Project Report

On

Online Restaurant Meal Reservation System

Supervisors

Noah Torp-Smith noah@itu.dk
Jens Christian Godskesen jcg@itu.dk

Submitted By

Akshaya Kumar Kanchinkoti akshayekumar@itu.dk
Rizwan Ali rialf03@itu.dk
Imran Walayat imwae03@itu.dk
Yasir Bashir yasjilla@itu.dk.

Duration: 3rd May – 28th May

Preface

This Report has been written by a group of students at the IT University of Copenhagen during the period 3rd May – 28th May, under the supervision of Noah Torp-Smith and Jens Christian Godskesen.

We would like to thank to our supervisors Noah Torp-Smith and Jens Christian Godskesen for their valuable comments and advices for this 4 week project.

We would also like to thank the System Administrator for keenly extending their help.

1. INTRODUCTION

1.1 Objective

The main objective of this 4 week project is to develop a client/server model, which deals with **"**Online Restaurant Meal Reservation System". The system has two parts first for the customers and the other for the management side.

The customer side allows the customer to view menu list according to the time of delivery he desires and reserve meal for that specific time, and at the management side the staff is allowed to edit information regarding menu list, price, assigning cook, maintain information regarding the orders placed, etc.

1.2 Scope

We are greatly constrained by "Time limitations" as only 4 weeks are given to complete the project. So we limited our scope of project to "Meal Reservation" only.

The sole purpose of the project is to learn the web application development in Java, how databases works for the web application and how to co-ordinate Java with databases to produce more commercial web applications for daily business.

We will try our best to make better usage of java (servlets, jsp), MySql and surely all in a better team work.

2. PROBLEM STATEMENT

2.1 Problem Area

Restaurant is a kind of business that serves people all over world with ready made food. Currently this industry is going on with lot of flare. People feel more comfortable with lot of variations in the selection and consumption of their food in their busy life.

One can see lot more restaurant in the world. Even in Denmark one can see thousands of restaurants with dishes from all over the world like from India, Pakistan, Mexican, etc fulfilling the needs of people with nourishments and enjoyments.

Let's concentrate on booking area in a restaurant. In traditional booking system, a customer has to make a phone call in order to get his meal reserved. If luckily the phone gets connected, then the customer does some formal conversation like hello, hi, etc. Than he demands for today's menu and do some discussion over menu items then he orders and he has to give some of this identification specifications. This process takes 5-8 minutes to complete. On the receiver side there is hardly one phone line and one operator. So he can cover around 15-20 orders maximum in an hour.

For each booking he has to register manually on paper and puts the order in a queue with specific priority according to time and quantity, and than a cook is assigned for the specific order to complete it.

There are lots of areas to be solved for current restaurants using modern IT World. Many areas come like human resource management, accounts management, etc. But our problem lies within domain of end customer and restaurant "Meal Reservation".

2.2 Problem Analysis

As discussed earlier our main problem area focuses on the “Meal reservation/booking system”, there are lot of problems in that area which are associated with both the customer and the restaurant staff.

We would like to analyze some of the problems here:

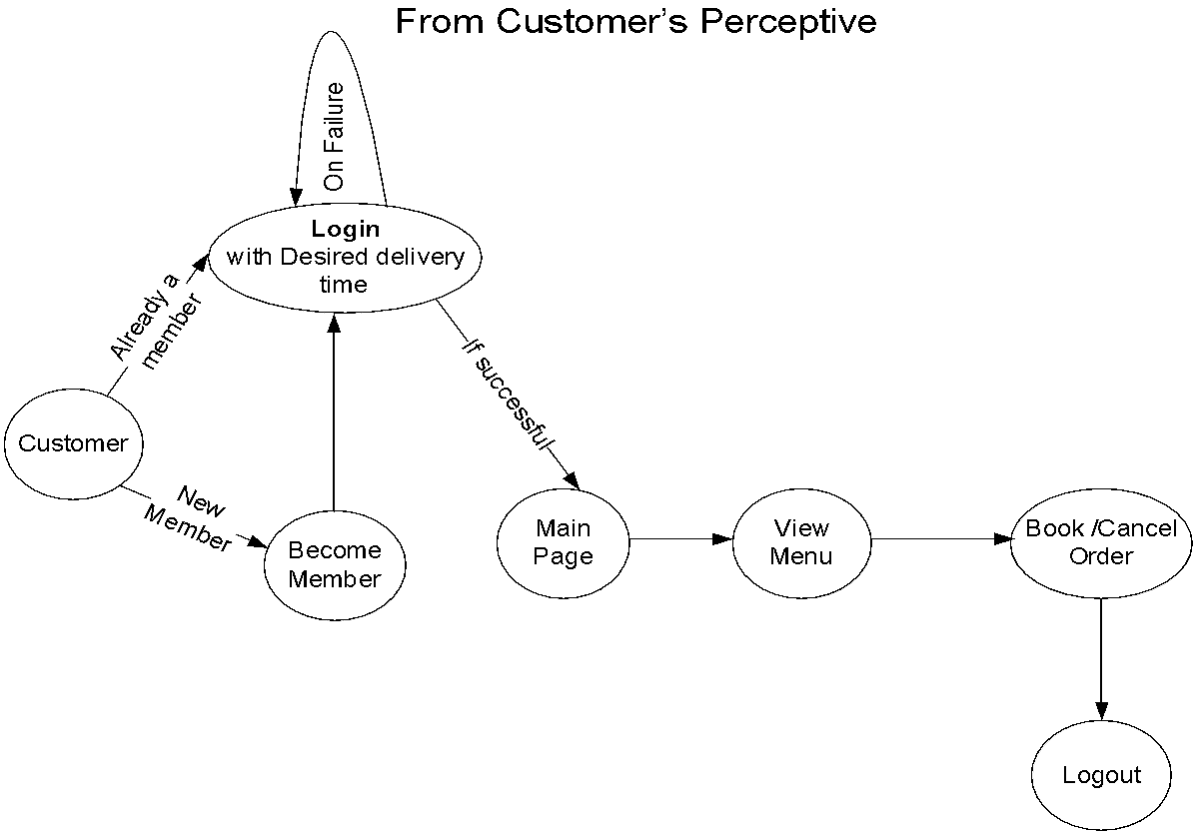
a. Initial problem is that the customer has to get connected over the phone, it would be harder if the restaurant is very popular and busy.
b. As customer won’t have the menu list with him, it would be harder for him to remember the entire list (with price as well...!) and come to a decision, i.e. customer is provided with less time to make decision.
c. The chances of committing mistakes at the restaurant side in providing a menu list for a specific time would be more.
d. There might be some communication problems or sometimes language might be a barrier.
e. As entire booking has to be done manually at the restaurant end, the chances of occurrence of mistakes is high as well.
f. Most of restaurants have single phone line and a single operator to handle incoming calls, so they can accept limited orders.
g. If the restaurant is of busy type, than the operator is left with no time to decide over the priority of the order fulfillment.
h. Even assigning orders (or some menu from the order) to a specific cook can be cumbersome if it is done parallel with the bookings of the order.
i. All the calls will not by intended for booking, as some calls might be for canceling the order or to fetch the status as well, this eats up the productive time at the restaurant side.

Still there might be many other problems associated with the traditional system of booking through telephone. So what should be the solution for these problems..?

2.3 Solutions for the Problems (How to solve this!!!!)

The solution for the above problems, so far we have thought, is client-server system that listen the requests online. We have the intensions to make the system that takes the customer reservations through the browser.

But how to make it..?

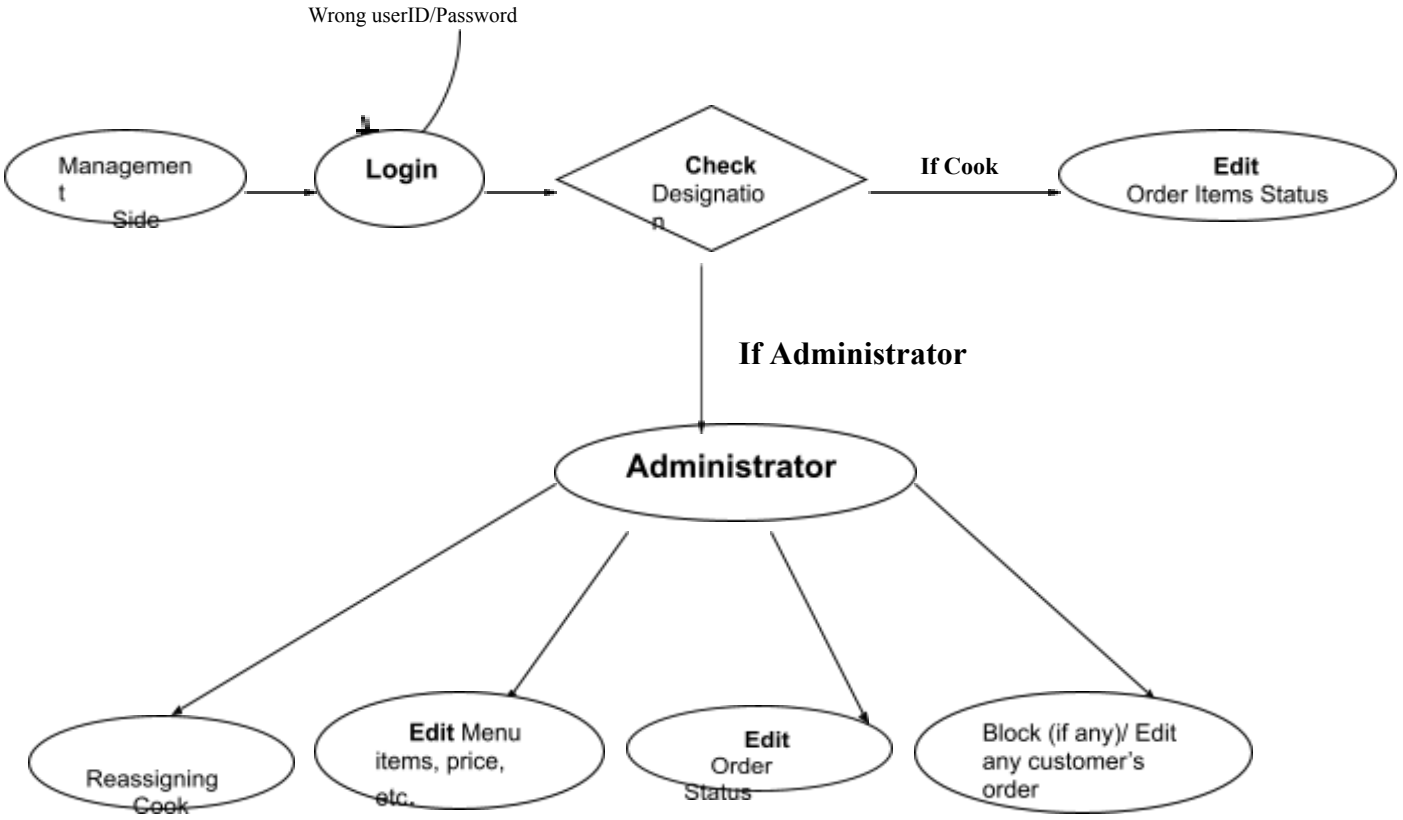


To resister a meal online, the customer has to become a member first then he can access the later part of the site. The option of becoming member was only an attempt to avoid (to some extent) placing the fake bookings.

During login the user as to enter his emailed/UserID, password and desired time of order delivery. After successful login the customer can access the menu page with the items listed according to the desired time. Later within the available items he can search for a menu according to his choice i.e. according to price range and category of food and later he can order a meal.

If the customer later wants to cancel the order, he is permitted to do this only within a specific time period. The customer is also given with the facility to view the status of the order and if the order is ready then he can go and get it.

From Management's Perspective



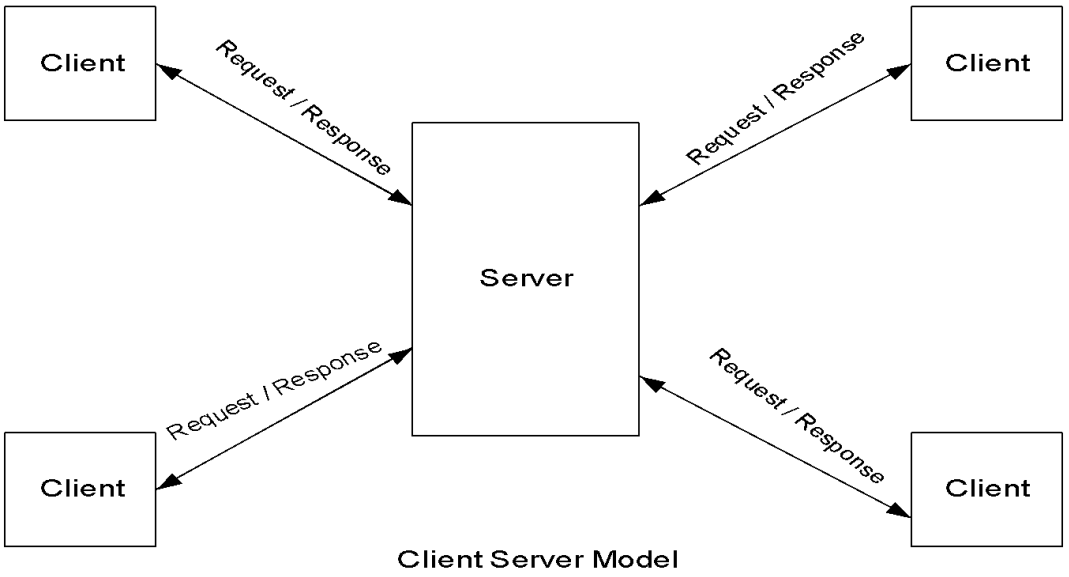
At Management side, initially the staff member has to login, and according to his designation the privileges are set. If the staff member is a cook, then he is allowed to edit only the order items status, indicating which menu items he has prepared.

If suppose the member is an administrator then, he is allowed to reassign the cook according to his priority, he can edit the menu information such as its price, items available currently, etc. He can also change the status of the order (in some special

cases), and can also block (if any customer exists)/Edit any customer's order according to his priority.

2.4 Technological Background

2.4.1 Client Server Architecture



Client Server Model

Client/server describes the relationship between two computer programs in which one program, the client, makes a service request from another program, the server, which fulfils the request.. In a network, the client/server model provides a convenient way to interconnect programs that are distributed efficiently across different locations. Computer transactions using the client/server model are very common.

Some of the characteristics of Client/Server Architecture

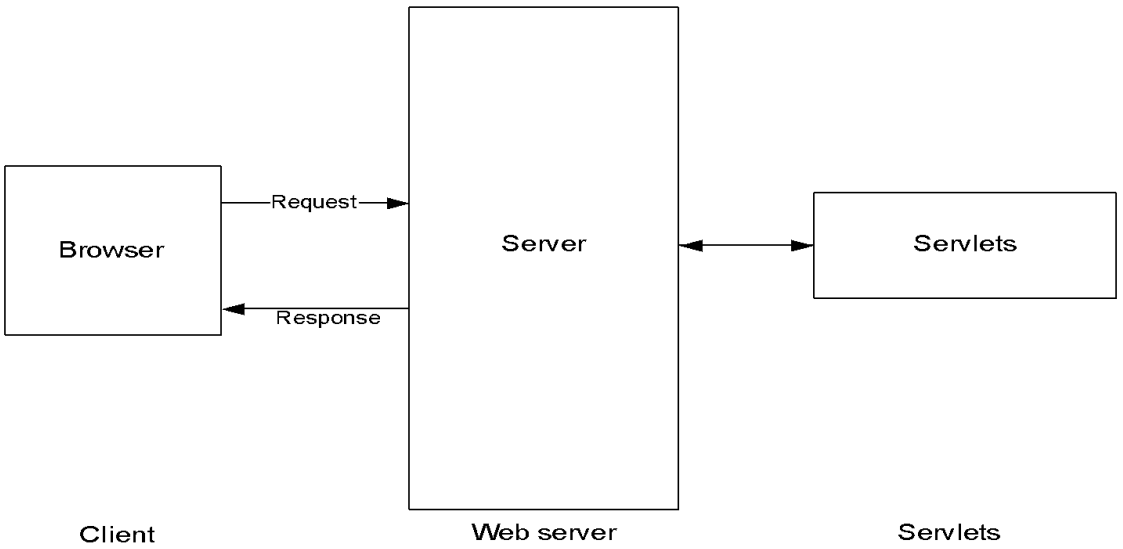
- *Asymmetrical protocols*—there is a many-to-one relationship between clients and a server. Clients always initiate a dialog by requesting a service. Servers wait passively for requests from clients.
- *Encapsulation of services*—the server is a specialist: when given a message requesting a service, it determines how to get the job done. Servers can be upgraded without affecting clients as long as the published message interface used by both is unchanged.

- *Integrity*—the code and data for a server are centrally maintained, which results in cheaper maintenance and the protection of shared data integrity. At the same time, clients remain personal and independent.
- *Location transparency*—the server is a process that can reside on the same machine as a client or on a different machine across a network. Client/server software usually hides the location of a server from clients by redirecting service requests. A program can be a client, a server, or both.
- *Message-based exchanges*—clients and servers are loosely-coupled processes that can exchange service requests and replies using messages.
- *Modular, extensible design*—the modular design of a client/server application enables that application to be fault-tolerant. In a fault-tolerant system, failures may occur without causing a shutdown of the entire application. In a fault-tolerant client/server application, one or more servers may fail without stopping the whole system as long as the services offered on the failed servers are available on servers that are still active. Another advantage of modularity is that a client/server application can respond automatically to increasing or decreasing system loads by adding or shutting down one or more services or servers.
- *Platform independence*—the ideal client/server software is independent of hardware or operating system platforms, allowing you to mix client and server platforms. Clients and servers can be deployed on different hardware using different operating systems, optimizing the type of work each performs.
- *Reusable code*—service programs can be used on multiple servers.
- *Scalability*—client/server systems can be scaled horizontally or vertically. Horizontal scaling means adding or removing client workstations with only a slight performance impact. Vertical scaling means migrating to a larger and faster server machine or adding server machines.
- *Separation of Client/Server Functionality*—client/server is a relationship between processes running on the same or separate machines. A server process is a provider of services. A client is a consumer of services. Client/server provides a clean separation of functions.

- *Shared resources*—one server can provide services for many clients at the same time, and regulate their access to shared resources.

2.4.2 About Servlets and JSP

Servlets



Servlets are loaded and executed by a web server. A servlet accepts requests from a client (via the web server), performs some task, and returns the results.

A servlet has to perform follows tasks:

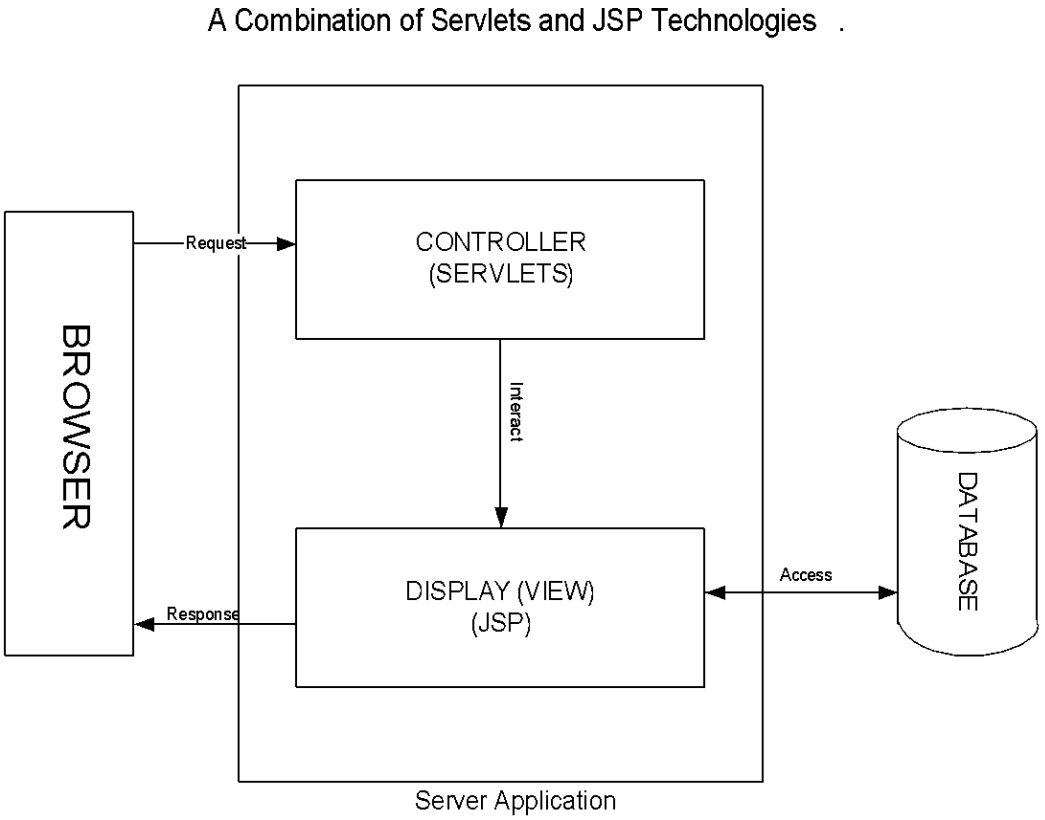
- Read Explicit data (form data) and implicit data (Request headers) sent by the client.
- Generate the results.
- Send explicit data back to client (in HTML format).
- Send implicit data back to client (status code and the response headers).

JSP (Java Server Page)

Java Server Pages (JSP) technology enables us to mix regular, static HTML with dynamically generated content from servlets. Java Server Pages (JSP) technology offers a simple way to create dynamic web pages that are both platform-independent and server-independent, giving you more freedom through Java technology's "Write Once, Run Anywhere" capability.

JSP technology separates content generation from presentation and takes advantage of reusable tags and objects, simplifying the maintenance of your web applications.

2.4.3 Integrating Servlets and JSP (why...?)



It may seem that servlets and JSPs are mutually exclusive, but in fact they work well together. You can reduce the amount of Java coding in your JSP by passing control from a servlet to a JSP.

The approach used by us is a hybrid approach for serving dynamic content, since it combines the use of both servlets and JSP. It takes advantage of the predominant strengths of both technologies, using JSP to generate the presentation layer and servlets to perform process-intensive tasks. Here, the servlet acts as the *controller* and is in charge of the request processing and the creation of any objects used by the JSP, as well as deciding, depending on the user's actions, which JSP page to forward the request to. Interestingly in particular there is no processing logic within the JSP page itself!! it is simply responsible for retrieving any objects that may have been previously created by the servlet, and extracting the dynamic content from that servlet

for insertion within static templates. In our opinion, this approach typically results in the cleanest separation of presentation from content.

3. SOFTWARE REQUIREMENT SPECIFICATIONS

3.1 Functional Specifications

3.1.1 Customer Specifications:

a. ***Search Menu according to the time of order delivery:***
During login, the costumer has to enter the time of delivery, and the menu items will appear accordingly. As availability of items differ from time to time.

b. Allow to search menu according to price and category:
Customers can search menu according to price range and the category as well, this functionality will be working as a search under search, i.e. first search criteria will be time of order delivery (which is default for all cases) and the later will be search within those menus according to price and category.

c. ***Allow to book/cancel order:***
Allowing customer to place an order or allowing him to cancel the placed booking.

d. ***Allow the customer to edit/create his personal account (membership):***
Here customer can create his new account or he can edit his personal information from the existing account.

e. ***Allow customer an option to change his password:***
Customers are given with a facility to change his existing password.

3.1.2 Management Specifications:

a. ***Edit Menu records:***

A manager can edit/create some or whole part of the menu record on daily basis. That is by changing the menu items, prices, description, etc.

b. Reassign Cook:

A manager can reassign the cook for a specific order or an item, originally the system assigns the cook automatically when an item is booked using some distribution fashion i.e. round robin fashion.

So a member can reassign the cook if such situation arrives.

c. Cancellation the order:

A manager can cancel of the order, this situation comes when the something goes wrong with working or in some unusual situations.

d. ***Create/Edit Cook's list:***

A manager can create a new entry of cook or edit the existing cook's information from the cook's list.

e. Blocking specific customer:

A manager can block a particular customer before signing up. This happens when the customer's previous history is bad, i.e. if he does not come up to take his orders which were ordered previously.

e. ***Allotting some privileges to cook:***

As previously stated a manager can create a cook, along with creation of a new cook he is given some privileges. The privileges include editing the status of the order, i.e. he can enter the status as ready if he has completed that item or can state the order is in process before preparing that item.

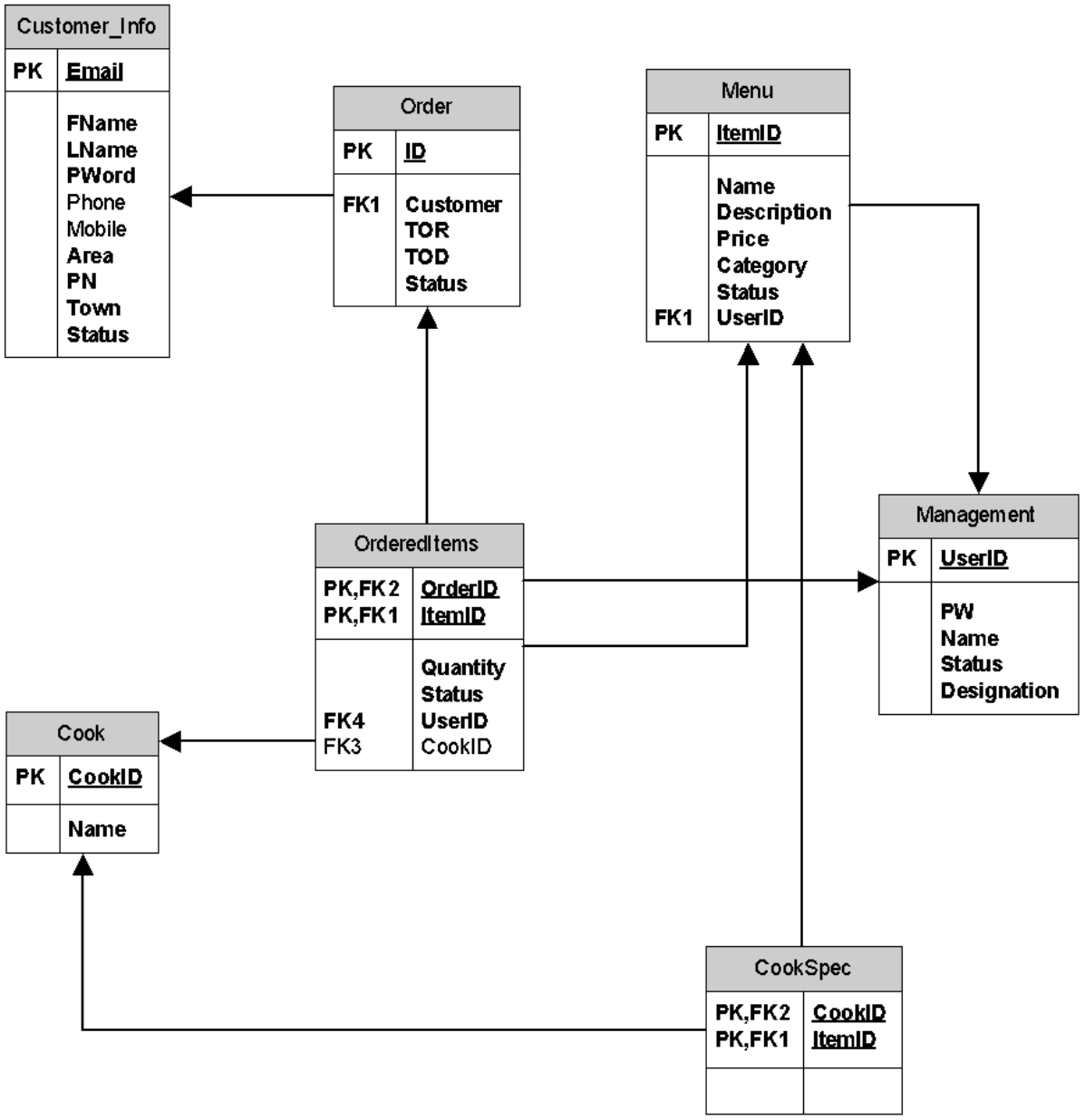
3.2 Software Specifications:

4. USER GUIDE

5. SYSTEM DESIGNING

5.1 Database Design:

ER Diagram for "On line Meal Reservation System"



Description

Customer_info:-

This table keeps the record of the customer's information before user logs on, he fills up a form that guides him how he can become a member.
EmailID is primary key in this table so we can recognize each member's emailID uniquely as it is used as their userID as well.
Other information includes customer Name, password , contact no, Address and status, the later tells him about whether the member is blacklisted or locked.
The entity shares a 1:N relation with order utilities.,

Menu :

The name insists, it contains the information of all menus and its related matter, Each menu is uniquely identified by its ItemID (Primary key). The purpose here is to provide customers all the information regarding menu such as Name (item), Description, Category, price and status (to check customer, whether that item is currently available or Not!!).
Later, at the management's point of view, we provide userID (uniquely selected by management staff) to alter the contents of table. This entity shares N:1 relationship with the ordered item entity.

Ordered Item:

This table provides information to the management staff regarding the uniquely generated orderID, which may contain one or more menu item uniquely identified by ItemID.
Apart from this, the management can check/alter the status of the order along with, they can alter the cook/s which was previously assigned to fulfill the order (as each cook is uniquely identified by cookID), this can be done by the management staff by logging in with UserID.
A cook can place status over this entity regarding the status of the specified item is ready for delivery or not !!, this is done by using uniquely provided cookID.

From customer's perspective, customer can check the status of his order in detail i.e. the status of each item in his order from this table as well as the quantity he ordered, the later can also be helpful at the management side.
This entity shares 1: N relationship with the Menu table

Order

This table tells about the OrderID (which is a primary key), who has placed the order and gives details about the time when order was placed and the time when the order will be delivered; along with the status of the order (usually some 5 status labels are assigned).
This table shares N:1 relationship with the Customer_info entity.

Cook:

Every cook is uniquely identified by his ID called 'cookID' and the other field "Name" is another step that will help management to recognize the specific cook. This entity shares N:N / N:1 relationship with the ordered item and 1:N relationship with cook spec entity.

Cook Spec:

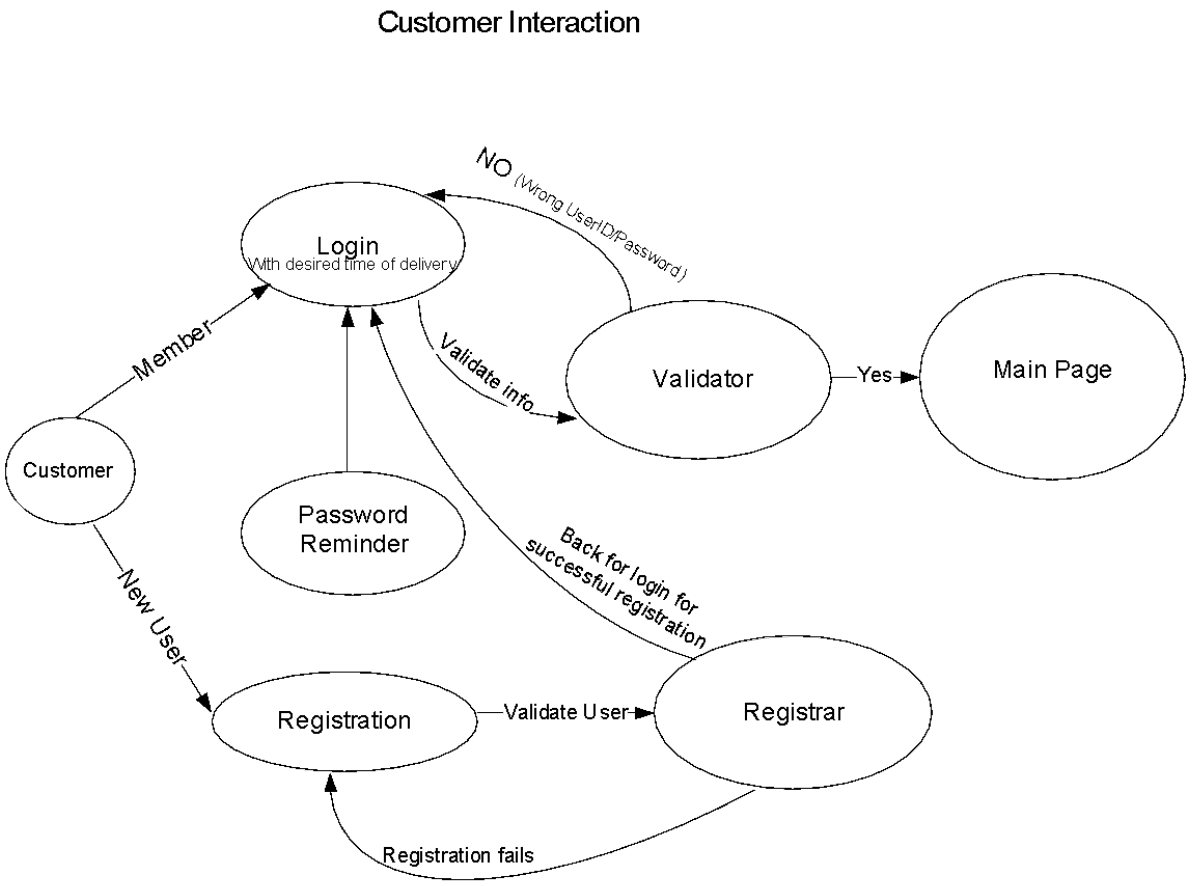
This table informs the management staff regarding the specific cook in his related items. Each cook is identified uniquely by ID same as before!!!

Management:

This Table represents the total management side of our project, UserID is the primary key in this table. Other fields include password and Name.
The status tells about which customer has to be blacklisted or blocked.
The "designation" field tells the level of user, say the administration side or a cook. The privileges are set according to the designation for eg : A cook cannot cancel the order and so.
The management staff can access the Menu and the ordered items and table.

5.2 System Design

5.2.1 System Design for Customer



For giving the order, the user should become a member initially. User would have to install his information like the address and other key information so that he doesn't have to give his information each time. For signing up every customer has to give some this details such as address, name, Contact_no etc and the most important is email ID which is the primary key to identify each customer uniquely, thus email becomes the User_ID for the customer, immediately after submitting the form, a password is sent to respective Email_ID so he can access the site and service.

The question that "why the user is not permitted to choose his own password initially?" And "the reason behind this approach is that, by doing this we can validate the user's email and later on he can access the site.

But again how Validation...?

Our only intension of doing this is that, only that person (user) should access the site who originally owns that emailID (as only that person can extract the password given by us, who knows the password for this emailID!!!!!)....and this way we can avoid users giving others emailID.

The only thing needed here is to sign in to the system through browser and from any place where internet is available. Now he would have the option to edit his current information and big thing to reserve the meal diminishing the human interaction. He would have today's menu (according to the time of delivery which he has entered) in front of him and he have the clear choices for order. He has variety of things to do here and have the option to cancel the order before the specific time of completion. For the first time to access the system, customer has to give his key information like identification and so on.

For the management side, it is quite possible to book many orders concurrently. System will be able to book nearly infinite number of orders at a time. Management side has more updated information and they can get the currently booked orders through the browser and all automatically. System will be able to deal with the customers who don't come to take there orders by blocking them and not letting them to signup again. This is done by maintaining some information regarding the status of order and the relative customer. So according to that the system can deal him. How system deals with him..? He should have to pay the amount of the last order in order to continue with his membership.

All that would be implemented in Java Server Pages and Java Servlets.

At Login page we will be checking the user's existence and mapping his userID/emailID with his password, if the user is valid then he is allowed to access further.

At Registration, it is checked that the userID/ emailID is not pre-existening, along with various general events/acts such as the customer had entered right format of the e-mail, or it is not entered NIL, all phone & mobile nos contain only the numbers, etc.

The option of password reminder is also included, so that when the user forgets his password then he can get a new password by giving his emailID, only if he already exists!!.

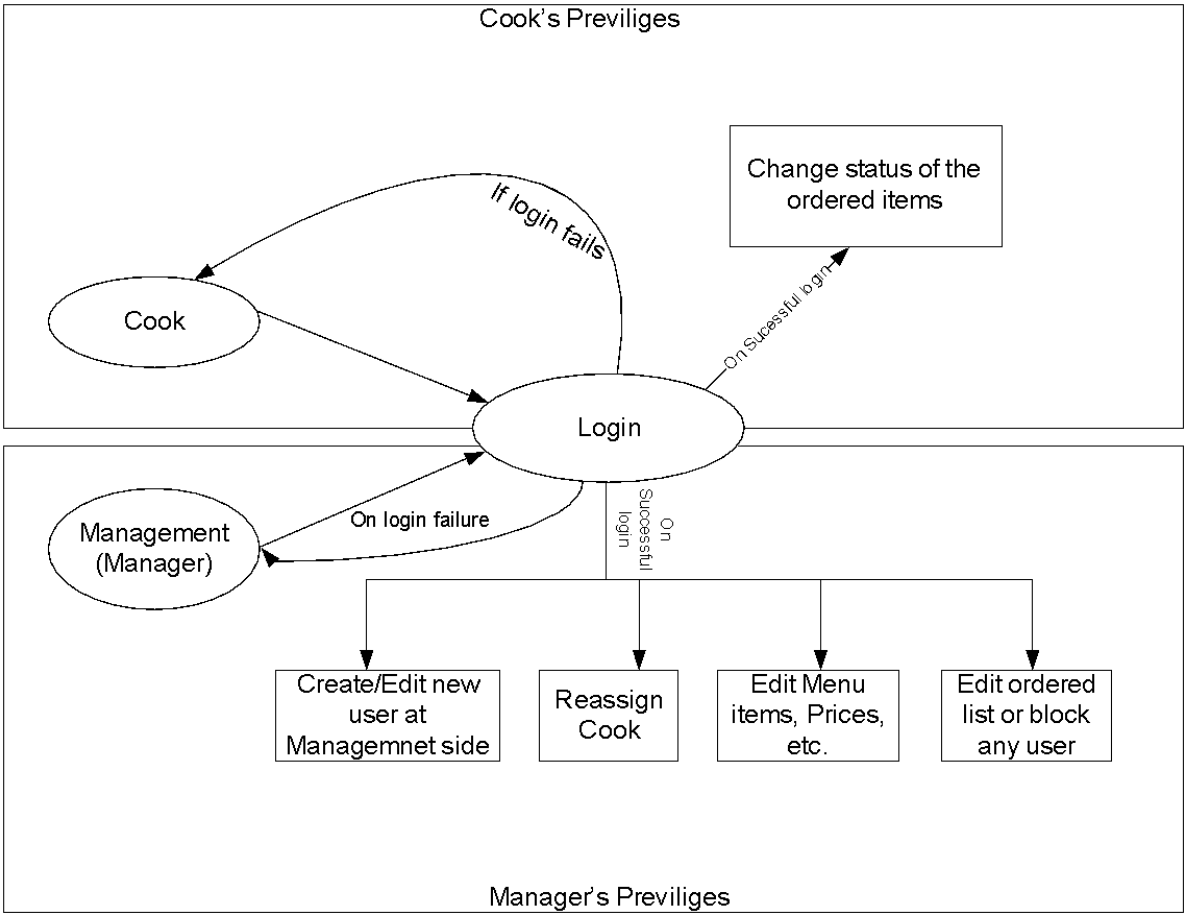
By the Management perspective, we will be imposing some prevailages so that only an authorized management staff/User can alter the contents of the site. The system also traces which user had altered the contents of the site, as each user at the management side is provided with unique UserID.

At the management side we have usually two types of the users, one the manager and the other is cook, the later can only alter or deal with the status of the ordered items, which he was given to complete. And the former can do all the updations required at the management side such as reassigning a cook, editing menu items, it's prices, descriptions, edit order status or can block some user if situation demands.

Data Flow diagram for Customer:

5.2.2 System Design for Management side:

Management side interaction



Explanation:

Data flow diagram for Management side: