

SOFTWARE ENGINEERING THEORY PROJECT

Fall Semester 2016-2017

TOPIC:

Restaurant Service System

Teacher:

Prof. Alok Chauhan

STUDENTS:

- Osho Agyeya(15BCE1326)
- Kashish Miglani(15BCE1003)
- Sanchay Gupta(15BCE1190)
- Sachin Gopal(15BCE1188)

[illegible]

TOPIC:

Restaurant Service System



Teacher:

Prof. Alok Chauhan

STUDENTS:

- Osho Agyeya(15BCE1326)
- Kashish Miglani(15BCE1003)
- Sanchay Gupta(15BCE1190)
- Sachin Gopal(15BCE1188)

I. Product Definition

Problem Statement

To create a site that automates the process of placing orders and reduces miscommunication and discrepancy that occur during the order process and generates the bill automatically.

Functions to be provided

1. Login and signup of the user
2. Admin login
3. Order placed from user to admin
4. Bill Calculation
5. Update and recover password

Processing environment: hardware/software

1. Notepad
2. Web Browser (Google Chrome 53)
3. XAMPP

User Characteristics

1. Customer
 - Place order
 - Retrieve password
 - Update password
 - Login and Signup
2. Owner
 - View orders
 - View customer details

Solution Strategy

Creating and developing a website which handles requests by the customer and provides the owner with appropriate controls.

Product Features

- Login page
- Signup page
- Forgot Password
- Menu Page
- About Us page
- Location
- Admin view
- Bill generation
- Order database management
- Customer database management

Acceptance Criteria

Simplifies the ordering process and reduces the time complexity.

Sources of Information

- W3Schools
- Google Material Design
- Stack Overflow
- GitHub

II. Project Plan

Life Cycle Model

The software requires the use of waterfall model for developmental processes.

Team Structure

Non-egoistical work model:

- Front-end designing – Sachin Gopal and Sanchay Gupta
- Back-end designing – Kashish Miglani and Osho Agyeya

Development Schedule

Milestones:

- Requirement elicitation
- Requirement analysis
- Preliminary design
- Final design
- Modular division
- Unit coding
- Module integration
- Testing

Reviews

- PFR (Product Feasibility Review)
- SRR (Software Requirement Review)
- PDR (Preliminary Design Review)
- CDR (Critical Design Review)
- SCR (Source Code Review)
- ATR (Acceptance Test Review)
- PRR (Product Release Review)
- PPM (Project Post-Mortem)

Programming languages and development tools

- HTML
- CSS
- JavaScript
- SQL
- PHP
- jQuery

- XAMPP

Documents to be prepared

- System Requirements
- Software Requirements Specification
- Design Document
- User's Manual
- Test Plan
- Source Code Documentation
- Project Legacy

Manner of demonstration

Hosting the website on a single system which acts as both the server and client system.

Sources of information

- W3Schools
 - Google Material Design
 - Stack Overflow
 - GitHub
 - Software Engineering: A Practitioner's Approach by Roger Pressman
-

Software Requirements Specification

for

Restaurant Service System

Version 1.0 approved

Prepared by

Osho Agyeya, Kashish Miglani, Sachin Gopal, Sanchay Gupta

VIT University

25/07/2016

Table of Contents

Table of Contents	ii
Revision History	ii
1. Introduction.....	1
1.1 Purpose	1
1.2 Document Conventions.....	1
1.3 Intended Audience and Reading Suggestions.....	1
1.4 Product Scope	1
1.5 References.....	1
2. Overall Description	2
2.1 Product Perspective	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment.....	3
2.5 Design and Implementation Constraints.....	3
2.6 User Documentation	3
2.7 Assumptions and Dependencies	3
3. External Interface Requirements	5
3.1 User Interfaces	3
3.2 Hardware Interfaces	4
3.3 Software Interfaces	4
3.4 Communications Interfaces	4
4. System Features.....	Error! Bookmark not defined.
5. Other Nonfunctional Requirements.....	Error! Bookmark not defined.
5.1 Performance Requirements.....	Error! Bookmark not defined.
5.2 Safety Requirements	Error! Bookmark not defined.
5.3 Security Requirements.....	Error! Bookmark not defined.
5.4 Software Quality Attributes.....	Error! Bookmark not defined.
5.5 Business Rules	Error! Bookmark not defined.
6. Other Requirements	7
Appendix A: Glossary.....	7
Appendix B: Analysis Models.....	8

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this document is to provide the software requirement specification report for restaurant service system being developed as a web application to facilitate online order placement and bill calculation.

This is the first release model of the product. Subsequent bugs will be removed after appropriate feedback from the clients.

1.2 Document Conventions

Certain abbreviations have been used in the SRS which have been provided in the glossary. All the headings of the SRS are **bold and black in color**. Critical terms have been highlighted with yellow. Emphasis is to be laid on bold letters for TLDR.

1.3 Intended Audience and Reading Suggestions

This is a college level project being implemented under the guidance of college professors. The targeted user domain is restaurants near VIT University, specifically Kunti restaurant. However, any restaurant/café/motel may make use of this service in order to augment their business by reaching out to more customers over the Internet.

Relevant app developers may make use of this document for developing apps as an alternative over functionalities to the current version. Testers may make use of this SRS to find bugs and contingencies emerging at runtime. Documentation writers may suggest relevant corrections to this SRS to enhance readability and comprehension while transforming the SRS into a document which follows global conventions of documentation.

The SRS documentation is supposed to be read in top-down manner. The document conventions have been specified above to clear any doubts regarding abbreviations/terminology used.

1.4 Product Scope (Problem Definition)

The project is focused upon developing a web application that will allow the specific restaurant administrators to handle various food orders that are placed by multitudes of customers over the Internet. The customers can make use of this web application to post their orders online and view the consequent bill that is generated. The administrator gets to decide which orders are verified by him/her.

1.5 References

For basic web development and specifications: https://en.wikipedia.org/wiki/Web_development

A brief note on SRS: https://en.wikipedia.org/wiki/Software_requirements_specification

2. Overall Description

2.1 Product Perspective

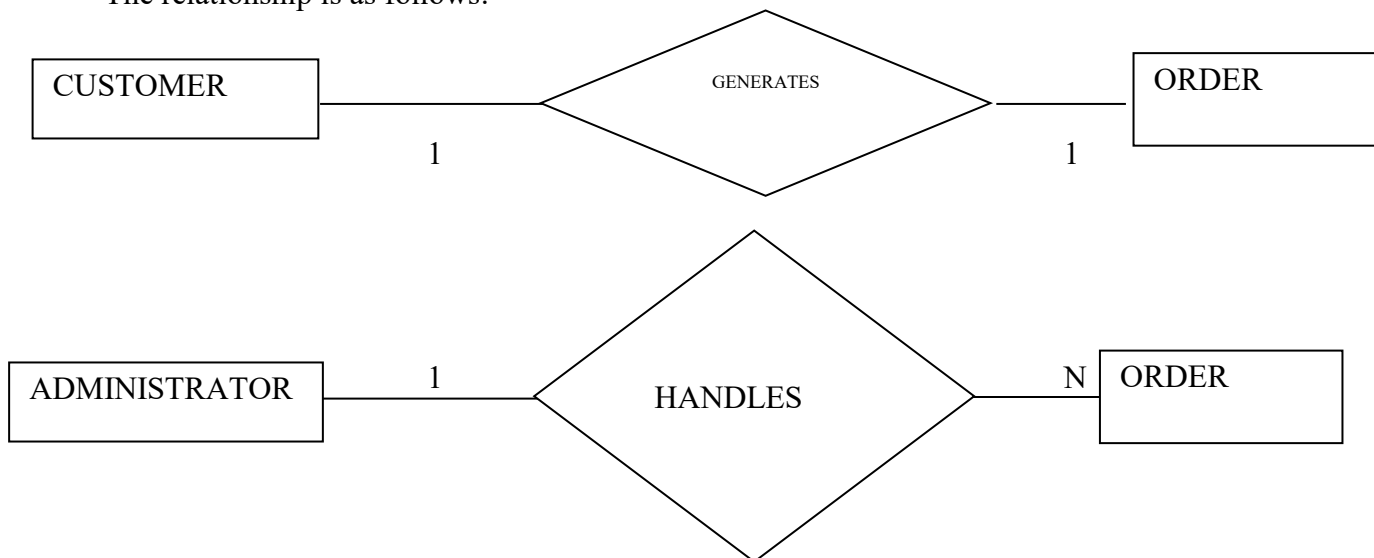
The traditional order recording process for any restaurant is the pen and paper mode along with telephonic conversation. However, this model has certain flaws such as non-simultaneous recording of orders for multiple customers since one telephone can support a binary sided call at a given point of time. Also, if the phone network is disturbed, then the traditional method fails. Therefore, a web based application will help to receive simultaneous orders online while helping the user to calculate the bill too.

2.2 Product Functions

- Create accounts for customer and administrator.
- Allow the customer to save the delivery address in his login along with certain other details.
- Enable the administrator to receive the order and verify it.
- Allow customers to post their orders online by choosing the dishes placed in the menu along with their quantity.
- Enable the administrator to calculate the bill for the received orders.
- Enable the customers to choose the mode of payment.

2.3 User Classes and Characteristics

First entity shall be the customer.
Second entity shall be the order.
Third entity will be administrator.
The relationship is as follows:



2.4 Operating Environment

The application is WAN based. The application shall run on all browsers like Google Chrome, IE, Mozilla, Safari, and Microsoft Edge. The application will work on mobile browsers too. The system needs to have stable Internet connection of at least 100 kbps for the entire order registering process to happen fluidly.

2.5 Design and Implementation Constraints

- The application can be operated only on a WAN connected environment(only online use is possible).
- No detailed security features have been incorporated.
- The web application is targeted at small scale food businesses.
- The food items that have been incorporated are not comprehensive. These might change according to the industry which is implementing the application.
- Insufficient internet connection might hamper the order recording process.
- GUI is only available in English.

2.6 User Documentation

Following online manuals/ebooks may be used to understand the basic nature of the web development process. Certain websites catering to the needs of web development for restaurant and similar food joints have also been incorporated.

<http://www.deitel.com/Books/InternetWebScripting/InternetWorldWideWebHowtoProgram4e/tabid/2048/Default.aspx>

<http://www.restaurantbyclick.com/>
<http://webdiner.com/>

2.7 Assumptions and Dependencies

- It is assumed that the customer and the administrator have working internet connection.
- Connection speed should be around 100kbps minimum for hassle free order placement.
- If the load on the website becomes too large (2000 users online), then the app might crash.
- The admin gets to decide whether any order is to be rejected or not.
- The list of food items have been restricted. Therefore, only the included items maybe ordered. For specific dishes, the admin needs to be communicated with telephonically.

3. External Interface Requirements

3.1 User Interfaces:

The basic user interface is based on advanced CSS designing. HTML 5 has been used in order to design the basic layout of the application. PHP has been used for server side scripting. HTML has

been used for client-side scripting. Certain images of delicious cuisines have been fetched from the Internet for aesthetic aspects of the website. Various options shall pop-up on the screen to collect the necessary information from the customer.

3.2 Hardware Interfaces

CPU: The processor should have minimum clock rate of 1.2 Ghz.

Internet Connection: Connection should be stable on minimum 100 kbps.

I/O: The mouse and keyboard are the only supported devices for receiving orders from the customers.

3.3 Software Interfaces

The application is supposed to be run on:

OS: Windows, MAC, UBUNTU (essentially all OS' supporting Internet)

Browsers supporting HTML 5: Google Chrome, Safari, Mozilla, IE, Edge.

Browsers supporting CSS 3.

<http://css3test.com/> : Developers can make use of this tool to find out whether CSS3 is supported on it or not.

Browser should support php for server side scripting.

3.4 Communications Interfaces

The web browsers that have been supported are Google Chrome, Internet Explorer, Mozilla Firefox, Safari. If the customer wants to connect with the admin, then he/she has to communicate with him/her telephonically. The mode of menu placement is an electronic form. The Email id is the unique login of the customer. HTTP is the communication standard that will be used in order to load the website. HTTP is the underlying protocol used by the World Wide Web and this protocol defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands.

4.0 System Features:

Functional requirements			
Requirement Index	Requirement title	Requirement definition	User inputs
1	login for registered customers	to enter details of account for login	given below as sub requirements
1.1	User login email-id	To decide who is accessing the account on the website	Email id
1.2	user password	To enter the student password to sign in	password
1.3	Sign up	To choose the option for sign up to create a new account	button click
1.03.01	name	To submit name for new account	name
1.03.02	email id	To submit unique email-id	email id
1.03.03	address	To give delivery address	address
1.03.04	contact no.	To submit contact no.	phone no.
1.03.05	password	To create a password	password
1.03.06	retype password	To confirm the password choice once again	password
1.03.07	submit	To submit all the details for account creation	Submit button
2.00.00	logout	To logout out of the website	logout button
2.00.1	details	To display user details	-
2.1	order details by customer	To enter the details of the order	given below as sub requirements
2.01.01	dish	To select the dish to be ordered	check box against dish name
2.01.02	quantity	To select the quantity for the dish selected	whole number
2.01.03	method of delivery	to select the mode of dining	either home delivery or dine in restaurant radio boxes
2.01.04	submit	To submit the order details	proceed to bill button
2.2	order management by admin	To manage the orders that have been placed by the customers	-
2.02.01	approve order	administrator selects the order to be approved	approve/reject button
2.02.02	approved orders	to allow the customers to view the orders that have been posted by him	View approved orders button
3.1	order approval details	to allow the customer to view whether the order has been placed or not	-
3.01.01	place new order	to allow the customer to give next order	button named new order
3.2	approved order management	to allow the admin to check off the approved orders that have been delivered	-
3.02.01	check off	to eliminate all the orders that have been cooked and delivered	completed button

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- **0.1 second** is about the limit for having the user feel that the system is reacting instantaneously, meaning that no special feedback is necessary except to display the result. These include requirement no 1.1-1.3,1.03,2.1.
- **1.0 second** is about the limit for the user's flow of thought to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second, but the user does lose the feeling of operating directly on the data. These include requirement no 1.3,2.00.00,3.02.01,2.00.01
- **10 seconds** is about the limit for keeping the user's attention focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect. These time will be spent while the attendance is being accepted/submitted. 1.3.07,2.01.04,2.02.01,2.02.02,3.01.01.

5.2 Safety Requirements

none

5.3 Security Requirements

The user can change his password and email at any point. Therefore, the user can change his/her attribute values as and when needed to ensure the security of his/her account.

5.4 Software Quality Attributes

Non-functional Requirements	
Platform Compatibility	Windows,MAC,UBUNTU(OS supporting Internet browsing)
Implementation logic	Website shall be created using appropriate tools in order to facilitate the order recording process.

Accessibility	Intended for food service institutions.
Availability	Source code shall be made public on GitHub.
	Initial availability is restricted to local restaurant in partnership with the developers.
Development platform	HTML 5,CSS 3,php
Scalability	Scalable to large scale food businesses like KFC
	as well as small scale food businesses like local restaurants.
Usability	Easy to grasp GUI
	Simplified menus
	Attractive+Informative styling
Price	Free
Performance	Short response time(maximum time taken by the app is to calculate the bill on the basis of items selected) Optimum throughput

5.5 Business Rules

This app is supposed to be used only for creating user and customer login, recording the orders of the customers, calculating the bill incurred and deciding the method of delivery. The admin shall keep on altering the data on the website according to his/her convenience/inconvenience.

6. Other Requirements

Additional requirements generated on the basis of user feedback shall be incorporated later.

Appendix A: Glossary

Following abbreviations have been used in the SRS:

Admin=administrator

HTML=hypertext markup language

S/w=software

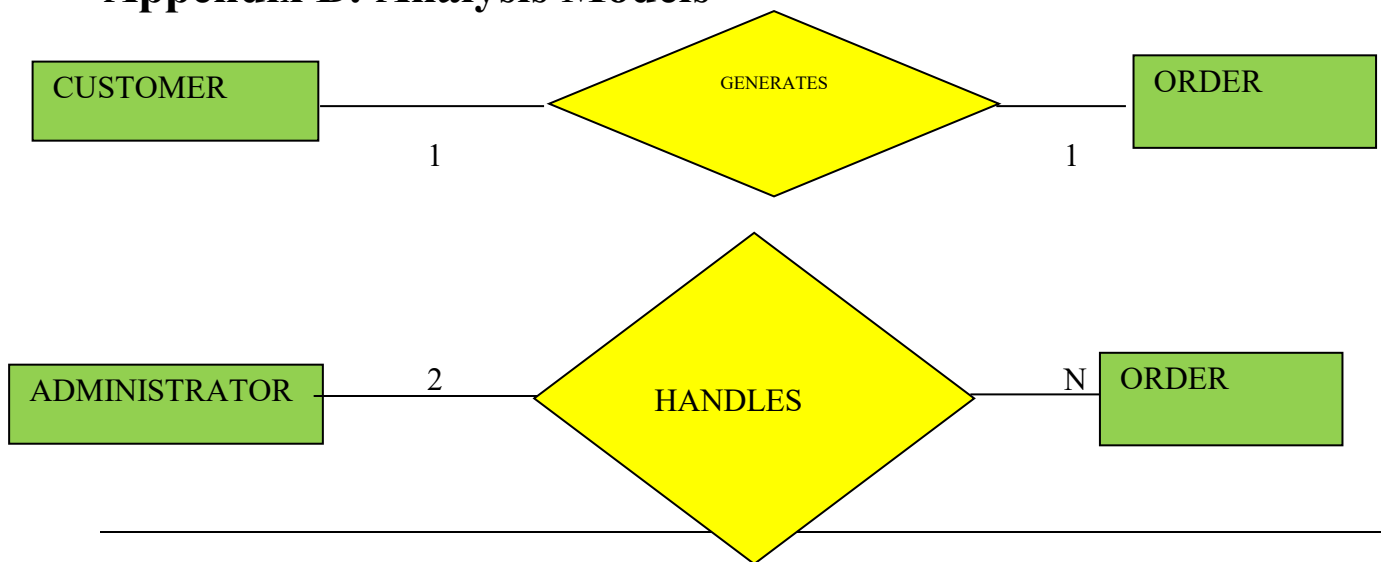
App=application

CSS: cascading style sheets

php: Hypertext Preprocessor

TLDR: Too long didn't read

Appendix B: Analysis Models



SOFTWARE ENGINEERING THEORY PROJECT DESIGN DOCUMENT



Fall Semester 2016-2017

TOPIC:

Restaurant Service System



Teacher:

Prof. Alok Chauhan

STUDENTS:

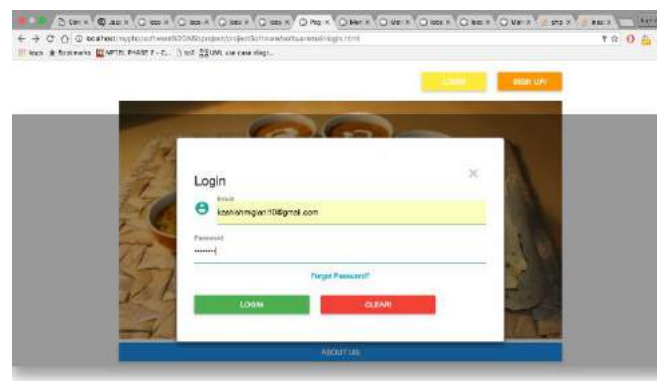
- Osho Agyeya(15BCE1326)
- Kashish Miglani(15BCE1003)
- Sanchay Gupta(15BCE1190)
- Sachin Gopal(15BCE1188)

Design Document

EXTERNAL DESIGN SPECIFICATIONS:

User Displays and report formats:

The user display is lucid and intuitive. The main home screen has been integrated with photos of lip smacking dishes to draw the attention of the user. The buttons are clear and sleek.



The menu has categorized the dishes according to ethnicities.

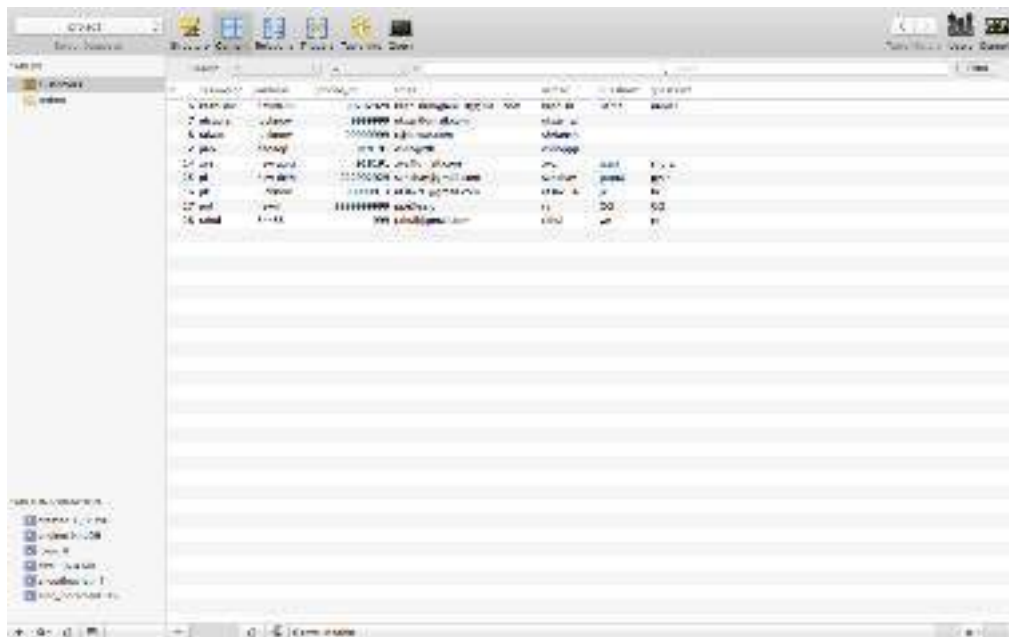
The screenshot displays the Oracle SQL Developer environment. At the top, there's a toolbar with various icons for file operations and database management. Below the toolbar, the 'TABLE1' table is selected in the left-hand pane. The main area shows the table's structure with 12 columns: C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, and C12. The table contains 6 rows of data. Below the table, the 'TABLE INFORMATION' section provides details about the table, including its name, owner, status, and other properties.

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	C11	C12
1	2	3	4	5	6	7	8	9	10	11	12
10	20	30	40	50	60	70	80	90	100	110	120
100	200	300	400	500	600	700	800	900	1000	1100	1200
1000	2000	3000	4000	5000	6000	7000	8000	9000	10000	11000	12000
10000	20000	30000	40000	50000	60000	70000	80000	90000	100000	110000	120000
100000	200000	300000	400000	500000	600000	700000	800000	900000	1000000	1100000	1200000

TABLE INFORMATION

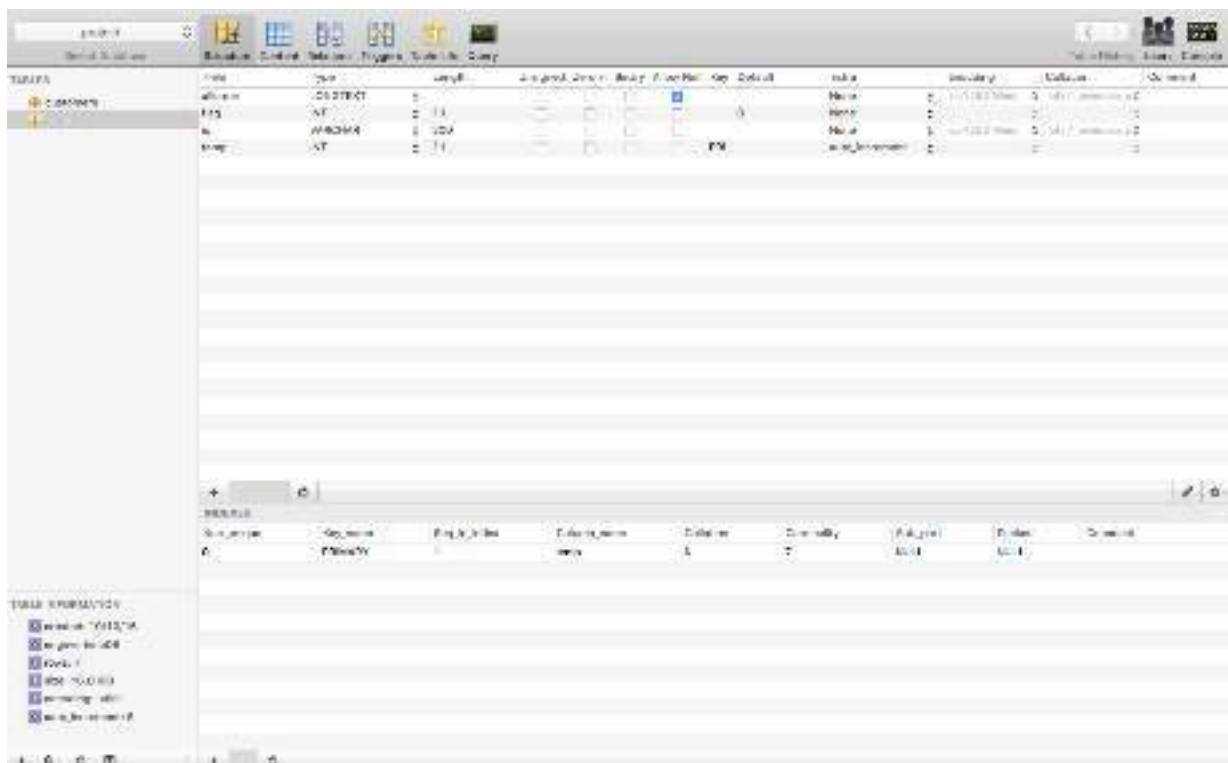
Table Name	Owner	Status	Tablespace	Clustering	Partitioning	Logging	Tablespace	Comments
TABLE1	SCOTT	VALID	USERS	YES	NO	YES	USERS	

The datatypes for customers table are visible above.



Column Name	Data Type	Nullable	Default Value	Comments
CUSTOMER_ID	NUMBER(4)	NOT NULL		
LAST_NAME	VARCHAR2(30)	NOT NULL		
FIRST_NAME	VARCHAR2(30)	NOT NULL		
EMAIL	VARCHAR2(100)	NOT NULL		
PHONE_NUMBER	VARCHAR2(20)	NOT NULL		
BIRTHDAY	DATE	NOT NULL		
CREDIT_LIMIT	NUMBER(10,2)	NOT NULL		
CREDIT_RATING	NUMBER(3)	NOT NULL		
PAYMENT_TERM	NUMBER(3)	NOT NULL		
TERMS_CONDITIONS	VARCHAR2(200)	NOT NULL		
MANAGER_ID	NUMBER(4)	NOT NULL		

The sample data for customers table are visible above.



CUSTOMER_ID	LAST_NAME	FIRST_NAME	EMAIL	PHONE_NUMBER	BIRTHDAY	CREDIT_LIMIT	CREDIT_RATING	PAYMENT_TERM	TERMS_CONDITIONS	MANAGER_ID
1	DEHAENE	FRANÇOIS	FRANCOIS.D@DEHAENE.COM	(31) (0)30 23 45 6789	1990-01-01	300000	AAA	30	Standard terms and conditions	7
2	BAERENTJEN	FRANÇOIS	FRANCOIS.BA@BAERENTJEN.COM	(31) (0)30 23 45 6789	1990-01-01	300000	AAA	30	Standard terms and conditions	7
3	BAERENTJEN	FRANÇOIS	FRANCOIS.BA@BAERENTJEN.COM	(31) (0)30 23 45 6789	1990-01-01	300000	AAA	30	Standard terms and conditions	7
4	BAERENTJEN	FRANÇOIS	FRANCOIS.BA@BAERENTJEN.COM	(31) (0)30 23 45 6789	1990-01-01	300000	AAA	30	Standard terms and conditions	7
5	BAERENTJEN	FRANÇOIS	FRANCOIS.BA@BAERENTJEN.COM	(31) (0)30 23 45 6789	1990-01-01	300000	AAA	30	Standard terms and conditions	7

The datatypes for customers table are visible above.

Windows Explorer window showing a folder named 'TABLE' containing a file named 'TABLE1'. The file is a CSV file with a table of data. The table has columns for 'Date', 'Time', 'Location', 'Status', and 'Count'. The data is organized into two main sections: 'TABLE1' and 'TABLE2'. The 'TABLE1' section contains 10 rows of data, and the 'TABLE2' section contains 10 rows of data. The file is highlighted in blue.

Date	Time	Location	Status	Count
2007-01-01	10:00	TABLE1	TABLE1	1
2007-01-01	10:01	TABLE1	TABLE1	2
2007-01-01	10:02	TABLE1	TABLE1	3
2007-01-01	10:03	TABLE1	TABLE1	4
2007-01-01	10:04	TABLE1	TABLE1	5
2007-01-01	10:05	TABLE1	TABLE1	6
2007-01-01	10:06	TABLE1	TABLE1	7
2007-01-01	10:07	TABLE1	TABLE1	8
2007-01-01	10:08	TABLE1	TABLE1	9
2007-01-01	10:09	TABLE1	TABLE1	10
2007-01-01	10:10	TABLE2	TABLE2	1
2007-01-01	10:11	TABLE2	TABLE2	2
2007-01-01	10:12	TABLE2	TABLE2	3
2007-01-01	10:13	TABLE2	TABLE2	4
2007-01-01	10:14	TABLE2	TABLE2	5
2007-01-01	10:15	TABLE2	TABLE2	6
2007-01-01	10:16	TABLE2	TABLE2	7
2007-01-01	10:17	TABLE2	TABLE2	8
2007-01-01	10:18	TABLE2	TABLE2	9
2007-01-01	10:19	TABLE2	TABLE2	10

Architectural Design Specifications:

Software architecture:

Architecture Genre:

- *Commercial,
- *Communications
- *Financial

Architecture Style:

Data-centered architectures. A data store (e.g., a file or database) resides at the center of this architecture and is accessed frequently by other components that

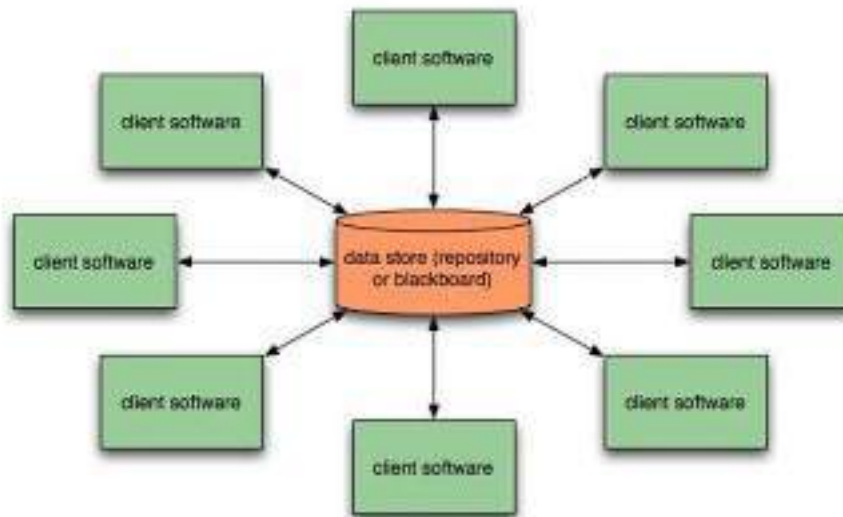
update, add, delete, or otherwise modify data within the store. Figure 9.1 illustrates a typical data-centered style. Client software accesses a central repository.

In some cases the data repository is passive. That is, client software accesses the

data independent of any changes to the data or the actions of other client software. A variation on this approach transforms the repository into a “blackboard”. that sends notifications to client software when data of interest to the client changes. Data-centered architectures promote integrability . That is, existing components can be changed and new client components added to the architecture without concern about other clients (because the client components operate independently). In addition, data can be passed among clients using the blackboard

mechanism (i.e., the blackboard component serves to coordinate the transfer

Data-Centered Architecture



32

of information between clients). Client components independently execute processes.

Since the customer and admin details are going to be stored in the database managed by the administrator, therefore a data centered architecture will be the best choice for this project. Plus, the database can be modified as and when required by the administrator.

Detailed Design Specifications:

Component interface specifications:

The interface of the site is aimed to be user friendly and easy to use. Using advance design library the site is easy upon the user eyes and appeal with its simplistic design.

The frontend component is designed using

- HTML
- CSS
- Java Script

The backend is made using

- PHP
- SQL

The different component in our site are:

- Register/create account
- Login
- Choose dishes according to the offerings in the menu
- Calculate the bill
- Delivery options
- Approve/reject order(For admin)
- Change password

Each of the components in the site is linked to one another through the user main page or the menu page using hyperlinking in the front end and using session and cookies in the backend. The entire backend is linked with a SQL database for the storing of data of the system like orders, user details and menu items

Documentation for each routine:

ROUTINE	Precise routine details	User inputs for each routine
login for registered customers	to enter details of account for login	given below as sub requirements
User login email-id	To decide who is accessing the account on the website	Email id
user password	To enter the student password to sign in	password
Sign up	To choose the option for sign up to create a new account	button click
name	To submit name for new account	name
email id	To submit unique email-id	email id
address	To give delivery address	address
contact no.	To submit contact no.	phone no.
password	To create a password	password
retype password	To confirm the password choice once again	password
submit	To submit all the details for account creation	Submit button
logout	To logout out of the website	logout button
details	To display user details	-
order details by customer	To enter the details of the order	given below as sub requirements
dish	To select the dish to be ordered	check box against dish name
quantity	To select the quantity for the dish selected	whole number
method of delivery	to select the mode of dining	either home delivery or dine in restaurant radio boxes
submit	To submit the order details	proceed to bill button
order management by admin	To manage the orders that have been placed by the customers	-
approve order	administrator selects the order to be approved	approve/reject button
approved orders	to allow the customers to view the orders that have been posted by him	View approved orders button
order approval details	to allow the customer to view whether the order has been placed or not	-
place new order	to allow the customer to give next order	button named new order
approved order management	to allow the admin to check off the approved orders that have been delivered	-
check off	to eliminate all the orders that have been cooked and delivered	completed button

Pseudocode for each routine:

The various routines that are covered under the site are

1. Register/create account:

Get the user following details from the user:

- Username
- Password
- Contact no:
- Email
- Address

When the user presses the submit button the system check whether the email address already exist or not and if it doesn't it creates a new user

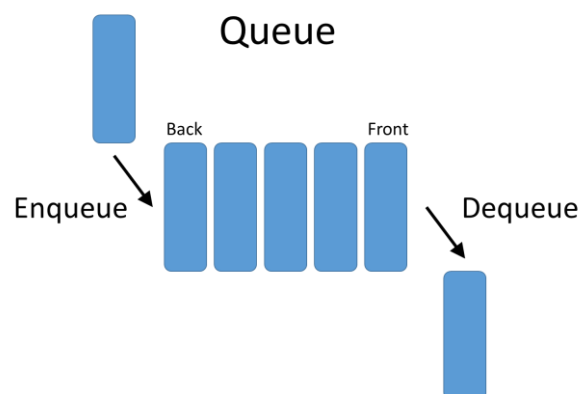
2. Login

A pop up is displayed and ask for the user to enter the username and password

- On click of the login button it redirect the user to his/her account
3. Choose dishes according to the offerings in the menu
Create navigable tabs for each section of the menu
List the item with their names and price and with a checkbox to check when the item is to be added to the cart
Get the user to enter the quantity of the food required for a checked item
 4. Calculate the bill
When the user place the order by pressing the submit button in the menu then subtotal of each item and grand total are to be displayed in a tabular manner
 5. Approve/reject order(For admin)
The admin can reject or accept a particular order on the basis of the real time inventory he has or the current status of the restaurant.
 6. Change password
The form ask the user for the following
 - His old password
 - His new password
 - Retyping his new password

Data Structure:

The data structure used for order processing is queue which operates in FIFO manner.



Packaging specifications:

Certain packaging principles have been followed in order to organize the entire system:

The Release Reuse Equivalency Principle (REP). *“The granule of reuse is the granule of release”.* When classes or components are designed for reuse, there is an implicit contract that is established between the developer of the reusable entity and the people who will use it. The developer commits to establish a release control system that supports and maintains older versions of the entity while the users slowly upgrade to the most current version. Rather than addressing each class individually, it is often advisable to group reusable classes into packages that can be managed and controlled as newer versions evolve.

The Common Closure Principle (CCP). *“Classes that change together belong together.”.* Classes should be packaged cohesively. That is, when classes are packaged as part of a design, they should address the same functional or behavioural area. When some characteristic of that area must change, it is likely that only those classes within the package will require modification. This leads to more effective change control and release management.

The Common Reuse Principle (CRP). *“Classes that aren’t reused together should not be grouped together”.* When one or more classes within a package changes, the release number of the package changes. All other classes or packages that rely on the package that has been changed must now update to the most recent release of the package and be tested to ensure that the new release operates without incident. If classes are not grouped cohesively, it is possible that a class with no relationship to other classes within a package is changed. This will precipitate unnecessary integration and testing. For this reason, only classes that are reused together should be included within a package.

SOFTWARE ENGINEERING THEORY PROJECT USER'S MANUAL



Fall Semester 2016-2017

TOPIC:

Restaurant Service System



Teacher:

Prof. Alok Chauhan

STUDENTS:

- Osho Agyeya(15BCE1326)
- Kashish Miglani(15BCE1003)
- Sanchay Gupta(15BCE1190)
- Sachin Gopal(15BCE1188)

MANUAL

INTRODUCTION:

Product rational and overview:

Restaurant service system has become a successful answer for efficient ordering in the fooding sector. Not only does it provide the convenience of ordering food from restaurants while sitting at home, it also helps the owner or the admin of the website to maintain a clean database of orders given. This generates efficiency in food resource planning while reducing the workload of both the admin and the customer. Online food ordering is a process of ordering food from a local restaurant or food cooperative through a web page or app. Much like ordering consumer goods online, many of these allow customers to keep accounts with them in order to make frequent ordering convenient. A customer will search for a favourite restaurant, usually filtered via type of cuisine and choose from available items, and choose delivery or pick-up. Payment can be amongst others either by credit card or cash, with the restaurant returning a percentage to the online food company.

TERMINOLOGY USED:

The basic terminology that will be followed during the entire software manual is given as follows:

A

Acceptance criteria: The exit criteria that a component or system must satisfy in order to be accepted by a user, customer.

Accuracy: The capability of the software product to provide the right or agreed results or effects with the needed degree of precision

Adaptability: The capability of the software product to be adapted for different specified environments without applying actions or means other than those provided for this purpose for the software considered

Assessment: Activity of determination of quantitative or qualitative value of a product, service, activity, process in regard to given quality or acceptance criteria.

Attractiveness: The capability of the software product to be attractive to the user .

Attribute: A characteristic of an object.

Availability: The degree to which a component or system is operational and accessible when required for use.

B

Baseline: A specification or software product that has been formally reviewed or agreed upon, that thereafter serves as the basis for further development, and that can be changed only through a formal change control process

Benefit: Value delivered to stakeholders

C

Changeability: The capability of the software product to enable specified modifications to be implemented.

Component: A minimal software item that e.g. can be tested in isolation.

Consistency: The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a component or system.

Constraint: A statement of restriction that modifies a requirement or set of requirements by limiting the range of acceptable solutions

Customer: Current or potential buyer or user of the products or service of an individual or organization, called the supplier, seller, or vendor

D

Defect: A flaw in a component or system that can cause the component or system to fail to perform its required function, e.g. an incorrect statement or data definition. A defect, if encountered during execution, may cause a failure of the component or system.

E

Efficiency: The capability of the software product to provide appropriate performance, relative to the amount of resources used under stated conditions

Error: A human action that produces an incorrect result

F

Failure: Deviation of the component or system from its expected delivery, service or result.

Feature: An attribute of a component or system specified or implied by requirements documentation (for example reliability, usability or design constraints)

Function: A description of “what” a system does.

Functional requirement: A requirement that specifies a function that a component or system must perform.

G

Goal: A desired state or result of an undertaken. Goals should be measurable and defined in time so that the progress can be monitored.

H

High-level: A position in a hierarchy of defined system components, which is closer to the top than the bottom, relative to the total defined set of those components

I

Impact: Estimated or actual numeric effect of a design idea on a requirement attribute under given conditions.

L

Learnability: The capability of the software product to enable the user to learn its application

M

Maintainability: The ease with which a software product can be modified to correct defects, modified to meet new requirements, modified to make future maintenance easier, or adapted to a changed environment .

Measure: The number or category assigned to an attribute of an entity by making a measurement.

N

Need: Something desired by a defined stakeholder. Satisfying that need would have some value for some stakeholder. A need might not be agreed as a formal requirement, and it might not be prioritized such that it is actually acted upon (designed and implemented). Need is a term often used as a stakeholder view of a problem before requirements specification is carried out .

Non-functional requirement: A requirement that does not relate to functionality, but to attributes such as reliability, efficiency, usability, maintainability and portability.

O

Operability: The capability of the software product to enable the user to operate and control it.

P

Performance: The degree to which a system or component accomplishes its designated functions within given constraints regarding processing time and throughput

Point of view: A certain perspective on the system or requirements.

Portability: The ease with which the software product can be transferred from one hardware or software environment to another

Priority: The level of (business) importance assigned to an item.

Process: A set of interrelated activities, which transform inputs into outputs .

Product: An output of a process.

Product requirement: A requirement related to the product of the development process. They affect quality of the product.

Q

Quality Assurance (QA): Part of quality management focused on providing confidence that quality requirements will be fulfilled

R

Reliability: The ability of the software product to perform its required functions under stated conditions for a specified period of time, or for a specified number of operations.

Requirement: (1) A condition or capability needed by a user to solve a problem or achieve an objective. (2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents. (3) A documented representation of a condition or capability as in (1) or (2) .

Requirements analysis: A set of tasks, activities and tools to determine whether the stated (elicited) requirements are unclear, incomplete, ambiguous, or contradictory, and then documenting the requirements in a form of consistent model.

Review: An evaluation of a product or project status to ascertain discrepancies from planned results and to recommend improvements. Examples include management review, informal review, technical review, inspection, and walkthrough .

Risk: (1) The effect of uncertainty on objectives, whether positive or negative (2) A factor that could result in future negative consequences; usually expressed as impact and likelihood.)

S

Safety: The capability of the software product to achieve acceptable levels of risk of harm to people, business, software, property or the environment in a specified context of use .

Scalability: The capability of the software product to be upgraded to accommodate increased loads

Scenario: (1) A projected course of action, events or situations leading to specified result. (2) An ordered sequence of interactions between specified entities (e.g. a system and an actor). (3) In UML: an execution trace of a use case.

Scope: The extent of influence of something. Scope can apply to anything, like a specification, or a specified system or project

Security: Attributes of software products that bear on its ability to prevent unauthorized access, whether accidental or deliberate, to programs and data [ISO/IEC 25000]. See also *Functionality*.

Software quality: The totality of functionality and features of a software product that bear on its ability to satisfy stated or implied needs.

Solution: (1) Solution is the implementation of the requirement. (2) A design idea which, if implemented, is expected to lead to the partial or full satisfaction of a set of attribute requirements; to solve a (defined) problem.

Specification: A document that specifies, ideally in a complete, precise and verifiable manner, the requirements, design, behaviour, or other characteristics of a component or system, and, often, the procedures for determining whether these provisions have been satisfied.

Stability: The capability of the software product to avoid unexpected effects from modifications in the software

Stakeholder: Any person who has an interest in an IT project. Project stakeholders are individuals and organizations that are actively involved in the project, or whose interests may be affected as a result of project execution or project completion.

Standard: Formal, possibly mandatory, set of requirements developed and used to prescribe consistent approaches to the way of working or to provide guidelines

System: A collection of components organized to accomplish a specific function or set of functions.

T

Testability: The capability of the software product to enable modified software to be tested.

Testable requirements: The degree to which a requirement is stated in terms that permit establishment of test designs (and subsequently test cases) and execution of tests to determine whether the requirements have been met.

Traceability: The ability to identify related items in documentation and software, such as requirements with associated tests.

U

Understandability: The capability of the software product to enable the user to understand whether the software is suitable, and how it can be used for particular tasks and conditions of use

Usability: The capability of the software to be understood, learned, used and attractive to the user when used under specified conditions

User: A person who uses a software product.

V

Value: Perceived benefit, it is the potential consequence of system attributes, for one or more stakeholders. Value is the perceived usefulness, worth, utility, or importance of a defined system component or system state, for defined stakeholders, under specified conditions. Value is relative to a stakeholder: it is not absolute.

Vendor: A person, group or organization providing the solution.

Verification: Confirmation by examination and through provision of objective evidence that specified requirements have been fulfilled

Version: A specific form or variation of something.

Vision: An image of the project's deliverable as the solution to the stated need or problem.

W

Walkthrough: A step-by-step presentation by the author of a document in order to gather information and to establish a common understanding of its content.

BASIC FEATURES:

SIGNUP: This feature enables new customers/admin to create their accounts.

LOGIN: This feature enables existing customers/admin to enter into their accounts.

RESTAURANT MENU: The entire restaurant food menu details have been displayed in the website along with dish type and their prices.

PLACE ORDER: The user is entitled to place his order according to his wishes along with the subsequent bill generation.

VIEW DETAILS (ORDER): This feature enables existing customers to view all the orders that had been submitted by them.

ORDER VIEW/APPROVAL: The admin is allowed to view the orders and approve them according to whether he has the time and necessary resources in order to prepare that order not.

DELIVERY OPTIONS: The customer has been given the option of choosing to whether he wants to dine at the restaurant itself, take the packed meal from the restaurant as a pickup or get the packed meal delivered to his home.

EDIT DETAILS : This feature enables existing customers to edit all the details that they had provided at the time of sign up.

SUMMARY OF DISPLAY AND REPORT FORMATS:

The format of the manual is in accordance with the conventional manual format used by major agencies. It can be described as follows:

Text written in this font style is indicative of the main headings that have been used in the manual.

Text written in this font style is indicative of the sub headings that have been used inside the main headings the manual.

Text written in this **font** style is indicative of the various points that have been listed inside the sub headings the manual.

A single page break has been provided after every sub heading and its description is complete.

A double page break has been provided after every main heading and its description is complete.

Text written in **bold** has been used for major emphasis as well as occasional headings.

Text which has been underlined has been used for mentioning key points.

OUTLINE OF THE MANUAL:

The outline of the manual serves the purpose of explaining in simple terminology the entire manual in brief. It is equivalent to just touching the points listed in the manual.

- Introduction
 1. Product rationale and overview: This section focuses on the basic purpose of the software product. A brief idea about the need of product, its basic functionalities and its advantages is mentioned. This is intended to give a general overview of the product aimed at acquainting the user with the software product.

2. Terminology: This section is dedicated to mentioning all the possible terms that might be used in this manual . The official definition of the terms has been mentioned which can be used as a reference hereafter.
 3. Basic features: This section describes the basic functionalities which the application provides for execution.
 4. Summary of display and report formats: The summary of display and report format is aimed at briefing the reader about the format in which the manual has been written. It includes the order that is followed during the entire
 5. Outline of the manual: This section brushes up with all the topics that have been covered while writing the manual . It is just a simple coverage of all the headings that have been covered in this manual.
- Getting started
 1. Sign on: Single sign-on (SSO) is a session and user authentication service that permits a user to use one set of login credentials (e.g., name and password) to access multiple applications. The service authenticates the end user for all the applications the user has been given rights to and eliminates further prompts when the user switches applications during the same session. On the back end, SSO is helpful for logging user activities as well as monitoring user accounts.
 2. Help mode: The help mode is needed when the user has doubts regarding the options that are available for implementation. This can be helpful when the user wants to dynamically learn from the help section.
 3. Sample run: This section contains screenshots of the software implementation. This helps the user to check his implementation view with the actual implementation
 - Modes of operation
 1. Commands/displays/options-This is aimed at showcasing the various options that are available to the user.
 - Advanced features-This states the special features available for certain peculiar tweaks.
 - Command syntax and system options-Syntax and settings options for the application.
-

GETTING STARTED:

SIGN ON:

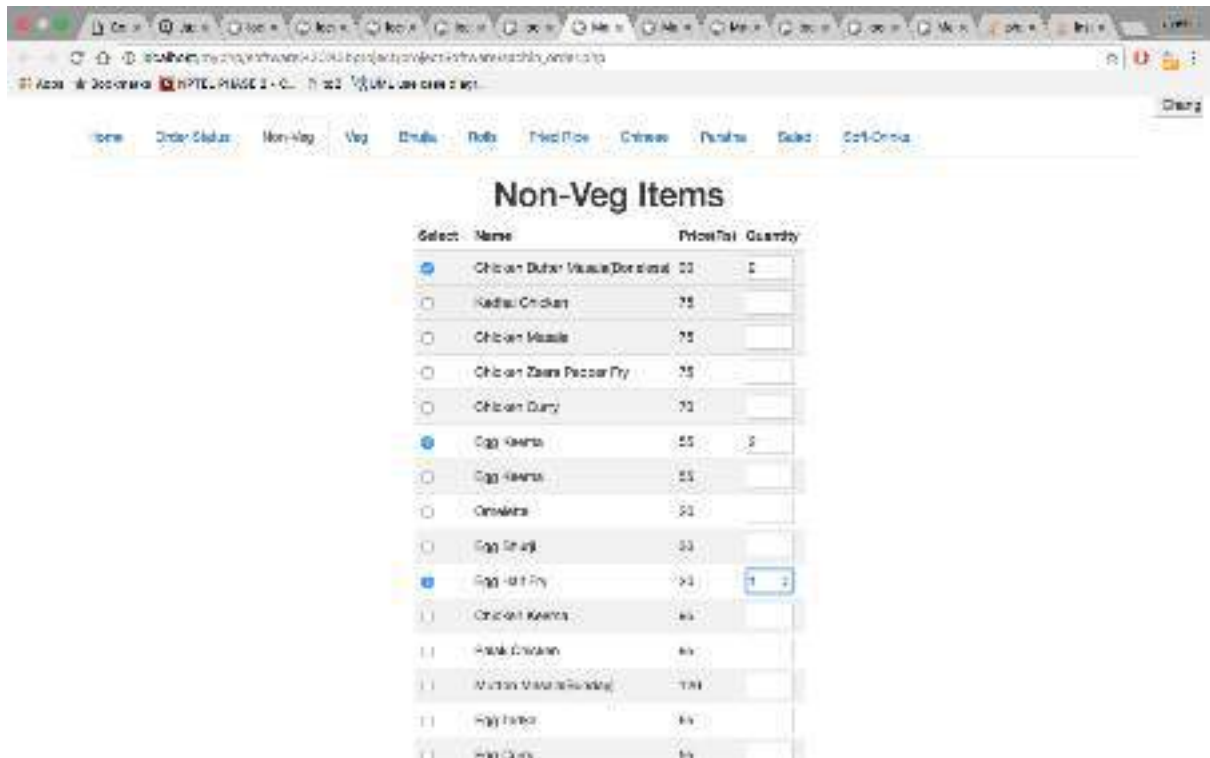
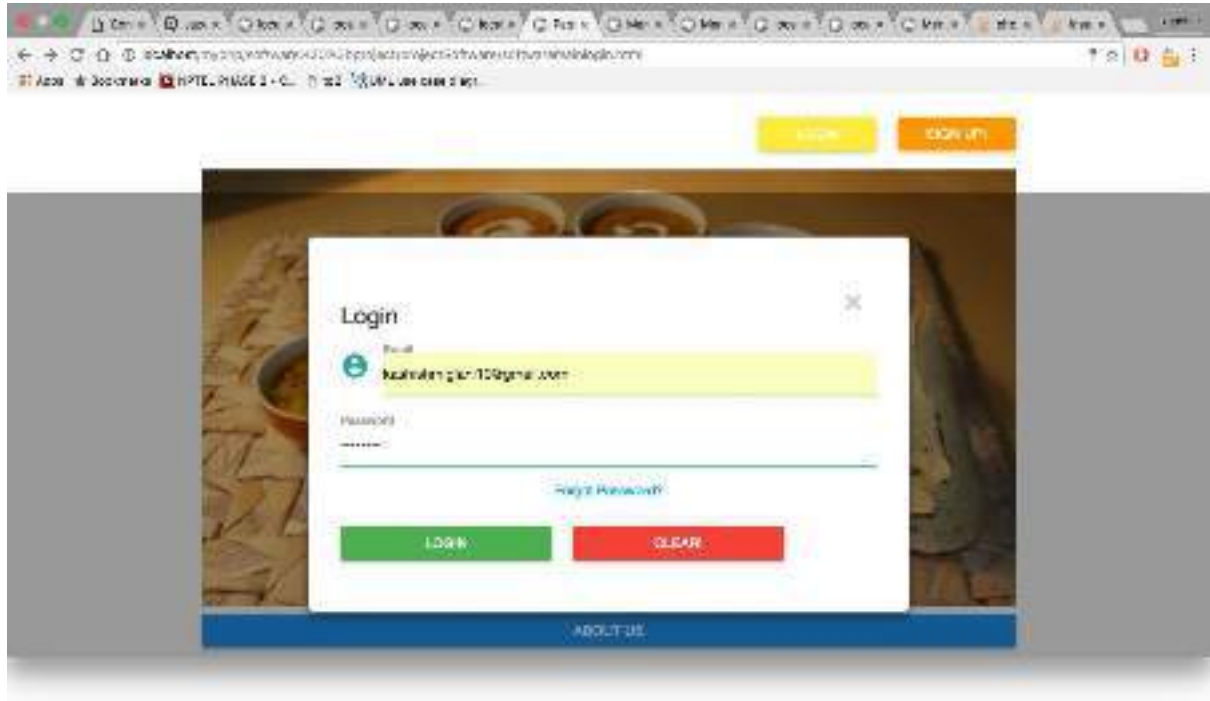
The account creation is a necessary activity in order to access the website. At the time of account creation, the user needs to provide his username, email id, and phone no, password, address and hint questions in order to create a customer account. In the same way, the admin needs to provide his email and password initially to create an account. Once the accounts have been created, the Single sign-On(SSO) property is exhibited ,that is the user can use the credentials provided by him in order to sign in on multiple machines (one at a time). As a result, a single email id and password are sufficient for a user to access his account.

HELP MODE:

The help mode is an important segment of this manual. This is mainly used for reference by the user in case of any contingency.

- **Forget password condition:** In case the user forgets his password, then he is given the option of entering the answer of two questions that he had mentioned at the time of signup. If the user gives the correct option then he/she is reminded of his password.
- **Reset password:** In case the user feels that the password chosen by him is inadequate, then he/she can change his password by choosing the option that has been provided.
- **Cancel order:** In case the user feels that the order provided by him can be edited or needs appending, then the user can cancel the present order and upload his new order.
- **Order not sent:** In case the order is not sent to the admin or the order processing stage encounters an error, then the user can resend the order without having to mention the order again and again.
- **Order approval not sent:** In case the order is not approved by the admin or the order approving stage encounters an error, then the admin can resend the order approval.
- **Order item confusion:** The names of the dishes have been made available in the most simple language as possible. If any confusion persists, then the customer can find the relevant information about the dish online simultaneously.
- **Bill calculation:** The script for calculating the price has been made with utmost accuracy. In case the final amount has some calculation error, then the order can be cancelled and mentioned again.

SAMPLE RUN :



localhost/myphp/software%20%5bproject/projectSoftware/order1.php

<input type="checkbox"/>	Chicken Masala	75	<input type="text"/>
<input type="checkbox"/>	Chicken Zarda Peas Fry	75	<input type="text"/>
<input type="checkbox"/>	Chicken Curry	75	<input type="text"/>
<input checked="" type="checkbox"/>	Egg Keema	55	<input type="text" value="2"/>
<input type="checkbox"/>	Egg Keema	55	<input type="text"/>
<input type="checkbox"/>	Chicken	25	<input type="text"/>
<input type="checkbox"/>	Egg Curry	55	<input type="text"/>
<input checked="" type="checkbox"/>	Egg Half Fry	25	<input type="text" value="1"/>
<input type="checkbox"/>	Chicken Keema	55	<input type="text"/>
<input type="checkbox"/>	Palak Chicken	55	<input type="text"/>
<input type="checkbox"/>	Mutton Masala(Boneless)	120	<input type="text"/>
<input type="checkbox"/>	Egg Tandoori	55	<input type="text"/>
<input type="checkbox"/>	Egg Curry	55	<input type="text"/>
<input type="checkbox"/>	Half Curry	75	<input type="text"/>
<input type="checkbox"/>	Chicken Fry	55	<input type="text"/>
<input type="checkbox"/>	Tandoori Chicken	55	<input type="text"/>

Proceed

Status

localhost/myphp/software%20%5bproject/projectSoftware/order1.php

The total order is
 Chicken Butter Masala(Boneless)-2
 Egg Keema-2
 Egg Half Fry-1
 and The total cost is 310

MODES OF OPERATION:

Commands/displays/options

The website adheres to conventional browser commands and shortcuts like refresh and return back to the previous page. The display works on varying display sizes and browsers normally. The website is optimized for an enhanced user experience without any difficulty in the process. The customer's order is saved for a short period of time, even after closing the browser to speed-up the checkout process. The customer can cancel the order in case of any changes in plans and subsequent changes are reflected in the restaurant manager's display.

The menu is designed in a simple, easy to use manner which is intuitive. It is categorized by type for easy navigation. After placing the order, the customer can view the status of the order which is managed by the restaurant's manager.

The restaurant manager can view all the orders placed in the restaurant in an easy to view, serialized fashion. The manager can accept or reject the orders based on availability and the customer can be informed of the same immediately. Finance management becomes easy through the total bill calculator.

ADVANCED FEATURES:

The website provides advanced features for enhanced user experience. These advanced options need not be necessarily employed by the user but act as optional features.

VIEW DETAILS (ORDER): This feature enables existing customers to view all the orders that had been submitted by them.

ORDER VIEW/APPROVAL: The admin is allowed to view the orders and approve them according to whether he has the time and necessary resources in order to prepare that order not.

EDIT DETAILS: This feature enables existing customers to edit all the details that they had provided at the time of sign up.

Command Syntax and System Options

Since the system is GUI based, there is no defined syntax that the user has to follow while giving instructions to the application. The basic structure of the application can be changed only by altering the very source code (php,html,javascript) of the application.

SOFTWARE ENGINEERING THEORY PROJECT TEST MANUAL



Fall Semester 2016-2017

TOPIC:

Restaurant Service System



Teacher:

Prof. Alok Chauhan

STUDENTS:

- Osho Agyeya(15BCE1326)
- Kashish Miglani(15BCE1003)
- Sanchay Gupta(15BCE1190)
- Sachin Gopal(15BCE1188)

Introduction:

The project is aimed at developing a web application that provides online order placement facility. The users have a variety of orders to choose from. They get the bill once the order has been placed. The administrator gets to decide which orders will be prepared or not.

The document is aimed at providing an efficient test plan to benchmark the credibility of the system.

Features to be tested:

The following features are going to be tested:

- login signup activity
- order placement activity
- bill calculation activity
- admin functionalities
- logout activity

Features not to be tested:

The following features shall not be tested:

- Register activity: This is a onetime activity which shall not be used once the user has registered successfully.
- Individual dishes: All the order options/various dishes on the menu are working perfectly as their source code has been reviewed several times
- About us: This is a single web page denoting the details of the restaurant. Since it is just conveying information, it does not need testing.

Approach:

The overall approach is as follows:

- Initially relevant test cases are provided.
- Then, the expected outcomes are assumed.
- Test execution
- Comparison of results of the test with assumed/expected results
- Recording the results for statistical evaluation

Item pass/fail criteria:

The software is supposed to have passed the test criteria if:

- All the inputs are accepted by the software
- The software performs all the steps of the algorithms defined.
- The software produces the desired outputs
- The outputs are provided in the desired time frame

All the other conditions are supposed to come under failure conditions.

Test deliverables:

After the test has been completed, a summary of the test report is provided which states the following conditions:

- Inputs are accepted by the software
- Outputs generated
- Time slice within which the output is provided
- The improvement that is expected for the next testing

Testing tasks:

The testing tasks are as follows:

Test Strategy

The purpose of testing is to find defects, not to pass easy tests. A test strategy basically tells you which types of testing seem best to do, the order in which to perform them, the proposed sequence of execution, and the optimum amount of effort to put into each test objective to make your testing most effective. A test strategy is based on the prioritized requirements and any other available information about what is important to the customers. Because you will always face time and resource constraints, a test strategy faces up to this reality and tells you how to make the best use of whatever resources you do have to locate most of the worst defects. Without a test strategy, you are apt to waste your time on less fruitful testing and miss using some of your most powerful testing options. You

should create the test strategy at about the middle of the design phase as soon as the requirements have settled down.

Testing Plan

A testing plan is simply that part of your project plan that deals with the testing tasks. It details who will do which tasks, starting when, ending when, taking how much effort, and depending on which other tasks. It provides a complete list of all the things that need to be done for testing, including all the preparation work during all of the phases before testing. It shows the dependencies among the tasks to clearly create a critical path without surprises. You will be able to start filling in the details of your testing plan as soon as your test strategy is completed. Both your test strategy and testing plan are subject to change as the project evolves. Modify your strategy first, if you need to, and then your testing plan.

Test Cases

Your test cases (and automated test scripts if called for by your strategy) are prepared based on the strategy which tells you how much of each type of testing to do. Test cases are developed based on prioritized requirements and acceptance criteria for the software, keeping in mind the customer's emphasis on quality dimensions and the project's latest risk assessment of what could go wrong. Except for a small amount of ad hoc testing, all of your test cases should be prepared in advance of the start of testing. There are many different approaches to developing test cases. Test case development is an activity performed in parallel with software development. It is just as difficult to do a good job of coming up with test cases as it is to program the system itself. In addition to figuring out what steps to take to test the system, you need to know the requirements and business rules well enough to predict exactly what the expected results should be. Without expected results to compare to actual results, you will not be able to say whether a test will pass or fail. A good test case checks to make sure requirements are being met and has a good chance of uncovering defects.

Test Data

In addition to the steps to perform to execute your test cases, you also need to systematically come up with test data to use. This often equals sets of names, addresses, product orders, or whatever other information the system uses. Since you are probably going to test query functions, change functions and delete functions, you will most likely need a starting database of data in addition to the examples to input. Consider how many times you might need to go back to the starting point of the database to restart the testing and how many new customer names you will need for all the testing in your plan. Test data development is usually done simultaneously with test case development.

Test Environment

You will need a place to do the testing and the right equipment to use. Unless the software is very simple, one PC will not suffice. You will need all of the components of the system as close as possible to what it will eventually be. Test environments may be scaled-down versions of the real thing, but all the parts need to be there for the system to actually run. Building a test environment usually involves setting aside separate regions on mainframe computers and/or servers, networks and PCs that can be dedicated to the test effort and that can be reset to restart testing as often as needed.

Sometimes lab rooms of equipment are set aside, especially for performance or usability testing. A wish list of components that will be needed is part of the test strategy, which then needs to be reality checked as part of the test planning process. Steps to set up the environment are part of the testing plan and need to be completed before testing begins.

Test cases

Machine Configuration:

Browser: Google Chrome Version 53.0.2785.143 m

Server: MAMP

Database System: SeQueL Pro

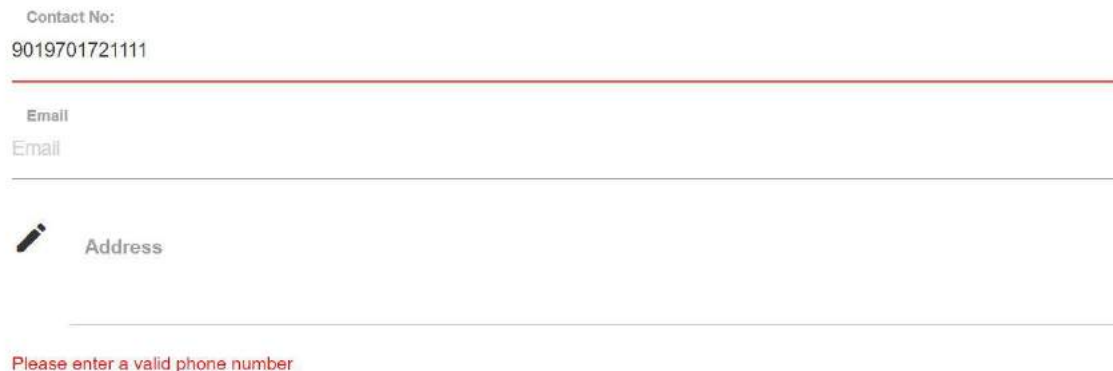
Requirements being tested:

The response of the following components of the system to varying stimuli:

- Sign up
 - Login
 - Order placement
 - Order status and administrator overview
-


Sign up Module:

Test case 1: Functional, Stress and Structural tests



Contact No:
9019701721111

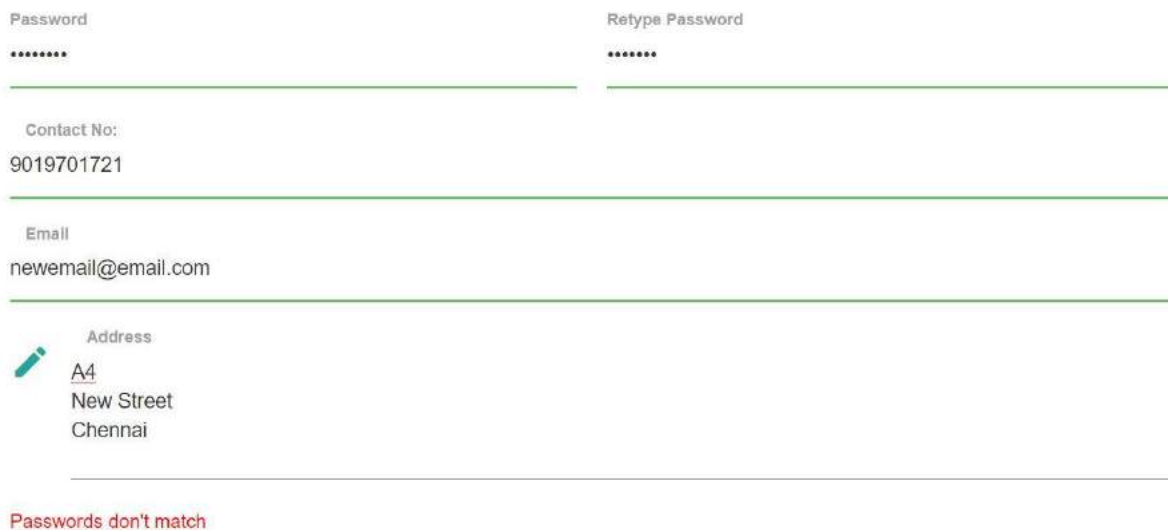
Email
Email

 Address

Please enter a valid phone number

On entering an invalid phone number (that is greater than 10 digits), we receive an error stating the same. The registration fails in the subsequent stage.

Test case 2: Functional, Stress and Structural tests




Password
.....

Retype Password
.....

Contact No:
9019701721

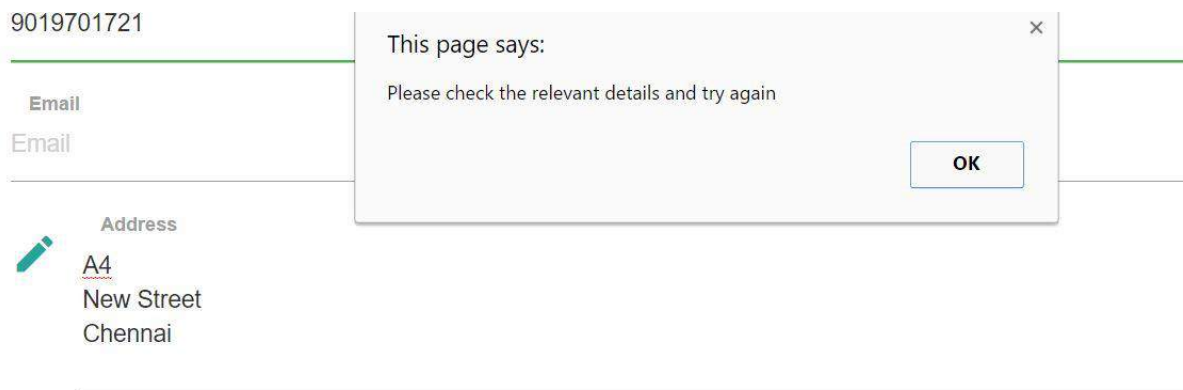
Email
newemail@email.com

 Address
A4
New Street
Chennai

Passwords don't match

On entering a different password in the retype password we get an error message stating the same. The registration fails in the subsequent stage.

Test case 3: Functional, Stress and Structural tests



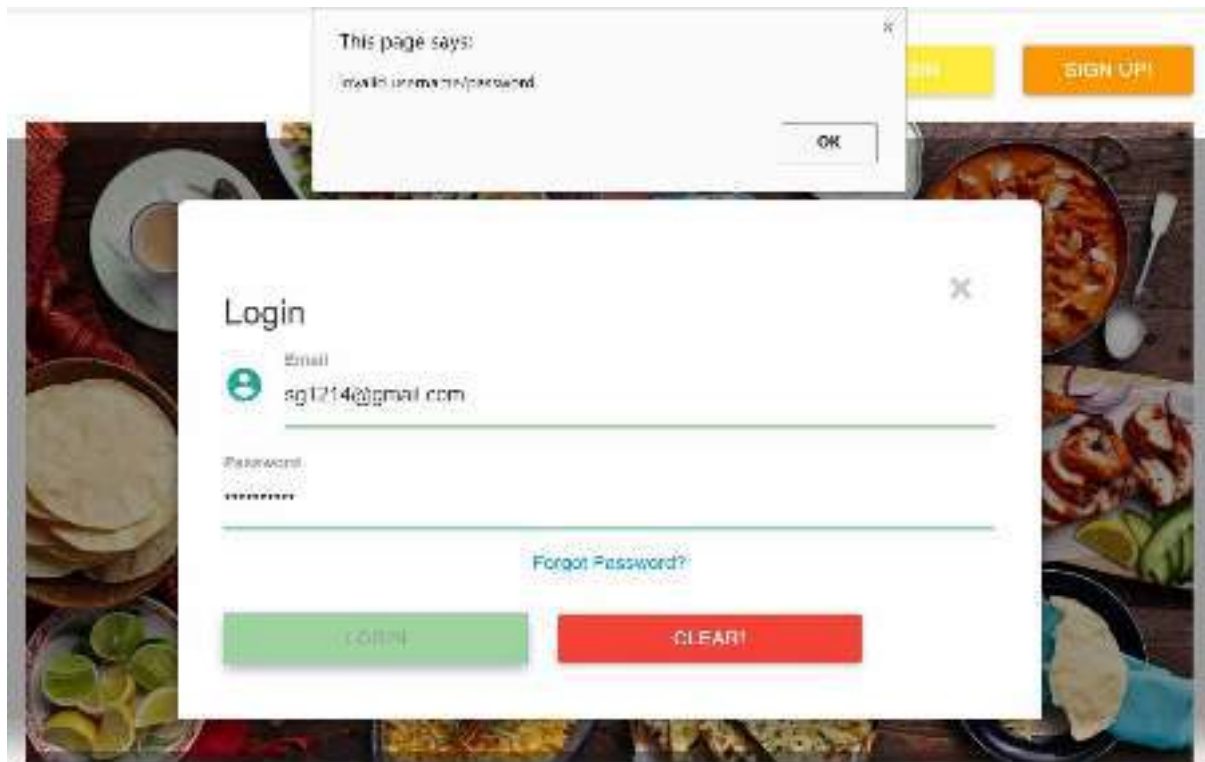
The screenshot shows a registration form with the following elements:

- A text input field at the top containing the number "9019701721".
- A label "Email" positioned above a second, empty text input field.
- A label "Address" positioned above a text area containing the text "A4", "New Street", and "Chennai".
- A green pencil icon to the left of the address text area.
- An alert dialog box is displayed in the center-right of the form. It has a title bar with a close button (X). The message inside reads: "This page says: Please check the relevant details and try again". There is an "OK" button at the bottom right of the dialog.

During the registration process, if a field is left empty, we get an alert to check the relevant details and try again. Database is not updated until valid inputs are provided.

Sign in Module:

Test case 4: Functional and Stress test



If a user provides an invalid username or password, that is tries to login before registration, an alert message notifies the user of this and the user can try signing in again or register on the website.

Order Placement Module:

Test case 5: Functional test

Input:

Non-Veg

Veg

Bhujia

Rolls

Fried Rice

Chinese

Paratha

Salad

Soft-Drinks

Non-Veg Items

Select	Name	Price(Rs)	Quantity
<input checked="" type="checkbox"/>	Chicken Butter Masala(Boneless)	90	3
<input type="checkbox"/>	Kadhai Chicken	75	
<input type="checkbox"/>	Chicken Masala	75	
<input checked="" type="checkbox"/>	Chicken Zeera Pepper Fry	75	4
<input type="checkbox"/>	Chicken Curry	70	
<input checked="" type="checkbox"/>	Egg Keema	55	3
<input type="checkbox"/>	Egg Keema	55	
<input type="checkbox"/>	Omelette	20	
<input type="checkbox"/>	Egg Bhurji	30	
<input type="checkbox"/>	Egg Half Fry	20	
<input type="checkbox"/>	Chicken Keema	95	

Output:

Item	quantity	subtotal
Chicken Butter Masala(Boneless)	3	270
Chicken Zeera Pepper Fry	4	300
Egg Keema	3	165
The Grand total :735		

When a user selects the check box and enters a positive quantity value, the order proceeds normally and the bill reflects this. Order calculation also proceeds as expected.

Test case 6: Functional and Stress test

Input:

Roll Items

Select	Name	Price(Rs)	Quantity
<input checked="" type="checkbox"/>	Egg Roll	35	<input type="text" value="0"/>
<input type="checkbox"/>	Chicken Roll	45	<input type="text" value="55"/>
<input type="checkbox"/>	Veg Roll	25	<input type="text"/>
<input checked="" type="checkbox"/>	Paneer Roll	45	<input type="text" value="20"/>

Proceed

Output:

Item	quantity	subtotal
Egg Roll	0	0
Paneer Roll	20	900

The Grand total :900

When a user checks a check box without entering the quantity or vice versa, the bill generated shows the quantity as 0 and handles the exception.

Test case 7: Functional and Stress test

Input:

Veg Items

Select	Name	Price(Rs)	Quantity
<input checked="" type="checkbox"/>	Paneer Butter Masala	80	<input type="text" value="13"/>
<input checked="" type="checkbox"/>	Paneer Masala	60	<input type="text" value="40"/>
<input checked="" type="checkbox"/>	Kadai Paneer	75	<input type="text" value="44"/>
<input checked="" type="checkbox"/>	Mutter Paneer	70	<input type="text" value="55"/>
<input checked="" type="checkbox"/>	Palak Paneer	70	<input type="text" value="22"/>
<input checked="" type="checkbox"/>	Paneer Bhurji	80	<input type="text" value="11"/>
<input checked="" type="checkbox"/>	Shahi Paneer	95	<input type="text" value="11"/>
<input checked="" type="checkbox"/>	Aloo Gobi Masala	55	<input type="text" value="50"/>
<input checked="" type="checkbox"/>	Gobi Masala	65	<input type="text" value="12"/>
<input checked="" type="checkbox"/>	Aloo Dum	55	<input type="text" value="11"/>
<input checked="" type="checkbox"/>	Aloo Zeera	50	<input type="text" value="60"/>
<input checked="" type="checkbox"/>	Bhindi Masala	50	<input type="text" value="40"/>
<input type="checkbox"/>	Bhindi Fry	50	<input type="text"/>

Output:

Item	quantity	subtotal
Paneer Butter Masala	13	1040
Paneer Masala	40	2400
Kadai Paneer	44	3300
Mutter Paneer	55	3850
Palak Paneer	22	1540
Paneer Bhurji	11	880
Shahi Paneer	11	1045
Aloo Gobi Masala	50	2750
Gobi Masala	12	780
Aloo Dum	11	605
Aloo Zeera	60	3000
Bhindi Masala	40	2000
The Grand total :23190		

When a user places a bulk order with extremely large quantities, the system computes the bill normally. The page disallows the user to add an item whose quantity is greater than 99. The admin ultimately has the choice to accept or reject the order.

Order status and administrator overview module

Test case 8: Functional test

Input

Chicken Fried Rice Schezwan 5 375
Green Peas Rice 7 420
Jeera Rice 7 350
YES

Chicken Butter Masala(Boneless) 2 180
Chicken Masala 2 150
Egg Bhurji 1 30
Gobi Manchurian 1 55
Paneer Chilly 5 350
Chicken Lollipop 6 540
Kheer 4 120
NO

Chicken Butter Masala(Boneless) 2 180
Chicken Masala 2 150
Egg Bhurji 1 30

Output

Chicken Fried Rice Schezwan 5 375

Green Peas Rice 7 420

Jeera Rice 7 350

ACCEPTED

Chicken Butter Masala(Boneless) 2 180

Chicken Masala 2 150

Egg Bhurji 1 30

Gobi Manchurian 1 55

Paneer Chilly 5 350

Chicken Lollipop 6 540

Kheer 4 120

REJECTED

Chicken Butter Masala(Boneless) 2 180

Chicken Masala 2 150

Egg Bhurji 1 30

PENDING

The administrator i.e. the restaurant manager has the option to Submit or Reject the order. This is reflected in the order status. This can further be accessed by the restaurant's employees to start preparing the order for faster deliveries.

SOFTWARE

ENGINEERING

THEORY PROJECT

SOURCE CODE

DOCUMENTATION



Fall Semester 2016-2017

TOPIC:

Restaurant Service System



Teacher:

Prof. Alok Chauhan

STUDENTS:

- Osho Agyeya(15BCE1326)
- Kashish Miglani(15BCE1003)
- Sanchay Gupta(15BCE1190)
- Sachin Gopal(15BCE1188)

SOURCE CODE DOCUMENTATION

All the source code files have been attached in this document. The purpose of the files has been added as a standard prologue. Internal commenting has been provided in order to make the code readable.

FILE 1:About Us.html

Prologue:

This file provides the information about the restaurant for which the website is being maintained. It has a short description about the restaurant providing relevant details like address, working hours and phone number.

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="bootstrap.min.css">
    <script src="jquerymin.js"></script>
    <script src="bootstrapmin.js"></script>
    <title>About Us</title>

</head>
<body>
    <!--
    <div class="container">
        <ul class="nav nav-tabs">
            <li><a href="060816_1.html">Home</a></li>
            <li><a href="060816_12.html">Sign Up</a></li>
        </ul>
    </div>
    -->
    <h1 align="center">About Us</h1>

    <div id="Image">
        
    </div>
    <!-- - This the information about KUNTI restraunt- - >
    <div style="padding:10px;margin:10px" id="About">
        <font size="3em" style="text-align:right-side;">
            Kunti Restaurants is a prime restaurant located near
VIT Chennai Campus and remains one of the most popular go-to spots for
students in the nearby colleges. Famous for its quality food at an
affordable rate, it is especially popular among the North Indian food
lovers in the college. It features a variety of North Indian and
Chinese delicacies.<br/>
        </font>
    </div>

    <div id="Footer">
        <font size="3em">
            <b>Phone number: </b>
```

```

        <font size="2em">
            80154 04303<br/><br/>
        </font>
        <b>Address: </b><br>
        <font size="2em">
            No 973, Annai Complex,<br/> Varadhapure
Amman,<br/> Kovil Street,<br/> Melakottaiyur,<br/> Madurapakkam,<br/>
Tamil Nadu 600048 <br/><br/>
        </font>
        <b>Working hours:</b>
        <font size="2em">
            8 a.m. to 9 p.m., 7 days a week<br/><br/>
        </font>
        <b><a align="center"
href="https://goo.gl/maps/pigVexSrAjB2">Location</a></b>
        </font>
    </div>

</body>
</html>

```

FILE 2:connection1.php

Prologue:

This php file is used to create a connection between the database and the php file so that queries may be used to access the database.

```

<?php
session_start();
$user = 'root';
$password = 'root';
$db = 'project';
$host = 'localhost';
$port = 3306;
#this is used for the connection between the mysql and php
$link = mysql_connect(
    "$host:$port",
    $user,
    $password
) or die(mysql_error());
#this is used to select the database out of the various databases
$db_selected = mysql_select_db(
    $db,
    $link
) or die(mysql_error());

?>

```

FILE 3:forget_password.php

Prologue:

This php file is used in case the user forgets the password which he was using previously. The user is asked 2 questions for which he has to provide the answers which he had given at the time of account creation. If the answers are correct, then the user is provided his password, else an error is displayed.

```

<?php
include 'connection1.php';
#First it will ask the user of the email id , if the email id is
already registered then it will ask two security questions
if(isset($_POST['user']))
{
$user2=$_POST['user'];
$_SESSION['name']=$user2;
$query="select * from customers where email='".$user2."'";
$r=mysql_query($query) or die(mysql_error());
if(mysql_num_rows($r)==1)
{
    echo "<form action='' method='post'> </br><b>Question1</b>:What
was the nickname of your first friend ?<br> <textarea rows='5'
cols='20' name='ques1' required></textarea><br><br><b>Question2</b>:What
was the first song which you heard ?<br><textarea rows='5' cols='20'
name='ques2' required></textarea><br><br><input type='submit'
name='sub2' value='submit' /></form>";
}

else
{
    echo 'username doesnt match';
}

}
#If the answer to both the questions matches , then it will show the
user its password.
if(isset($_POST['sub2']))
{
    $ans1=$_POST['ques1'];
    $ans2=$_POST['ques2'];
    $user2=$_SESSION['name'];
    $query2="select * from customers where question1='".$ans1.'" and
question2='".$ans2.'" and email='".$user2."'";
    $v=mysql_query($query2) or die(mysql_error());
    if(mysql_num_rows($v)==1)
    {
        $query3="select passwordc from customers where
question1='".$ans1.'" and question2='".$ans2.'" and
email='".$user2."'";
        $tf=mysql_query($query3) or die(mysql_error());
        $t=mysql_fetch_array($tf) or die(mysql_error());
        echo "Your password was : <h1>".$t['passwordc']. "</h1>";
    }
    else
    {
        echo "Answers of the questions doesnt match";
    }
}

?>

```

FILE 4:login_project.php

Prologue:

This php file is executed when the user is trying to log in into his account. If the email and the password that are entered are the same as the data which is present in the database, then the user is successfully logged into his account. Otherwise, an error is raised that the "username/password doesn't match. TRY AGAIN" .

```
<?php
include 'connection1.php';

#since we are assuming only one owner of the kunti restraunt so rather
than having a separate database for the owner we are just assigning him
an username and a default password , If the username and the password
matches then it will redirect to the next page
if(isset($_POST['submit']))
{
    $email=$_POST['email'];
    $pass=$_POST['password'];

if($email=="user@gmail.com" && $pass=="password")
    {
        ?>
        <script>
        window.open('user.php');
        </script>
        <?php
    }

    else{
#this is for checking the customer sign in

        $query1="select * from customers where email='".$email.'"and
passwordc='".$pass.'"";
        $r=mysql_query($query1) or die(mysql_error());
        if(mysql_num_rows($r)==1)
        {
            $_SESSION['pass']=$pass;
            $_SESSION['user']=$email;
            ?>
            <script>
            window.open('sachin_order.php');
            </script>

            <?php
        }

        else
        {
            ?>
            <script>window.alert('username/password doesnt match. TRY
AGAIN');
            window.open('softwaremainlogin.html');</script>

            <?php
        }
    }
    ?>
```

FILE 5:logout.php

Prologue:

This php file is executed when the user wants to log out of his account. As soon as the user presses the logout button which has been provided on the html page, then the user is logged out of his account and the control returns to the main page.

```
<?php
if(isset($_POST['logout']))
{
    session_destroy();#it will destroy the current session and will
    redirect the user to the homepage.
    ?>
    <script>
        window.open('softwaremainlogin.html');
    </script>
    <?php
    exit();
}
?>
```

FILE 6:order_status.php

Prologue:

This php file is used when the administrator wants to accept or reject the orders which have been placed by the customers. Also, the list of orders is updated according to whether the orders have been accepted or rejected.

```
<?php
include 'connection1.php';

if(isset($_POST['status']))
{
    #it is for the user to view the status of the orders already
    placed
    $email=$_SESSION['user'];
    $query="select * from orders order by temp desc";
    $r=mysql_query($query) or die('NN');
    while($row=mysql_fetch_array($r))
    {
        if($row['id']==$email)
        {
            #if the flag will be negative then it means that the admin has rejected
            the order.
            if($row['flag']==-1)
            {
                echo $row['allorder']." REJECTED";
            }
            #if the flag is positive then it means that the admin has accepted the
            order.
            else if($row['flag']==1)
            {
                echo $row['allorder']." ACCEPTED";
            }
        }
    }
}
```

```

# if the flag is zero then it means that order is still pending.
    else if($row['flag']==0)
    {
        echo $row['allorder']." PENDING";
    }
    echo "<br><br>";
}
}
}
?>

```

FILE 7:order2.php

Prologue:

This php file is used to calculate the bill according to the choices that have been entered by the user. The price and quantity of the selected dishes are used for this purpose.

```

<?php
include 'connection1.php';
echo "<head>
    <title>Forgot Password</title>
    <meta name='viewport' content='width=device-width, initial-
scale=1'>
    <link rel='stylesheet' href='external/css1.css'>
    <link rel='stylesheet' href='external/css2.css'>
    <link rel='stylesheet' href='external/css3.css'>
    <script src='external/jq1.js'></script>
    <script src='external/jq2.js'></script>
    <script src='external/jq3.js'></script>
    <script src='external/jq4.js'></script>
    <script>
        $(document).ready(function() {
            $('select').material_select();
        });</script>
</head>
$order="<table>";
<div class='container' padding:10px>
<div class='row'>
<div class='col 12'> <p></p></div>
<div class='col 18'><table class='striped'>
<tr><th>Item</th><th>quantity</th><th>subtotal</th></tr>";
$cost=0;
# if the submit is pressed then it will check for the selected
orders and will sum up their cost in a variable.
if(isset($_POST['submit']))
{

    if(isset($_POST['101']))
    {
        if(isset($_POST['101q']))
        {

            $quantity=$_POST['101q'];
            $cost=$cost+(90*$quantity);
            $current=$_POST['101'];

```

```

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(90*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['102']))
{
    if(isset($_POST['102q']))
    {

        $quantity=$_POST['102q'];
        $cost=$cost+(75*$quantity);
        $current=$_POST['102'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(75*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['103']))
{
    if(isset($_POST['103q']))
    {

        $quantity=$_POST['103q'];
        $cost=$cost+(75*$quantity);
        $current=$_POST['103'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(75*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['104']))
{
    if(isset($_POST['104q']))
    {

        $quantity=$_POST['104q'];
        $cost=$cost+(75*$quantity);
        $current=$_POST['104'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(75*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['105']))
{
    if(isset($_POST['105q']))
    {

        $quantity=$_POST['105q'];

```

```

        $cost=$cost+(70*$quantity);
        $current=$_POST['105'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['106']))
{
    if(isset($_POST['106q']))
    {

        $quantity=$_POST['106q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['106'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['107']))
{
    if(isset($_POST['107q']))
    {

        $quantity=$_POST['107q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['107'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['108']))
{
    if(isset($_POST['108q']))
    {

        $quantity=$_POST['108q'];
        $cost=$cost+(20*$quantity);
        $current=$_POST['108'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(20*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['109']))
{
    if(isset($_POST['109q']))
    {

```



```

        $quantity=$_POST['109q'];
        $cost=$cost+(30*$quantity);
        $current=$_POST['109'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(30*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['110']))
{
    if(isset($_POST['110q']))
    {

        $quantity=$_POST['110q'];
        $cost=$cost+(20*$quantity);
        $current=$_POST['110'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(20*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['111']))
{
    if(isset($_POST['111q']))
    {

        $quantity=$_POST['111q'];
        $cost=$cost+(95*$quantity);
        $current=$_POST['111'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(95*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['112']))
{
    if(isset($_POST['112q']))
    {

        $quantity=$_POST['112q'];
        $cost=$cost+(95*$quantity);
        $current=$_POST['112'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(95*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['113']))
{

```

```

        if(isset($_POST['113q']))
        {

            $quantity=$_POST['113q'];
            $cost=$cost+(120*$quantity);
            $current=$_POST['113'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(120*
$quantity)."</td></tr>";

        }

    }
    if(isset($_POST['114']))
    {
        if(isset($_POST['114q']))
        {

            $quantity=$_POST['114q'];
            $cost=$cost+(65*$quantity);
            $current=$_POST['114'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(65*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['115']))
    {
        if(isset($_POST['115q']))
        {

            $quantity=$_POST['115q'];
            $cost=$cost+(55*$quantity);
            $current=$_POST['115'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['116']))
    {
        if(isset($_POST['116q']))
        {

            $quantity=$_POST['116q'];
            $cost=$cost+(70*$quantity);
            $current=$_POST['116'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

        }

    }

```

```

if(isset($_POST['117']))
{
    if(isset($_POST['117q']))
    {

        $quantity=$_POST['117q'];
        $cost=$cost+(80*$quantity);
        $current=$_POST['117'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['118']))
{
    if(isset($_POST['118q']))
    {

        $quantity=$_POST['118q'];
        $cost=$cost+(95*$quantity);
        $current=$_POST['118'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(95*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['201']))
{
    if(isset($_POST['201q']))
    {

        $quantity=$_POST['201q'];
        $cost=$cost+(80*$quantity);
        $current=$_POST['201'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['202']))
{
    if(isset($_POST['202q']))
    {

        $quantity=$_POST['202q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['202'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}

```

```

}
if(isset($_POST['203']))
{
    if(isset($_POST['203q']))
    {

        $quantity=$_POST['203q'];
        $cost=$cost+(75*$quantity);
        $current=$_POST['203'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(75*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['204']))
{
    if(isset($_POST['204q']))
    {

        $quantity=$_POST['204q'];
        $cost=$cost+(70*$quantity);
        $current=$_POST['204'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['205']))
{
    if(isset($_POST['205q']))
    {

        $quantity=$_POST['205q'];
        $cost=$cost+(70*$quantity);
        $current=$_POST['205'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['206']))
{
    if(isset($_POST['206q']))
    {

        $quantity=$_POST['206q'];
        $cost=$cost+(80*$quantity);
        $current=$_POST['206'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

```

```

    }

}

if(isset($_POST['207']))
{
    if(isset($_POST['207q']))
    {

        $quantity=$_POST['207q'];
        $cost=$cost+(95*$quantity);
        $current=$_POST['207'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(95*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['208']))
{
    if(isset($_POST['208q']))
    {

        $quantity=$_POST['208q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['208'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['209']))
{
    if(isset($_POST['209q']))
    {

        $quantity=$_POST['209q'];
        $cost=$cost+(65*$quantity);
        $current=$_POST['209'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(65*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['210']))
{
    if(isset($_POST['210q']))
    {

        $quantity=$_POST['210q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['210'];
    }
}

```

```

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['211']))
{
    if(isset($_POST['211q']))
    {

        $quantity=$_POST['211q'];
        $cost=$cost+(50*$quantity);
        $current=$_POST['211'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['212']))
{
    if(isset($_POST['212q']))
    {

        $quantity=$_POST['212q'];
        $cost=$cost+(50*$quantity);
        $current=$_POST['212'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['213']))
{
    if(isset($_POST['213q']))
    {

        $quantity=$_POST['213q'];
        $cost=$cost+(50*$quantity);
        $current=$_POST['213'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['214']))
{
    if(isset($_POST['214q']))
    {

        $quantity=$_POST['214q'];

```

```

        $cost=$cost+(45*$quantity);
        $current=$_POST['214'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(45*$
quantity)."</td></tr>";
    }

}
if(isset($_POST['215']))
{
    if(isset($_POST['215q']))
    {

        $quantity=$_POST['215q'];
        $cost=$cost+(45*$quantity);
        $current=$_POST['215'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(45*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['216']))
{
    if(isset($_POST['216q']))
    {

        $quantity=$_POST['216q'];
        $cost=$cost+(80*$quantity);
        $current=$_POST['216'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['217']))
{
    if(isset($_POST['217q']))
    {

        $quantity=$_POST['217q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['217'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['218']))
{
    if(isset($_POST['218q']))
    {

```

```

        $quantity=$_POST['218q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['218'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['219']))
{
    if(isset($_POST['219q']))
    {

        $quantity=$_POST['219q'];
        $cost=$cost+(50*$quantity);
        $current=$_POST['219'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['220']))
{
    if(isset($_POST['220q']))
    {

        $quantity=$_POST['220q'];
        $cost=$cost+(65*$quantity);
        $current=$_POST['220'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(65*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['221']))
{
    if(isset($_POST['221q']))
    {

        $quantity=$_POST['221q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['221'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['222']))
{
    if(isset($_POST['222q']))

```



```

        {

            $quantity=$_POST['222q'];
            $cost=$cost+(55*$quantity);
            $current=$_POST['222'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['223']))
    {
        if(isset($_POST['223q']))
        {

            $quantity=$_POST['223q'];
            $cost=$cost+(60*$quantity);
            $current=$_POST['223'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['224']))
    {
        if(isset($_POST['224q']))
        {

            $quantity=$_POST['224q'];
            $cost=$cost+(60*$quantity);
            $current=$_POST['224'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['225']))
    {
        if(isset($_POST['225q']))
        {

            $quantity=$_POST['225q'];
            $cost=$cost+(60*$quantity);
            $current=$_POST['225'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['226']))

```

```

{
    if(isset($_POST['226q']))
    {

        $quantity=$_POST['226q'];
        $cost=$cost+(70*$quantity);
        $current=$_POST['226'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['227']))
{
    if(isset($_POST['227q']))
    {

        $quantity=$_POST['227q'];
        $cost=$cost+(70*$quantity);
        $current=$_POST['227'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['228']))
{
    if(isset($_POST['228q']))
    {

        $quantity=$_POST['228q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['228'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['229']))
{
    if(isset($_POST['229q']))
    {

        $quantity=$_POST['229q'];
        $cost=$cost+(80*$quantity);
        $current=$_POST['229'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

    }

```

```

    }
    if (isset($_POST['301']))
    {
        if (isset($_POST['301q']))
        {

            $quantity=$_POST['301q'];
            $cost=$cost+(50*$quantity);
            $current=$_POST['301'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

        }

    }
    if (isset($_POST['302']))
    {
        if (isset($_POST['302q']))
        {

            $quantity=$_POST['302q'];
            $cost=$cost+(50*$quantity);
            $current=$_POST['302'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

        }

    }
    if (isset($_POST['303']))
    {
        if (isset($_POST['303q']))
        {

            $quantity=$_POST['303q'];
            $cost=$cost+(50*$quantity);
            $current=$_POST['303'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

        }

    }
    if (isset($_POST['304']))
    {
        if (isset($_POST['304q']))
        {

            $quantity=$_POST['304q'];
            $cost=$cost+(50*$quantity);
            $current=$_POST['304'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

```

```

    }

}

if(isset($_POST['305']))
{
    if(isset($_POST['305q']))
    {

        $quantity=$_POST['305q'];
        $cost=$cost+(50*$quantity);
        $current=$_POST['305'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['401']))
{
    if(isset($_POST['401q']))
    {

        $quantity=$_POST['401q'];
        $cost=$cost+(35*$quantity);
        $current=$_POST['401'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(35*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['402']))
{
    if(isset($_POST['402q']))
    {

        $quantity=$_POST['402q'];
        $cost=$cost+(45*$quantity);
        $current=$_POST['402'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(45*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['403']))
{
    if(isset($_POST['403q']))
    {

        $quantity=$_POST['403q'];
        $cost=$cost+(25*$quantity);
        $current=$_POST['403'];
    }
}

```

```

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(25*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['404']))
{
    if(isset($_POST['404q']))
    {

        $quantity=$_POST['404q'];
        $cost=$cost+(45*$quantity);
        $current=$_POST['404'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(45*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['501']))
{
    if(isset($_POST['501q']))
    {

        $quantity=$_POST['501q'];
        $cost=$cost+(75*$quantity);
        $current=$_POST['501'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(75*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['502']))
{
    if(isset($_POST['502q']))
    {

        $quantity=$_POST['502q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['502'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['503']))
{
    if(isset($_POST['503q']))
    {

        $quantity=$_POST['503q'];

```

```

        $cost=$cost+(50*$quantity);
        $current=$_POST['503'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['504']))
{
    if(isset($_POST['504q']))
    {

        $quantity=$_POST['504q'];
        $cost=$cost+(70*$quantity);
        $current=$_POST['504'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['505']))
{
    if(isset($_POST['505q']))
    {

        $quantity=$_POST['505q'];
        $cost=$cost+(70*$quantity);
        $current=$_POST['505'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['506']))
{
    if(isset($_POST['506q']))
    {

        $quantity=$_POST['506q'];
        $cost=$cost+(50*$quantity);
        $current=$_POST['506'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['507']))
{
    if(isset($_POST['507q']))
    {

```

```

        $quantity=$_POST['507q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['507'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['508']))
{
    if(isset($_POST['508q']))
    {

        $quantity=$_POST['508q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['508'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['509']))
{
    if(isset($_POST['509q']))
    {

        $quantity=$_POST['509q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['509'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['510']))
{
    if(isset($_POST['510q']))
    {

        $quantity=$_POST['510q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['510'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['511']))
{

```

```

        if(isset($_POST['511q']))
        {

            $quantity=$_POST['511q'];
            $cost=$cost+(65*$quantity);
            $current=$_POST['511'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(65*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['512']))
    {
        if(isset($_POST['512q']))
        {

            $quantity=$_POST['512q'];
            $cost=$cost+(70*$quantity);
            $current=$_POST['512'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['513']))
    {
        if(isset($_POST['513q']))
        {

            $quantity=$_POST['513q'];
            $cost=$cost+(75*$quantity);
            $current=$_POST['513'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(75*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['514']))
    {
        if(isset($_POST['514q']))
        {

            $quantity=$_POST['514q'];
            $cost=$cost+(50*$quantity);
            $current=$_POST['514'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

        }

    }

```



```

if(isset($_POST['515']))
{
    if(isset($_POST['515q']))
    {

        $quantity=$_POST['515q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['515'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['516']))
{
    if(isset($_POST['516q']))
    {

        $quantity=$_POST['516q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['516'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['515']))
{
    if(isset($_POST['515q']))
    {

        $quantity=$_POST['515q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['515'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['516']))
{
    if(isset($_POST['516q']))
    {

        $quantity=$_POST['516q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['516'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}

```

```

    }
    if (isset($_POST['601']))
    {
        if (isset($_POST['601q']))
        {
            $quantity=$_POST['601q'];
            $cost=$cost+(55*$quantity);
            $current=$_POST['601'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

        }

    }
    if (isset($_POST['602']))
    {
        if (isset($_POST['602q']))
        {
            $quantity=$_POST['602q'];
            $cost=$cost+(60*$quantity);
            $current=$_POST['602'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

        }

    }
    if (isset($_POST['603']))
    {
        if (isset($_POST['603q']))
        {
            $quantity=$_POST['603q'];
            $cost=$cost+(80*$quantity);
            $current=$_POST['603'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

        }

    }
    if (isset($_POST['604']))
    {
        if (isset($_POST['604q']))
        {
            $quantity=$_POST['604q'];
            $cost=$cost+(70*$quantity);
            $current=$_POST['604'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

```

```

    }

    }
    if(isset($_POST['605']))
    {
        if(isset($_POST['605q']))
        {

            $quantity=$_POST['605q'];
            $cost=$cost+(70*$quantity);
            $current=$_POST['605'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['606']))
    {
        if(isset($_POST['606q']))
        {

            $quantity=$_POST['606q'];
            $cost=$cost+(80*$quantity);
            $current=$_POST['606'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['607']))
    {
        if(isset($_POST['607q']))
        {

            $quantity=$_POST['607q'];
            $cost=$cost+(80*$quantity);
            $current=$_POST['607'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

        }

    }
    if(isset($_POST['608']))
    {
        if(isset($_POST['608q']))
        {

            $quantity=$_POST['608q'];
            $cost=$cost+(80*$quantity);
            $current=$_POST['608'];

```

```

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(80*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['609']))
{
    if(isset($_POST['609q']))
    {

        $quantity=$_POST['609q'];
        $cost=$cost+(90*$quantity);
        $current=$_POST['609'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(90*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['610']))
{
    if(isset($_POST['610q']))
    {

        $quantity=$_POST['610q'];
        $cost=$cost+(50*$quantity);
        $current=$_POST['610'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(50*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['611']))
{
    if(isset($_POST['611q']))
    {

        $quantity=$_POST['611q'];
        $cost=$cost+(55*$quantity);
        $current=$_POST['611'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['612']))
{
    if(isset($_POST['612q']))
    {

        $quantity=$_POST['612q'];

```

```

        $cost=$cost+(60*$quantity);
        $current=$_POST['612'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['613']))
{
    if(isset($_POST['613q']))
    {

        $quantity=$_POST['613q'];
        $cost=$cost+(65*$quantity);
        $current=$_POST['613'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(65*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['614']))
{
    if(isset($_POST['614q']))
    {

        $quantity=$_POST['614q'];
        $cost=$cost+(70*$quantity);
        $current=$_POST['614'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['615']))
{
    if(isset($_POST['615q']))
    {

        $quantity=$_POST['615q'];
        $cost=$cost+(75*$quantity);
        $current=$_POST['615'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(75*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['616']))
{
    if(isset($_POST['616q']))
    {

```

```

        $quantity=$_POST['616q'];
        $cost=$cost+(60*$quantity);
        $current=$_POST['616'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(60*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['617']))
{
    if(isset($_POST['617q']))
    {

        $quantity=$_POST['617q'];
        $cost=$cost+(65*$quantity);
        $current=$_POST['617'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(65*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['618']))
{
    if(isset($_POST['618q']))
    {

        $quantity=$_POST['618q'];
        $cost=$cost+(70*$quantity);
        $current=$_POST['618'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(70*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['619']))
{
    if(isset($_POST['619q']))
    {

        $quantity=$_POST['619q'];
        $cost=$cost+(75*$quantity);
        $current=$_POST['619'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(75*$
quantity)."</td></tr>";

    }

}

if(isset($_POST['620']))
{

```

```

        if(isset($_POST['620q']))
        {

            $quantity=$_POST['620q'];
            $cost=$cost+(90*$quantity);
            $current=$_POST['620'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(90*$
quantity)."</td></tr>";

        }

    }

    if(isset($_POST['621']))
    {
        if(isset($_POST['621q']))
        {

            $quantity=$_POST['621q'];
            $cost=$cost+(65*$quantity);
            $current=$_POST['621'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(65*$
quantity)."</td></tr>";

        }

    }

    if(isset($_POST['622']))
    {
        if(isset($_POST['622q']))
        {

            $quantity=$_POST['622q'];
            $cost=$cost+(55*$quantity);
            $current=$_POST['622'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(55*$
quantity)."</td></tr>";

        }

    }

    if(isset($_POST['701']))
    {
        if(isset($_POST['701q']))
        {

            $quantity=$_POST['701q'];
            $cost=$cost+(20*$quantity);
            $current=$_POST['701'];

            $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(20*$
quantity)."</td></tr>";

        }

    }

```

```

if(isset($_POST['702']))
{
    if(isset($_POST['702q']))
    {

        $quantity=$_POST['702q'];
        $cost=$cost+(20*$quantity);
        $current=$_POST['702'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(20*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['703']))
{
    if(isset($_POST['703q']))
    {

        $quantity=$_POST['703q'];
        $cost=$cost+(20*$quantity);
        $current=$_POST['703'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(20*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['704']))
{
    if(isset($_POST['704q']))
    {

        $quantity=$_POST['704q'];
        $cost=$cost+(20*$quantity);
        $current=$_POST['704'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(20*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['705']))
{
    if(isset($_POST['705q']))
    {

        $quantity=$_POST['705q'];
        $cost=$cost+(35*$quantity);
        $current=$_POST['705'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(35*$
quantity)."</td></tr>";

    }

}

```



```

}
if(isset($_POST['706']))
{
    if(isset($_POST['706q']))
    {

        $quantity=$_POST['706q'];
        $cost=$cost+(20*$quantity);
        $current=$_POST['706'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(20*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['707']))
{
    if(isset($_POST['707q']))
    {

        $quantity=$_POST['707q'];
        $cost=$cost+(35*$quantity);
        $current=$_POST['707'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(35*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['708']))
{
    if(isset($_POST['708q']))
    {

        $quantity=$_POST['708q'];
        $cost=$cost+(40*$quantity);
        $current=$_POST['708'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(40*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['709']))
{
    if(isset($_POST['709q']))
    {

        $quantity=$_POST['709q'];
        $cost=$cost+(8*$quantity);
        $current=$_POST['709'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(8*$q
uantity)."</td></tr>";

```

```

    }

}
if(isset($_POST['710']))
{
    if(isset($_POST['710q']))
    {

        $quantity=$_POST['710q'];
        $cost=$cost+(6*$quantity);
        $current=$_POST['710'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(6*$q
uantity)."</td></tr>";

    }

}
if(isset($_POST['711']))
{
    if(isset($_POST['711q']))
    {

        $quantity=$_POST['711q'];
        $cost=$cost+(10*$quantity);
        $current=$_POST['711'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(10*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['712']))
{
    if(isset($_POST['712q']))
    {

        $quantity=$_POST['712q'];
        $cost=$cost+(8*$quantity);
        $current=$_POST['712'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(8*$q
uantity)."</td></tr>";

    }

}
if(isset($_POST['713']))
{
    if(isset($_POST['713q']))
    {

        $quantity=$_POST['713q'];
        $cost=$cost+(8*$quantity);
        $current=$_POST['713'];
    }
}

```

```

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(8*$q
quantity)."</td></tr>";

    }

}
if(isset($_POST['714']))
{
    if(isset($_POST['714q']))
    {

        $quantity=$_POST['714q'];
        $cost=$cost+(30*$quantity);
        $current=$_POST['714'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(30*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['715']))
{
    if(isset($_POST['715q']))
    {

        $quantity=$_POST['715q'];
        $cost=$cost+(30*$quantity);
        $current=$_POST['715'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(30*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['802']))
{
    if(isset($_POST['802q']))
    {

        $quantity=$_POST['802q'];
        $cost=$cost+(35*$quantity);
        $current=$_POST['802'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(35*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['801']))
{
    if(isset($_POST['801q']))
    {

        $quantity=$_POST['801q'];

```

```

        $cost=$cost+(20*$quantity);
        $current=$_POST['801'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(20*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['901']))
{
    if(isset($_POST['901q']))
    {

        $quantity=$_POST['901q'];
        $cost=$cost+(30*$quantity);
        $current=$_POST['901'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(30*$
quantity)."</td></tr>";

    }

}
if(isset($_POST['902']))
{
    if(isset($_POST['902q']))
    {

        $quantity=$_POST['902q'];
        $cost=$cost+(25*$quantity);
        $current=$_POST['902'];

        $order=$order."<tr><td>$current</td><td>$quantity</td><td>".(25*$
quantity)."</td></tr>";

    }

}

#here it will print the total bill and will store the order in the
history of the customer as well as the owner's history.
$order.="</table>";
$email=$_SESSION['user'];
$query="insert into orders values('$order','','$email','')";
mysql_query($query) or die('problem occurred');
echo $order;
echo "<tr><td colspan=3 align='right'>The Grand total :".$cost
."</td></tr></table>";
echo "</div><div class='col 12'><p></p></div></div></div>";
}
?>

```

FILE 8:sachin_order.php

Prologue:

This php file displays the entire menu that is available to the customer to choose the dishes. The prices are listed. The user can select the desired dishes and their quantities. The dishes have been

classified into various categories for the convenience of searching the respective dish. Logout button can be used to exit from the website. Proceed button can be used to go to the next page.

```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="bootstrap.min.css">
    <script src="jquerymin.js"></script>
    <script src="bootstrapmin.js"></script>
    <style>
        th,td{
            padding:5px;
            text-align:left;
            border-bottom: 1px solid #ddd;
        }

        tr:hover{
            background-color: #f0f0f0;
        }

        input{
            width: 50px;
        }

        input.Submit{
            width: 75px;
        }

        tr:nth-child(even){
            background-color: #f2f2f2
        }

        h1{
            text-align:center;
        }
        body{
            padding: 10px;
        }
    </style>
    <title>Menu Page</title>
</head>
<?php
session_start();
    $email=$_SESSION['user'];

    echo $email." you have succesfully logged in</br>";
    $_SESSION['user']=$email;
?>
<body>
<!-- this is for updating the current password(LAYOUT) -- >
    <form action="update_pass.php" method="post" align="right">
        <input type="submit" value="Change Password" name="change"/>
</form>

    <div class="container">
        <ul class="nav nav-tabs">
            <li><a href="060816_1.html">Home</a></li>
```

```

                <li><a href="order_status.php" text-align="right">Order Status</a></li>
                <li class="active"><a data-toggle="tab" href="#Non-Veg">Non-Veg</a></li>
                <li><a data-toggle="tab" href="#Veg">Veg</a></li>
                <li><a data-toggle="tab" href="#Bhujia">Bhujia</a></li>
                <li><a data-toggle="tab" href="#Rolls">Rolls</a></li>
                <li><a data-toggle="tab" href="#Fried-Rice">Fried Rice</a></li>
                <li><a data-toggle="tab" href="#Chinese">Chinese</a></li>
                <li><a data-toggle="tab" href="#Paratha">Paratha</a></li>
                <li><a data-toggle="tab" href="#Salad">Salad</a></li>
                <li><a data-toggle="tab" href="#Soft-Drinks">Soft-Drinks</a></li>
            </ul>
            <div class="tab-content">
                <div id="Non-Veg" class="tab-pane fade in active">
                    <h1>Non-Veg Items</h1>

<form method="post" action="order2.php">

    <table align="center">

        <tr>

            <th>Select</th><th>Name</th><th>Price (Rs)</th><th>Quantity</th>
            </tr>
            <tr>
                <td><input type="checkbox" name="101" value="Chicken Butter Masala (Boneless)"/></td><td>Chicken Butter Masala (Boneless)</td><td>90</td><td><input type="number" name="101q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="102" value="Kadhai Chicken"/></td><td>Kadhai Chicken</td><td>75</td><td><input type="number" name="102q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="103" value="Chicken Masala"/></td><td>Chicken Masala</td><td>75</td><td><input type="number" name="103q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="104" value="Chicken Zeera Pepper Fry"/></td><td>Chicken Zeera Pepper Fry</td><td>75</td><td><input type="number" name="104q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="105" value="Chicken Curry"/></td><td>Chicken Curry</td><td>70</td><td><input type="number" name="105q" min="0"/></td>
            </tr>
            <tr>

```

<input name="106" type="checkbox" value="Egg Keema"/>	Egg Keema	55	<input min="0" name="106q" type="number"/>
<input name="107" type="checkbox" value="Fish Fry"/>	Egg Keema	55	<input min="0" name="107q" type="number"/>
<input name="108" type="checkbox" value="Omelette"/>	Omelette	20	<input min="0" name="108q" type="number"/>
<input name="109" type="checkbox" value="Egg Bhurji"/>	Egg Bhurji	30	<input min="0" name="109q" type="number"/>
<input name="110" type="checkbox" value="Egg Half Fry"/>	Egg Half Fry	20	<input min="0" name="110q" type="number"/>
<input name="111" type="checkbox" value="Chicken Keema"/>	Chicken Keema	95	<input min="0" name="111q" type="number"/>
<input name="112" type="checkbox" value="Palak Chicken"/>	Palak Chicken	95	<input min="0" name="112q" type="number"/>
<input name="113" type="checkbox" value="Mutton Masala (Sunday)"/>	Mutton Masala (Sunday)	120	<input min="0" name="113q" type="number"/>
<input name="114" type="checkbox" value="Egg Tadka"/>	Egg Tadka	65	<input min="0" name="114q" type="number"/>
<input name="115" type="checkbox" value="Egg Curry"/>	Egg Curry	55	<input min="0" name="115q" type="number"/>
<input name="116" type="checkbox" value="Fish Curry"/>	Fish Curry	70	<input min="0" name="116q" type="number"/>

```

        <td><input type="checkbox" name="117" value="Lever
Fry"/></td><td>Lever Fry</td><td>80</td><td><input type="number"
name="117q" min="0"/></td>
    </tr>
    <tr>
        <td><input type="checkbox" name="118" value="Tawa
Chicken"/></td><td>Tawa Chicken</td><td>95</td><td><input type="number"
name="118q" min="0"/></td>
    </tr>
</table>

</div>
<div id="Veg" class="tab-pane fade">
    <h1>Veg Items</h1>
    <table align="center">

<tr>

    <th>Select</th><th>Name</th><th>Price (Rs)</th><th>Quantity</th>
    </tr>
    <tr>
        <td><input type="checkbox" name="201" value="Paneer
Butter Masala"/></td><td>Paneer Butter Masala</td><td>80</td><td><input
type="number" name="201q" min="0"/></td>
    </tr>
    <tr>
        <td><input type="checkbox" name="202" value="Paneer
Masala"/></td><td>Paneer Masala</td><td>60</td><td><input type="number"
name="202q" min="0"/></td>
    </tr>
    <tr>
        <td><input type="checkbox" name="203" value="Kadai
Paneer "/></td><td>Kadai Paneer</td><td>75</td><td><input type="number"
name="203q" min="0"/></td>
    </tr>
    <tr>
        <td><input type="checkbox" name="204" value="Mutter
Paneer"/></td><td>Mutter Paneer</td><td>70</td><td><input type="number"
name="204q" min="0"/></td>
    </tr>
    <tr>
        <td><input type="checkbox" name="205" value="Palak
Paneer"/></td><td>Palak Paneer</td><td>70</td><td><input type="number"
name="205q" min="0"/></td>
    </tr>
    <tr>
        <td><input type="checkbox" name="206" value="Paneer
Bhurji"/></td><td>Paneer Bhurji</td><td>80</td><td><input type="number"
name="206q" min="0"/></td>
    </tr>
    <tr>
        <td><input type="checkbox" name="207" value="Shahi
Paneer"/></td><td>Shahi Paneer</td><td>95</td><td><input type="number"
name="207q" min="0"/></td>
    </tr>
    <tr>
        <td><input type="checkbox" name="208" value="Aloo Gobi
Masala"/></td><td>Aloo Gobi Masala</td><td>55</td><td><input
type="number" name="208q" min="0"/></td>
    </tr>

```



```

        <tr>
            <td><input type="checkbox" name="209" value="Gobi
Masala"/></td><td>Gobi Masala</td><td>65</td><td><input type="number"
name="209q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="210" value="Aloo
Dum"/></td><td>Aloo Dum</td><td>55</td><td><input type="number"
name="210q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="211" value="Aloo
Zeera"/></td><td>Aloo Zeera</td><td>50</td><td><input type="number"
name="211q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="212" value="Bhindi
Masala"/></td><td>Bhindi Masala</td><td>50</td><td><input type="number"
name="212q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="213" value="Bhindi
Fry"/></td><td>Bhindi Fry</td><td>50</td><td><input type="number"
name="213q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="214" value="Daal
Fry"/></td><td>Daal Fry</td><td>45</td><td><input type="number"
name="214q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="215" value="Daal
Tadka"/></td><td>Daal Tadka</td><td>45</td><td><input type="number"
name="215q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="216" value="Palak
Mushroom"/></td><td>Palak Mushroom</td><td>80</td><td><input
type="number" name="216q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="217" value="Channa
Paneer"/></td><td>Channa Paneer</td><td>60</td><td><input type="number"
name="217q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="218" value="Aloo Gobi
Matar"/></td><td>Aloo Gobi Matar</td><td>60</td><td><input
type="number" name="218q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="219" value="Aloo
Masala"/></td><td>Aloo Masala</td><td>50</td><td><input type="number"
name="219q" min="0"/></td>
        </tr>
        <tr>

```

<input type="checkbox"/>	Green Peas Masala	65	<input type="number"/>
<input type="checkbox"/>	Aloo Matar	60	<input type="number"/>
<input type="checkbox"/>	Aloo Bhindi Masala	55	<input type="number"/>
<input type="checkbox"/>	Channa Masala	60	<input type="number"/>
<input type="checkbox"/>	Aloo Palak	60	<input type="number"/>
<input type="checkbox"/>	Aloo Parwal	60	<input type="number"/>
<input type="checkbox"/>	Mushroom Masala	70	<input type="number"/>
<input type="checkbox"/>	Matar Mushroom	70	<input type="number"/>
<input type="checkbox"/>	Rajma Masala	60	<input type="number"/>
<input type="checkbox"/>	Mushroom Chilly	80	<input type="number"/>

Bhujia

Select	Name	Price (Rs)	Quantity
--------	------	------------	----------

```

        </tr>
        <tr>
            <td><input type="checkbox" name="301" value="Aloo
Bhujia"/></td><td>Aloo Bhujia</td><td>50</td><td><input type="number"
name="301q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="302" value="Bhindi
Bhujia"/></td><td>Bhindi Bhujia</td><td>50</td><td><input type="number"
name="302q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="303" value="Gobi
Bhujia"/></td><td>Gobi Bhujia</td><td>50</td><td><input type="number"
name="303q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="304" value="Aloo Gobi
Bhujia"/></td><td>Aloo Gobi Bhujia</td><td>50</td><td><input
type="number" name="304q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="305" value="Aloo Parwal
Bhujia"/></td><td>Aloo Parwal BHujia</td><td>50</td><td><input
type="number" name="305q" min="0"/></td>
        </tr>
    </table>

    </div>
    <div id="Rolls" class="tab-pane fade">
        <h1>Roll Items</h1>
        <table align="center">

        <tr>

            <th>Select</th><th>Name</th><th>Price (Rs)</th><th>Quantity</th>
        </tr>
        <tr>
            <td><input type="checkbox" name="401" value="Egg
Roll"/></td><td>Egg Roll</td><td>35</td><td><input type="number"
name="401q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="402" value="Chicken
Roll"/></td><td>Chicken Roll</td><td>45</td><td><input type="number"
name="402q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="403" value="Veg
Roll"/></td><td>Veg Roll</td><td>25</td><td><input type="number"
name="403q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="404" value="Paneer
Roll"/></td><td>Paneer Roll</td><td>45</td><td><input type="number"
name="404q" min="0"/></td>
        </tr>
    </table>

    </div>
    <div id="Fried-Rice" class="tab-pane fade">

```

```

        <h1>Fried Rice Items</h1>
        <table align="center">
        <tr>

        <th>Select</th><th>Name</th><th>Price (Rs)</th><th>Quantity</th>
        </tr>
        <tr>
                <td><input type="checkbox" name="501" value="Chicken
Fried Rice Schezwan"/></td><td>Chicken Fried Rice
Schezwan</td><td>75</td><td><input type="number" name="501q"
min="0"/></td>
                </tr>
                <tr>
                <td><input type="checkbox" name="502" value="Green
Peas Rice"/></td><td>Green Peas Rice</td><td>60</td><td><input
type="number" name="502q" min="0"/></td>
                </tr>
                <tr>
                <td><input type="checkbox" name="503" value="Jeera
Rice"/></td><td>Jeera Rice</td><td>50</td><td><input type="number"
name="503q" min="0"/></td>
                </tr>
                <tr>
                <td><input type="checkbox" name="504" value="Mushroom
Fried Rice"/></td><td>Mushroom Fried Rice</td><td>70</td><td><input
type="number" name="504q" min="0"/></td>
                </tr>
                <tr>
                <td><input type="checkbox" name="505" value="Chicken
Fried Rice"/></td><td>Chicken Fried Rice</td><td>70</td><td><input
type="number" name="505q" min="0"/></td>
                </tr>
                <tr>
                <td><input type="checkbox" name="506" value="Veg Fried
Rice"/></td><td>Veg Fried Rice</td><td>50</td><td><input type="number"
name="506q" min="0"/></td>
                </tr>
                <tr>
                <td><input type="checkbox" name="507" value="Veg
Schezwan Fried Rice"/></td><td>Veg Schezwan Fried
Rice</td><td>55</td><td><input type="number" name="507q" min="0"/></td>
                </tr>

        <tr>
                <td><input type="checkbox" name="508" value="Potato
Fried Rice"/></td><td>Potato Fried Rice</td><td>55</td><td><input
type="number" name="508q" min="0"/></td>
                </tr>
                <tr>
                <td><input type="checkbox" name="509" value="Potato
Schezwan Fried Rice"/></td><td>Potato Schezwan Fried
Rice</td><td>60</td><td><input type="number" name="509q" min="0"/></td>
                </tr>
                <tr>
                <td><input type="checkbox" name="510" value="Gobi
Fried Rice"/></td><td>Gobi Fried Rice</td><td>60</td><td><input
type="number" name="510q" min="0"/></td>
                </tr>

```

<input type="checkbox"/>	Gobi Schezwan Fried Rice	65	<input type="number"/>
<input type="checkbox"/>	Paneer Fried Rice	70	<input type="number"/>
<input type="checkbox"/>	Paneer Schezwan Fried Rice	75	<input type="number"/>
<input type="checkbox"/>	Chilly Potato	50	<input type="number"/>
<input type="checkbox"/>	Egg Fried Rice	55	<input type="number"/>
<input type="checkbox"/>	Egg Schezwan Fried Rice	60	<input type="number"/>

Chinese Items

Select	Name	Price (Rs)	Quantity
<input type="checkbox"/>	Gobi Manchurian	55	<input type="number"/>
<input type="checkbox"/>	Gobi 65	60	<input type="number"/>
<input type="checkbox"/>	Paneer 65	80	<input type="number"/>

```
 <input type="checkbox" name="604" value="Paneer Chilly"/></td><td>Paneer Chilly</td><td>70</td><td><input type="number" name="604q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="605" value="Paneer Manchurian"/></td><td>Paneer Manchurian</td><td>70</td><td><input type="number" name="605q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="606" value="Chicken Manchurian"/></td><td>Chicken Manchurian</td><td>80</td><td><input type="number" name="606q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="607" value="Chilly Chicken"/></td><td>Chilly Chicken</td><td>80</td><td><input type="number" name="607q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="608" value="Chicken 65"/></td><td>Chicken 65</td><td>80</td><td><input type="number" name="608q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="609" value="Chicken Lollipop"/></td><td>Chicken Lollipop</td><td>90</td><td><input type="number" name="609q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="610" value="Veg Noodles"/></td><td>Veg Noodles</td><td>50</td><td><input type="number" name="610q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="611" value="Veg Schezwan Noodles"/></td><td>Veg Schezwan Noodles</td><td>55</td><td><input type="number" name="611q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="612" value="Egg Noodles"/></td><td>Egg Noodles</td><td>60</td><td><input type="number" name="612q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="613" value="Egg Schezwan Noodles"/></td><td>Egg Schezwan Noodles</td><td>65</td><td><input type="number" name="613q" min="0"/></td> </tr> <tr>  <input type="checkbox" name="614" value="Chicken Noodles"/></td><td>Chicken | | | | | | | | | | |
```

```

Noodles</td><td>70</td><td><input type="number" name="614q"
min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="615"
value="Chicken Schezwan Noodles"/></td><td>Chicken Schezwan
Noodles</td><td>75</td><td><input type="number" name="615q"
min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="616"
value="Gobi Noodles"/></td><td>Gobi Noodles</td><td>60</td><td><input
type="number" name="616q" min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="617"
value="Gobi Schezwan Noodles"/></td><td>Gobi Schezwan
Noodles</td><td>65</td><td><input type="number" name="617q"
min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="618"
value="Paneer Noodles"/></td><td>Paneer
Noodles</td><td>70</td><td><input type="number" name="618q"
min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="619"
value="Paneer Schezwan Noodles"/></td><td>Paneer Schezwan
Noodles</td><td>75</td><td><input type="number" name="619q"
min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="620"
value="Paneer Makkan Wala"/></td><td>Paneer Makkan
Wala</td><td>90</td><td><input type="number" name="620q" min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="621"
value=""/></td><td>Rajama Makkani</td><td>65</td><td><input
type="number" name="621q" min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="622"
value="Dal Makhani"/></td><td>Dal Makhani</td><td>55</td><td><input
type="number" name="622q" min="0"/></td>
</tr>
<tr>
<td><input type="checkbox" name="623"
value="Paneer Schezwan Noodles"/></td><td>Paneer Schezwan
Noodles</td><td>75</td><td><input type="number" name="623q"
min="0"/></td>
</tr>
</table>
</div>
<div id="Paratha" class="tab-pane fade">
<h1>Paratha</h1>
<table align="center">

```

```

        <tr>
            <th>Select</th><th>Name</th><th>Price (Rs) </th><th>Quantity</th>
            </tr>
            <tr>
                <td><input type="checkbox" name="701"
value="Aloo Paratha"/></td><td>Aloo Paratha</td><td>20</td><td><input
type="number" name="701q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="702"
value="Gobi Paratha"/></td><td>Gobi Paratha</td><td>20</td><td><input
type="number" name="702q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="703"
value="Pudina Paratha"/></td><td>Pudina
Paratha</td><td>20</td><td><input type="number" name="703q"
min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="704"
value="Sattu Paratha"/></td><td>Sattu Paratha</td><td>20</td><td><input
type="number" name="704q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="705"
value="Onion Paratha"/></td><td>Onion Paratha</td><td>20</td><td><input
type="number" name="705q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="706"
value="Paneer Paratha"/></td><td>Paneer
Paratha</td><td>35</td><td><input type="number" name="706q"
min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="707" value="Egg
Paratha"/></td><td>Egg Paratha</td><td>35</td><td><input type="number"
name="707q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="708"
value="Chicken Paratha"/></td><td>Chicken
Paratha</td><td>40</td><td><input type="number" name="708q"
min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="709"
value="Plain Paratha"/></td><td>Plain Paratha</td><td>8</td><td><input
type="number" name="709q" min="0"/></td>
            </tr>
            <tr>
                <td><input type="checkbox" name="710"
value="Butter Paratha"/></td><td>Butter
Paratha</td><td>10</td><td><input type="number" name="710q"
min="0"/></td>
            </tr>
        </table>
    
```



```

        <tr>
            <td><input type="checkbox" name="711"
value="Roti"/></td><td>Roti</td><td>6</td><td><input type="number"
name="711q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="712"
value="Butter Roti"/></td><td>Butter Roti</td><td>8</td><td><input
type="number" name="712q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="713"
value="Roti Sabzi"/></td><td>Roti Sabzi</td><td>30</td><td><input
type="number" name="713q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="714"
value="Paratha Sabzi"/></td><td>Paratha Sabzi</td><td>30</td><td><input
type="number" name="714q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="715"
value="Puri Sabzi"/></td><td>Puri Sabzi</td><td>30</td><td><input
type="number" name="715q" min="0"/></td>
        </tr>
    </table>
</div>
<div id="Salad" class="tab-pane fade">
    <h1>Salads</h1>
    <table align="center">
        <tr>
            <th>Select</th><th>Name</th><th>Price (Rs)</th><th>Quantity</th>
        </tr>
        <tr>
            <td><input type="checkbox" name="801"
value="Green Salad"/></td><td>Green Salad</td><td>35</td><td><input
type="number" name="801q" min="0"/></td>
        </tr>
        <tr>
            <td><input type="checkbox" name="802"
value="Onion Salad"/></td><td>Onion Sabzi</td><td>20</td><td><input
type="number" name="802q" min="0"/></td>
        </tr>
    </table>
</div>
<div id="Soft-Drinks" class="tab-pane fade">
    <h1>Soft-Drinks</h1>
    <table align="center">
        <tr>
            <th>Select</th><th>Name</th><th>Price (Rs)</th><th>Quantity</th>
        </tr>
        <tr>
            <td><input type="checkbox" name="901"
value="Kheer"/></td><td>Kheer</td><td>30</td><td><input type="number"
name="901q" min="0"/></td>
        </tr>
    </table>

```

```

                                <td><input type="checkbox" name="902"
value="Lassi"/></td><td>Lassi</td><td>25</td><td><input type="number"
name="902q" min="0"/></td>
                                </tr>
                                </table></div>
                                </div>
                                <br/>
                                <center><input class="Submit" type="submit"
name="submit" value="Proceed"/></center>
                                <br/>
                                </div>
                                </form>
<!--this Is for creating the logout button. -->
    <form action="logout.php" method="post"><input type="submit"
value="logout" name="logout"/></form>
<!--this for creating the order_Status button. -->
    <form action="order_status.php" method="post">
        <input type="submit" value="Status" name="status"/>
    </form>

</body>
</html>

```

FILE 9:signup_project.php

Prologue:

This php file is used when the user wants to create a new account. The details of the table are entered into the database. The user is supposed to provide the fields given in the form like name, password, address, phone no, email, password1, ques1 answer, ques2 answer. If the user name already exists , then the user is asked to enter a new name. Registration is successful if all the processes are carried out successfully.

```

<?php
include 'connection1.php';
if(isset($_POST['submit']))
{
    $name=$_POST['name'];
    $pass=$_POST['password'];
    $address=$_POST['address'];
    $phone=$_POST['phoneno'];
    $email=$_POST['email'];
    $passr=$_POST['password1'];
    $ans1=$_POST['ques1'];
    $ans2=$_POST['ques2'];
    if($passr!=$pass)
    {
        echo "Password is not matching.Try agian";
        return false;
    }
    else
    {
        #it will insert the following values in the customer table
        $query2="insert into customers
values('','$pass','$address','$phone','$email','$name','$ans1','$ans2')
";
    }
}

```

```

        mysql_query($query2) or die("Change the username </br> This
username is already in use");

        echo "$name ." your registration is succesfull";
    }
}
#this is the button for going back to the home page
if(isset($_POST['move']))
{
    ?>
    <script>
        window.open('login_project.php');

    </script>
    <?php
}
mysql_close();
?>

```

FILE 10:update_pass.php

Prologue:

This php file is used when the user wants update his password. The user is asked about his old password and new password. The relevant changes are made to the database. If the current password value that has been entered does not match with the actual current value, then the an alert is raised.

```

<html>
<head>
<!--this is javascript for checking that wether the password in the
password and re-type password field matches.
    <script>
        function check()
        {
            var p=document.getElementById('i').value;
            var pr=document.getElementById('p').value;
            if(pr!='') {
                if(p!=pr)
                {
                    document.getElementById('o').innerHTML='<font
color="red">password doesnt match</font>';
                    return false;
                }
            }
            else
            {
                document.getElementById('o').innerHTML='<font>✔</font>';
            }
        }
    </script>
</head>
<body><pre>
<form action="" method="post">
    Current Password: <input type="password" name="current" />

```

```

        New Password: <input type="password" id='i' name="pass1"/>
        Retype New Password: <input type="password" id='p' name="pass2"
onblur="check();" /> <span id='o'></span>
        <input type="submit" name="sub"/>
</form>
</pre></body>
<?php

include 'connection1.php';
$email=$_SESSION['user'];

if(isset($_POST['sub'])) {
    $cur=$_POST['current'];
    $pass=$_SESSION['pass'];

    $p=$_POST['pass1'];
    $p1=$_POST['pass2'];
    #this is for updating the current password if the old password entered
    is correct.
    if($cur==$pass && $p==$p1 )
    {
        $query="update customers set passwordc='".$p1."' where
passwordc='".$cur."' and email='".$email."'";
        mysql_query($query) or die('check the entered current password
or your phone number');
        echo "Your password has been successfully updated";
    }
    else
    {
        echo "Either current password is incorrect or the password isnt
matching";
    }
}
?>
</html>

```

FILE 11:user.php

Prologue:

This file is the most important file of the entire project. It operates in the background. Some or the other parts of this file keep on getting implemented every now and then. The file provides certain commands necessary for properly updating records table.

```

<?php
include 'connection1.php';
#this for howing the owner all the orders which have been been placed
$query1="select * from orders order by temp desc";
$r=mysql_query($query1);
$queryw="select * from orders;";
$a= mysql_query($queryw);
$num=mysql_num_rows($a);

$i=$num;
echo "<form action='' method='post'>";

#this is for providing the accept, reject and reset button for every
order.
while($row=mysql_fetch_array($r))

```

[illegible]

```

mysql_query($query4) or die('NNNN');
echo "<meta http-equiv='refresh' content='0'>";
}

```

?>

FILE 12:softwaremainlogin.html

Prologue:

This file is used along with login_project.php . It provides the necessary layout for the implementation of the mentioned php file .

```

<!DOCTYPE html>
<html lang="en">
<head>
<head>
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link rel="stylesheet" href="external/css1.css">
    <link rel="stylesheet" href="external/css2.css">
    <link rel="stylesheet" href="external/css3.css">
    <script src="external/jq1.js"></script>
    <script src="external/jq2.js"></script>
    <script src="external/jq3.js"></script>
    <script src="external/jq4.js"></script>
    <script>
        $(document).ready(function() {
            $('select').material_select();
        });</script>
    <title>Page 1</title>
<style>
/* Full-width input fields */

/* Set a style for all buttons */
button {
    background-color: #4CAF50;
    color: white;
    padding: 14px 20px;
    margin: 8px 0;
    border: none;
    cursor: pointer;
    width: 100%;
}

/* Extra styles for the cancel button */
.cancelbtn {
    width: auto;
    padding: 10px 18px;
    background-color: #f44336;
}
.cancelbtn2{
    background-color: #f44336;
    color: white;
    padding: 10px 18px;
    margin: 8px 0;
}

```

```

        border: none;
        cursor: pointer;
        width: 100%;
    }

    /* Center the image and position the close button */
    .imgcontainer {
        text-align: center;
        margin: 24px 0 12px 0;
        position: relative;
    }

    img.avatar {
        width: 40%;
        border-radius: 50%;
    }

    .container {
        padding: 16px;
    }

    span.psw {
        float: right;
        padding-top: 16px;
    }

    /* The Modal (background) */
    .modal {
        display: none; /* Hidden by default */
        position: fixed; /* Stay in place */
        z-index: 1; /* Sit on top */
        left: 0;
        top: 0;
        width: 100%; /* Full width */
        height: 100%; /* Full height */
        overflow: hidden; /* Enable scroll if needed */
        background-color: rgb(0,0,0); /* Fallback color */
        background-color: rgba(0,0,0,0.4); /* Black w/ opacity */
        padding-top: 20px;
    }

    /* Modal Content/Box */
    .modal-content {
        background-color: #fefefe;
        margin: 2% auto 2% auto; /* 5% from the top, 15% from the bottom
and centered */
        border: 1px solid #888;
        width: 50%; /* Could be more or less, depending on screen size */
    }

    /* The Close Button (x) */
    .close {
        position: absolute;
        right: 25px;
        top: 0;
        color: #000;
        font-size: 35px;
    }

```

```

        font-weight: bold;
    }

    .close:hover,
    .close:focus {
        color: red;
        cursor: pointer;
    }

    /* Add Zoom Animation */
    .animate {
        -webkit-animation: animatezoom 0.6s;
        animation: animatezoom 0.6s
    }

    @-webkit-keyframes animatezoom {
        from {-webkit-transform: scale(0)}
        to {-webkit-transform: scale(1)}
    }

    @keyframes animatezoom {
        from {transform: scale(0)}
        to {transform: scale(1)}
    }

    /* Change styles for span and cancel button on extra small screens */
    @media screen and (max-width: 300px) {
        span.psw {
            display: block;
            float: none;
        }
        .cancelbtn {
            width: 100%;
        }
    }

    .mySlides {
        display:none;
    }

    .margins-custom{
        margin-top:16px!important;
    }

</style>
</head>
<body class="container">
<div class="row" style="background-color:'#fefefe'">
    <div class="col 18"><p></p></div>
    <div class="col 12"><button
onclick="document.getElementById('id01').style.display='block'"
class='btn waves-effect waves-light yellow'>Login</button></div>
    <div class="col 12"><button
onclick="location.href='softwareuserssignup.html'" class="btn waves-
effect waves-light orange">Sign Up!</button></div>
</div>

<div class="margins-custom" style="height:100%; max-width:100%">
    
    

```



```

    
</div>

<div style="text-align:center">
<button onclick="location.href='About Us.html'" class="btn waves-effect
waves-light blue">About Us</button>
</div>

<div id="id01" class="modal">
<div class="modal-content animate">
    <form action="login_project.php" method="post">
        <div class="imgcontainer">
            <span
onclick="document.getElementById('id01').style.display='none'"
class="close" title="Close Modal">&times;</span>
        </div>
        <div style="padding:10px">
            <h5>Login</h5>
            <div class="row">
                <div class="input-field col s12">
                    <i class="material-icons prefix">account_circle</i>
                    <input placeholder="Email" id="email" type="email"
class="active validate" name="email" required>
                    <label for="email">Email</label>
                </div>
                <div class="input-field col s12">
                    <label for="password">Password</label>
                    <input id="password" type="password" placeholder="Password"
class="validate" name="password" required>
                </div>
                <div style="text-align:center">
                    <a href="softwareuserforgetpass.html">Forgot
Password?</p>
                </div>
            </div>
            <div class="row" >
                <div class="col l5"><button type="submit" name="submit"
class='btn waves-effect waves-light green'>Login</button></div>
                <div class="col l5"><button type="reset" class="btn waves-
effect waves-light red">Clear!</button></div>
            </div>
        </form>
    </div>
</div>

<script>
    var myIndex = 0;
    carousel();

    function carousel() {
        var i;
        var x = document.getElementsByClassName("mySlides");
        for (i = 0; i < x.length; i++) {
            x[i].style.display = "none";
        }
        myIndex++;
        if (myIndex > x.length) {myIndex = 1}
        x[myIndex-1].style.display = "block";

```

```

        setTimeout(carousel, 2000); // Change image every 2 seconds
    }
</script>

</body>
</html>

```

FILE 13:softwareusersignup.html

Prologue:

This file is used along with signup_project.php . It provides the necessary layout for the implementation of the mentioned php file .

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Sign Up!</title>
    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link rel="stylesheet" href="external/css1.css">
    <link rel="stylesheet" href="external/css2.css">
    <link rel="stylesheet" href="external/css3.css">
    <script src="external/jq1.js"></script>
    <script src="external/jq2.js"></script>
    <script src="external/jq3.js"></script>
    <script src="external/jq4.js"></script>
    <script>
        $(document).ready(function() {
            $('select').material_select();
        });</script>
</head>
<body class='container'>
    <form method="post" action="signup_project.php" method="post">
        <h2>Sign Up!</h2>
        <div class="row">
            <div class="input-field col s12">
                <i class="material-icons prefix">account_circle</i>
                <input placeholder="Username" id="name" name="name"
name="name" type="text" class="active validate" required>
                <label for="name">Username</label>
            </div>
        </div>
        <div class="row">
            <div class="input-field col s6">
                <label for="password">Password</label>
                <input name="password" id="password" name="password"
type="password" placeholder="Password" class="validate" required>
            </div>
            <div class="input-field col s6">
                <label for="password">Retype Password</label>
                <input name="password1" id="password1" onblur="check()"
name="password1" type="password" placeholder="Retype Password"
class="validate" required>
            </div>
        </div>
        <div class="row">
            <div class="input-field col s12">

```

```

        <input placeholder="Phone/Mobile No" name="phoneno"
id="phone" type="number" class="validate">
        <label for="phone">Contact No:</label>
    </div>
</div>
<div class="row">
    <div class="input-field col s12">
        <input placeholder="Email" id="email" name="email"
type="email" class="validate">
        <label for="email">Email</label>
    </div>
</div>
<div class="row">
    <div class="input-field col s12">
        <i class="material-icons prefix">mode_edit</i>
        <textarea name="address" id="address"
class="materialize-textarea"></textarea>
        <label for="address">Address</label>
    </div>
</div>
<div class="row">
    <h5>Security Questions:</h5><br/>
    <b>Question1</b>: What was the nickname of your first friend ?<br>
    <div class="input-field col s12">
        <input placeholder="Answer" name="ques1" id="ques1"
type="text" class="active validate">
    </div>
</div>
<div class='row'>
    <b>Question2</b>: What was the first song which you heard ?<br>
    <div class="input-field col s12">
        <input id="ques2" type="text" name="ques2"
placeholder="Answer" class="validate" >
    </div>
</div>
<div class="row" >
    <div class="col l6"><button type="submit" name="submit"
class='btn waves-effect waves-light green'>Register</button></div>
    <div class="col l6"><button type="reset" class="btn waves-
effect waves-light red">Clear!</button></div>
</div>
</div>
</form>
<script>
function check()
{
    var p=document.getElementById('i').value;
    var pr=document.getElementById('p').value;
    if(pr!='') {
        if(p!=pr)
        {
            document.getElementById('o').innerHTML='<font
color="red">password doesnt match</font>';
            return false;
        }
    }
    else
    {

```

```
        document.getElementById('o').innerHTML='<font>✓</font>';
    }}

}
</script>
</body>
</html>
```

FILE 13:css2.css

Prologue:

This file has been used for styling.

```
/* fallback */
@font-face {
  font-family: 'Material Icons';
  font-style: normal;
  font-weight: 400;
  src: local('Material Icons'), local('MaterialIcons-Regular'),
  url(https://fonts.gstatic.com/s/materialicons/v18/2fcrYFNaTjcS6g4U3t-
  Y5ZjZjT5FdEJ140U2DJYC3mY.woff2) format('woff2');
}

.material-icons {
  font-family: 'Material Icons';
  font-weight: normal;
  font-style: normal;
  font-size: 24px;
  line-height: 1;
  letter-spacing: normal;
  text-transform: none;
  display: inline-block;
  white-space: nowrap;
  word-wrap: normal;
  direction: ltr;
  -moz-font-feature-settings: 'liga';
  -moz-osx-font-smoothing: grayscale;
}
```

FILE 14:css1.css

Prologue:

This file has been used for styling.

```
/*!

* Bootstrap v3.3.7 (http://getbootstrap.com)

* Copyright 2011-2016 Twitter, Inc.

* Licensed under MIT (https://github.com/twbs/bootstrap/blob/master/LICENSE)

*/
```

FILE 15:css3.css

Prologue:

This file has been used for styling.

/*!

* Materialize v0.97.3 (<http://materializecss.com>)

* Copyright 2014-2015 Materialize

* MIT License (<https://raw.githubusercontent.com/Dogfalo/materialize/master/LICENSE>)

*/

FILE 16:jq1.js

Prologue:

This file has been used for styling.

/*! jQuery v1.12.4 | (c) jQuery Foundation | jquery.org/license */

FILE 17:jq2.js

Prologue:

This file has been used for styling.

/*!

* Bootstrap v3.3.7 (<http://getbootstrap.com>)

* Copyright 2011-2016 Twitter, Inc.

* Licensed under the MIT license

*/

FILE 18:jq3.js

Prologue:

This file has been used for styling.

/*! jQuery v2.1.1 | (c) 2005, 2014 jQuery Foundation, Inc. | jquery.org/license */

FILE 19:jq4.js

Prologue:

/*!

* Materialize v0.97.3 (<http://materializecss.com>)

* Copyright 2014-2015 Materialize

* MIT License (<https://raw.githubusercontent.com/Dogfalo/materialize/master/LICENSE>)

*/

FILE 19:forget_password.php

Prologue:

This file is used when the user forgets his/her password.

```
<?php
include 'connection1.php';

if(isset($_POST['user']))
{
    $user2=$_POST['user'];
    $_SESSION['name']=$user2;
    $query="select * from customers where email='".$user2."'";
    $r=mysql_query($query) or die(mysql_error());
    if(mysql_num_rows($r)==1)
    {
        //echo "<form action='' method='post'> </br><b>Question1</b>:What
        was the nickname of your first friend ?<br> <textarea rows='5'
        cols='20' name='ques1' required></textarea><br><br><b>Question2</b>:What
        was the first song which you heard ?<br><textarea rows='5' cols='20'
        name='ques2' required></textarea><br><br><input type='submit'
        name='sub2' value='submit' /></form>";
        echo '<head>
        <title>Forgot Password</title>
        <meta name="viewport" content="width=device-width, initial-
        scale=1">
        <link rel="stylesheet" href="external/css1.css">
        <link rel="stylesheet" href="external/css2.css">
        <link rel="stylesheet" href="external/css3.css">
        <script src="external/jq1.js"></script>
        <script src="external/jq2.js"></script>
        <script src="external/jq3.js"></script>
        <script src="external/jq4.js"></script>
        <script>
            $(document).ready(function() {
                $(select).material_select();
            });</script>
        </head>
        <body class="container" padding:10px>
        <div class=row>
        <div class=col 17> <h4>Forgot Password</h4></div>
        <div class="col 15" ><br><button
        onclick="location.href=softwaremainlogin.html" class="btn waves-effect
        waves-light red">Home</button></div>
        </div>
        <form action="forget_password.php" method="post">
        <div class="row">
        <b>Question1</b>:What was the nickname of your first friend ?<br>
        <div class="input-field col s12">
            <input placeholder="Answer" id="ques1" name="ques1"
            type="text" class="active validate">
        </div>
        </div>
        </div>
```

```

        <div class=row>
        <b>Question2</b>:What was the first song which you heard ?<br>
        <div class="input-field col s12">
            <input id="ques2" type="text" name="ques2"
placeholder="Answer" class="validate" >
        </div>
        </div>
        <div class="col l5"><button type="submit" name="sub2" class=btn
waves-effect waves-light amber>Get Password</button></div>
    </form>
</body>';

}

else
{
    echo 'username doesnt match';
}

}

if(isset($_POST['sub2']))
{
    $ans1=$_POST['ques1'];
    $ans2=$_POST['ques2'];
    $user2=$_SESSION['name'];
    $query2="select * from customers where question1='".$ans1."' and
question2='".$ans2."' and email='".$user2."'";
    $v=mysql_query($query2) or die(mysql_error());
    if(mysql_num_rows($v)==1)
    {
        $query3="select passwordc from customers where
question1='".$ans1."' and question2='".$ans2."' and
email='".$user2."'";
        $tf=mysql_query($query3) or die(mysql_error());
        $t=mysql_fetch_array($tf) or die(mysql_error());
        echo "Your password was : <h1>".$t['passwordc']. "</h1>";
    }
    else
    {
        echo "Answers of the questions doesnt match";
    }
}

?>

```

FILE 20 : softwareuserforgetpass.html

Prologue:

This file is used when the user forgets his/her password.

```

<!DOCTYPE html>
<html lang="en">
<head>
    <title>Forgot Password</title>

```

```

    <meta name="viewport" content="width=device-width, initial-
scale=1">
    <link rel="stylesheet" href="external/css1.css">
    <link rel="stylesheet" href="external/css2.css">
    <link rel="stylesheet" href="external/css3.css">
    <script src="external/jq1.js"></script>
    <script src="external/jq2.js"></script>
    <script src="external/jq3.js"></script>
    <script src="external/jq4.js"></script>
    <script>
        $(document).ready(function() {
            $('select').material_select();
        });</script>
</head>
<body class='container' padding:10px>
<div class='row'>
<div class='col 17'> <h4>Forgot Password</h4></div>
<div class="col 15" ><br><button
onclick="location.href='softwaremainlogin.html'" class="btn waves-
effect waves-light red">Home</button></div>
</div>
<form action="forget_password.php" method="post">
    <div class="row">
        <b>Question1</b>:What was the nickname of your first friend ?<br>
        <div class="input-field col s12">
            <input placeholder="Answer" id="ques1" name="ques1"
type="text" class="active validate">
        </div>
    </div>
    <div class='row'>
        <b>Question2</b>:What was the first song which you heard ?<br>
        <div class="input-field col s12">
            <input id="ques2" type="text" name="ques2"
placeholder="Answer" class="validate" >
        </div>
    </div>
    <div class="col 15"><button type="submit" name="submit"
class='btn waves-effect waves-light amber'>Get Password</button></div>
</form>
</body>
</html>

```

SOFTWARE ENGINEERING THEORY PROJECT LEGACY



Fall Semester 2016-2017

TOPIC:

Restaurant Service System



Teacher:

Prof. Alok Chauhan

STUDENTS:

- Osho Agyeya(15BCE1326)
- Kashish Miglani(15BCE1003)
- Sanchay Gupta(15BCE1190)
- Sachin Gopal(15BCE1188)

SECTION 1: PROJECT DESCRIPTION:

The project is aimed at developing a website as a restaurant service system. The customers are provided with the functionality of being able to send their orders online to the restaurant administrator. They are supposed to create an account on the website. Same is expected from the restaurant administrator. Customers have been provided various dishes in the menu from which they can choose the desired dishes and their respective quantities. The bill is calculated and displayed on the following screen. The administrator has the choice of accepting or rejecting the orders that have been placed by the customers. The administrator can reset his preferences in case of any predicament.

SECTION 2: INITIAL EXPECTATIONS:

The description above provides the basic functionalities of the application. The initial expectation of the project were as follows:

- The website was supposed to be launched on a proper web server.
- The customer was supposed to have all the relevant functionalities in order to make changes to his/her account.
- The styling of the project website was supposed to be done perfectly using CSS, Bootstrap.
- The admin was supposed to have all the functionalities to enable him to send a message to his customers via chat.
- The customers were supposed to have this functionality of being able to decide whether they want the food to be delivered to their home or to eat it in restaurant.
- An additional feature of being able to book a table at the restaurant.
- A la carte and buffet options were supposed to be incorporated in the project.
- The current location of the customer was supposed to be tracked by the website in order to tell how much amount of

time will be needed in order to deliver their food items to them if they opt for delivery.

- An inventory telling the admin about how many resources(vegetables,syrups,spices) are left in the stock was supposed to be included in the project
- An option of availing a discount by ordering within a specific time range.
- Specifying all the necessary details about what ingredients have been used in the preparation of each dish.

SECTION 3:CURRENT STATUS OF THE PROJECT:

As far as the current status of the project is concerned, a major part of the proposed expectations have been achieved. The following functionalities have been completed successfully:

- The user is successfully able to launch the website.
- The user can create a secure account.
- The user has the option of changing his account password.
- Customers are able to view the menu perfectly.
- The names of the dishes and their price has been listed in an immaculate manner.
- The bill is calculated perfectly and is displayed tidily.
- The admin has the option of accepting or rejecting the orders that have been sent by the customers.
- The admin has the option of resetting his personal choice.
- The page pertaining to the location and description of the restaurant has been made attractive and pleasing.
- The database has been created perfectly which maintains all the customer order records and admin record processing.

SECTION 4:REMAINING AREAS OF CONCERN

- The customer login needs to be simpler and less congested.
- The inventory needs to be created and updated.
- The timing for discount option needs to be activated.

- A la carte v/s buffet option needs to be incorporated.
- Booking a table at a restaurant
- Chatting with the customers and admin needs to be included.
- Customer location tagging needs to be looked at with greater detail.
- Dine in and home delivery option is to be included.

SECTION 5:ACTIVITIES/TIME LOG(S)

The project was started on Tuesday, 26 July 2016, 20:30:00.It has been completed on Friday, 21 October 2016, 19:03:18.

The duration is 87 days, 19 hours, 3 minutes and 18 seconds

Or 2 months, 25 days excluding the end date.

Alternative time units:

87 days, 19 hours, 3 minutes and 18 seconds can be converted to one of these units:

7,585,398 seconds

126,423 minutes (rounded down)

2107 hours (rounded down)

87 days (rounded down)

12 weeks (rounded down)

23.99% of 2016

The time slice that has been allotted to various activities are as follows(for 2016):

Creation of all the HTML pages/basic layout: 30 July – August 10. Time period between July 30, 2016 and August 10, 2016 (exclusive) equals to:

11 days

... or 1 weeks & 4 days

Database creation complete: 2 August – August 20. Time period between August 2, 2016 and August 20, 2016 (exclusive) equals to:
18 days

... or 2 weeks & 4 days

CSS Styling: August 20-September 15. Time period between August 20, 2016 and September 15, 2016 (exclusive) equals to:
26 days

... or 3 weeks & 5 days

BOOTSTRAP Styling : September 12- October 10. Time period between September 12, 2016 and October 10, 2016 (exclusive) equals to:
28 days

... or 4 weeks

Final linking and necessary formatting: Done till 21st October

SECTION 6: TECHNICAL LESSONS LEARNED:

A plethora of technical lessons were imbibed while making this project.

- Softwares like XAMP or WAMP have a high probability of crashing. So it is necessary keep background processes to a minimum while running these softwares.
- Extensive knowledge about HTML, CSS, JAVASCRIPT, PHP, BOOTSTRAP.
- The database should have a key entry in order to keep the rows distinct and achieve a standard level of normalisation from the beginning.
- The CSS should have tags properly defined in order to keep the formatting exclusive to the elements.
- The BOOTSTRAP needs proper internet connection for desired working.

- The connections made via sql should be properly done so that the data is sent to and received from the database without any hassle.
- The priority level of development is as follows:HTML,PHP,JAVASCRIPT,CSS,BOOTSTRAP
- The database should be checked at regular intervals.
- Proper commenting is necessary for easy readability of the code.

SECTION 7: MANAGERIAL LESSONS LEARNED

Management is significant in order to keep proper track of resources,schedule,quality. The following managerial lessons have been learnt in the course of this project:

005. A manager who is his own systems engineer or financial manager is one who will probably try to do open heart surgery on himself.

1. Wrong decisions made early can be salvaged, but "right" decisions made late cannot.
2. Never make excuses; instead, present plans of actions to be taken.
3. Managers who rely on the paperwork to do the reporting of activities are known failures.
4. Not all successful managers are competent and not all failed managers are incompetent. Luck still plays a part in success or failure, but luck favors the competent, hard-working manager.
5. Documentation does not take the place of knowledge. There is a great difference in what is supposed to be, what is thought to have been, and what the reality is. Documents are normally a static picture in time which is outdated rapidly.

6. People have reasons for doing things the way they do them. Most people want to do a good job, and if they don't, the problem is they probably don't know how or exactly what is expected.
7. A puzzle is hard to discern from just one piece, so don't be surprised if team members deprived of information reach the wrong conclusion.
8. Reviews are for the reviewed and not the reviewer. The review is a failure if the reviewed learn nothing from it.
9. Management principles are still the same. It is just the tools that have changed. You still should find the right people to do the work and get out of the way so they can do
10. It is mainly the incompetent that don't like to show off their work.
11. A working meeting has about six people attending. Meetings larger than this are for information transfer.
12. All problems are solvable in time, so make sure you have enough schedule contingency - if you don't, the next project manager that takes your place will.
13. Sometimes the best thing to do is nothing. It is also occasionally the best help you can give. Just listening is all that is needed on many occasions. You may be the boss but, if you constantly have to solve someone's problems, you are working for him.
14. Integrity means your subordinates trust you.
15. People who monitor work and don't help get it done, never seem to know exactly what is going on.

16. The seeds of problems are laid down early. Initial planning is the most vital part of a project. Review of most failed projects or of project problems indicates that the disasters were well planned to happen from the start.

17. Whoever said beggars can't be choosers doesn't understand project management. Many times it is better to trust to luck than to get known poor support.

18. There is only one solution to a weak project manager in industry — get rid of him fast. The main job of a project manager in industry is to keep the customer happy. Make sure the one working with you knows that "on schedule, on cost, and a good product"--not flattery--is all that makes you happy.

19. Talk is not cheap. The best way to understand a personnel or technical problem is to talk to the right people. Lack of talk at the right levels is deadly.

20. Projects require teamwork to succeed. Remember most teams have a coach and not a boss, but the coach still has to call some of the plays.

21. In political decisions, do not look for logic - look for politics.

22. Reviews, meetings, and reality have little in common.

23. The project manager who is the smartest man on his project has done a lousy job of recruitment.

20. Client wants the best. Once you tell him what the best costs, he asks if you can scale back.

SECTION 8: RECOMMENDATIONS TO FUTURE PROJECTS

Recommendations to the future project will be a comprehensive set of points.

1. **Emphasize the importance of teamwork**—Before the groups are formed and the task is set out, teachers should make clear why this particular assignment is being done in groups. Students are still regularly reporting in survey data that teachers use groups so they don't have to teach or have as much work to grade. Most of us are using groups because employers in many fields want employees who can work with others they don't know, may not like, who hold different views, and possess different skills and capabilities.
2. **Teach teamwork skills**—Most students don't come to group work knowing how to function effectively in groups. Whether in handouts, online resources, or discussions in class, teachers need to talk about the responsibilities members have to the group (such as how sometimes individual goals and priorities must be relinquished in favor of group goals) and about what members have the right to expect from their groups. Students need strategies for dealing with members who are not doing their fair share. They need ideas about constructively resolving disagreement. They need advice on time management.
3. **Use team-building exercises to build cohesive groups**—Members need the chance to get to know each other, and they should be encouraged to talk about how they'd like to work together. Sometimes a discussion of worst group experiences makes clear to everyone that there are behaviors to avoid. This might be followed with a discussion of what individual members need from the group in order to do their best work. Things like picking a group name and creating a logo also help create a sense of identity for the group, which in turn fosters the commitment groups need from their members in order to succeed.
4. **Thoughtfully consider group formation**—Most students prefer forming their own groups, and in some studies these groups are more productive. In other research, students in these groups “enjoy” the experience of working together, but they don't always get a lot done. In most professional contexts, people don't get to choose their project partners. If the goal is for students to learn how to work with others

whom they don't know, then the teacher should form the groups. There are many ways groups can be formed and many criteria that can be used to assemble groups. Groups should be formed in a way that furthers the learning goals of the group activity.

5. **Make the workload reasonable and the goals clear**—Yes, the task can be larger than what one individual can complete. But students without a lot of group work experience may struggle with large, complex tasks. Whatever the task, the teacher's goals and objectives should be clear. Students shouldn't have to spend a lot of time trying to figure out what they are supposed to be doing.
6. **Consider roles for group members**—Not all the literature recommends assigning roles, although some does. Roles can emerge on their own as members see what functions the group needs and step up to fill those roles. However, this doesn't always happen when students are new to group work. The teacher can decide on the necessary roles and suggest them to a group with the group deciding who does what. The teacher can assign the roles, but should realize that assigning roles doesn't guarantee that students will assume those roles. Assigned roles can stay the same or they can rotate. However they're implemented, roles are taken more seriously if groups are required to report who filled what role in the group.
7. **Provide some class time for meetings**—It is very hard for students to orchestrate their schedules. Part of what they need to be taught about group work is the importance of coming to meetings with an agenda—some expectation about what needs to get done. They also need to know that significant amounts of work can be done in short periods of time, provided the group knows what needs to be done next. Working online is also increasingly an option, but being able to convene even briefly in class gives groups the chance to touch base and get organized for the next steps.
8. **Request interim reports and group process feedback**—One of the group's first tasks ought to be the creation of a time line—what they expect to have done by when. That time line should guide instructor requests for progress reports from the group, and the reports should be supported with evidence. It's not good enough for the group to say it's collecting references. A list of references collected should be submitted with the report. Students should report individually on how well the group is working together, including their contributions to the group.

Ask students what else could they contribute that would make the group function even more effectively.

9. **Require individual members to keep track of their contributions**—The final project should include a report from every member identifying their contribution to the project. If two members report contributing the same thing, the teacher defers to the student who has evidence that supports what the student claims to have done.
 10. **Include peer assessment in the evaluation process**—What a student claims to have contributed to the group and its final product can also be verified with a peer assessment in which members rate or rank (or both) the contributions of others. A formative peer assessment early in the process can help members redress what the group might identify as problems they are experiencing at this stage.
-



Restaurant Service System

15BCE1003: Kashish Miglani, **15BCE1188:** Sachin Gopal, **15BCE1190:** Sanchay Gupta, **15BCE1326:** Osho Agyeya | **Prof. Alok Chauhan** | SCSE

Motivation/Innovation

We had the motivation to make this website after observing and undergoing the confusion and miscommunication that occurs in placing telephonic orders to local restaurants. Our site aims to digitize the process of placing orders, making the entire process simpler for the customer and the restaurant manager. The customer can also go through the menu online before placing an order and thus have an idea about the expected billing amount. This will help the customer make an informed decision.

Scope of the Project

The project is focused upon developing a web application that will allow the specific restaurant administrators to handle various food orders that are placed by multitudes of customers over the Internet. The customers can make use of this web application to post their orders online and view the consequent bill that is generated. The administrator gets to decide which orders are verified by him/her. The customer can also view the order status and review the order.

Methodology

The site was created using the concept studied in Software Engineering, Database Management Systems and Internet and Web Programming and using the following languages/services: **HTML, CSS, JavaScript, PHP, MySQL, WAMP and jQuery.**

Screenshot



Results

The website will allow the user to do the following:

Customer:

- > Register into the web application and save the delivery address
- > Browse menu of the associated restaurant
- > Determine the bill before placing the order
- > Place order through the web application
- > View order status and past orders

Restaurant Manager:

- > The restaurant manager will be able to receive a list of the orders placed
- > On the basis of inventory and demand for the dishes, the manager can plan the resources and appropriately confirm or reject the order
- > The restaurant manager can directly view a log of the orders placed and earnings from the orders. This eases the management process
- > Based on the demand for a product, the manager can allocate more resources for it in subsequent days

The system will help the customer in placing orders by removing discrepancies in communication. The manager will be able to allocate more time in delivering the order to the customer due to automated order receiving. This will increase the profitability of the restaurant.

Conclusion/Summary

The site is ready to be deployed and in the future can be extended to similar local restaurants to further their interests through digitalization.

Acknowledgments/References

1. www.w3school.com
2. www.w3resource.com
3. www.stackoverflow.com
4. www.tutorialpoint.com
5. www.materializecss.com
6. www.developer.google.com
7. Head First PHP and MySQL by Lynn Beighley and Michael Morrison
8. Head First HTML and CSS by Elisabeth Robson and Eric Freeman