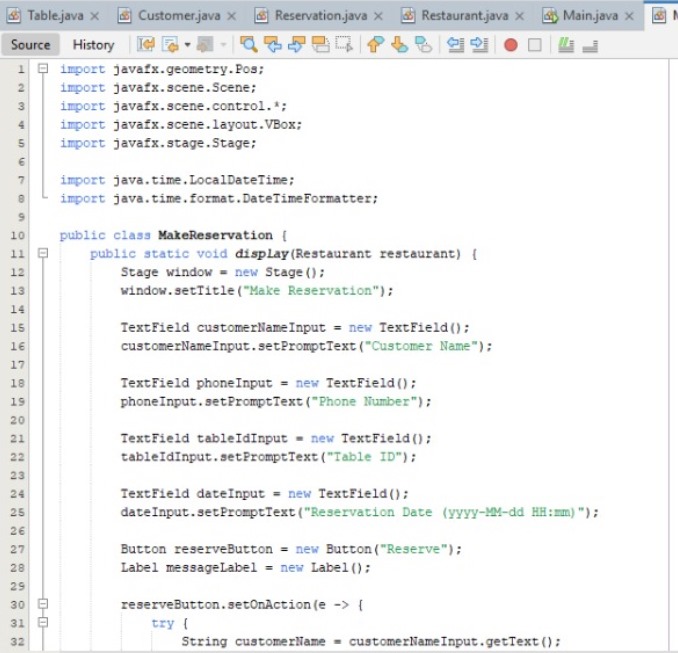


* ViewTables.java





* MakeReservation





**PEMBAHASAN**

Proyek ini dirancang untuk mengelola reservasi restoran secara efektif dengan menggunakan teknologi modern. Sistem ini mempermudah staf restoran dalam mengelola meja, pelanggan, dan reservasi dengan menyediakan antarmuka berbasis GUI (JavaFX) yang intuitif. Selain itu, sistem ini terintegrasi dengan database MySQL untuk memastikan data disimpan dengan aman dan dapat diakses kapan saja.

Fitur utama Melihat Daftar Meja ,Fitur ini memungkinkan pengguna untuk melihat daftar meja yang tersedia di restoran, termasuk informasi seperti ID meja, kapasitas, dan status reservasi.

Menambah Meja Baru, Admin dapat menambahkan meja baru ke database restoran dengan memasukkan ID meja dan kapasitasnya. Meja yang baru ditambahkan akan memiliki status "tidak terpesan".

Membuat Reservasi, Pelanggan dapat melakukan reservasi dengan mengisi data seperti nama, nomor telepon, ID meja, dan waktu reservasi. Sistem secara otomatis menandai meja yang dipesan sebagai "terpesan" dan menyimpan data pelanggan.

Membatalkan Reservasi, Fitur ini memungkinkan pengguna untuk membatalkan reservasi berdasarkan ID reservasi. Setelah reservasi dibatalkan, status meja akan diubah menjadi "tidak terpesan".

Melihat Daftar Reservasi, Sistem menampilkan daftar reservasi yang telah dilakukan, termasuk informasi pelanggan, ID meja, dan waktu reservasi.

Proyek ini menggunakan arsitektur berbasis MVC (Model-View-Controller) untuk memisahkan logika bisnis, data, dan antarmuka pengguna. Berikut adalah deskripsi singkat dari komponen utama:

Model, Berisi kelas yang merepresentasikan data utama, seperti Table, Customer, dan Reservation. Setiap kelas memiliki atribut dan metode untuk memanipulasi data.

Controller, Kelas Restaurant bertindak sebagai controller utama, yang menangani logika aplikasi, seperti membaca/menulis data ke database, membuat reservasi, dan mengubah status meja.

View, Antarmuka GUI dibuat menggunakan JavaFX, dengan kelas seperti ViewTables, AddTable, MakeReservation, dan CancelReservation. Setiap kelas ini bertanggung jawab untuk menampilkan data kepada pengguna dan menangkap input pengguna.

Teknologi yang Digunakan :

Java, Bahasa pemrograman utama yang digunakan untuk membangun aplikasi ini.

JavaFX, Framework untuk membuat antarmuka pengguna yang interaktif dan user-friendly.

MySQL, Database relasional yang digunakan untuk menyimpan data pelanggan, meja, dan reservasi.

JDBC (Java Database Connectivity), Library untuk menghubungkan aplikasi Java dengan database MySQL.

Sistem ini menggunakan tiga tabel utama dalam database:

**Tabel tables**

Kolom: tableId (String), capacity (Integer), isReserved (Boolean)

Deskripsi: Menyimpan data tentang meja di restoran.

**Tabel customers**

Kolom: customerId (Integer), name (String), phoneNumber (String)

Deskripsi: Menyimpan data pelanggan yang melakukan reservasi.

**Tabel reservations**

Kolom: reservationId (Integer), customerId (Foreign Key), tableId (Foreign Key), reservationDate (Timestamp)

Deskripsi: Menyimpan data reservasi, termasuk informasi pelanggan dan meja yang dipesan.

Alur Kerja Sistem yakni Admin memulai aplikasi, Admin dapat memilih fitur utama, seperti Melihat daftar meja yang tersedia, Menambah meja baru ke database, Membuat reservasi pelanggan, Membatalkan reservasi berdasarkan ID reservasi.

Sistem berkomunikasi dengan database MySQL untuk membaca/mengubah data sesuai permintaan pengguna.

Perubahan data (seperti status meja atau penambahan reservasi) disinkronkan antara database dan antarmuka GUI.

Data ditampilkan kembali kepada pengguna dalam bentuk daftar meja atau daftar reservasi.

Keunggulan Sistem,seperti User-Friendly Antarmuka berbasis JavaFX dirancang agar mudah digunakan bahkan oleh pengguna non-teknis. Efisien, Mengurangi kebutuhan pencatatan manual dan mempercepat proses reservasi. Real-Time Updates, Data selalu diperbarui secara real-time melalui koneksi ke database MySQL. Modularitas, Struktur berbasis objek dan MVC membuat sistem ini mudah untuk diubah atau ditambahkan fitur baru di masa depan.

**PENUTUP**

**KESIMPULAN**

Proyek Sistem Reservasi Restoran yang telah dikembangkan bertujuan untuk menyediakan solusi berbasis teknologi dalam mengelola reservasi meja secara lebih terstruktur, efisien, dan akurat. Sistem ini menggabungkan kemampuan pemrograman berbasis Java dengan antarmuka grafis JavaFX dan penyimpanan data menggunakan database MySQL. Dengan pendekatan arsitektur MVC (Model-View-Controller), sistem berhasil memisahkan logika bisnis, antarmuka pengguna, dan pengelolaan data sehingga lebih mudah dikembangkan dan dikelola di masa depan.

Beberapa poin kesimpulan penting dari proyek ini adalah sebagai berikut:

Efisiensi Pengelolaan Data

Sistem ini mampu meminimalkan kesalahan yang sering terjadi dalam proses manual, seperti kesalahan pencatatan meja yang terpesan atau duplikasi data pelanggan. Dengan integrasi MySQL, data tersimpan secara terstruktur, dapat diakses kapan saja, dan mudah diperbarui.

Kemudahan Penggunaan

Antarmuka pengguna berbasis JavaFX dirancang agar intuitif, sehingga staf restoran dapat dengan mudah mengelola meja, pelanggan, dan reservasi. Fitur-fitur seperti pembuatan reservasi, pembatalan reservasi, dan penambahan meja baru dapat diakses dengan langkah-langkah sederhana.

Fleksibilitas dan Skalabilitas

Sistem dirancang dengan struktur modular sehingga dapat dikembangkan lebih lanjut, misalnya dengan menambahkan fitur baru seperti integrasi pembayaran, sistem notifikasi, atau pelaporan statistik. Selain itu, kemampuan sistem untuk mengelola banyak data memungkinkan implementasi pada restoran dengan skala besar.

Penerapan Teknologi yang Relevan

Penggunaan Java sebagai bahasa pemrograman utama memberikan stabilitas dan fleksibilitas, sementara MySQL sebagai sistem basis data memastikan data dikelola secara terorganisir. Dengan JDBC sebagai penghubung, sistem mampu mengolah data secara real-time.

Manfaat untuk Bisnis Restoran

Implementasi sistem ini mendukung peningkatan produktivitas restoran dengan mengurangi waktu yang dihabiskan untuk pencatatan manual dan mempermudah staf dalam memastikan meja terorganisir dengan baik. Hal ini juga berpotensi meningkatkan kepuasan pelanggan karena reservasi dapat dilakukan dengan lebih cepat dan akurat.

**DAFTAR PUSTAKA**

Course-Net, M.

Course-Net, M. (2023) *Netbeans Adalah: Pengertian , Definisi & Kelebihan Kekurangan*, *Course*. Available at: https://course-net.com/blog/netbeans-adalah/ (Accessed: 19 December 2024).

*NetBeans IDE*, www.oracle.com/tools/technologies/netbeans-ide.html. Accessed 19 Dec. 2024.

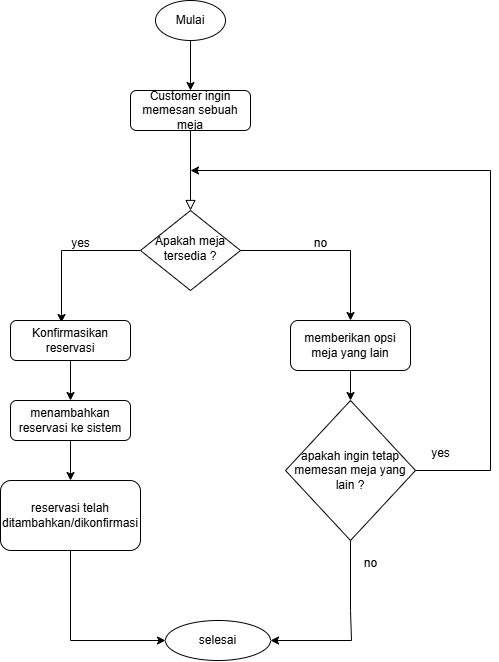
NetBeans, A.

NetBeans, Apache. “Apache NetBeans 24.” About Apache NetBeans, netbeans.apache.org/front/main/about/. Accessed 19 Dec. 2024.

aws.amazon.com/id/what-is/java/. Accessed 19 Dec. 2024.

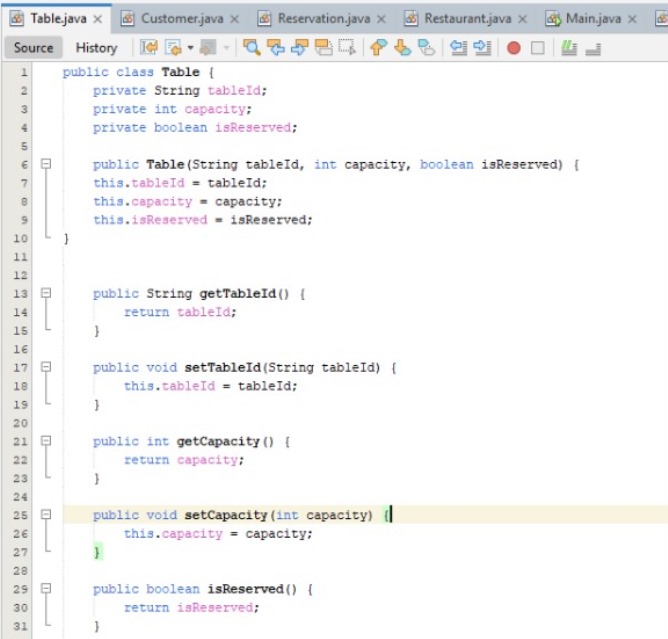
**LAMPIRAN 1**

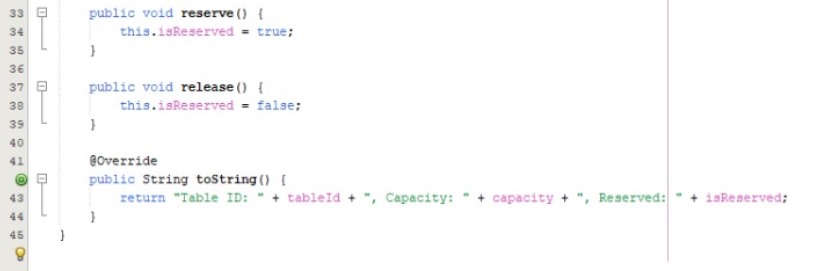
* **Flowchart**

****

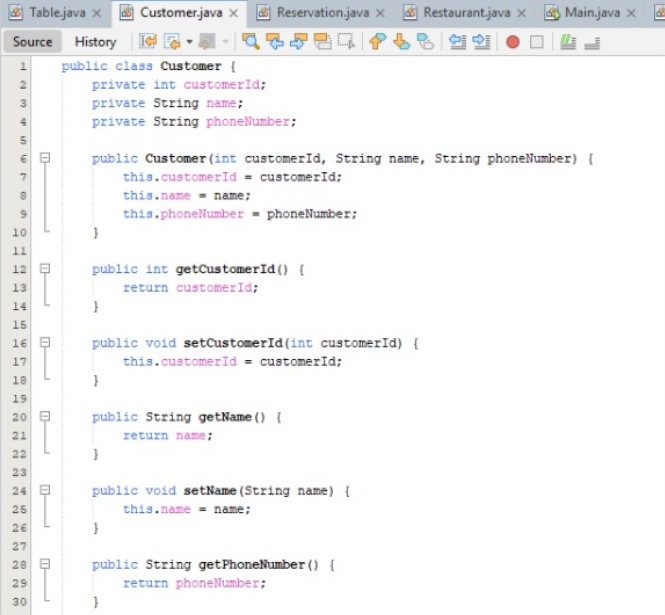
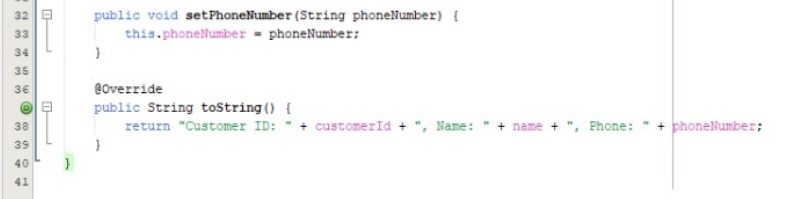
**LAMPIRAN 2**

* **Class Table.java**

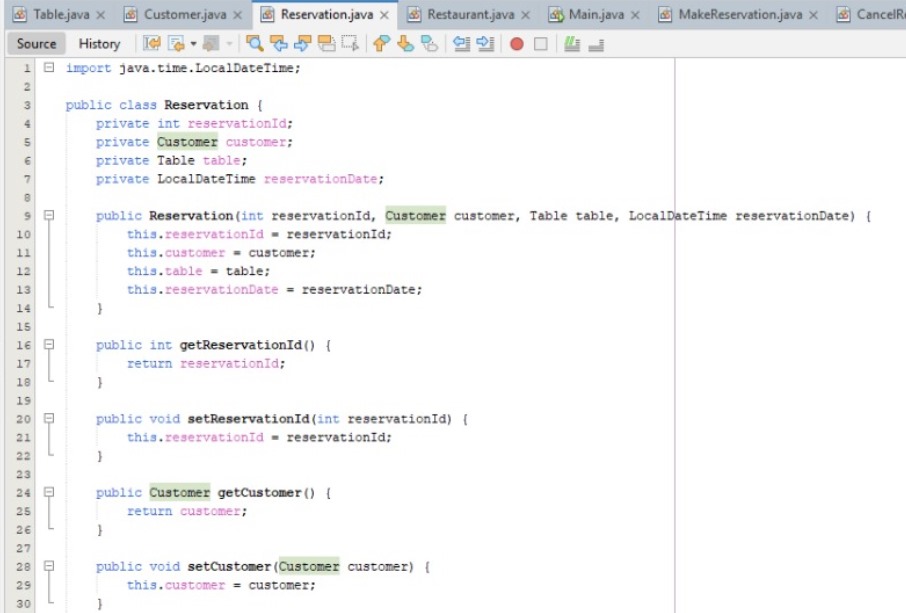
****

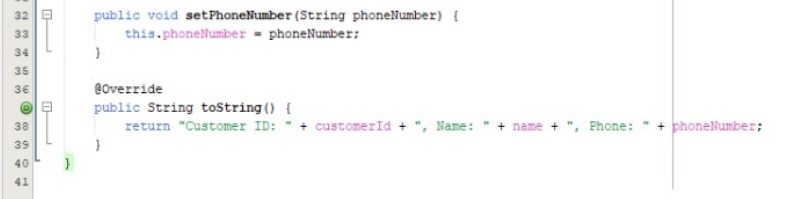
****

* Class Customer.java

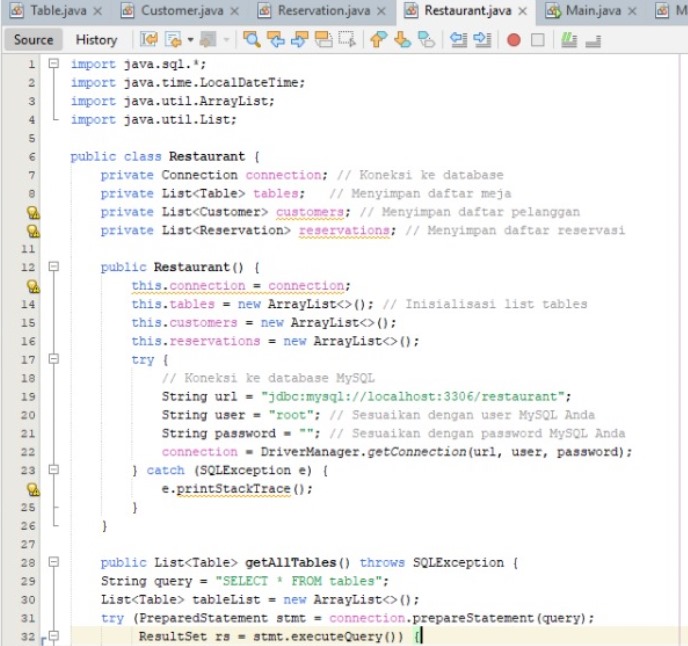


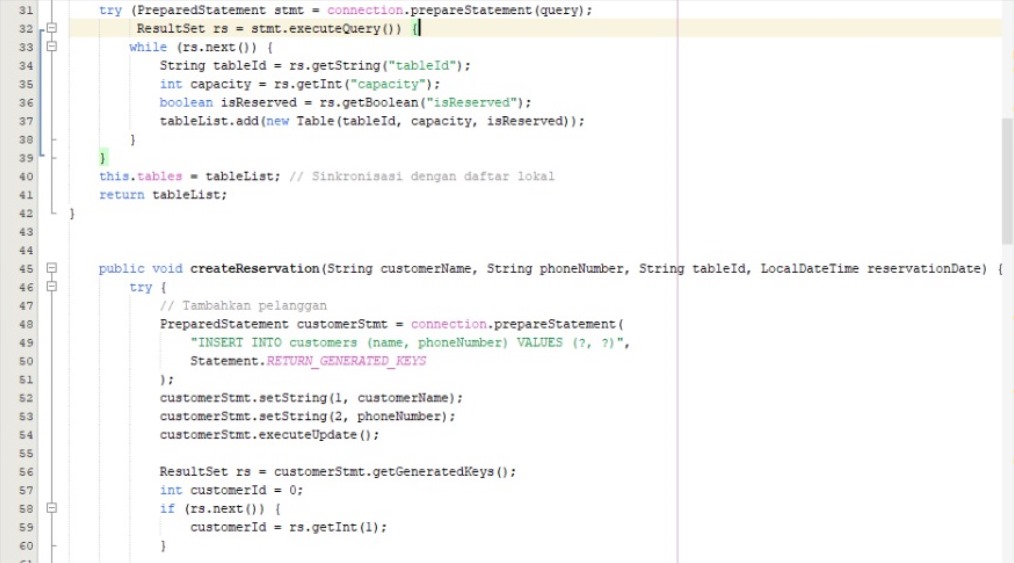
* Class Reservasi.java



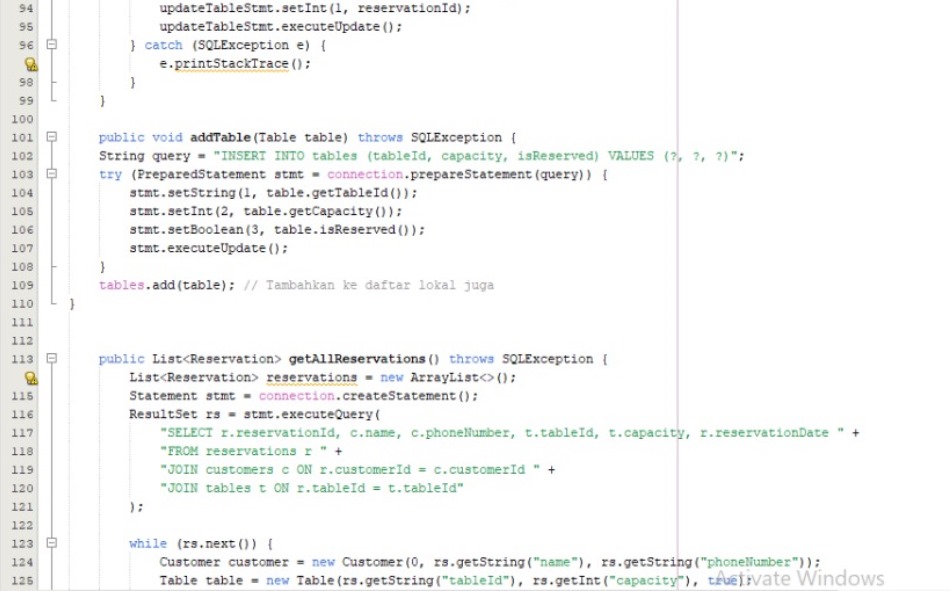


* Class Restaurant.java



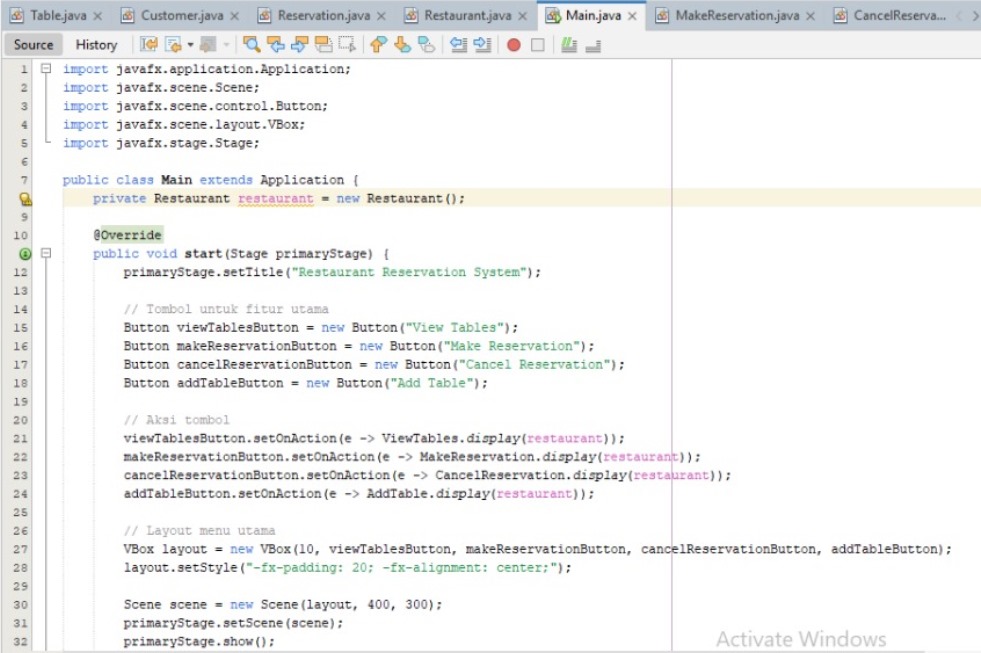


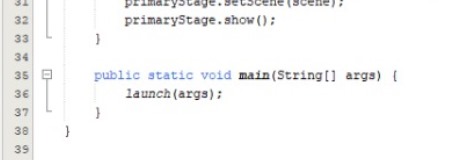


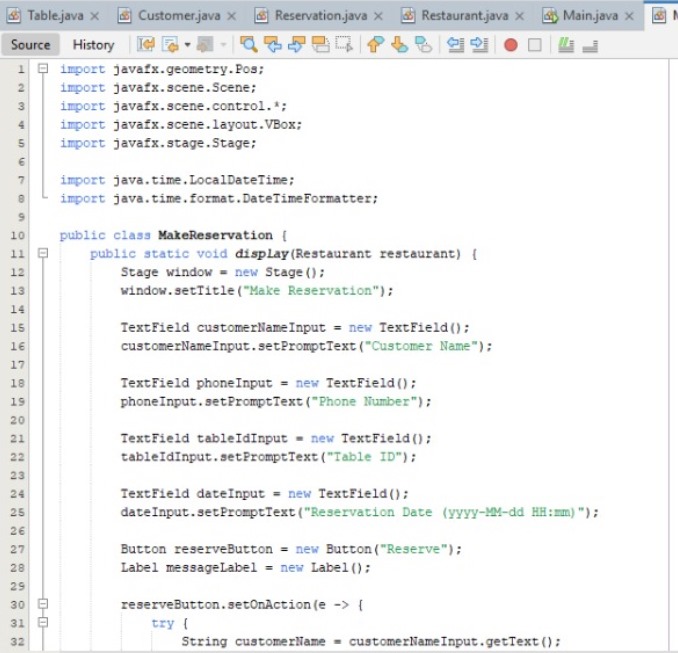




* Class Main.java

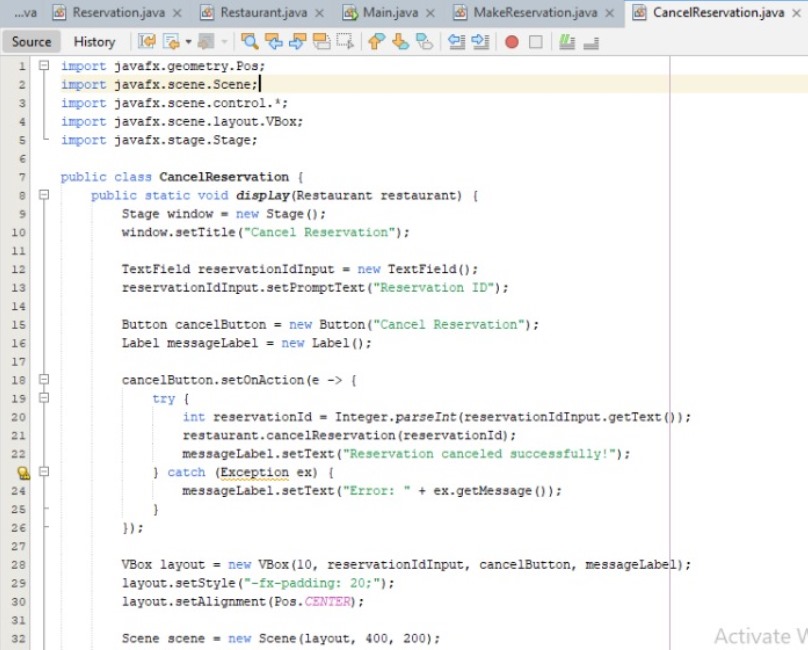


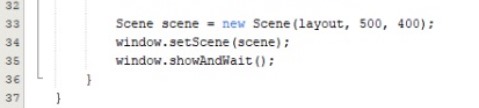


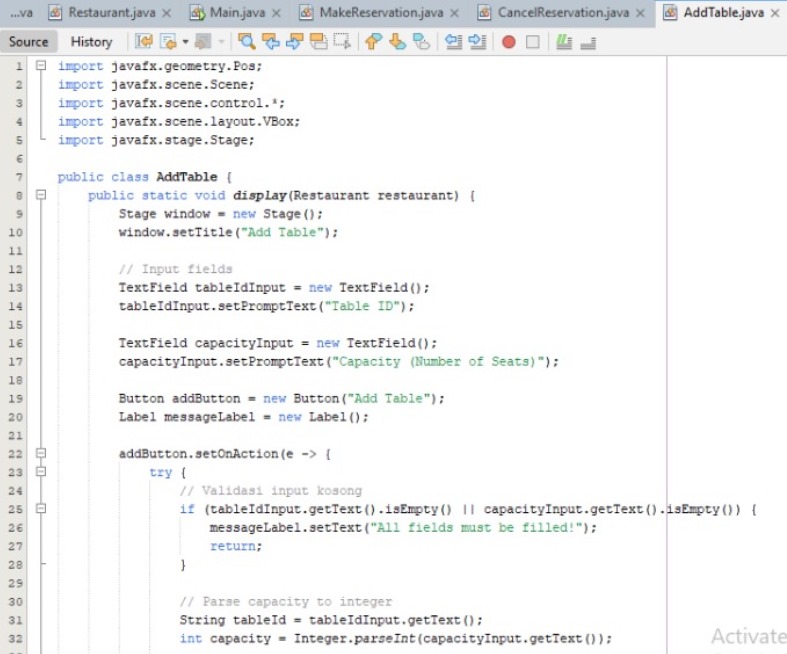
* Class MakeReservation.java

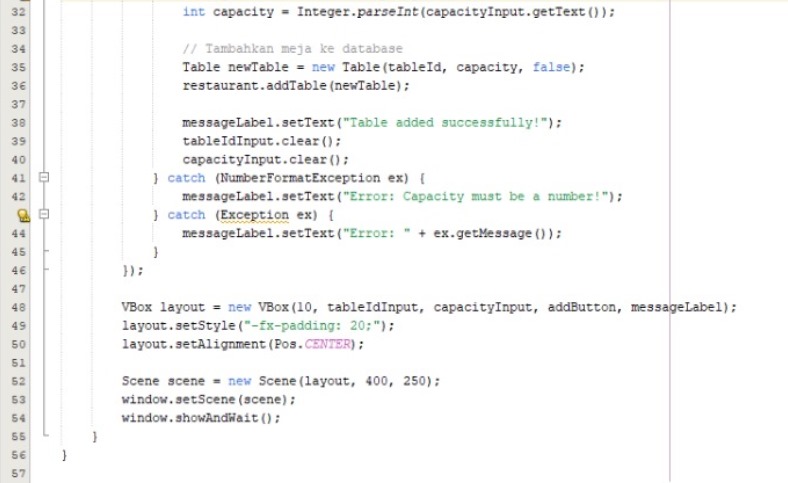


* Class CancelReservation.java

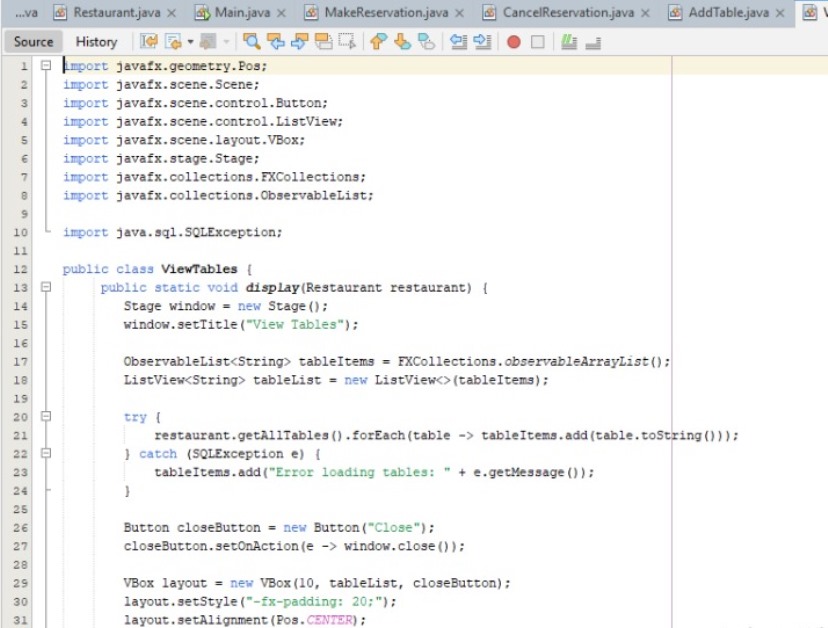


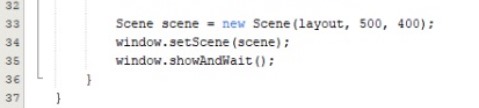


* Class AddTable.java



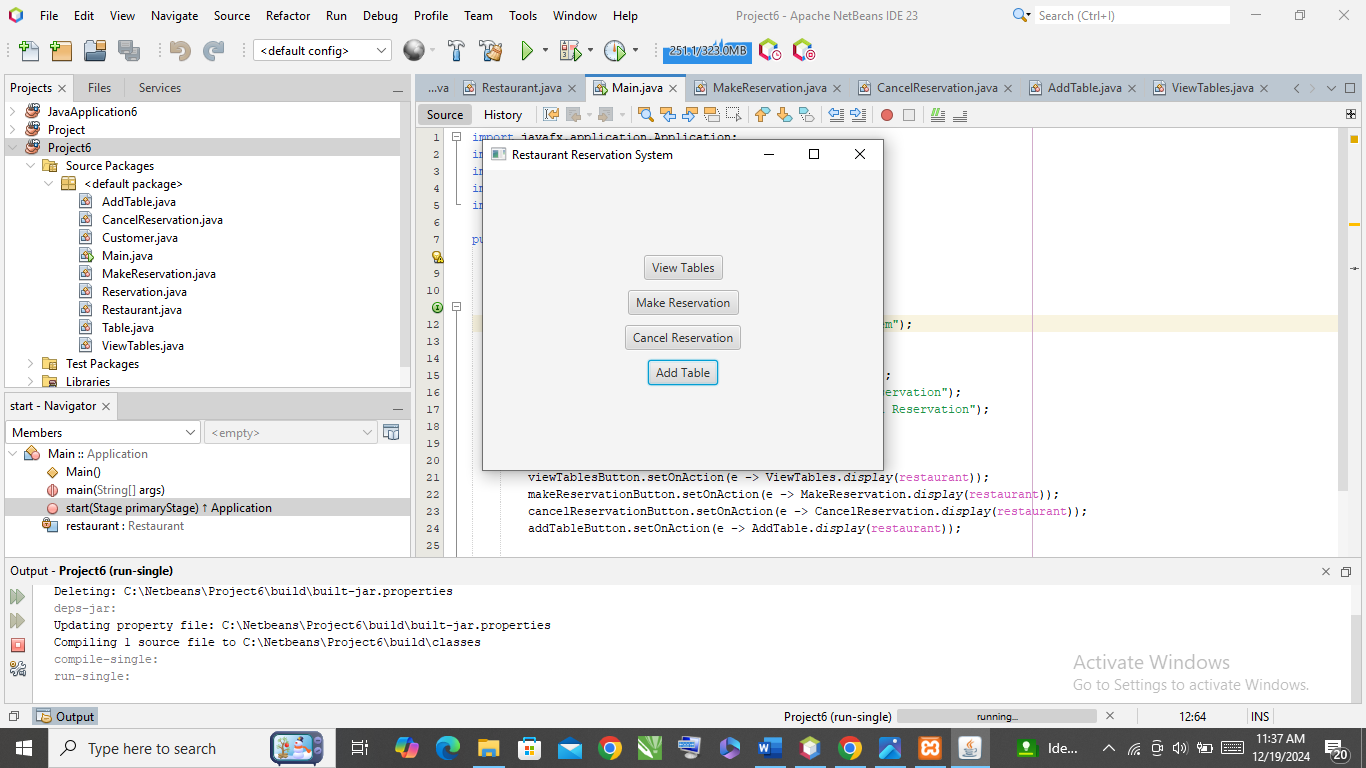
* Class ViewTable.java



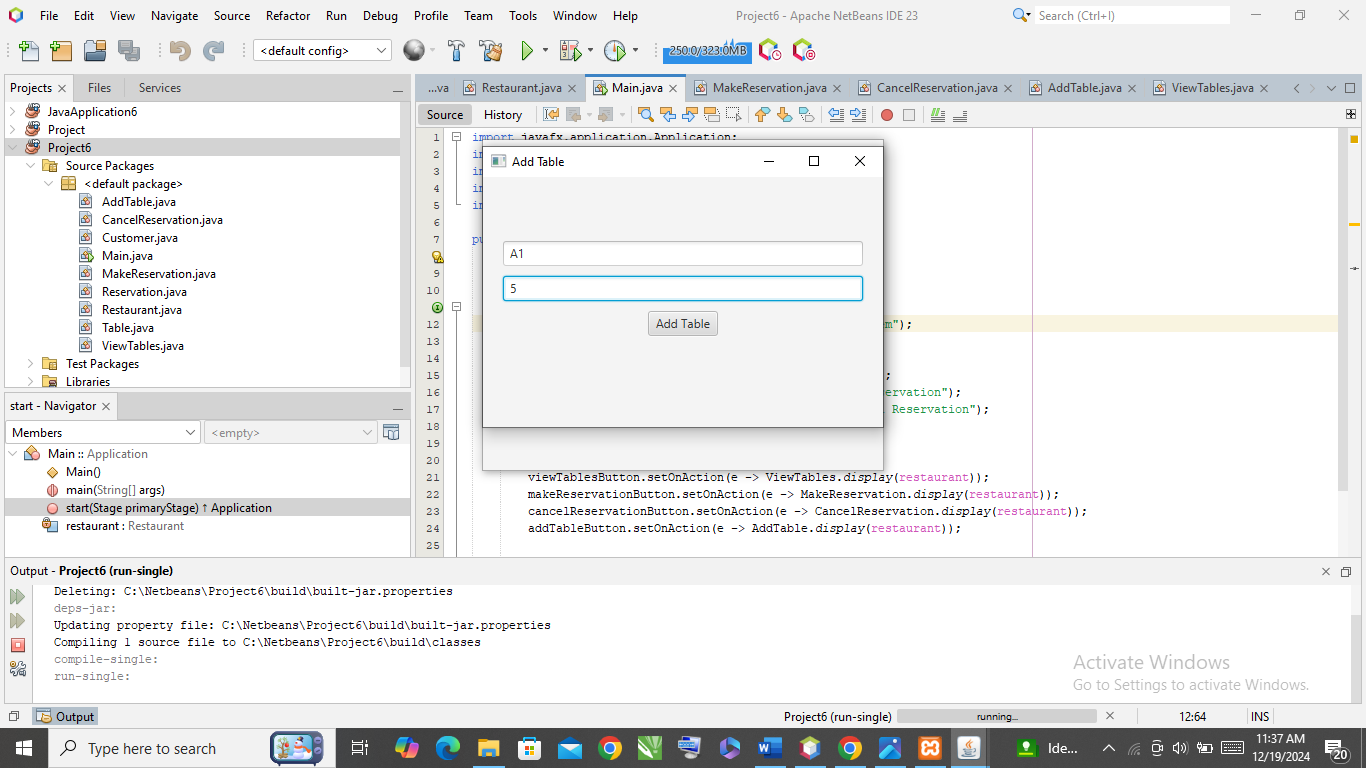


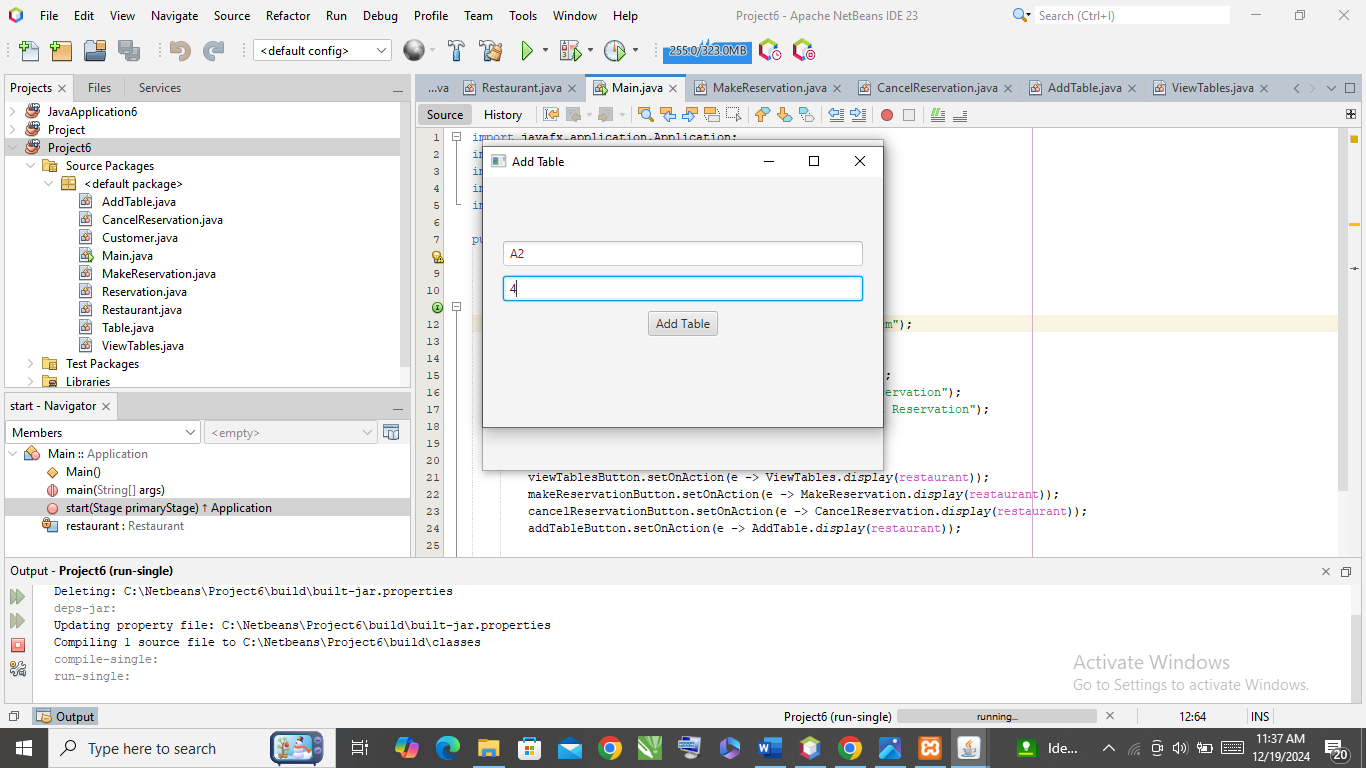
**LAMPIRAN 3**

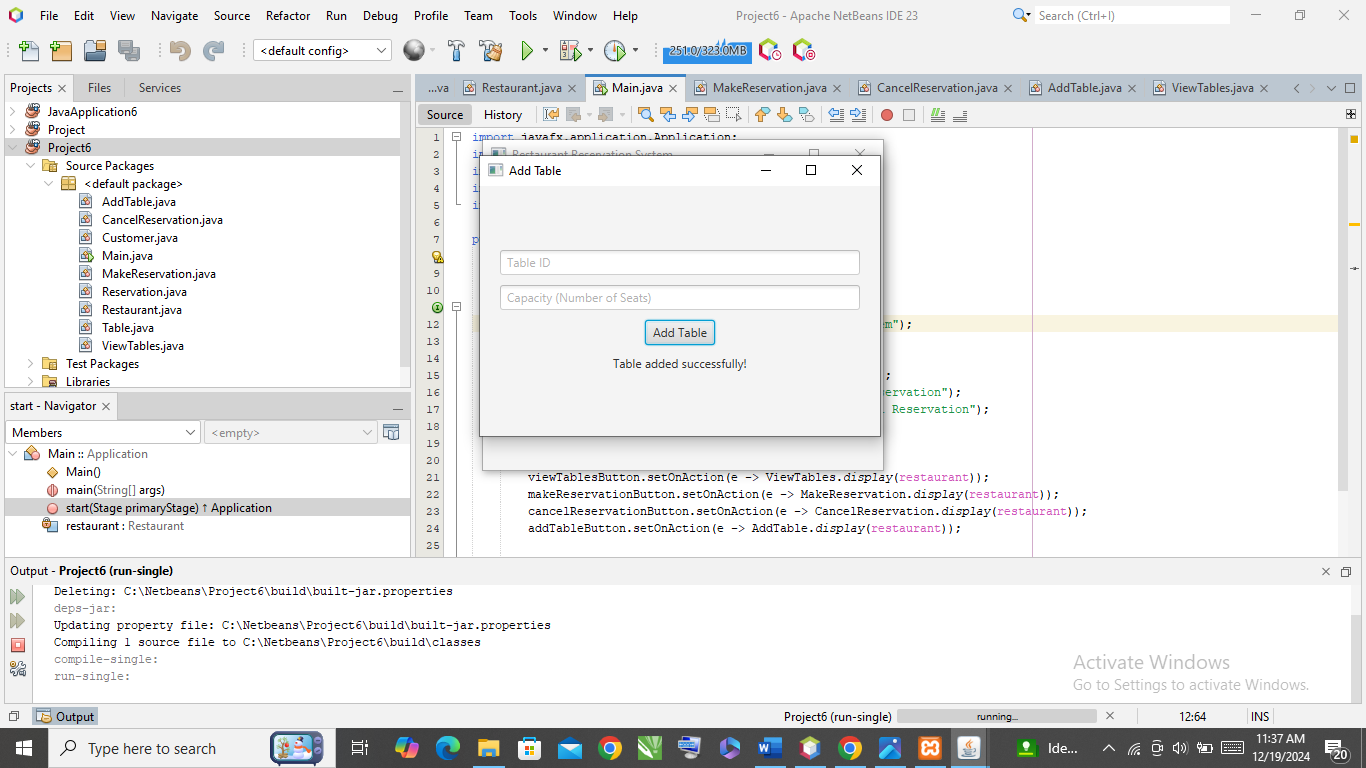
* **Tampilan awal**

****

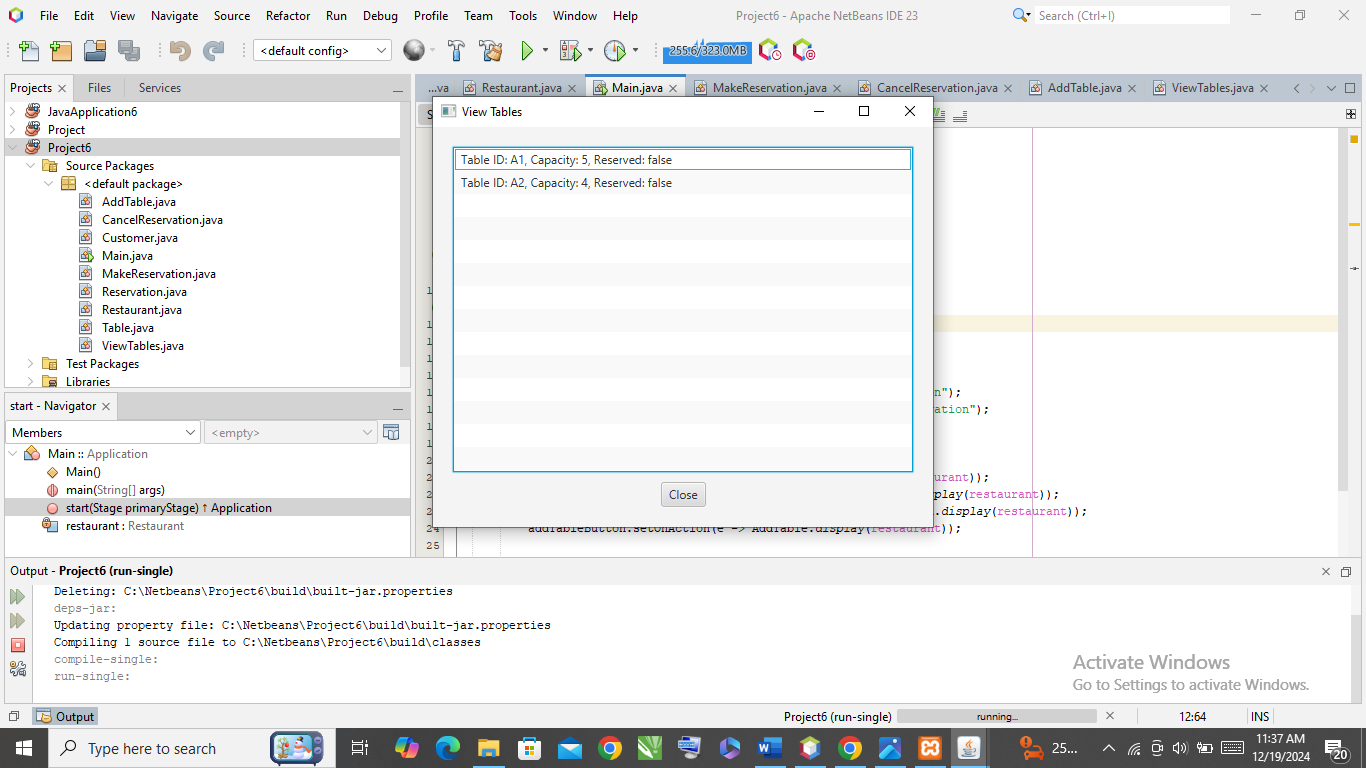
* **Menambahkan Meja**

****

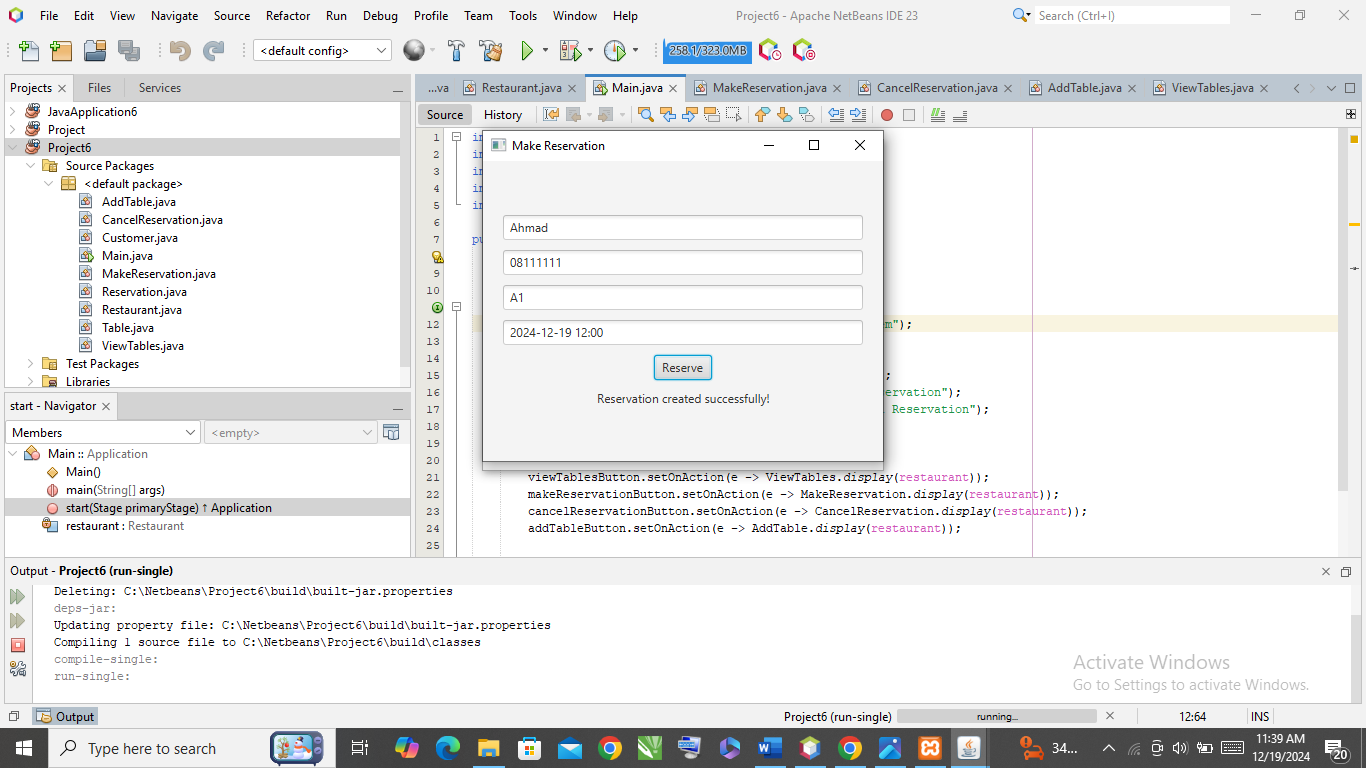




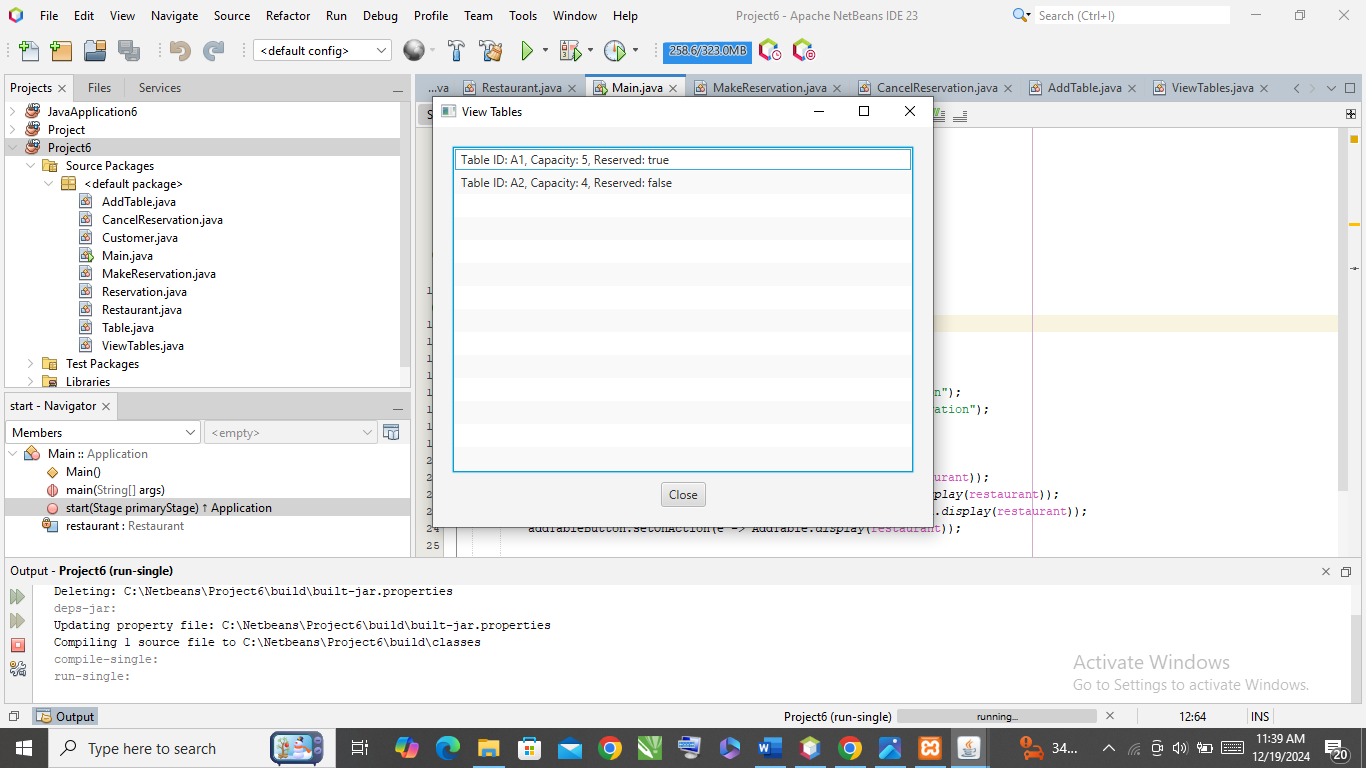
* **Tampilan Meja Saat Meja telah di tambahkan**



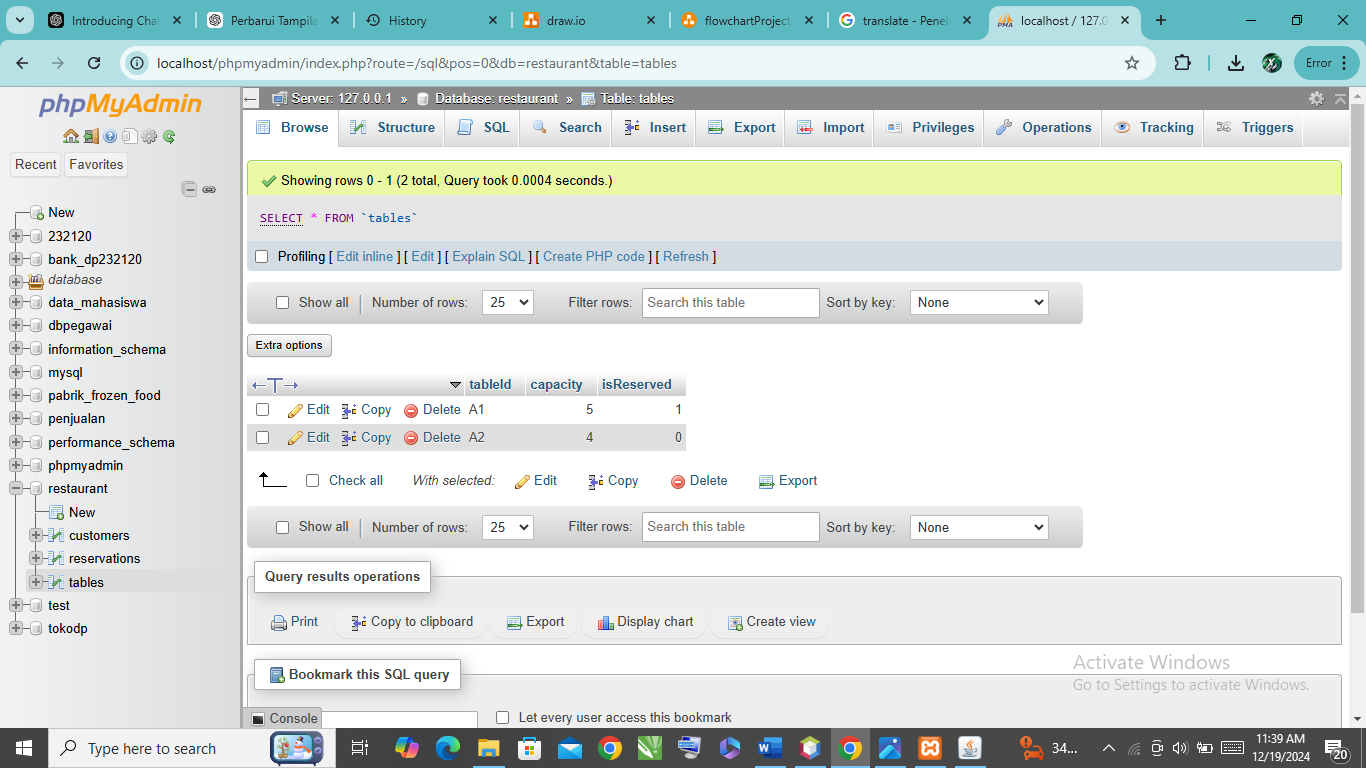
* **Membuat Reservasi**



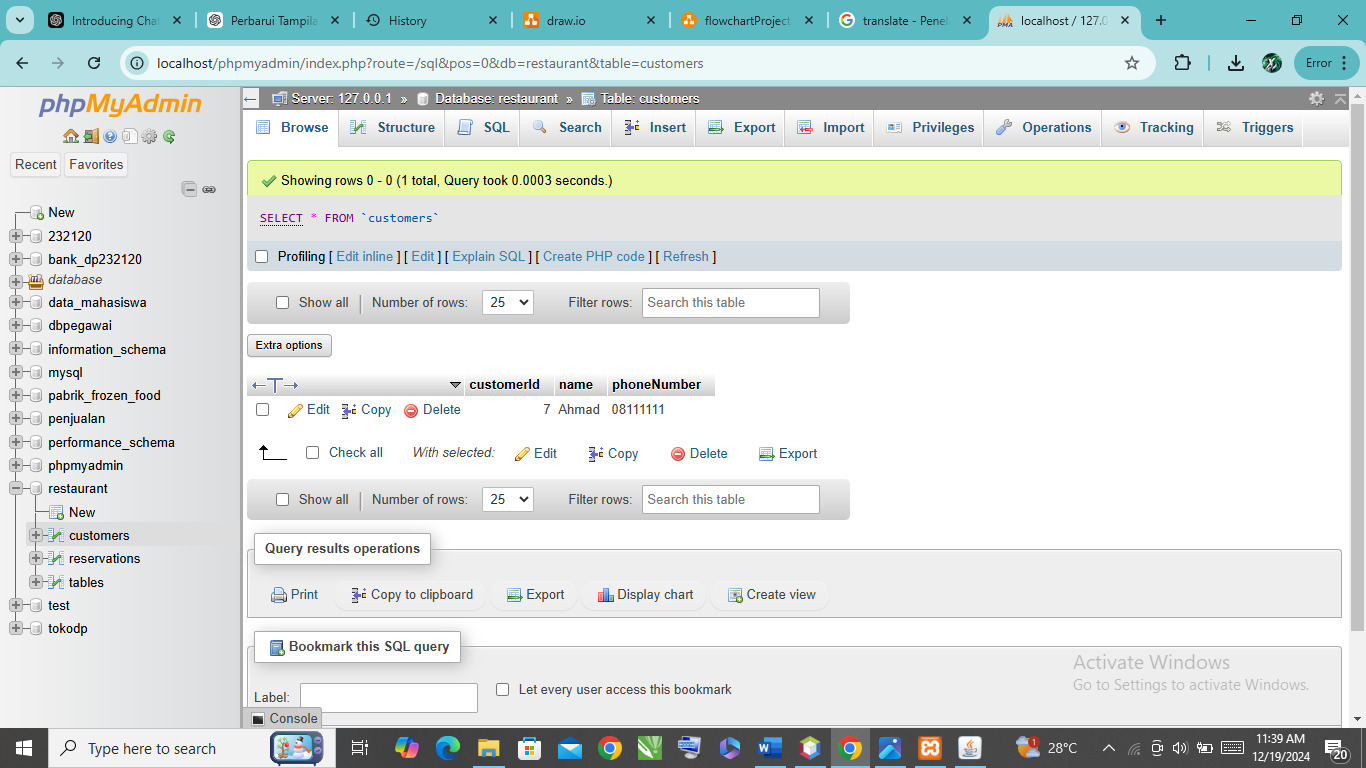
* **Tampilan View tables Setelah Menambahkan Reservasi**

****

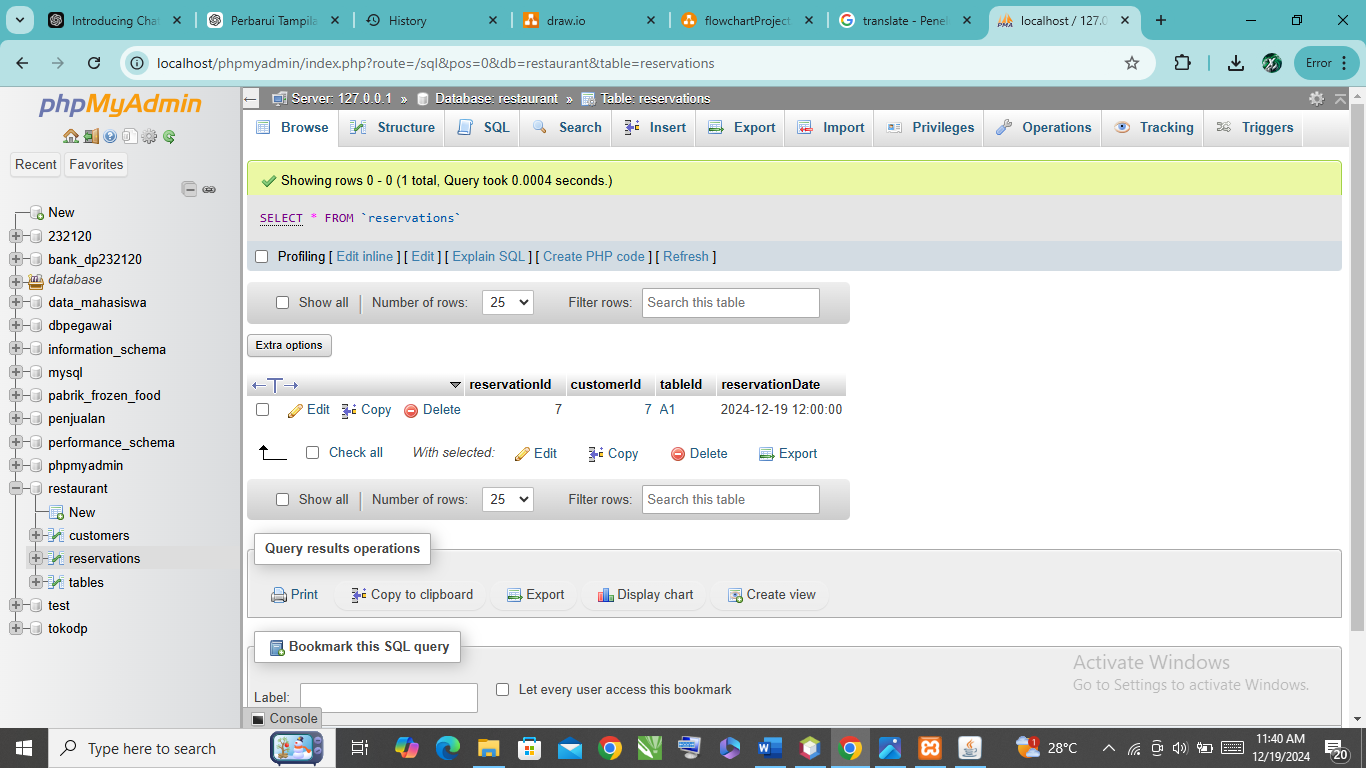
* **Tampilan Database Pada Tabel Tables**

****

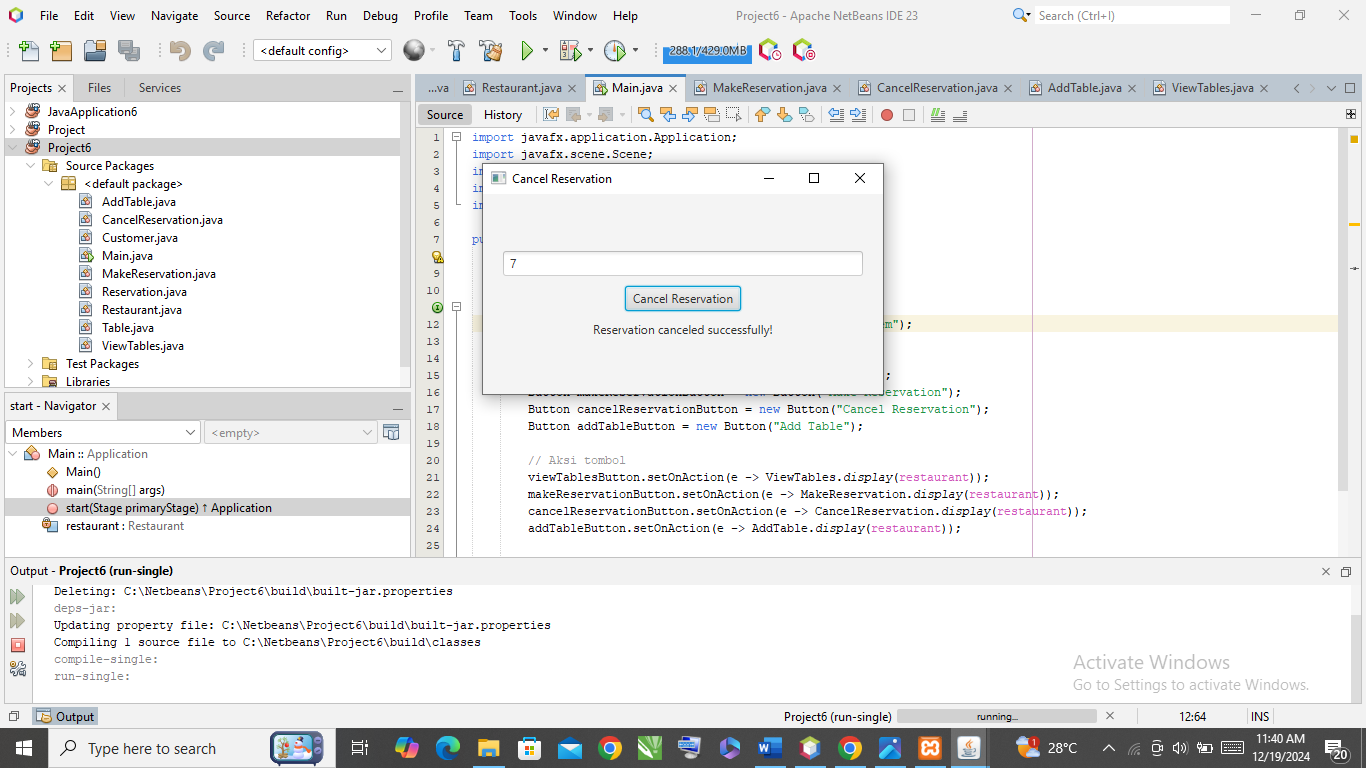
* **Tampilan Database Pada Tabel Customers**



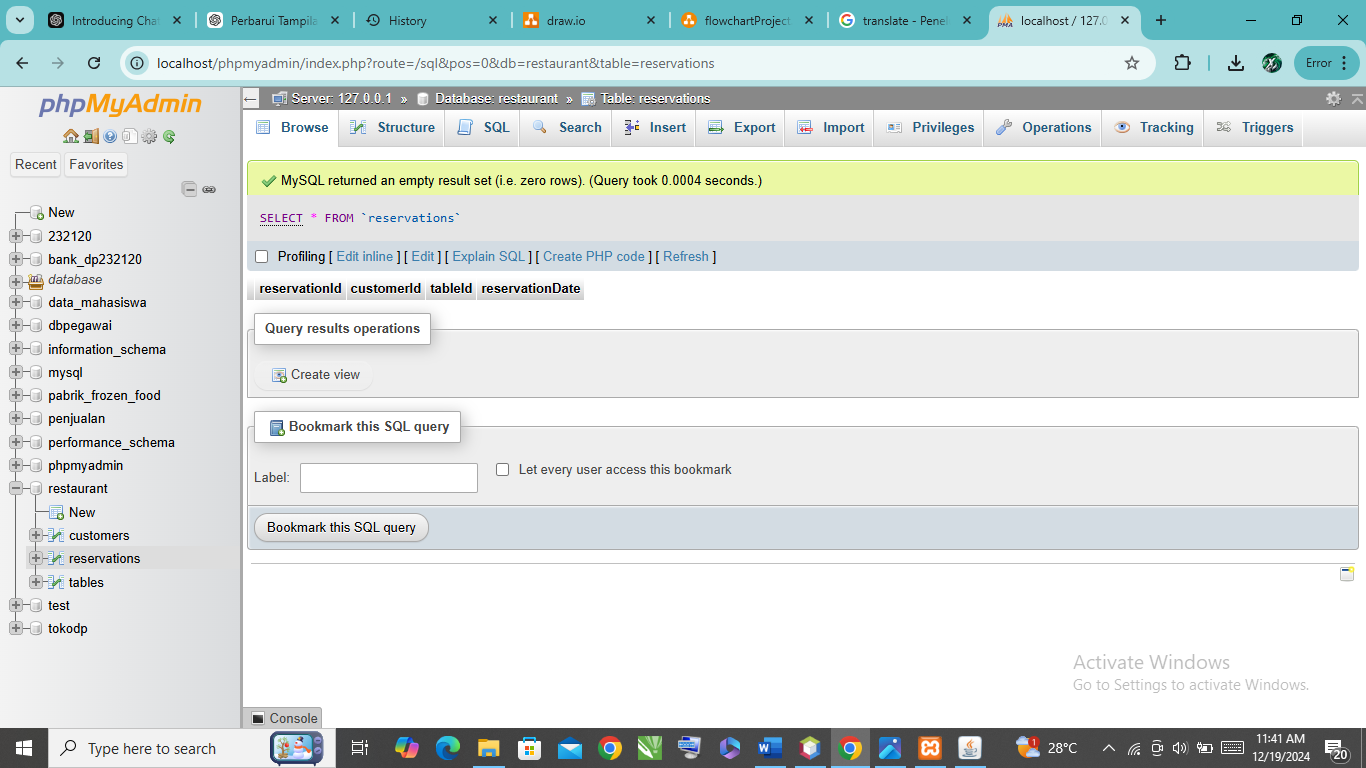
* **Tampilan Database Pada Tabel Reservations**



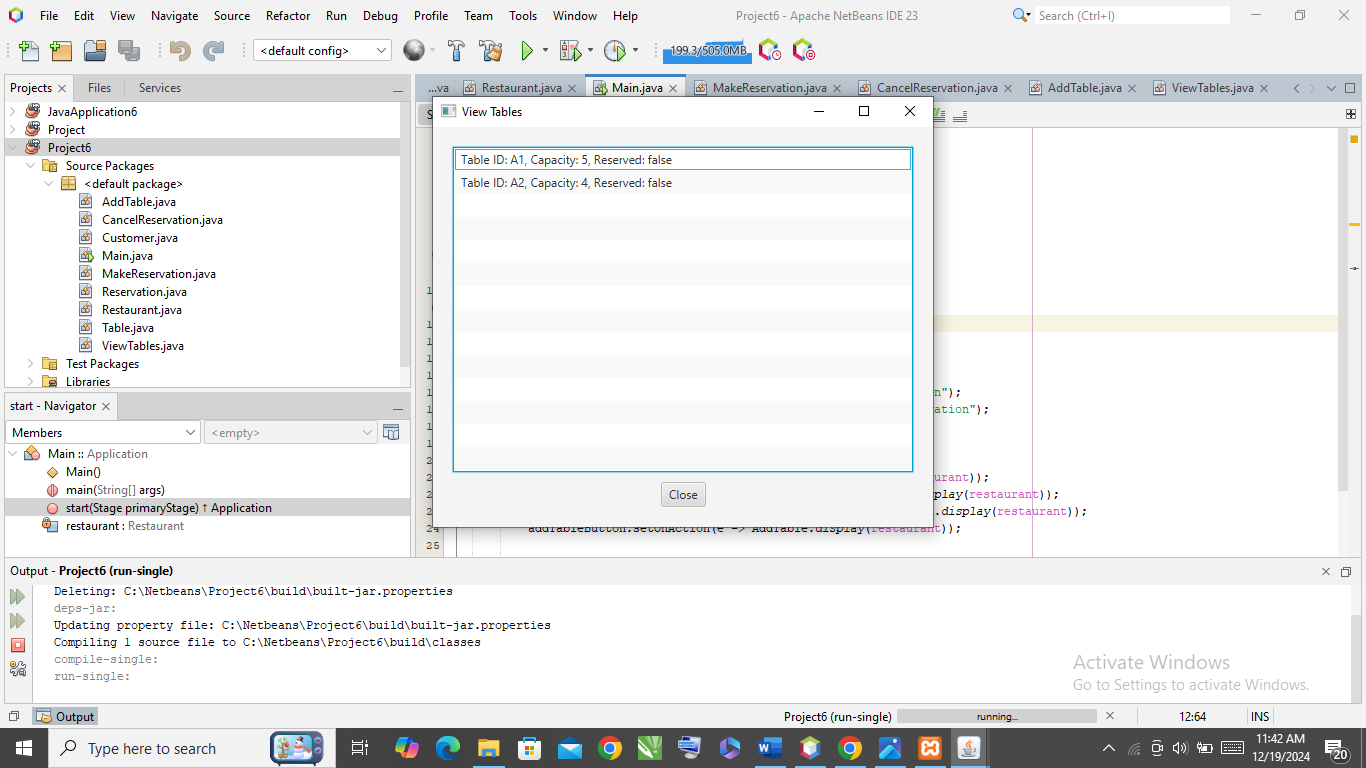
* **Membatalkan Reservasi, sesuaikan dengan reservationid pada database**

****

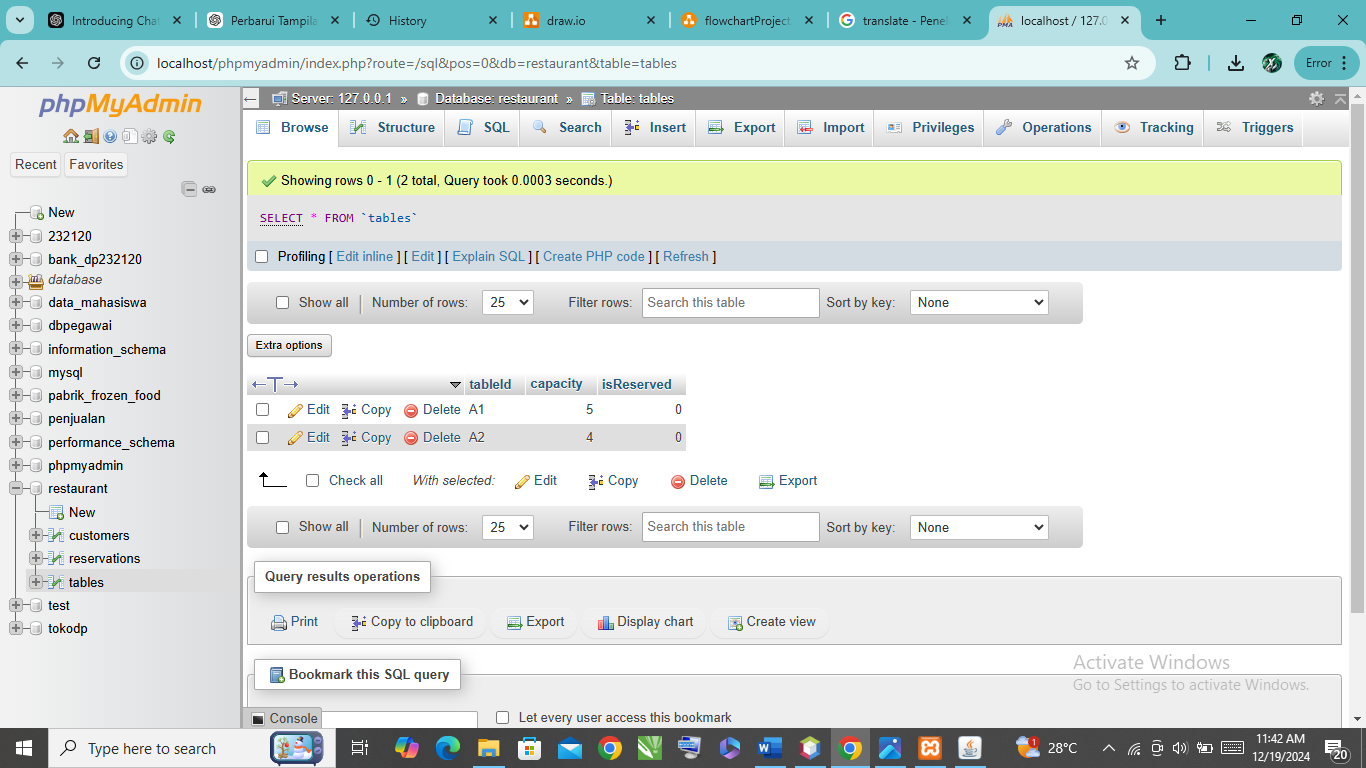
* **Tampilan di database pada table reservations setelah membatalkan reservasi**

****

* **Tampilan di view tables setelah membatalkan reservasi**



* **Tampilan database pada tabel tables setelah membatalkan reservasi**

****