**ASSIGNMENT # 04**

**CUI Cafe Management System**

**SP24-BSE-020**

**FA23-BSE-039**

**SP24-BSE-100**

**CUI Cafe Management System**

**Overview**

The CUI Cafe Management System is a JavaFX-based application designed to facilitate the management of a cafe. It allows students, faculty, and staff to place orders, view menus, and submit reviews. The system consists of a server that handles requests from multiple clients (students and faculty) and an admin interface for managing orders and menu items. The application uses two text files for data storage: one for the cafe menu and another for customer reviews.

**Key Components**

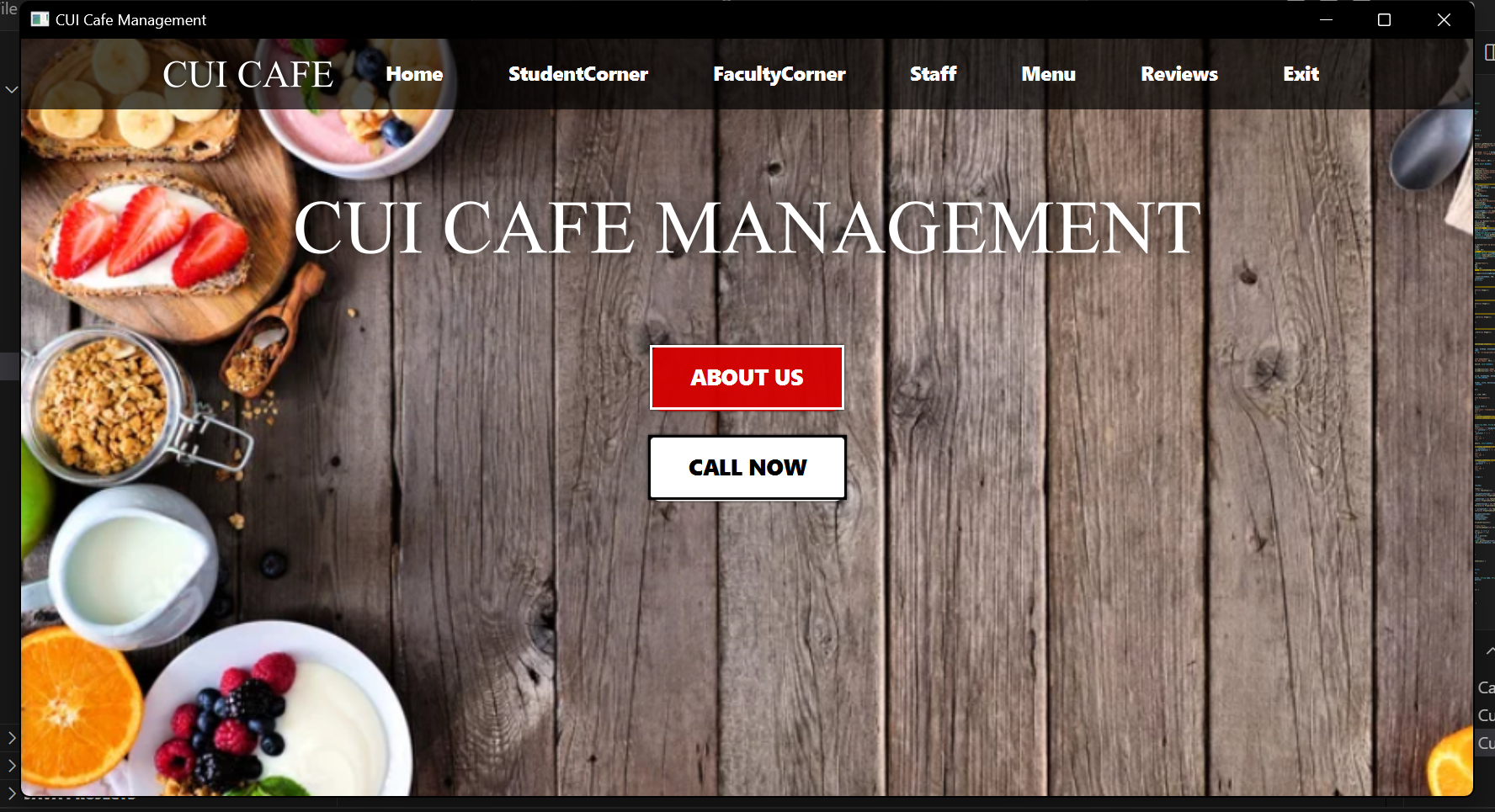
1. **Server**: Manages connections from multiple clients, processes orders, and sends bills.
2. **Client Applications**:
   * **StudentClient**: Allows students to view the menu, place orders, and submit reviews.
   * **FacultyClient**: Similar to StudentClient but tailored for faculty members.
   * **AdminClient**: Provides an interface for administrators to view orders and generate bills.
3. **Data Storage**:
   * **cafemenu.txt**: Contains the menu items, their quantities, and prices.
   * **CafeReviews.txt**: Stores customer reviews, including designation, name, comments, and ratings.

**Important Methods**

* **CuiCafe**: The main application class that sets up the user interface and navigation.
* **CafeReviews**: Handles the display and filtering of reviews.
* **AdminClient**: Connects to the server, retrieves orders, and generates bills.
* **StudentClient** and **FacultyClient**: Connect to the server to place orders and receive bills.

**Steps to Run the Application**

1. **Set Up the Server**:
   * Create a server class that listens for incoming connections from clients.
   * Handle incoming messages and manage orders.
2. **Create the Text Files**:
   * Create **cafemenu.txt** with the following format:
   * Create **CafeReviews.txt** with the following format:
3. **Compile and Run the Server**:
   * Compile the server code and run it to start listening for client connections.
4. **Compile and Run the Clients**:
   * Compile the **StudentClient**, **FacultyClient**, and **AdminClient** classes.
   * Run the clients to interact with the server.

****

**Source Code :**

**AdminClient:**

import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.HBox;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.io.\*;

import java.net.Socket;

import java.util.HashMap;

import java.util.Map;

public class AdminClient extends Application {

private static final String SERVER\_ADDRESS = "localhost";

private static final int SERVER\_PORT = 12345;

private static final String ADMIN\_PASSWORD = "123";

private PrintWriter out;

private TableView<Order> studentOrdersTable;

private TableView<Order> facultyOrdersTable;

private Map<String, MenuItem> menuItems;

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("CUI Cafe Admin Client");

// Load menu items and their quantities

loadMenuItems();

//Password prompt

TextInputDialog passwordDialog = new TextInputDialog();

passwordDialog.setTitle("Admin Login");

passwordDialog.setHeaderText("Enter Admin Password");

passwordDialog.setContentText("Password:");

passwordDialog.showAndWait().ifPresent(password -> {

if (ADMIN\_PASSWORD.equals(password)) {

showAdminInterface(primaryStage);

new Thread(this::connectToServer).start();

} else {

Alert alert = new Alert(Alert.AlertType.ERROR);

alert.setTitle("Error");

alert.setHeaderText("Invalid Password");

alert.setContentText("The password you entered is incorrect.");

alert.showAndWait();

}

});

}

private void showAdminInterface(Stage primaryStage) {

VBox root = new VBox(10);

root.setStyle("-fx-padding: 10;");

HBox ordersBox = new HBox(10);

VBox studentBox = new VBox(10);

Label studentOrdersLabel = new Label("Student Orders:");

studentOrdersTable = new TableView<>();

setupOrderTable(studentOrdersTable);

studentBox.getChildren().addAll(studentOrdersLabel, studentOrdersTable);

VBox facultyBox = new VBox(10);

Label facultyOrdersLabel = new Label("Faculty Orders:");

facultyOrdersTable = new TableView<>();

setupOrderTable(facultyOrdersTable);

facultyBox.getChildren().addAll(facultyOrdersLabel, facultyOrdersTable);

ordersBox.getChildren().addAll(studentBox, facultyBox);

Button generateBillButton = new Button("Generate Bill");

generateBillButton.setOnAction(e -> generateBill());

root.getChildren().addAll(ordersBox, generateBillButton);

Scene scene = new Scene(root, 800, 600);

primaryStage.setScene(scene);

primaryStage.show();

}

private void setupOrderTable(TableView<Order> table) {

TableColumn<Order, String> itemColumn = new TableColumn<>("Item");

itemColumn.setCellValueFactory(new PropertyValueFactory<>("item"));

itemColumn.setPrefWidth(200);

TableColumn<Order, Integer> quantityColumn = new TableColumn<>("Quantity");

quantityColumn.setCellValueFactory(new PropertyValueFactory<>("quantity"));

quantityColumn.setPrefWidth(100);

table.getColumns().addAll(itemColumn, quantityColumn);

}

private void connectToServer() {

try {

Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);

out = new PrintWriter(socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String message;

while ((message = in.readLine()) != null) {

System.out.println("Received from server: " + message);

if (message.startsWith("Order: Student")) {

String[] parts = message.split(":");

String item = parts[2].trim();

int quantity = Integer.parseInt(parts[3].trim());

studentOrdersTable.getItems().add(new Order(item, quantity));

} else if (message.startsWith("Order: Faculty")) {

String[] parts = message.split(":");

String item = parts[2].trim();

int quantity = Integer.parseInt(parts[3].trim());

facultyOrdersTable.getItems().add(new Order(item, quantity));

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

private void generateBill() {

ObservableList<Order> studentOrders = studentOrdersTable.getItems();

ObservableList<Order> facultyOrders = facultyOrdersTable.getItems();

String studentBill = calculateBill(studentOrders);

String facultyBill = calculateBill(facultyOrders);

// Send bills to clients

sendMessage("Bill: Student: " + studentBill);

sendMessage("Bill: Faculty: " + facultyBill);

// Clear orders and bills

studentOrdersTable.getItems().clear();

facultyOrdersTable.getItems().clear();

}

private String calculateBill(ObservableList<Order> orders) {

StringBuilder bill = new StringBuilder();

int total = 0;

for (Order order : orders) {

String item = order.getItem();

int quantity = order.getQuantity();

if (menuItems.containsKey(item) && menuItems.get(item).getQuantity() >= quantity) {

int price = menuItems.get(item).getPrice();

total += quantity \* price;

bill.append(item).append(" x ").append(quantity).append(" = $").append(quantity \* price).append("\n");

menuItems.put(item, new MenuItem(item, menuItems.get(item).getQuantity() - quantity, price));

} else {

Alert alert = new Alert(Alert.AlertType.WARNING);

alert.setTitle("Warning");

alert.setHeaderText("Item Unavailable");

alert.setContentText("The item " + item + " is unavailable or insufficient quantity.");

alert.showAndWait();

}

}

bill.append("Total: $").append(total);

return bill.toString();

}

private void sendMessage(String message) {

if (out != null) {

out.println(message);

}

}

private void loadMenuItems() {

menuItems = new HashMap<>();

try {

File file = new File("cafemenu.txt");

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine()) != null) {

String[] parts = line.split(":", 4);

if (parts.length >= 4) {

String item = parts[1].trim();

int quantity = Integer.parseInt(parts[2].trim());

int price = Integer.parseInt(parts[3].trim());

menuItems.put(item, new MenuItem(item, quantity, price));

}

}

br.close();

} catch (IOException e) {

e.printStackTrace();

}

}

public static class Order {

private final String item;

private final int quantity;

public Order(String item, int quantity) {

this.item = item;

this.quantity = quantity;

}

public String getItem() {

return item;

}

public int getQuantity() {

return quantity;

}

}

public static class MenuItem {

private final String name;

private final int quantity;

private final int price;

public MenuItem(String name, int quantity, int price) {

this.name = name;

this.quantity = quantity;

this.price = price;

}

public String getName() {

return name;

}

public int getQuantity() {

return quantity;

}

public int getPrice() {

return price;

}

}

public static void main(String[] args) {

launch(args);

}

}

**CafeReviews:**

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

public class CafeReviews {

private ObservableList<Review> reviews;

public TableView<Review> showReviews() {

TableView<Review> tableView = new TableView<>();

TableColumn<Review, String> designationColumn = new TableColumn<>("Designation");

designationColumn.setCellValueFactory(new PropertyValueFactory<>("designation"));

TableColumn<Review, String> commentsColumn = new TableColumn<>("Comments");

commentsColumn.setCellValueFactory(new PropertyValueFactory<>("comments"));

TableColumn<Review, Integer> ratingColumn = new TableColumn<>("Rating");

ratingColumn.setCellValueFactory(new PropertyValueFactory<>("rating"));

tableView.getColumns().add(designationColumn);

tableView.getColumns().add(commentsColumn);

tableView.getColumns().add(ratingColumn);

reviews = FXCollections.observableArrayList();

File file = new File("CafeReviews.txt");

try (BufferedReader br = new BufferedReader(new FileReader(file))) {

String line;

while ((line = br.readLine()) != null) {

String[] parts = line.split(":", 3);

if (parts.length == 3) {

String designation = parts[0];

String comments = parts[1];

int rating = Integer.parseInt(parts[2].trim());

reviews.add(new Review(designation, comments, rating));

}

}

} catch (IOException e) {

e.printStackTrace();

}

tableView.setItems(reviews);

return tableView;

}

public ObservableList<Review> getReviews() {

return reviews;

}

public static class Review {

private final String designation;

private final String comments;

private final int rating;

public Review(String designation, String comments, int rating) {

this.designation = designation;

this.comments = comments;

this.rating = rating;

}

public String getDesignation() {

return designation;

}

public String getComments() {

return comments;

}

public int getRating() {

return rating;

}

}

}

**CafeServer:**

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.List;

public class CafeServer {

private static final int PORT = 12345;

private static List<ClientHandler> clients = new ArrayList<>();

public static void main(String[] args) {

try (ServerSocket serverSocket = new ServerSocket(PORT)) {

System.out.println("Server started on port " + PORT);

while (true) {

Socket clientSocket = serverSocket.accept();

ClientHandler clientHandler = new ClientHandler(clientSocket);

clients.add(clientHandler);

new Thread(clientHandler).start();

}

} catch (IOException e) {

e.printStackTrace();

}

}

public static void broadcast(String message) {

for (ClientHandler client : clients) {

client.sendMessage(message);

}

}

private static class ClientHandler implements Runnable {

private Socket socket;

private PrintWriter out;

public ClientHandler(Socket socket) {

this.socket = socket;

}

@Override

public void run() {

try {

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

out = new PrintWriter(socket.getOutputStream(), true);

String message;

while ((message = in.readLine()) != null) {

System.out.println("Received: " + message);

if (message.startsWith("Review: ")) {

saveReview(message.substring(8));

}

broadcast(message);

}

} catch (IOException e) {

e.printStackTrace();

}

}

public void sendMessage(String message) {

out.println(message);

}

private void saveReview(String review) {

try (FileWriter fw = new FileWriter("CafeReviews.txt", true);

BufferedWriter bw = new BufferedWriter(fw);

PrintWriter out = new PrintWriter(bw)) { //it will be automatically closed

out.println(review);

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

**CuiCafe:**

import java.io.\*;

import java.util.Comparator;

import java.util.stream.Collectors;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Button;

import javafx.scene.control.ComboBox;

import javafx.scene.control.TableColumn;

import javafx.scene.effect.DropShadow;

import javafx.scene.layout.HBox;

import javafx.scene.layout.Pane;

import javafx.scene.layout.StackPane;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.Text;

import javafx.stage.Stage;

public class CuiCafe extends Application {

@Override

public void start(Stage primaryStage) {

// Root Pane

StackPane root = new StackPane();

// Background image path

String backgroundImage = getClass().getResource("/background.png") != null

? getClass().getResource("/background.png").toExternalForm()

: "file:path/to/default/image.png";

// Set background style

root.setStyle("-fx-background-image: url('" + backgroundImage + "');" +

"-fx-background-size: cover; -fx-background-position: center;");

// Logo

Text logo = new Text("CUI CAFE");

logo.setFont(Font.font("Times New Roman", 30)); // Logo font and size

logo.setFill(Color.WHITE);

logo.setEffect(new DropShadow(5, Color.BLACK));

// Navigation Buttons

Button btnHome = createNavButton("Home");

Button btnStudent = createNavButton("Student\_Corner");

Button btnFaculty = createNavButton("Faculty\_Corner");

Button btnStaff = createNavButton("Staff");

Button btnMenu = createNavButton("Menu");

Button btnReviews = createNavButton("Reviews");

Button btnExit = createNavButton("Exit");

//buttons action

btnReviews.setOnAction(e -> {

CafeReviews reviews = new CafeReviews();

TableView<CafeReviews.Review> tableView = reviews.showReviews();

Stage reviewStage = new Stage();

reviewStage.setTitle("Cafe Reviews");

Pane reviewPane = new Pane();

tableView.setPrefSize(750, 370);

reviewPane.getChildren().add(tableView);

Text selectingDesignation = new Text();

selectingDesignation.setText("Select Designation:");

selectingDesignation.setLayoutX(20);

selectingDesignation.setLayoutY(400);

selectingDesignation.setFill(Color.BLACK);

selectingDesignation.setFont(Font.font("Times New Roman", 16));

ComboBox<String> designationComboBox = new ComboBox<>();

designationComboBox.getItems().addAll("Student", "Faculty");

designationComboBox.setLayoutX(20);

designationComboBox.setLayoutY(420);

designationComboBox.setPrefSize(150, 30);

Button filterByDesignation = new Button("Filter by Designation");

filterByDesignation.setLayoutX(180);

filterByDesignation.setLayoutY(420);

filterByDesignation.setPrefSize(150, 30);

filterByDesignation.setOnAction(event -> {

String selectedDesignation = designationComboBox.getValue();

if (selectedDesignation != null) {

ObservableList<CafeReviews.Review> filteredReviews = reviews.getReviews().stream()

.filter(review -> review.getDesignation().equalsIgnoreCase(selectedDesignation))

.collect(Collectors.toCollection(FXCollections::observableArrayList));

tableView.setItems(filteredReviews);

}

});

Button sortByRating = new Button("Sort by Rating");

sortByRating.setLayoutX(400);

sortByRating.setLayoutY(420);

sortByRating.setPrefSize(150, 30);

sortByRating.setOnAction(event -> {

ObservableList<CafeReviews.Review> sortedReviews = reviews.getReviews().stream()

.sorted(Comparator.comparingInt(CafeReviews.Review::getRating).reversed())

.collect(Collectors.toCollection(FXCollections::observableArrayList));

tableView.setItems(sortedReviews);

});

Button backButton = new Button("Back");

backButton.setLayoutX(560);

backButton.setLayoutY(420);

backButton.setPrefSize(150, 30);

backButton.setOnAction(event -> reviewStage.close());

reviewPane.getChildren().addAll(selectingDesignation, designationComboBox, filterByDesignation, sortByRating, backButton);

Scene reviewScene = new Scene(reviewPane, 750, 500);

reviewStage.setScene(reviewScene);

reviewStage.setResizable(false);

reviewStage.show();

});

//displaying menu

btnMenu.setOnAction(e -> {

try {

new MenuClient().start(new Stage());

} catch (Exception ex) {

ex.printStackTrace();

}

});

//admin

btnStaff.setOnAction(e -> {

try {

new AdminClient().start(new Stage());

} catch (Exception ex) {

ex.printStackTrace();

}

});

//student corner

btnStudent.setOnAction(e -> {

try{

new StudentClient().start(new Stage());

}

catch(Exception ex){

ex.printStackTrace();

}

});

//faculty corner

btnFaculty.setOnAction(e -> {

try{

new FacultyClient().start(new Stage());

}

catch(Exception ex){

ex.printStackTrace();

}

});

// Exit Button Action

btnExit.setOnAction(e -> primaryStage.close());

// Navigation Bar

HBox navBar = new HBox(20, logo, btnHome, btnStudent, btnFaculty, btnStaff, btnMenu, btnReviews, btnExit);

navBar.setAlignment(Pos.CENTER);

navBar.setStyle("-fx-padding: 10; -fx-background-color: rgba(0, 0, 0, 0.7);");

navBar.setSpacing(30);

// Title

Text title = new Text("CUI CAFE MANAGEMENT");

title.setFont(Font.font("Times New Roman", 60)); // Updated font and size

title.setFill(Color.WHITE);

title.setEffect(new DropShadow(20, Color.BLACK));

// Buttons

Button btnAboutUs = createStyledButton("ABOUT US", "rgba(255, 0, 0, 0.8)", "white");

Button btnCallNow = createStyledButton("CALL NOW", "white", "black");

// Layout for Buttons

VBox buttonsLayout = new VBox(20, btnAboutUs, btnCallNow);

buttonsLayout.setAlignment(Pos.TOP\_CENTER);

// Main Layout

VBox layout = new VBox(60, navBar, title, buttonsLayout);

layout.setAlignment(Pos.TOP\_CENTER);

// Add to Root

root.getChildren().add(layout);

// Scene

Scene scene = new Scene(root, 1150, 600);

primaryStage.setTitle("CUI Cafe Management");

primaryStage.setScene(scene);

primaryStage.show();

}

private Button createNavButton(String text) {

Button button = new Button(text);

button.setStyle("-fx-background-color: transparent;" +

"-fx-text-fill: white;" +

"-fx-font-size: 16;" +

"-fx-font-weight: bold;");

button.setOnMouseEntered(e -> button.setStyle("-fx-background-color: white; -fx-text-fill: black; -fx-font-size: 16; -fx-font-weight: bold;"));

button.setOnMouseExited(e -> button.setStyle("-fx-background-color: transparent; -fx-text-fill: white; -fx-font-size: 16; -fx-font-weight: bold;"));

return button;

}

private Button createStyledButton(String text, String backgroundColor, String textColor) {

Button button = new Button(text);

button.setStyle("-fx-background-color: " + backgroundColor + ";" +

"-fx-border-color: " + textColor + ";" +

"-fx-border-width: 2;" +

"-fx-text-fill: " + textColor + ";" +

"-fx-font-size: 18;" +

"-fx-font-weight: bold;" +

"-fx-background-radius: 10;" +

"-fx-padding: 10 30;");

button.setEffect(new DropShadow(3, Color.BLACK));

button.setOnMouseEntered(e -> button.setStyle("-fx-background-color: " + textColor + ";" +

"-fx-border-color: " + textColor + ";" +

"-fx-text-fill: " + backgroundColor + ";" +

"-fx-font-size: 18;" +

"-fx-font-weight: bold;" +

"-fx-background-radius: 10;" +

"-fx-padding: 10 30;"));

button.setOnMouseExited(e -> button.setStyle("-fx-background-color: " + backgroundColor + ";" +

"-fx-border-color: " + textColor + ";" +

"-fx-text-fill: " + textColor + ";" +

"-fx-font-size: 18;" +

"-fx-font-weight: bold;" +

"-fx-background-radius: 10;" +

"-fx-padding: 10 30;"));

return button;

}

public static void main(String[] args) {

launch(args);

}

}

class CafeReviews {

private ObservableList<Review> reviews;

public TableView<Review> showReviews() {

TableView<Review> tableView = new TableView<>();

TableColumn<Review, String> designationColumn = new TableColumn<>("Designation");

designationColumn.setCellValueFactory(new PropertyValueFactory<>("designation"));

TableColumn<Review, String> nameColumn = new TableColumn<>("Name");

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

TableColumn<Review, String> commentsColumn = new TableColumn<>("Comments");

commentsColumn.setCellValueFactory(new PropertyValueFactory<>("comments"));

TableColumn<Review, Integer> ratingColumn = new TableColumn<>("Rating");

ratingColumn.setCellValueFactory(new PropertyValueFactory<>("rating"));

tableView.getColumns().add(designationColumn);

tableView.getColumns().add(nameColumn);

tableView.getColumns().add(commentsColumn);

tableView.getColumns().add(ratingColumn);

reviews = FXCollections.observableArrayList();

File file = new File("CafeReviews.txt");

try (BufferedReader br = new BufferedReader(new FileReader(file))) {

String line;

while ((line = br.readLine()) != null) {

String[] parts = line.split(":", 4);

if (parts.length >= 4) {

String designation = parts[0];

String name = parts[1];

String comments = parts[2];

int rating = Integer.parseInt(parts[3]);

reviews.add(new Review(designation, name, comments, rating));

}

}

} catch (IOException e) {

e.printStackTrace();

}

tableView.setItems(reviews);

return tableView;

}

public ObservableList<Review> getReviews() {

return reviews;

}

public static class Review {

private final String designation;

private final String name;

private final String comments;

private final int rating;

public Review(String designation, String name, String comments, int rating) {

this.designation = designation;

this.name = name;

this.comments = comments;

this.rating = rating;

}

public String getDesignation() {

return designation;

}

public String getName() {

return name;

}

public String getComments() {

return comments;

}

public int getRating() {

return rating;

}

}

}

**FacultyClient:**

import javafx.application.Application;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.io.\*;

import java.net.Socket;

public class FacultyClient extends Application {

private static final String SERVER\_ADDRESS = "localhost";

private static final int SERVER\_PORT = 12345;

private PrintWriter out;

private TextArea billArea;

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("CUI Cafe Faculty Client");

VBox root = new VBox(10);

root.setStyle("-fx-padding: 10;");

Label menuLabel = new Label("Menu:");

TableView<MenuItem> menuTable = new TableView<>();

setupMenuTable(menuTable);

TextField quantityField = new TextField();

quantityField.setPromptText("Quantity");

Button orderButton = new Button("Order");

orderButton.setOnAction(e -> {

MenuItem selectedItem = menuTable.getSelectionModel().getSelectedItem();

String quantity = quantityField.getText();

if (selectedItem != null && !quantity.isEmpty()) {

sendMessage("Order: Faculty: " + selectedItem.getName() + ": " + quantity);

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Order Placed");

alert.setHeaderText(null);

alert.setContentText("Your order has been placed.");

alert.showAndWait();

}

});

Label billLabel = new Label("Bill:");

billArea = new TextArea();

billArea.setEditable(false);

Label reviewLabel = new Label("Submit Review:");

TextField reviewField = new TextField();

TextField ratingField = new TextField();

ratingField.setPromptText("Rating (1-5)");

Button submitReviewButton = new Button("Submit");

submitReviewButton.setOnAction(e -> {

String review = reviewField.getText();

String rating = ratingField.getText();

if (!review.isEmpty() && !rating.isEmpty()) {

sendMessage("Review: Faculty: " + review + ": " + rating);

reviewField.clear();

ratingField.clear();

}

});

root.getChildren().addAll(menuLabel, menuTable, quantityField, orderButton, billLabel, billArea, reviewLabel, reviewField, ratingField, submitReviewButton);

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

primaryStage.setScene(scene);

primaryStage.show();

new Thread(this::connectToServer).start();

}

private void setupMenuTable(TableView<MenuItem> table) {

TableColumn<MenuItem, String> nameColumn = new TableColumn<>("Item");

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

nameColumn.setPrefWidth(150);

TableColumn<MenuItem, Integer> priceColumn = new TableColumn<>("Price");

priceColumn.setCellValueFactory(new PropertyValueFactory<>("price"));

priceColumn.setPrefWidth(100);

table.getColumns().addAll(nameColumn, priceColumn);

table.setItems(loadMenuItems());

}

private ObservableList<MenuItem> loadMenuItems() {

ObservableList<MenuItem> menuItems = FXCollections.observableArrayList();

try {

File file = new File("cafemenu.txt");

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine()) != null) {

String[] parts = line.split(":", 4);

if (parts.length >= 4) {

String item = parts[1].trim();

int price = Integer.parseInt(parts[3].trim());

menuItems.add(new MenuItem(item, price));

}

}

br.close();

} catch (IOException e) {

e.printStackTrace();

}

return menuItems;

}

private void connectToServer() {

try {

Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);

out = new PrintWriter(socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String message;

while ((message = in.readLine()) != null) {

final String finalMessage = message;

System.out.println("Received from server: " + finalMessage);

if (finalMessage.startsWith("Bill: Faculty")) {

Platform.runLater(() -> showBillWindow(finalMessage));

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

private void showBillWindow(String bill) {

Stage billStage = new Stage();

billStage.setTitle("Faculty Bill");

VBox root = new VBox(10);

root.setStyle("-fx-padding: 10;");

TextArea billArea = new TextArea(bill);

billArea.setEditable(false);

Label reviewLabel = new Label("Submit Review:");

TextField reviewField = new TextField();

TextField ratingField = new TextField();

ratingField.setPromptText("Rating (1-5)");

Button submitReviewButton = new Button("Submit");

submitReviewButton.setOnAction(e -> {

String review = reviewField.getText();

String rating = ratingField.getText();

if (!review.isEmpty() && !rating.isEmpty()) {

sendMessage("Review: Faculty: " + review + ": " + rating);

billStage.close();

}

});

root.getChildren().addAll(billArea, reviewLabel, reviewField, ratingField, submitReviewButton);

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

billStage.setScene(scene);

billStage.show();

}

private void sendMessage(String message) {

if (out != null) {

out.println(message);

}

}

public static void main(String[] args) {

launch(args);

}

public static class MenuItem {

private final String name;

private final int price;

public MenuItem(String name, int price) {

this.name = name;

this.price = price;

}

public String getName() {

return name;

}

public int getPrice() {

return price;

}

}

}

**Menu Client:**

import javafx.application.Application;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.Scene;

import javafx.scene.control.ScrollPane;

import javafx.scene.control.TableColumn;

import javafx.scene.control.TableView;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.VBox;

import javafx.scene.paint.Color;

import javafx.scene.text.Font;

import javafx.scene.text.Text;

import javafx.stage.Stage;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

public class MenuClient extends Application {

@Override

public void start(Stage primaryStage) {

//setting title

primaryStage.setTitle("Cafe Menu");

VBox root = new VBox(10);

root.setStyle("-fx-background-color: black; -fx-padding: 20;");

root.setPrefSize(800, 600);

Text title = new Text("Cafe Menu");

title.setFill(Color.WHITE);

title.setFont(Font.font("Times New Roman", 30));

root.getChildren().add(title);

TableView<MenuItem> tableView = new TableView<>();

tableView.setPrefWidth(750);

TableColumn<MenuItem, String> juicesColumn = new TableColumn<>("Juices");

juicesColumn.setCellValueFactory(new PropertyValueFactory<>("juices"));

juicesColumn.setPrefWidth(250);

TableColumn<MenuItem, String> fastFoodColumn = new TableColumn<>("Fast Food");

fastFoodColumn.setCellValueFactory(new PropertyValueFactory<>("fastFood"));

fastFoodColumn.setPrefWidth(250);

TableColumn<MenuItem, String> desiFoodColumn = new TableColumn<>("Desi Food");

desiFoodColumn.setCellValueFactory(new PropertyValueFactory<>("desiFood"));

desiFoodColumn.setPrefWidth(250);

tableView.getColumns().addAll(juicesColumn, fastFoodColumn, desiFoodColumn);

ObservableList<MenuItem> menuItems = FXCollections.observableArrayList();

List<String> juices = new ArrayList<>();

List<String> fastFood = new ArrayList<>();

List<String> desiFood = new ArrayList<>();

try {

File file = new File("cafemenu.txt");

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine()) != null) {

String[] parts = line.split(":", 4);

if (parts.length >= 4) {

String category = parts[0];

String item = parts[1] + " - $" + parts[2];

switch (category) {

case "Juices":

juices.add(item);

break;

case "Fast Food":

fastFood.add(item);

break;

case "Desi Food":

desiFood.add(item);

break;

}

}

}

br.close();

} catch (IOException e) {

e.printStackTrace();

}

//adding items to the table of menu

for (int i = 0; i < juices.size(); i++) {

String juiceItem = juices.get(i);

String fastFoodItem = fastFood.get(i);

String desiFoodItem = desiFood.get(i);

menuItems.add(new MenuItem(juiceItem, fastFoodItem, desiFoodItem));

}

tableView.setItems(menuItems);

root.getChildren().add(tableView);

//for scrooling if contents in table exceeds

ScrollPane scrollPane = new ScrollPane(root);

scrollPane.setFitToWidth(true);

//setting scene

Scene scene = new Scene(scrollPane,750,500);

primaryStage.setScene(scene);

primaryStage.show();

}

public static class MenuItem {

private final String juices;

private final String fastFood;

private final String desiFood;

public MenuItem(String juices, String fastFood, String desiFood) {

this.juices = juices;

this.fastFood = fastFood;

this.desiFood = desiFood;

}

public String getJuices() {

return juices;

}

public String getFastFood() {

return fastFood;

}

public String getDesiFood() {

return desiFood;

}

}

}

**StudentClient:**

import javafx.application.Application;

import javafx.application.Platform;

import javafx.collections.FXCollections;

import javafx.collections.ObservableList;

import javafx.scene.Scene;

import javafx.scene.control.\*;

import javafx.scene.control.cell.PropertyValueFactory;

import javafx.scene.layout.VBox;

import javafx.stage.Stage;

import java.io.\*;

import java.net.Socket;

public class StudentClient extends Application {

private static final String SERVER\_ADDRESS = "localhost";

private static final int SERVER\_PORT = 12345;

private PrintWriter out;

private TextArea billArea;

@Override

public void start(Stage primaryStage) {

primaryStage.setTitle("CUI Cafe Student Client");

VBox root = new VBox(10);

root.setStyle("-fx-padding: 10;");

Label menuLabel = new Label("Menu:");

TableView<MenuItem> menuTable = new TableView<>();

setupMenuTable(menuTable);

TextField quantityField = new TextField();

quantityField.setPromptText("Quantity");

Button orderButton = new Button("Order");

orderButton.setOnAction(e -> {

MenuItem selectedItem = menuTable.getSelectionModel().getSelectedItem();

String quantity = quantityField.getText();

if (selectedItem != null && !quantity.isEmpty()) {

sendMessage("Order: Student: " + selectedItem.getName() + ": " + quantity);

Alert alert = new Alert(Alert.AlertType.INFORMATION);

alert.setTitle("Order Placed");

alert.setHeaderText(null);

alert.setContentText("Your order has been placed.");

alert.showAndWait();

}

});

Label billLabel = new Label("Bill:");

billArea = new TextArea();

billArea.setEditable(false);

Label reviewLabel = new Label("Submit Review:");

TextField reviewField = new TextField();

TextField ratingField = new TextField();

ratingField.setPromptText("Rating (1-5)");

Button submitReviewButton = new Button("Submit");

submitReviewButton.setOnAction(e -> {

String review = reviewField.getText();

String rating = ratingField.getText();

if (!review.isEmpty() && !rating.isEmpty()) {

sendMessage("Review: Student: " + review + ": " + rating);

reviewField.clear();

ratingField.clear();

}

});

root.getChildren().addAll(menuLabel, menuTable, quantityField, orderButton, billLabel, billArea, reviewLabel, reviewField, ratingField, submitReviewButton);

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

primaryStage.setScene(scene);

primaryStage.show();

//new thread to start the connection

new Thread(this::connectToServer).start();

}

private void setupMenuTable(TableView<MenuItem> table) {

TableColumn<MenuItem, String> nameColumn = new TableColumn<>("Item");

nameColumn.setCellValueFactory(new PropertyValueFactory<>("name"));

nameColumn.setPrefWidth(150);

TableColumn<MenuItem, Integer> priceColumn = new TableColumn<>("Price");

priceColumn.setCellValueFactory(new PropertyValueFactory<>("price"));

priceColumn.setPrefWidth(100);

table.getColumns().addAll(nameColumn, priceColumn);

table.setItems(loadMenuItems());

}

private ObservableList<MenuItem> loadMenuItems() {

ObservableList<MenuItem> menuItems = FXCollections.observableArrayList();

try {

File file = new File("cafemenu.txt");

BufferedReader br = new BufferedReader(new FileReader(file));

String line;

while ((line = br.readLine()) != null) {

String[] parts = line.split(":", 4);

if (parts.length >= 4) {

String item = parts[1].trim();

int price = Integer.parseInt(parts[3].trim());

menuItems.add(new MenuItem(item, price));

}

}

br.close();

} catch (IOException e) {

e.printStackTrace();

}

return menuItems;

}

private void connectToServer() {

try {

Socket socket = new Socket(SERVER\_ADDRESS, SERVER\_PORT);

out = new PrintWriter(socket.getOutputStream(), true);

BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String message;

while ((message = in.readLine()) != null) {

final String finalMessage = message;

System.out.println("Received from server: " + finalMessage);

if (finalMessage.startsWith("Bill: Student")) {

Platform.runLater(() -> showBillWindow(finalMessage));

}

}

} catch (IOException e) {

e.printStackTrace();

}

}

private void showBillWindow(String bill) {

Stage billStage = new Stage();

billStage.setTitle("Student Bill");

VBox root = new VBox(10);

root.setStyle("-fx-padding: 10;");

TextArea billArea = new TextArea(bill);

billArea.setEditable(false);

Label reviewLabel = new Label("Submit Review:");

TextField reviewField = new TextField();

TextField ratingField = new TextField();

ratingField.setPromptText("Rating (1-5)");

Button submitReviewButton = new Button("Submit");

submitReviewButton.setOnAction(e -> {

String review = reviewField.getText();

String rating = ratingField.getText();

if (!review.isEmpty() && !rating.isEmpty()) {

sendMessage("Review: Student: " + review + ": " + rating);

billStage.close();

}

});

root.getChildren().addAll(billArea, reviewLabel, reviewField, ratingField, submitReviewButton);

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

billStage.setScene(scene);

billStage.show();

}

private void sendMessage(String message) {

if (out != null) {

out.println(message);

}

}

public static void main(String[] args) {

launch(args);

}

public static class MenuItem {

private final String name;

private final int price;

public MenuItem(String name, int price) {

this.name = name;

this.price = price;

}

public String getName() {

return name;

}

public int getPrice() {

return price;

}

}

}

**Cafereviews.txt:**

Student:Great food and service:5

Faculty:Nice ambiance but a bit noisy:4

Student:Affordable prices and tasty meals:5

Faculty:Good variety of dishes:4

Student:Friendly staff and quick service:5

Student:It was nice:3

Student:its nice:4

Faculty: Juicy!: 4

Student: Very tasty!: 4

Faculty: nice: 2

Faculty: Hey its yummy: 4

Student: yummy: 4

**Cafemenu.txt:**

Juices:Orange Juice:10:3

Juices:Apple Juice:15:3

Juices:Mango Juice:8:3

Juices:Pineapple Juice:12:4

Juices:Grape Juice:9:4

Juices:Banana Juice:10:10

Juices:Watermelon Juice:7:5

Fast Food:Cheeseburger:20:5

Fast Food:Chicken Burger:18:5

Fast Food:Veggie Burger:15:4

Fast Food:Pepperoni Pizza:10:8

Fast Food:Margherita Pizza:12:7

Fast Food:Chicken Pasta:14:6

Fast Food:Veggie Pasta:16:5

Desi Food:Chicken Biryani:10:7

Desi Food:Mutton Biryani:8:8

Desi Food:Palao:12:6

Desi Food:Roti:50:1

Desi Food:Chicken Karahi:7:9

Desi Food:Mutton Karahi:5:10

Desi Food:Chicken Tikka:6:7

**Conclusion**

The CUI Cafe Management System is a comprehensive solution for managing cafe operations, allowing for efficient order processing and customer feedback. By utilizing JavaFX for the user interface and socket programming for client-server communication, the system provides a seamless experience for users. The use of text files for data storage ensures that the application remains lightweight and easy to manage.