CENG 490

FINAL DESIGN REPORT

Team Name : QuartetSoft

Project Name : Sofr@m

Team Members :
Evren Merih Tan  - 1298314
Guner Mutlu      - 1298017
Mustafa Zengin  - 1250984
Rovsen Gasimov  - 1222173
1.VOCABULARY

Admin: Person who uses server

Customer: Person who is willing to benefit from restaurant services

PDA: Handheld device that is connected to the wireless network and has its own software

Server: Machine on which all software are stored and distributed

User: Is the person who uses the clients

2.INTRODUCTION

Sofr@m is a Restaurant Automation Software which handles various types of user actions with assistance of pure Graphical User Interfaces. User intuitiveness and ease of use were the design principals of Sofr@m software solution that can help you cut restaurant losses, reduce employee mistakes, improve customer service, increase profitability, and much more. It is the right software to handle sale and management needs for all kind of eating enterprise. It ensures efficient restaurant conduction and direction via PDA that will significantly improve staff's efficiency and customer service. Thus our software will be time saving by improving restaurant operations.

Employees working in the restaurant will not face any difficulties in using it since our software is easy to use with its screen layouts and windows based interface. Sofr@m will also allow customers give orders via internet.

2.1 Goals and Objectives

As indicated above, our software aims to maintain efficient restaurant management by minimizing ordering time.

The goals of Sofr@m are as follows:

1. Versatile and fast restaurant management
2. Eliminating double order entry by the staff.
3. Managing scheduling of employees with ease.
4. Preventing unaffordable customer orderings and placing orders easily.
5. Keeping track of sale and employee statistics.
6. Improving wait staff's efficiency and customer service.

7. Better tracking of inventory stock, and notification alert when an ingredient of a product is running low and needs to be reordered.

8. Evaluating labor costs, number of items sold, number of guests served through reports

9. Reducing employee mistakes

3. DATAFLOW DIAGRAMS

Level 0 Dataflow Diagram:

Sofram takes inputs from two different sources which are PDAs and customers. The first input type PDAs are located in the restaurant and used by waiters and waiters send customers' orders to the main computer by using these PDAs. The second input is from the internet which is used by customers. Then sofram produces its output according to these inputs in order to make restaurant management more efficient.

Level 1 Dataflow Diagram:

When customer logs into sofram, first of all their username and password are checked through the database and if they are valid, the system lets the customer log into sofram web page. As soon as he/she goes into the system, his/her profile is opened in the database. After that the customer can make online orders or reservation without coming to the system and these orders or reservation are uploaded to the database and the customer is warned if his/her order or reservation can be served or not. When the customer logs out from the system, a message is shown on the screen to say good-bye. Another input source is the PDA. These are used by
waiters and when the waiter take an order from a table, he transmit these order to the main computer by PDA and the database is updated according to the order. Finally sofiram produces its output according to these inputs and provide an efficient restaurant management.

Level 2 Dataflow Diagram (Online Orders):
A customer can add a new item to his/her order list or can delete an item from his/her list or update the amount of an item which is already in his/her list by using the online part of sofram. While customer is doing these operations, database is updated according to his/her processes. Also when the customer logs into sofram, his/her profile information is taken from the database to understand the identity of the customer.

**Level 2 Dataflow Diagram (Online Reservation):**
When the customer logs into the system and if his/her username and password is valid, a page is shown to help the customer to do online reservation. He/she can see the status of the tables in the restaurant, see the menu and also day’s special offer is shown to the customer. After making his/her decision, the customer logs out from the system. And also the customer can go to the online order page from that reservation page.

Level 2 Dataflow Diagram (PDA):
When the waiter takes an order, he adds this new order to the database by using the ‘add’ module of his/her PDA. Also the waiter can delete an order or can update an order just by using his/her PDA. After pressing the confirmation button, the database is updated and also the waiter can see these orders in the screen of his/her PDA.

4. SEQUENCE DIAGRAMS

Online Request Sequence Diagram (Registering into sofam):
When the customer presses the registration part, he/she fills a form in order to take his/her info to the system. There are parts that must be filled such as username, password, name, address, credit card number, so the system controls if these parts are filled. If all of them are filled, the user is added to the system and if not the user is warned and the system wants him/her to fill the form again.

Online Request Sequence Diagram (Display Personal Info):
When the customer goes into the system after checking his/her username and password, he/she can make a request to see his/her personal info. This request is taken into account by the personal account system and the personal info of the customer is displayed on the screen.

**Online Request Sequence Diagram (Updating User Info):**
After logging into the system through the security login system, the user can see his/her info and change that. Personal account system updated this new info after taking the new data from the user.

**Online Request Sequence Diagram (Reserving a Table):**
First of the username submit his/her username and password by using the user interface then the username and password is checked by the system and if they are valid the user can log into the system. After logging into sofram, the user can make a request in order to request a table and to do that he/she can look for available tables in the restaurant. If there is an available table, the customer can reserve that table.

**Online Request Sequence Diagram (Request Food):**
The user logs into the system through the security login system. After loading his/her info to the system from the database, the customer can request food via sofaram web page without coming to the restaurant. But when he/she makes a request, the system controls the database in order to see if there is enough stock for that request. And the customer is warned after that control if his/her request can be served or not according to the stock status.

**Cashier Sequence Diagram:**
Cashier can display the active orders which are the orders that have been waiting to be served. So the cashier can control the status of all orders.

**Statistics Sequence Diagram:**

Cashier can see statistics by using the passive order list which is the list that holds the orders that have been served so far. By using these orders, the cashier can generate statistics and can see these statistics in graphical form.

**View Waiter’s Personal Info via PDA Sequence Diagram:**
The waiter can see his/her personal info on the screen of his/her PDA. To do that he/she sends a request to the personal account system via PDA and the personal account system uploads his/her personal info to his/her PDA. And the waiter can log out from the system as soon as he/she sends a log out wish signal to the system via his/her PDA.

**Update Waiter’s Personal Info via PDA Sequence Diagram:**
The waiter can update his/her personal info whenever he/she wants if he/she logs into the system via his/her PDA. When he/she sends a signal to update his/her personal info to personal account system and then personal account system is updated. When the waiter wants to logs out from the system, he sends a logout signal to the system and the system takes that request into account and the waiter logs out from the system.

**Take Order via PDA Sequence Diagram:**

First of all the waiter logs into his/her PDA, then the system controls his/her username and password in order to let him log into the system or not. Then the personal info of the waiter is loaded to the PDA via the personal account system. After these processes, the waiter can take new orders and transmit them to the main computer and these orders are put into the active
order list and that list is updated. When the order is ready, the waiter is warned by the kitchen, so he/she can know when to serve that order.
After logging into the system of the waiter, the waiter can open a new table and take orders from that table. Maybe customers sitting on that table want to cancel their orders. In that situation, waiter sends a cancel order signal to the main computer and the main computer
cancels that order and at the same time updates the active order list because that cancelled order is not an active order now.

**Send Bill via PDA Sequence Diagram:**

When the customers decide to leave the restaurant and want to take the bill, the waiter transmits that request to the cashier via his/her PDA, and then the main computer adds that bill to the bill list and then sends a request to the printer to print the bill of customers in order.

**Complete Send Bill via PDA Sequence Diagram:**
The username and the password of the waiter are checked by the security login system. If they are valid, the personal info of the waiter is uploaded to his/her PDA from the main computer. After that the waiter can open a new table for new coming customers or select a table or close a table. These operations are checked by the table management system. Then
the waiter sends a signal to the cashier who sits on the main computer to print the bill of a table. And whenever he/she wants, he/she can log out from the system.

PDA-Kitchen Sequence Diagram:

PDA transmits a new order request to the kitchen, then the kitchen monitor is updated as soon as this new order request reaches the kitchen from the PDA. And when the order is ready, kitchen monitor sends a signal to the PDA to tell that the order is ready for serving.

5.DATABASE DESIGN

5.1 ER Diagrams
5.2 Database schema

**Cashier**

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>username</td>
</tr>
<tr>
<td>password</td>
</tr>
<tr>
<td>keyword</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>surname</td>
</tr>
<tr>
<td>email</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>phone_number</td>
</tr>
<tr>
<td>mobile_number</td>
</tr>
<tr>
<td>user_type</td>
</tr>
</tbody>
</table>

**Employees**

<table>
<thead>
<tr>
<th>Field</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>surname</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>security_number</td>
</tr>
<tr>
<td>payment_amount</td>
</tr>
<tr>
<td>payment_date</td>
</tr>
<tr>
<td>telephone_number</td>
</tr>
<tr>
<td>mobilephone_number</td>
</tr>
<tr>
<td>date_of_employment</td>
</tr>
<tr>
<td>employment_type</td>
</tr>
</tbody>
</table>
**Customers**

<table>
<thead>
<tr>
<th>username</th>
<th>password</th>
<th>keyword</th>
<th>name</th>
<th>surname</th>
<th>telephone_number</th>
<th>mobilephone_number</th>
<th>address</th>
<th>birthday</th>
<th>credit_card</th>
<th>email</th>
</tr>
</thead>
</table>

**Credit_Cards**

<table>
<thead>
<tr>
<th>card_number</th>
<th>Type</th>
<th>password</th>
</tr>
</thead>
</table>

**Tables**

<table>
<thead>
<tr>
<th>table_id</th>
<th>status</th>
</tr>
</thead>
</table>

**Ingredients**

<table>
<thead>
<tr>
<th>name</th>
<th>price</th>
<th>amount</th>
<th>last_time_bought</th>
<th>company</th>
</tr>
</thead>
</table>

**Meal**

<table>
<thead>
<tr>
<th>meal_name</th>
<th>category</th>
<th>unit_price</th>
<th>ingredient_name</th>
<th>ingredient_amount</th>
</tr>
</thead>
</table>
Meal_Categories

| category_name |

Reservations

<table>
<thead>
<tr>
<th>Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_id</td>
</tr>
<tr>
<td>beg_time</td>
</tr>
<tr>
<td>end_time</td>
</tr>
</tbody>
</table>

Drinks

<table>
<thead>
<tr>
<th>drink_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>dimension</td>
</tr>
<tr>
<td>price</td>
</tr>
</tbody>
</table>

Online_Orders

<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>request_time</td>
</tr>
<tr>
<td>due_time</td>
</tr>
<tr>
<td>user_id</td>
</tr>
<tr>
<td>request_id</td>
</tr>
<tr>
<td>request_type</td>
</tr>
<tr>
<td>address</td>
</tr>
</tbody>
</table>

Order

<table>
<thead>
<tr>
<th>id</th>
</tr>
</thead>
<tbody>
<tr>
<td>table_id</td>
</tr>
<tr>
<td>request_id</td>
</tr>
<tr>
<td>request_type</td>
</tr>
<tr>
<td>request_date</td>
</tr>
<tr>
<td>order_status</td>
</tr>
<tr>
<td>current_price</td>
</tr>
</tbody>
</table>
5.3 Tables, attributes and their explanations

**Table Cashier**: This table holds the information of the user, who logged in as the cashier.

1. *username*: Username that the user uses to enter the system. All usernames will be different from each other to avoid from confusion. This is the primary key for the table. Type is string.

2. *password*: Password that the user uses to enter the system. This will be an encrypted string.

3. *keyword*: User specified string that will help the user to remind his/her password, in case the password is forgotten. Type is string.

4. *name*: Real name of the user. Type is string.

5. *surname*: Real surname of the user. Type is string.

6. *email*: e-mail address. Type is string.

7. *phone_number*: Telephone number of the user. Type is string.

8. *mobile_number*: Mobile phone number of the user. Type is string.

9. *user_type*: Specifies type of the user. Two types of user is eligible, normal and super user. Super user is able to change some features or settings that normal user is not. Type is integer.
**Table Employees:** This table will hold the information about employee.

1. **name**: Name of the employee. Type is string.
2. **surname**: Surname of the employee. Type is string.
3. **address**: Address of the employee. Type is string.
4. **security_number**: Social security number of the employee. This is the primary key. Type is string.
5. **payment_amount**: Amount of the payment paid to the employee. Type is real number.
6. **payment_date**: Date of the payment. Type is date.
7. **telephone_number**: Telephone number of the employee. Type is string.
8. **mobilephone_number**: Mobilephone number of the employee. Type is string.
9. **employment_type**: Type of the employment. Whether the employee is a cook or waiter. Type is string.

**Table Customers:** This table holds the information of the customers.

1. **username**: Username of the customer that will enable him/her to use online reservation or order facility. This is the primary key. Type is string.
2. **password**: Password of the customer that is required to enter the online system. This is an encrypted string.
3. **keyword**: User specified string that will help the user to remind his/her password, incase the password is forgotten. Type is string.
4. **name**: Real name of the customer. Type is string.
5. **surname**: Surname of the customer. Type is string.
6. **telephone_number**: Telephone number of the customer. Type is string.
7. **mobilephone_number**: Mobile phone number of the customer. Type is string.
8. **address**: Address of customer, that will be used for deliveries of online orders. Type is string.
9. **birthday**: Birthday of the customer.
10. **credit_card**: Credit card number of the user, that will be used for credit card payment type. This is the foreign key referencing to Credit_Cards.
11. **email**: e-mail address of the customer, that will be verified at the registration phase by sending a cookie. Type is string.

**Table Credit_Cards**: This table holds information about credit cards of customer.

1. **card_number**: Specifies credit card number. Type is string. This and type attribute are the foreign keys.

2. **type**: Type of the credit card. Type is integer.

3. **password**: Password for the specified card. This is an encrypted string.

**Table Online_Orders**: This table holds the required information about each online order.

1. **id**: Unique id. This is the primary key.

2. **request_time**: Time that request is delivered. Type is string.

3. **due_time**: Due time that the order must be delivered to the address. Type is string.

4. **user_id**: Id of the user who gave order. This is a foreign key referencing to Customers table. If the user is deleted all his/her orders will be deleted. Type is string.

5. **request_type**: Type of the request. That is it specifies whether a drink or meal is requested. Type is string.

6. **request_id**: Id of request, which is unique in its table. Type is string.

7. **address**: Address that the order must be delivered. Type is string.

**Table Tables**: Table that holds information of tables.

1. **table_id**: Unique id of tables. This is the primary key. Type is integer.

2. **status**: Specifies current status of the table. At any time a table may be in one of three states: Opened, free, reserved. Type is integer.

**Table Ingredients**: Table that holds information of ingredients.
1. *name* : Name of ingredient which is supposed to be unique. This is the primary key. Type is string.

2. *price* : Price of the ingredient for a given amount of ingredient. Type is real number.

3. *amount* : Amount of ingredient when bought last time. Type is real number.

4. *last_time_bought* : Time that ingredient bought last time. Type is date.

5. *company* : Specifies the supplier company. This is a foreign key referencing to the Suppliers table. Type is string.

**Table Meal** : Table that holds information of meals in the menu.

1. *meal_name* : Name of meal. Supposed to be unique. This is the primary key. Type is string.

2. *category* : The category that meal belongs to. This is a foreign key that references to the Meal_Categories table. Type is string.

3. *unit_price* : Price for meal of one portion. Type is real number.

4. *ingredient_name* : Ingredient name that is involved in that meal. This is another foreign key referencing to Ingredients table. Type is string.

5. *ingredient_amount* : Amount of specified ingredient that involved in one portion of meal. Type is string.

**Table Meal_Categories** : Table that holds types of meal.

1. *category_name* : Name of meal type, which is supposed to be unique. This is the primary key. Type is string.

**Table Reservations** : Table that holds reservation information.

1. *id* : Unique id for reservation. This is the primary key. Type is string.

2. *table_id* : Id of the table. This is a foreign key referencing to the Tables table. Type is integer.

3. *beg_time* : Beginning time of reservation. Type is string.

4. *end_time* : End time of reservation. Type is string.
**Table Drinks** : Table that holds drinks, their prices.

1. *drink_name* : Name of drink, which is supposed to be unique. This is the primary key. Type is string.

2. *dimension* : Specifies type of drink, that may be one of three: Little, normal or big. Type is integer.

3. *price* : Price of the drink. Type is real.

**Table Order** : Holds information of the orders.

1. *id* : Unique id for the order. This is the primary key. Type is string.

2. *table_id* : Id of the table that order belongs to. This is a foreign key referencing to the Tables table. Type is integer.

3. *request_id* : Id of request, which is unique in its table. That is if a drink is requested request_id is referencing to the Drinks table, where it is unique. If a meal is requested then request_id is referencing to Meal table. Type of it is string.

4. *request_type* : Specifies type of the food, if it is a drink or meal. Type is integer.

5. *request_date* : Date of the order. It will later be used for generating statistics. Type is date.

6. *order_status* : Specifies current status of the order. It maybe in one of two states: Order is being prepared at the moment or Order is finished. Type of order_status is integer.

7. *current_price* : Specifies price of the order. Why to hold it again if we already have the prices? Because those prices may change and when generating statistics later we would need that old price to calculate the total amount of sell for a given period of time.

**Table Suppliers** : This table hold information about the companies that supply ingredients or other kind of food to our restaurant.

1. *id* : Unique id for the company. This is the primary key. Type is integer.

2. *company_name* : Name of the company. Type is string.
3. *person_name*: Name of the person to contact when needed. Type is string.

4. *address*: Address of the company. Type is string.

5. *telephone_number1*: Telephone number to contact if needed. Type is string.

6. *telephone_number2*: Second telephone number to contact if needed. Type is string.

7. *mobilephone_number1*: Mobile phone number to contact when required. Type is string.

8. *mobilephone_number2*: Second mobile phone number to contact when required. Type is string.

9. *fax*: Fax number of the company. Type is string.

10. *email*: Email address to mail if required. Type is string.

11. *notes*: Notes about company, entered by the user of cash desk module. Type is string.

12. *url*: Home web page address of the company. Type is string.

6. USE CASE DIAGRAMS

1) Web Part
2) PDA Part

- Anonymous Waiter
  - Register
  - View Menu
  - View Food Detail
  - Search Food
  - Make Order
  - Cancel Order
  - Log In
  - View Info
    - Send Bill
    - Edit Info

- Waiter
  - Alert Of Lack
    - «uses» Stock Management System
  - Stock Update
  - Show Tables
  - Open Table
  - Close Table
  - View Order
  - Tables Update
  - «uses» Stock Management System
  - «extends» Edit Info
3) Kitchen Part
4) Cash-Desk Part

6.1 Description of Diagrams

1) Web Part

In this use case diagram, the actors are customers and stock management system. If a customer did not sign in to system it is named as anonymous user and can reach to restricted area such as search food, view menu which can be seen from use case diagram of customer. A registered user can use all the capabilities of the system, can give orders, can see tables and make a reservation.

The other actor, stock management system does checking the database for available meal. That means it makes a check for ingredients of meal and warns the customer if the order can not be done. Another
capability of the system is, it searches the database for a lack of material if it finds some material is under a value, it warns the manager (or admin).

2) PDA Part

In this use case diagram the actor is waiter. Without registering nothing can be done as seen from the use case. Again there is a system, exactly the same system that I mention about.

3) Kitchen Part

In this use case diagram the actor is PDA that is, the order is comes to kitchen and updates the kitchen monitor. The order is ready a signal is send to waiter when the meal is cooked.

Active Order List: This list is for orders that are not served.
Passive Order List: This list is for orders that are served.

4) Cash-Desk Part

In this use case diagram the actors are PDA and cashier. PDA(waiter) makes a request for bill and his responsibility for this use case finishes. When the bill is paid, the order is moved from active order list to passive order list. The cashier can see the orders, prints bill and prints end of the day information. Addition to these a manager can generate statistics and have an idea about the sold items.
7. CLASS DIAGRAMS
7.1 Description Of Classes and Function Descriptions

**Person Class:**

This class is an abstract class and manager, user, cashier, waiter classes are inherited from it.

**Manager Class:**

This class is inherited from Person class and has some extra capabilities other than cashier. It has a method `show_Statistics(type, category, beg date, end date)` in order to see statistics for a time period. Type and Category is for type of the drawn diagram (eg: histogram) and for the selected (from the GUI) Category (eg: drinks).
**Waiter Class:**

This class is inherited from Person class and have methods
openTable(tableID), getTable(tableID),
make_Order(tableID,orderID), cancel_Order(orderID),
send_Bill(tableID),setTable(tableID),getTable(tableID),
Briefly explaining these methods, openTable() selects the
table from tables_List and puts the ID of table into
responsibilityTables array. getTable() method is for selecting the table, for an operation.
make_Order() method is sending the orders that are taken from customers with an ID to database in
order to adding it to Active_Orders_List. cancel_Order() method is for deleting the order from
Active_Orders_List without putting the order to Passive_Orders_List. send_Bill() method is for
sending the bill of table to cash desk after this method, the order is deleted from Active_Orders_List
and inserted to Passive_Orders_List. close_Table() method is changing the status of table to free.

**User Class:**

This class is inherited from Person class and have methods
make_Reservation(userID,tableID,timeInterval,reservationID),
make_Order(userID,orderID), show_Menu(),
cancel_Order(orderID),cancel_Reservation(reservationID),
search_Food(). Briefly explaining these methods
make_Reservation method selects a table for a time period and
changes its status to busy.make_Order() method is takes the
orders of customer and puts the order into Active_Order_List.
show_Menu() method displays the menu class’ Items[] which is list of meals.cancel_Order() method
deletes the order from Active_Order_List. cancel_Reservation method changes the status of table that
is reserved from the user to free. search_Food() method is for searching a food and its ingredients.
Anonymous user class is not showed here but, if explaining briefly it have show_Menu() and
search_Food() methods only.

**Cashier Class:**

This class is inherited from Person class. The methods;
show_Orders_List() is for displaying all the orders.
send_bill_to_print(tableID) is for printing the bill of a table.
add_Item_To_Menu() is for adding new meals or drinks to menu.
remove_Item_From_Menu() is for deleting the item (meal or drink) from menu.
edit_Ingredients(foodID) is for changing the ingredient of meal.

**Order Class:**

This class is for wrapping and packaging the orders of customer that is making meal request and a drink request one thing.
select_Food() method is for selecting the items from a list and putting it into a basket. addto_Active_Order_List() method is called by make_Order() which is a method of user class. addto_Passive_Order_List() method is moving the item from Active_Order_List to Passive_Order_List and called when the bill request is done. remove_From_Active_Order_List() method is for deleting the item from Active_Order_List, it is called several times.

**Reservation Class:**

This class is for making customer reservations, it has methods select_Table(tableID) and show_Tables(). If we briefly explain the methods select_Table() method selects the table from a list and changes the table’s status to busy. show_Tables() method displays all the tables with their status.

**Order_List Class:**

This is an abstract class, have an attribute Orders[] which is for keeping the orders in database and when needed retrieving from Database. The method Show_Orders() is for displaying all orders.

**Active_Order_List Class:**

This class is for keeping the orders that’s bill is not paid yet. display_List() is for monitoring active orders. remove_From_List() is for deleting the order from list.
**Passive_Order_List Class:**

This class is for keeping the orders that’ bill is paid. This is needed because in order to generate statistics. The method `show_Statistics()` is called from manager class and the request is sent to statistics class. The `display_List()` method is monitoring the list.

- `Passive_Order_List` class
  - `show_Statistics()` method
  - `display_List()` method

**Statistics Class:**

Statistics class is for generating several types of diagrams (e.g., histogram). The method `draw_Statistics()` is called by `Passive_Order_List` class’ method `show_Statistics()` and draws the diagrams for a selected type for a time interval.

- `Statistics` class
  - `type`: int
  - `category`: int
  - `startDate`: Date
  - `endDate`: Date
  - `sets and get()`
  - `draw_Statistics()`: void

**Menu Class:**

This class is for generating a menu (real life entity). The `display_Menu()` method is monitoring the menu items. `add_Item()` is for adding a new food or a drink to `Items[]`. `remove_Item()` is for deleting a food from the `Items[]`. `show_Ingredients()` method is for displaying the food’s ingredients.

- `Menu` class
  - `display_Menu()`: void
  - `add_Item()`: void
  - `remove_Item()`: void
  - `show_Ingredients()` method

**Food Class:**

This is an abstract class and other classes such as meals and drinks are inherited from it.

- `Food` class
  - `itemName`: string
  - `price`: string
  - `sets and get()`

**Meal Class:**

This class is for meals that are in menu. The class is keeping ingredients of meal and have methods `add_Ingredients()` is for adding new materials to a meal. `remove_Ingredients()` is for removing the material from ingredients list.

- `Meals` class
  - `ingredients`: (Ingredients)
  - `sets and get(Ingredients)
  - `add_Ingredients()`: void
  - `remove_Ingredients()` method
Drink Class:

This class is for drinks in menu. It keeps the type of drink and price.

<table>
<thead>
<tr>
<th>Drink</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type: int – t</td>
</tr>
<tr>
<td>numberOfTypes: int – n</td>
</tr>
<tr>
<td>sets and get()</td>
</tr>
</tbody>
</table>

Stock Class:

This class is for keeping the raw materials of meals and drinks. It is updated regularly when a change is occur with updateStock() method. warnLackOfMaterial() is checking the stock and warning if a material decreases under sum value. add_Material() and remove_From_List() is for adding or removing material to materials list.

<table>
<thead>
<tr>
<th>Stock</th>
</tr>
</thead>
<tbody>
<tr>
<td>material[]: RawMaterials</td>
</tr>
<tr>
<td>updateStock() : void</td>
</tr>
<tr>
<td>warnLackOfMaterial() : string</td>
</tr>
<tr>
<td>sets and get()</td>
</tr>
<tr>
<td>remove_From_List() : void</td>
</tr>
<tr>
<td>add_Material() : void</td>
</tr>
</tbody>
</table>

Note: We did not showed some classes(some lists) and methods such as sets and gets.

8. GRAPHICAL USER INTERFACES

8.1 Cash-Desk User Interfaces

Our Cash-Desk module has two users: Normal-user and super-user. Different than normal-user, super-user is able to make changes on some important settings. But super-user does not have some permissions that normal-user has, like getting end-of-day, displaying table statuses.

8.1.1 Normal-User Interface: Below some of normal-user graphical user interfaces are given.

Login Page: In this page one can enter the system as a normal-user by filling user id, password textboxes and then clicking Log In button or by pressing return key on the keyboard. When the user logs in he/she goes to the main menu page. Clicking Cancel button user finalizes the program. In the top-right corner there is a link to the admin login page, where user can log in as a super-user.
8.1.2 **Admin user interface**: After logging in admin can take some actions that normal-user can not. Viewing or changing admin information (id and password),

**Login Page**: Admin can enter the system from the page like below. After logging in the first page he/she faces is the menu page. From the link at the top-right page user can switch to the normal-user login interface.
**Menu Page:** Menu page is given below. On the menu page super-user has some actions that can be taken. By clicking Staff Management button super-user is taken to a page, where he/she can manage the staff: Add new staff, change properties, wages, due dates of payments. Stock Management enables super-user to add ingredient, change amount of ingredient per meal. In user management new user(normal-user only) can be added. Details of users can be viewed or changed. In addition to these super-user can generate statistics in many categories. Weekly or monthly statistics for online orders, normal restaurant orders, ingredients used, total amount of sells are possible chioses. Super-user can change the settings of the meal categories or define new category for payment type, meal type, job. Last button finalizes the program.

In addition to all these user can take any action listed in menu bar without a need of going to the menu page. Help facility is provided to give information about any action or the program itself. There is a search utility that enables user to search for some topic with a given keyword.
**User Management Page:** Here super-user can add only normal-user, change or view details of any user. New normal-user registration page is given below.
Fields with asterisks (*) must be filled:

*User ID (At least 5 chars)
*Password (At least 5 chars)
*Retype Password
*Name
*Surname
*E-Mail Address
Address
Telephone Number
Mobile Phone Number
*Keyword (Password Remainder)

Submit  Cancel
WELCOME TO HASAN USTA’S ONLINE FOOD SERVICE

username
password

Submit  Sign In
name: mustafa
surname: zengin
username: mzet
address: 100 yıl isci sitesi, 15. yıldız 49
phone: 532 608 5992
birthday: 21.05.1982
edit info
Show My Basket
Submit My Basket
Cancel My Basket
Show My Reservation
Submit My Reservation
Cancel My Reservation

Free Tables 12:00-13:00
Time Interval 12:00-13:00

Reserve Cancel
Kebaps  Adana Kebap  View Ingredients  Add to Basket

Credit Card Number

Credit Card Password

If you want the order to a different address please fill the textbox:


Kebaps & Turkish Kitchen
Fast Food & Sandwiches
Pizzas
World Kitchen
9.ARCHITECTURAL DESIGN

9.1 Project Environment

9.1.1 Development Environment

- Operating System : Windows CE
- Platform: Microsoft .NET Framework
- Programming Language : C#
- Abstraction Layer : XML
- Database : Microsoft SQL Server 2000 or later
- Microsoft Internet Information Services 5.0 installed and running

Sofr@m utilizes Microsoft Windows Server and Terminal Services which is the virtual application layer technology to deliver the handheld version of the interfaces and its contents. Reliability and speed factors played the most important role in choosing this architecture. All wireless software data and program will reside on server computer that ensures us faster and more tolerable system.

Since PDA devices are subjective to external disconnections such as wireless range and battery lifetime, Terminal Services detach these factors. Due to the fact that all data resides on the server computer, if somehow PDA device disconnects, the order information that was taken and changes that were made afterwards need not have to re-entered, all that should be done is to reconnect the device to the network and retrieve the information, which you were working on, from the server.

Online reservation part allows customers check and reserve tables via internet without coming to the restaurant.

Windows will be our operating system for the project. The development environment is Microsoft .NET Framework since it is compatible with the programming language C#.

9.1.2 Working Environment
• Minimum 450 MHz Pentium II-class or compatible processors
• Minimum 512MB RAM
• 10 GB Harddisk
• Windows 9x/2000/nt/XP

10. PROJECT PLAN

<table>
<thead>
<tr>
<th>Sub Tasks</th>
<th>Start Time</th>
<th>Finish Time</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>30/11/04</td>
<td>01/06/05</td>
<td>7 months</td>
</tr>
<tr>
<td>Test 1</td>
<td>11/05/05</td>
<td>13/05/05</td>
<td>2 days</td>
</tr>
<tr>
<td>Debugging</td>
<td>13/01/05</td>
<td>17/01/05</td>
<td>4 days</td>
</tr>
<tr>
<td>Test 2</td>
<td>21/05/05</td>
<td>30/05/05</td>
<td>9 days</td>
</tr>
<tr>
<td>Finalization</td>
<td>01/06/05</td>
<td>15/06/05</td>
<td>2 weeks</td>
</tr>
</tbody>
</table>