DETAILED DESIGN

Project Code:	POS-01
Project Name:	Pizza Ordering System

Revision History

Version (x.yy)	Date of Revision	Description of Change	Reason for Change	Affected Sections	Approved By
1.0	16/04/2013	Initial Draft			
1.1	10/05/2013	Revision			
2.10	Aug 2013	Revision			
2.20	Sept 2013	Revision	Mapping with CPC		
2.3	Dec 2013	Revision	Mapping with UCF		

Affected Groups

Development Engineering
Quality Assurance
XYZ Automation Ltd

List of Reference Documents

Name	Version No.
1.RS_POS	2.20
2.FS_POS	2.20
3.	
4.	

Prepared by/Date **Reviewed by/Date** **Approved by/Date**

Table of Contents

1. Introduction

1.1 Background

XYZ Automation Ltd focuses on automating various systems that have been working manually since years**.**

1.2 Purpose

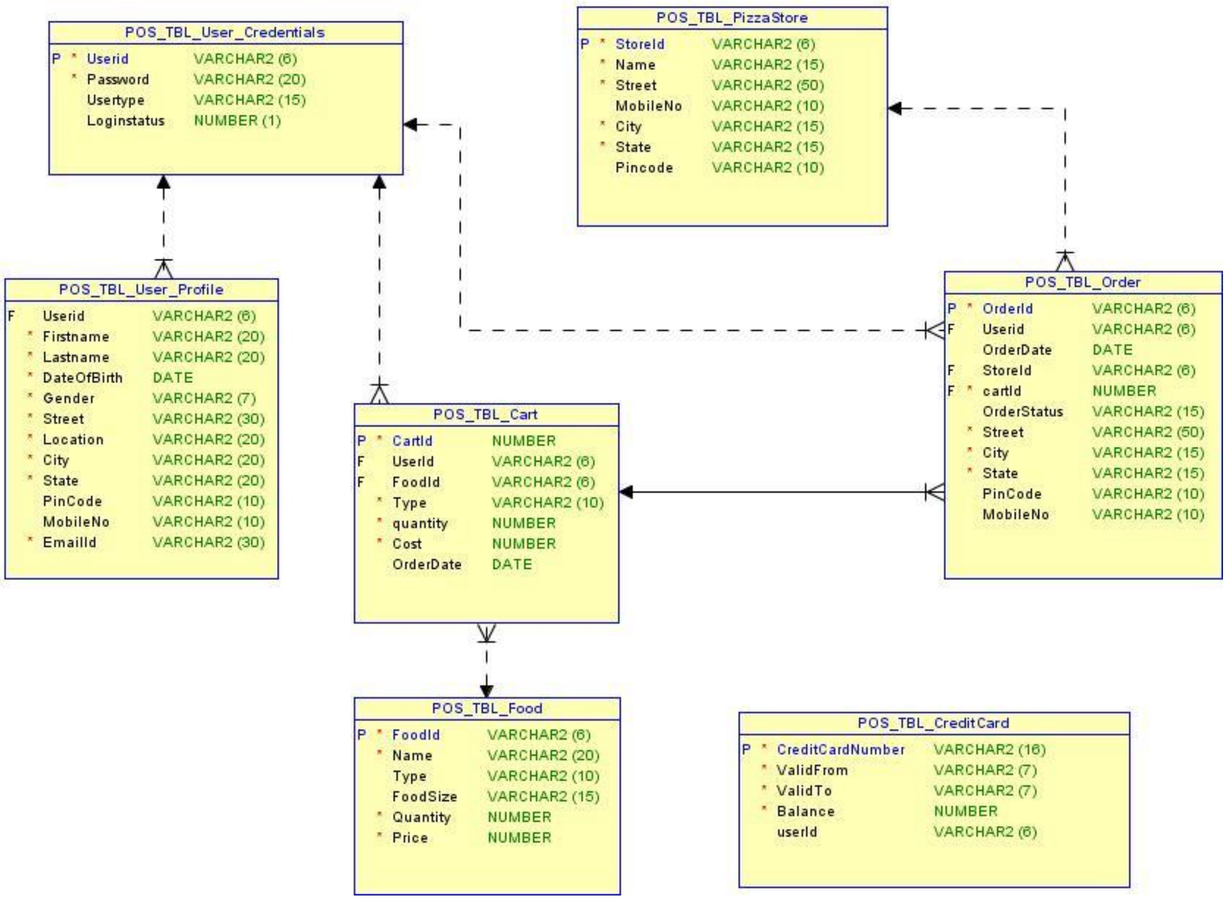
XYZ Automation Ltd recently planned to automate “ Pizza Ordering System” - a Web application intended to be used by user for ordering of pizza.

1.3 Scope

The scope of the Pizza Ordering System (POS) will be to provide the functionality as described in Functional Requirements document. The system will be developed on a Windows XP/Windows 7 machine using Java/J2EE technology and Oracle database.

2. Global Data Structures and Shared Data Functions

This section describes the structure of 7 tables to be used for the implementation of requirements as stated in the specification.



POS_TBL_User_Credentials

P	*	Userid	VARCHAR2 (6)
	*	Password	VARCHAR2 (20)
		Usertype	VARCHAR2 (15)
		Loginstatus	NUMBER (1)

POS_TBL_PizzaStore

P	*	StoreId	VARCHAR2 (6)
	*	Name	VARCHAR2 (15)
	*	Street	VARCHAR2 (50)
		MobileNo	VARCHAR2 (10)
	*	City	VARCHAR2 (15)
	*	State	VARCHAR2 (15)
		Pincode	VARCHAR2 (10)

POS_TBL_User_Profile

F		Userid	VARCHAR2 (6)
	*	Firstname	VARCHAR2 (20)
	*	Lastname	VARCHAR2 (20)
	*	DateOfBirth	DATE
	*	Gender	VARCHAR2 (7)
	*	Street	VARCHAR2 (30)
	*	Location	VARCHAR2 (20)
	*	City	VARCHAR2 (20)
	*	State	VARCHAR2 (20)
		PinCode	VARCHAR2 (10)
		MobileNo	VARCHAR2 (10)
	*	EmailId	VARCHAR2 (30)

POS_TBL_Cart

P	*	CartId	NUMBER
F		UserId	VARCHAR2 (6)
F		FoodId	VARCHAR2 (6)
	*	Type	VARCHAR2 (10)
	*	quantity	NUMBER
	*	Cost	NUMBER
		OrderDate	DATE

POS_TBL_Order

P	*	OrderId	VARCHAR2 (6)
F		Userid	VARCHAR2 (6)
		OrderDate	DATE
F		StoreId	VARCHAR2 (6)
F	*	cartId	NUMBER
		OrderStatus	VARCHAR2 (15)
	*	Street	VARCHAR2 (50)
	*	City	VARCHAR2 (15)
	*	State	VARCHAR2 (15)
		PinCode	VARCHAR2 (10)
		MobileNo	VARCHAR2 (10)

POS_TBL_Food

P	*	FoodId	VARCHAR2 (6)
	*	Name	VARCHAR2 (20)
		Type	VARCHAR2 (10)
		FoodSize	VARCHAR2 (15)
	*	Quantity	NUMBER
	*	Price	NUMBER

POS_TBL_CreditCard

P	*	CreditCardNumber	VARCHAR2 (16)
	*	ValidFrom	VARCHAR2 (7)
	*	ValidTo	VARCHAR2 (7)
	*	Balance	NUMBER
		userId	VARCHAR2 (6)

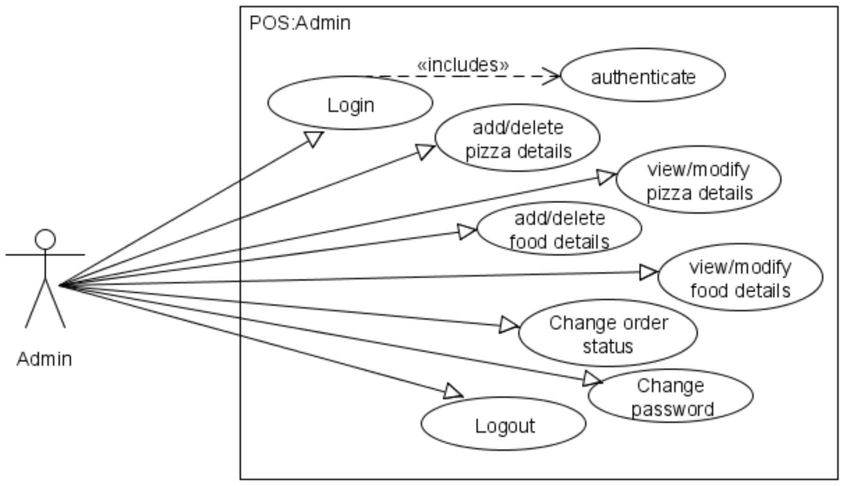
3. High Level Design

This section describes the high level design diagrams Use case diagram with Use Case definition, Sequence Diagram and Class Diagram which provides a visual representation of the requirements , logical flow and their class representations.

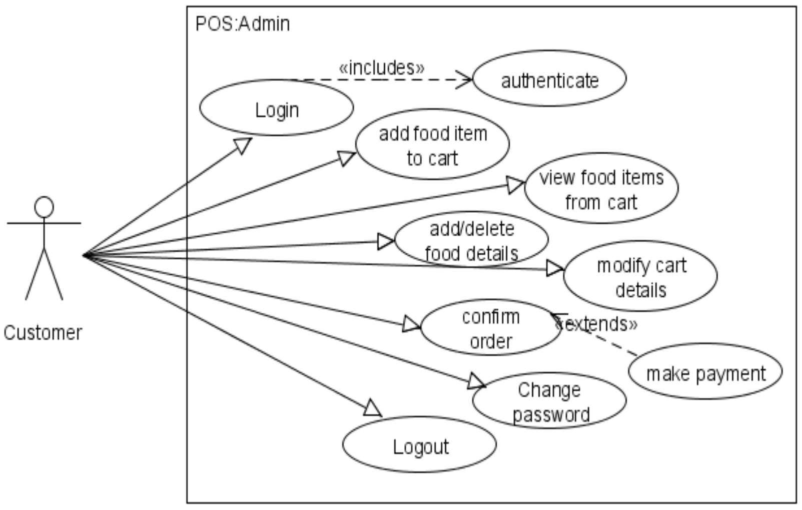
3.1 Use Case Diagrams

The requirements of a system can be represented using a use case model in the Use Case Diagram. The use case diagrams for the actors of this case study are given as below.

3.1.1 Use Case Diagram for Administrator



3.1.2 Use Case Diagram for Customer



3.2 Use case Definition

Generally, In a design document, Use case definitions should be written for all the *Requirements* of the system. Below shown is an example for Use Case Definition.

3.2.1 Add Food Item Details

Admin performing add food item fuctionality

USE CASE #	AD-005	
Goal	*The admin should be able to add food items details*	
Preconditions	*Admin should be successfully logged in.*	
Success End Condition	*Food item details are added successfully and a FoodID is auto-generated.*	
Failed End Condition	*Not able to add a food item and a proper message is shown.*	
Primary, Secondary Actors	*Primary: Administrator*	
Trigger	*Admin Initiative*	
DESCRIPTION	**Step**	**Action**
	1	*Click on 'Add Food Item' link.*
	2	*Enter valid Food Item details*
	3	*Click on 'Add FoodItem' button*
	Step	**Branching Action**
	1	*Enter invalid/incomplete item details*
	2	*Click on 'Add FoodItem' button*
Related Information/Use cases		
Priority	*P1*	
Performance	*1sec (excluding user input)*	
Frequency	*2 / week*	
Assumptions	*All Pizza Stores will have the same food items.*	

3.2.2 Modify Food Item Details

Administrators performing modify food item functionality

USE CASE #	*AD-008*	
Goal	*The admin should be able to modify food items details*	
Preconditions	*Food Items are present in the database.*	
Success End Condition	*Food Items details modified successfully. A proper message is shown.*	
Failed End Condition	*Unable to modify Food Item details*	
Primary, Secondary Actors	*Primary-Administrator*	
Trigger	*Admin Initiative*	
5DESCRIPTION	**Step**	**Action**
	1	*Click on 'Edit Food Item Details' link*
	2	*Enter FoodID. Click on Submit. Food details are shown.*
	3	*Enter new Food Item details.*
	4	*Click on 'Modify FoodItem' button.*
	5	*On the Confirmation box, Select OK.*
	Step	**Branching Action**
	1	*Enter invalid FoodID. Click on Submit.*
	2	*Select Cancle on confirmation.*
Related Information/Use cases		
Priority	*P2*	
Performance	*1sec (excluding user interaction)*	
Frequency	*1 / week*	
Assumptions	*The updation affects all pizza stores.*	

3.2.3 Confirm Order and Make Payment

Customer performing Order Confirmation and Make payment functionality

USE CASE #	*CU-004*	
Goal	The user should be able to confirm order by making payment	
Preconditions	*Food Items should be available in shopping cart.*	
Success End Condition	*OrderID should be auto-generated and shipment details should be entered in the database.*	
Failed End Condition	*No order generated. Shipment details also not stored in the database.*	
Primary, Secondary Actors	*Primary-Customer*	
Trigger	*User Initiative.*	
DESCRIPTION	**Step**	**Action**
	1	*Click on 'Confirm Order' button on the shopping cart page.*
	2	*Enter the shipment details.*
	3	*Click on 'Make Payment' button.*
	4	*Enter Credit Card details. Click on Pay.*
	Step	**Branching Action**
	1	*Enter incomplete shipment details*
	2	*Or, Enter invalid Credit Card details*
	3	*Or, Purchase worth more than Credit Balance.*
Related Information/Use cases		
Priority	*P1*	
Performance	*2 sec*	
Frequency	*30/day per Customer*	
Assumptions	*Credit Card details of the customer must be available in the database.*	

3.3 Class Diagram

The class diagram is a very basic concept in object-oriented world. Class diagrams demonstrate a model, describing what attributes and behavior it has rather than describing the methods for accomplishing operations. Class diagrams are very useful in representing relationships between classes and interfaces. The below example is for a Web Application using JSP/Servlet.

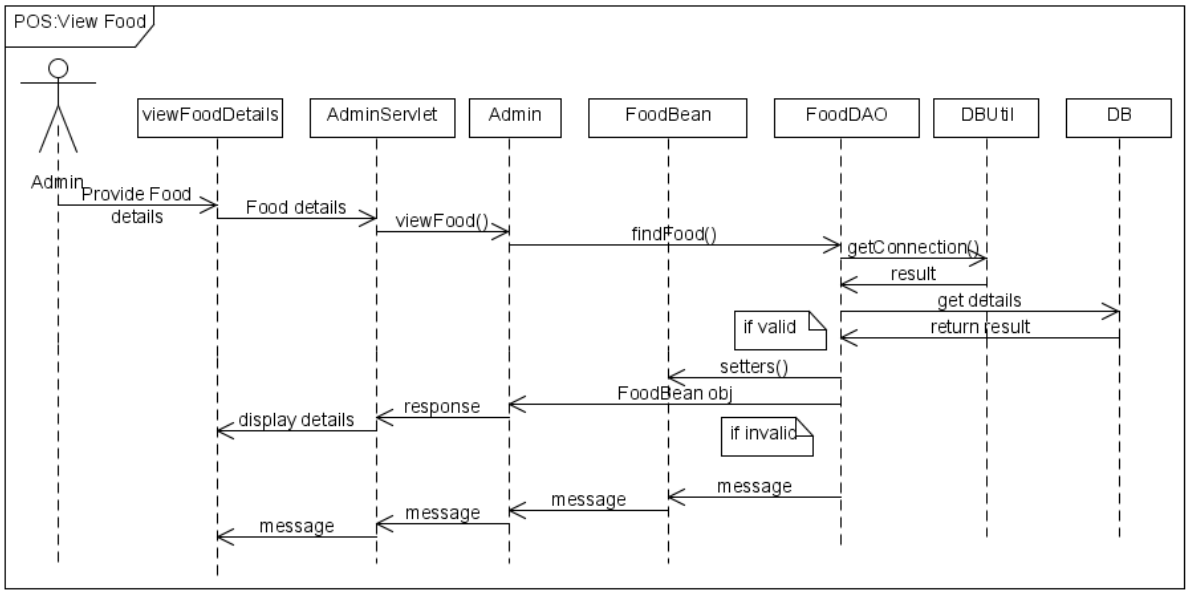
<to be designed by the participant………..>

3.4 Sequence Diagram

A graphical representation of a module's function invoking functions of other modules in order to achieve a task (specific user requirement) is called a sequence diagram. A sequence diagram for the authentication process is given below for reference. The below example is for a Web Application using servlets/jsp.

3.4.1 View Food details

Admin performing view food details



3.5 Packages / Interface / Classes

This section provides a brief outlook on the packaging hierarchy along with the respective classes to be used for the implementation.

The 4 packages mentioned below are for both standalone and Web Application.

Packages	
Package	**Description**
com.wipro.pos.service	This package contains all the Service Classes
com.wipro.pos.bean	This package contains all the bean Classes
com.wipro.pos.dao	This package contains all the DAO functionality classes
com.wipro.pos.util	This package contains all the generic functionality classes

This package is used only for a standalone application.

com.wipro.pos.ui	This package contains all the UI related classes [For Core Java]

The package for the controller class should be used as below based on the type of application

com.wipro.pos.listener	**listener -** core java
or	
com.wipro.pos.servlet	**servlet -** Web Applications
or	
com.wipro.pos.action	**action -** Struts
or	
com.wipro.pos.controller	**controller -** Spring

Package com.wipro.pos.bean

Class Name	Attributes	Data Type
ProfileBean	userID	String
	firstName	String
	lastName	String
	dateOfBirth	Date
	gender	String
	street	String
	location	String
	city	String
	state	String
	pincode	String
	mobileNo	String
	emailID	String
	password	String

CredentialsBean	userID	String
	password	String
	userType	String
	loginStatus	int
StoreBean	storeID	String
	name	String
	street	String
	mobileNo	String
	city	String
	state	String
	pincode	String
FoodBean	foodID	String
	name	String
	type	String
	foodSize	String
	quantity	int
	price	double
OrderBean	orderID	String
	userID	String
	orderDate	Date
	storeID	String
	cartID	int
	totalPrice	double
	orderStatus	String
	street	String
	city	String
	state	String
	pincode	String
	mobileNo	String
CartBean	cartID	int
	userID	String
	foodID	String
	type	String
	quantity	int
	cost	double
	orderDate	Date

Package com.wipro.pos.service

Interface Summary	
Interface	**Description**
Administrator	Entity class for Admin dealing with the admin process functionalities
	Methods
	String **addStore**(**StoreBean** storebean) Return value must be either: "SUCCESS", "FAIL", “ERROR”
	boolean **modifyStore**(**StoreBean** storebean)
	Int **removeStore**(ArrayList<String> storeId)
	StoreBean **viewStore**(String storeId)
	ArrayList <StoreBean> **viewAllStore**()
	String **addFood**(**FoodBean** foodbean) Return value must be either: "SUCCESS", "FAIL", “ERROR”
	boolean **modifyFood**(**FoodBean** foodbean)
	boolean **removeFood**(String storeId, String foodId)
	FoodBean viewFood(String foodId)
	ArrayList<FoodBean> **viewAllFood**(String storeId)
	String **changeOrderStatus**(String orderId) Return value must be either: "SUCCESS", "FAIL"
Customer	Entity class of Customer for dealing with the Customer process functionalities
	Methods
	int **addToCart**(CartBean cartBean)
	boolean **modifyCart**(CartBean cartBean)
	String **confirmOrder**(OrderBean orderBean, ArrayList<CartBean> cartbean)
	String **cancelOrder**(String orderId)
	ArrayList<StoreBean> viewStore(String city)
	ArrayList<CartBean> **viewCart**(String userid)
	ArrayList <OrderBean> **viewOrder**()

Package com.wipro.pos.dao

Find below the suggestive approach for CRUD operations [method naming & signature] for the DAO classes. Create the necessary DAO Interface/classes .

Interface/class Summary	
Interface	**Description**
xyzDAO	DAO interface to deal with operations related to the specific table.
	Method Summary
	String createXYZ(BeanObject)
	int deleteXYZ(ArrayList<String>)
	boolean updateXYZ(BeanObject)
	BeanObject findByID(String)
	ArrayList<BeanObject> findAll()

- If required, additional find methods can be created.

Package com.wipro.pos.util

Interface/class Summary	
Interface	**Description**
Authentication	This interface is responsible for performing the Authentication and Authorization process.
	Methods
	boolean **authenticate**(CredentialsBean credentialsBean)
	String **authorize**(String userId)
	boolean **changeLoginStatus**(CredentialsBean credentialsBean, int loginStatus)
DBUtil	This interface is responsible for the Database connection establishment.
	Methods
	static Connection **getDBConnection**(String driverType)
User	Interface for handling different types of users
	Methods
	String **login**(CredentialsBean credentialsBean) Return value must be either: "A", "C", "FAIL", “INVALID” A->Admin, C->Customer **Wrong username/password should return INVALID.**
	boolean **logout**(String userId)
	String **changePassword**(CredentialsBean credentialsBean, String newPassword) Return value must be either: "SUCCESS", "FAIL", “INVALID”
	String **register**(ProfileBean profileBean) Return value must be either: <userId of lenght 6>, "FAIL" Note: userId-> first 2 letter of first name followed by 4 digit auto generated number

Payment	Interface for handling payment related information String creditCardNumber, validFrom, validTo int balance
	Methods boolean findByCardNumber(String userid, String cardnumber) String **process**(Payment payment)

3.6 UI Templates

3.6.1 UI Principle

The UI [Presentation Layer] should be designed with the below mentioned principles which helps easy interaction by the user to the application.

3.6.2 UI controls and Usage Principle

Provides the information on UI Controls, which type of control should be used when and where.

UI Type	Controls	Description
Direct Entry	Text Box, Text Area	Any input that cannot be predicted and needs the user to key in. e.g. Name, Street, contact no etc.
Static Selection	Option Button, Check Box, Drop Down	Should be used where the input can be predefined. e.g. gender, month [Jan – Dec] etc. If number of items is more, drop down is preferred.
Dynamic Selection	Drop Down	The items for the drop down should be retrieved from a stored data. e.g. Displaying BranchId in a drop down from branch table.
Automation	Label Text Field [Read Only]	Data that are calculative or an output of a function. e.g. : Displaying system date, showing total amount etc.
Decision Control	Button	Operations like submit, save, clear should be executed only upon clicking respective buttons.

3.6.3 UI Template

This section contains the design template for the website home page [Fig. 1] that will be displayed at the time of opening this web application and Actor specific home page [Fig. 2].

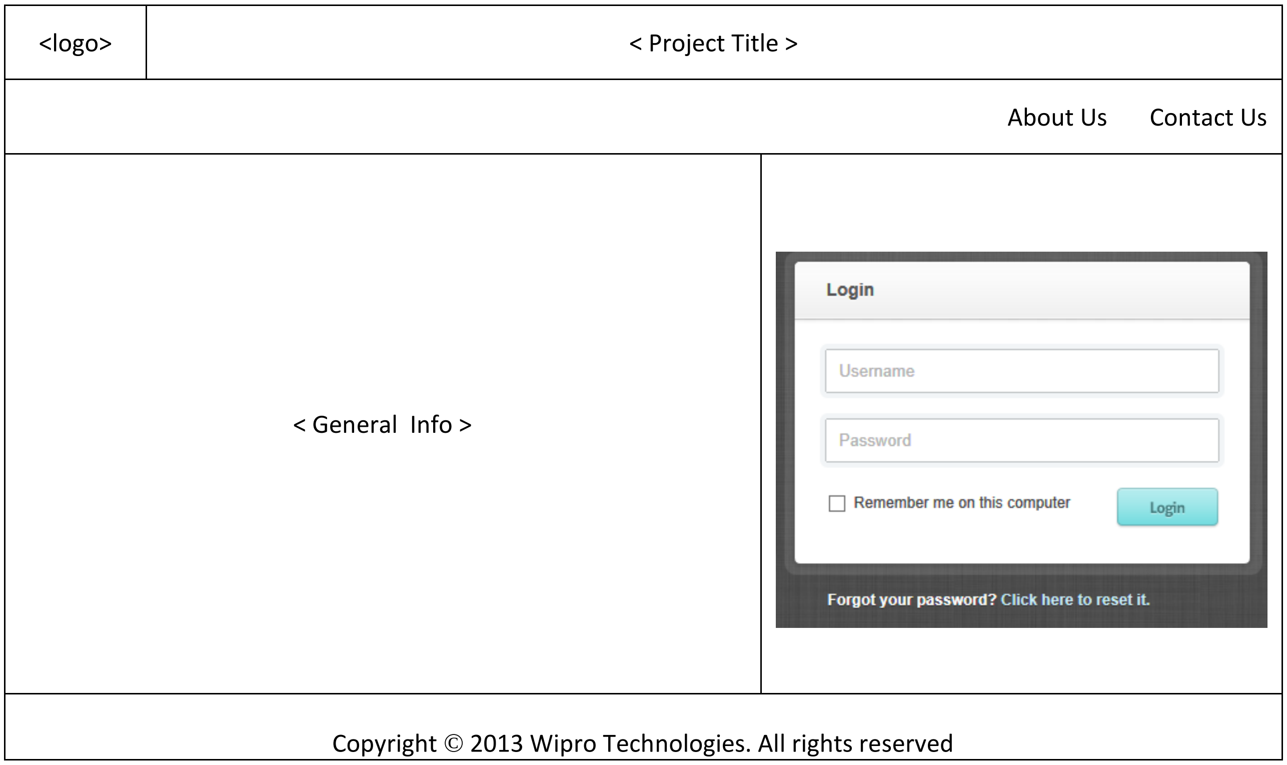

<logo>
< Project Title >
About Us
Contact Us
< General Info >
Login
Username
Password
Remember me on this computer
Login
Forgot your password? Click here to reset it.
Copyright © 2013 Wipro Technologies. All rights reserved

Fig. 1 - Main Page [First Page to open]

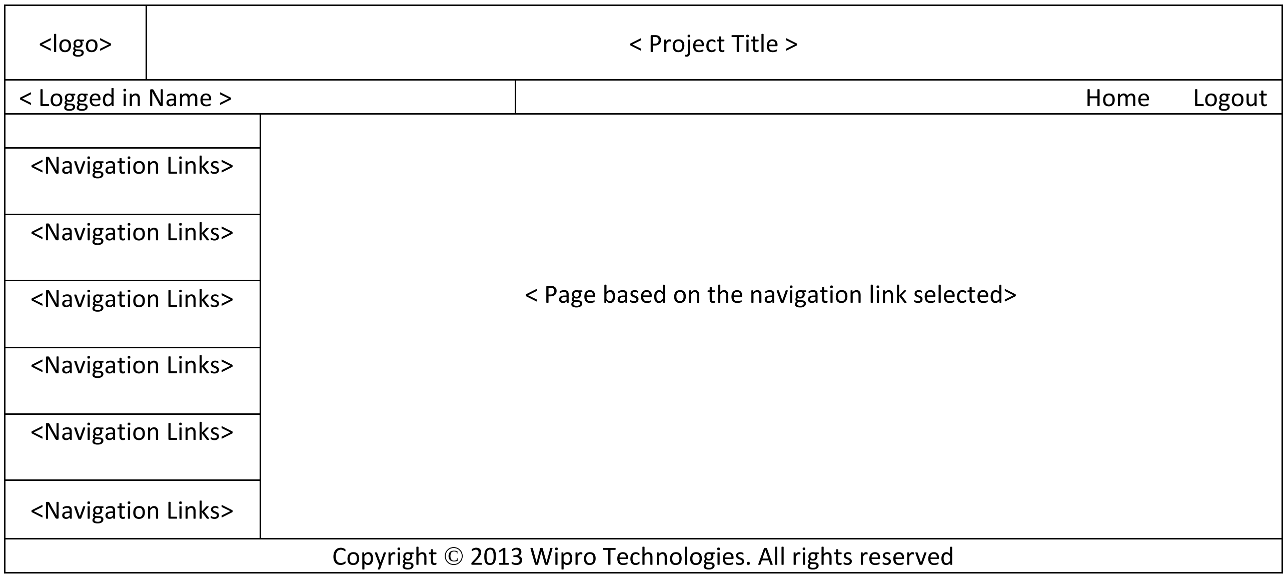
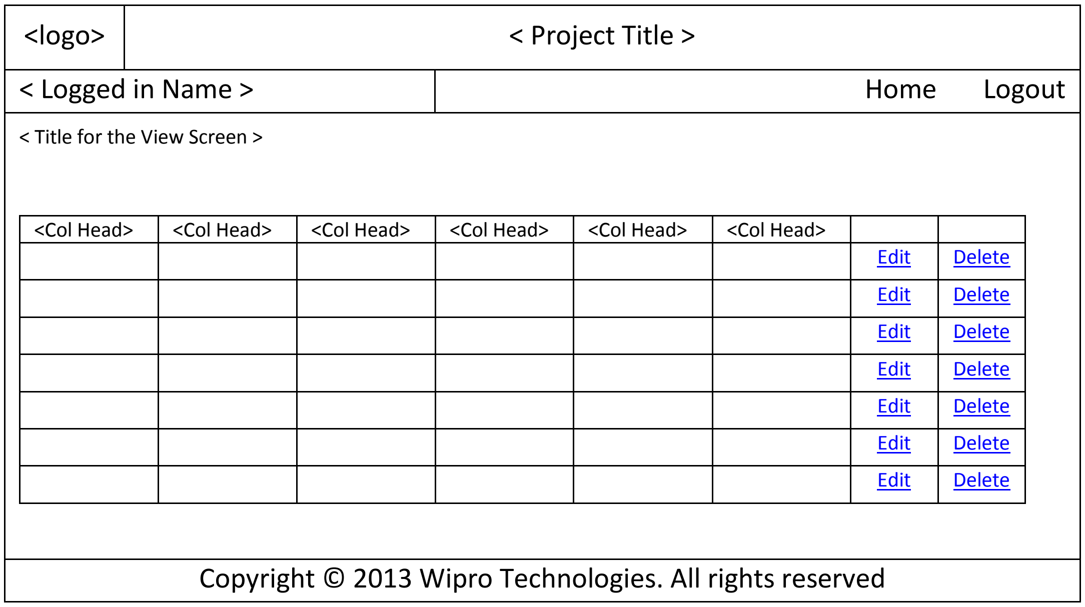


Fig. 2 - Home Page for Actor



<logo> < Project Title >

< Logged in Name > Home Logout

< Title for the View Screen >

<Col Head>	<Col Head>	<Col Head>	<Col Head>	<Col Head>	<Col Head>		
						Edit	Delete
						Edit	Delete
						Edit	Delete
						Edit	Delete
						Edit	Delete
						Edit	Delete
						Edit	Delete

Copyright © 2013 Wipro Technologies. All rights reserved

Fig. 3 – View Screen with Edit and Delete Functionality

4. Critical Functions and Focus for Testing

Authorization & Authentication are the critical functions need to be implemented before performing the tasks.

5. Limitations

- The scope of the application is limited to only one country.
- Users should be registered or logged in for performing Order related operations.
- Customer’s credit card details have to be already entered in the database.
- The number of administrators is limited to 2.

6. APPENDIX

1 Table: POS_TBL_User_Profile

This table contains User specific details entered during User Registration.

Field Name	Data Type	Description
Userid	VARCHAR2(6)	Auto-Generated, Primary Key*
Firstname	VARCHAR2(20)	Not Null
Lastname	VARCHAR2(20)	Not Null
DateOfBirth	DATE	Not Null
Gender	VARCHAR2(7)	Not Null
Street	VARCHAR2(30)	Not Null
Location	VARCHAR2(20)	Not Null
City	VARCHAR2(20)	Not Null
State	VARCHAR2(20)	Not Null
PinCode	VARCHAR2(10)	
MobileNo	VARCHAR2(10)	Exact 10 digit only
EmailId	VARCHAR2(30)	Not Null

* First 2 letters of First Name followed by 4 digits auto generated number

2 Table: POS_TBL_User_Credentials

This table contains Authentication Information for Administrator and Customer

Field Name	Data Type	Description
Userid	VARCHAR2(6)	Primary Key
Password	VARCHAR2(20)	Not Null
Usertype	VARCHAR2(15)	Either ['A','C']
Loginstatus	NUMBER(1)	Either[1 ,0]

3 Table: POS_TBL_PizzaStore

This table contains Pizza Store information added by the Admin.

Field Name	Data Type	Description
StoreId	VARCHAR2(6)	Primary Key, Auto Generated*
Name	VARCHAR2(15)	Not Null
Street	VARCHAR2(50)	Not Null
MobileNo	VARCHAR2(10)	Must be 10-digit exactly
City	VARCHAR2(15)	Not Null
State	VARCHAR2(15)	Not Null
Pincode	VARCHAR2(10)	

* First 2 letters of Store name followed by 4 digits auto generated number

4 Table: POS_TBL_Food

This table contains Food Item details as defined below.

Field Name	Data Type	Description
FoodId	VARCHAR(6)	Primary Key, Auto Generated**
Name	VARCHAR(20)	Not Null
Type	VARCHAR2(10)	Either [Veg | Non-Veg]
FoodSize	VARCHAR2(15)	Either [Small | Medium | Large]
Quantity	NUMBER	Not Null
Price	NUMBER	Not Null

** First 2 letters of Food name followed by 4 digits auto generated number

5 Table: POS_TBL_Order

This table contains Order details provided by the customer.

Field Name	Data Type	Description
OrderId	VARCHAR2(6)	Primary Key
Userid	VARCHAR2(6)	FK
OrderDate	DATE	Sysdate
StoreId	VARCHAR2(6)	FK
TotalPrice	NUMBER	Not Null, >0
OrderStatus	VARCHAR(15)	Confirmed, Delivered, Pending, Cancelled
cartId	NUMBER	
Street	VARCHAR(50)	Not Null
City	VARCHAR(15)	Not Null
State	VARCHAR(15)	Not Null
PinCode	VARCHAR(10)	
MobileNo	VARCHAR(10)	

6 Table: POS_TBL_Cart

This table contains cart details added by the customer.

Field Name	Data Type	Description
CartId	NUMBER	
UserId	VARCHAR2(6)	FK
FoodId	VARCHAR2(6)	FK
Type	VARCHAR2(10)	Not Null [Veg | Non Veg]
quantity	NUMBER	Not Null, >0
Cost	NUMBER	Not Null
OrderDate	Date	

7 Table: POS_TBL_CreditCard

This table contains CreditCard details of the user for ticket reservation.

Field Name	Data Type	Description
CreditCardNumber	VARCHAR2(16)	Primary Key
ValidFrom	VARCHAR2(7)	Not Null
ValidTo	VARCHAR2(7)	Not Null
Balance	NUMBER	Not Null
userId	VARCHAR2(6)	

Database Sequences

Sequence Name	Purpose	Start With
pos_seq_userid	User ID	1000
pos_seq_storeId	Store ID	1000
pos_seq_foodId	Food ID	1000
pos_seq_orderId	Order ID	1000
pos_seq_cartId	Cart ID	1000