WIPRO
Applying Thought

DETAILED DESIGN

Project Code:	OCS-01
Project Name:	Online Clinic 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	30/04/2013	Revision			
2.10	10/12/2013	Revision	Mapping with CPC		
2.2	12/06/2014	Revision	Mapping with UCF		

Affected Groups

List of Reference Documents

Name	Version No.
1.RS_OCS	2.20
2.FS_OCS	2.20
3.	
4.	

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

Table of Contents

1. Introduction

1.1 Background

XYZ Clinic provides health service facilities to patients across many cities.

1.2 Purpose

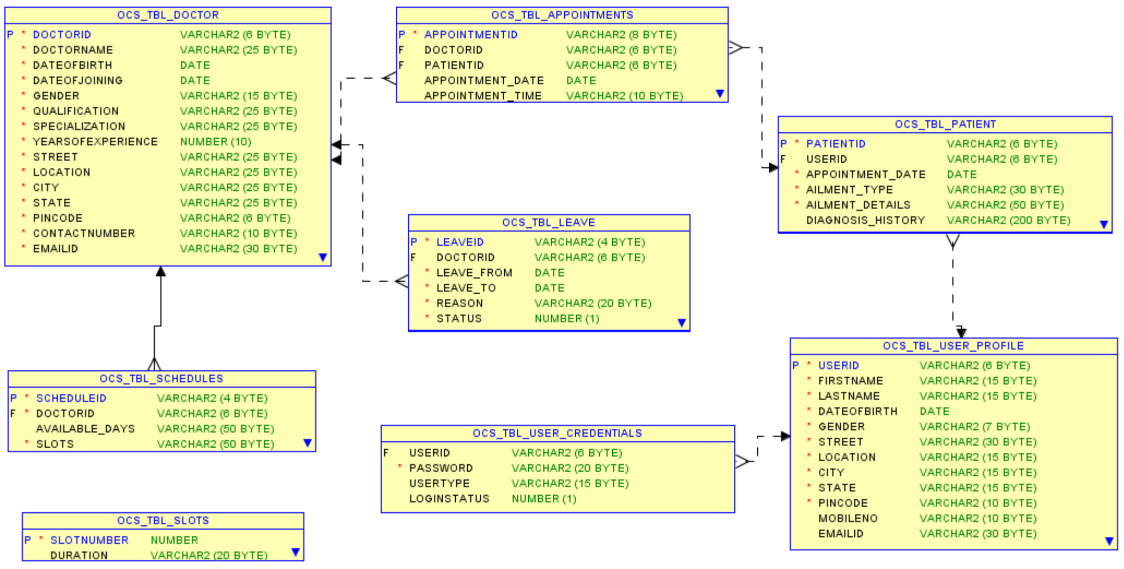
XYZ Clinic plans to develop “Online Clinic System” - where a patient can fix up an appointment with a Doctor over the network, record his/her ailments and choose a Specialized Doctor depending on his/her ailments or health complaints.

1.3 Scope

The scope of the Online Clinic System (OCS) will be to provide the functionality as described below. The system will be developed on a Windows operating system using Java/J2EE and Oracle database.

2. Global Data Structures and Shared Data Functions

This section describes the structure of 8 tables to be used for the implementation of requirements as stated in the specification.


OCS_TBL_DOCTOR
P * DOCTORID VARCHAR2 (6 BYTE)
* DOCTORNAME VARCHAR2 (25 BYTE)
* DATEOFBIRTH DATE
* DATEOFJOINING DATE
* GENDER VARCHAR2 (15 BYTE)
* QUALIFICATION VARCHAR2 (25 BYTE)
* SPECIALIZATION VARCHAR2 (25 BYTE)
* YEARSOFEXPERIENCE NUMBER (10)
* STREET VARCHAR2 (25 BYTE)
* LOCATION VARCHAR2 (25 BYTE)
* CITY VARCHAR2 (25 BYTE)
* STATE VARCHAR2 (25 BYTE)
* PINCODE VARCHAR2 (6 BYTE)
* CONTACTNUMBER VARCHAR2 (10 BYTE)
* EMAILID VARCHAR2 (30 BYTE)
OCS_TBL_APPOINTMENTS
P * APPOINTMENTID VARCHAR2 (8 BYTE)
F DOCTORID VARCHAR2 (6 BYTE)
F PATIENTID VARCHAR2 (6 BYTE)
APPOINTMENT_DATE DATE
APPOINTMENT_TIME VARCHAR2 (10 BYTE)
OCS_TBL_PATIENT
P * PATIENTID VARCHAR2 (6 BYTE)
F USERID VARCHAR2 (6 BYTE)
* APPOINTMENT_DATE DATE
* AILMENT_TYPE VARCHAR2 (30 BYTE)
* AILMENT_DETAILS VARCHAR2 (50 BYTE)
DIAGNOSIS_HISTORY VARCHAR2 (200 BYTE)
OCS_TBL_LEAVE
P * LEAVEID VARCHAR2 (4 BYTE)
F DOCTORID VARCHAR2 (6 BYTE)
* LEAVE_FROM DATE
* LEAVE_TO DATE
* REASON VARCHAR2 (20 BYTE)
* STATUS NUMBER (1)
OCS_TBL_SCHEDULES
P * SCHEDULEID VARCHAR2 (4 BYTE)
F * DOCTORID VARCHAR2 (6 BYTE)
AVAILABLE_DAYS VARCHAR2 (50 BYTE)
* SLOTS VARCHAR2 (50 BYTE)
OCS_TBL_USER_CREDENTIALS
F USERID VARCHAR2 (6 BYTE)
* PASSWORD VARCHAR2 (20 BYTE)
USERTYPE VARCHAR2 (15 BYTE)
LOGINSTATUS NUMBER (1)
OCS_TBL_USER_PROFILE
P * USERID VARCHAR2 (6 BYTE)
* FIRSTNAME VARCHAR2 (15 BYTE)
* LASTNAME VARCHAR2 (15 BYTE)
* DATEOFBIRTH DATE
* GENDER VARCHAR2 (7 BYTE)
* STREET VARCHAR2 (30 BYTE)
* LOCATION VARCHAR2 (15 BYTE)
* CITY VARCHAR2 (15 BYTE)
* STATE VARCHAR2 (15 BYTE)
* PINCODE VARCHAR2 (10 BYTE)
MOBILENO VARCHAR2 (10 BYTE)
EMAILID VARCHAR2 (30 BYTE)
OCS_TBL_SLOTS
P * SLOTNUMBER NUMBER
DURATION VARCHAR2 (20 BYTE)

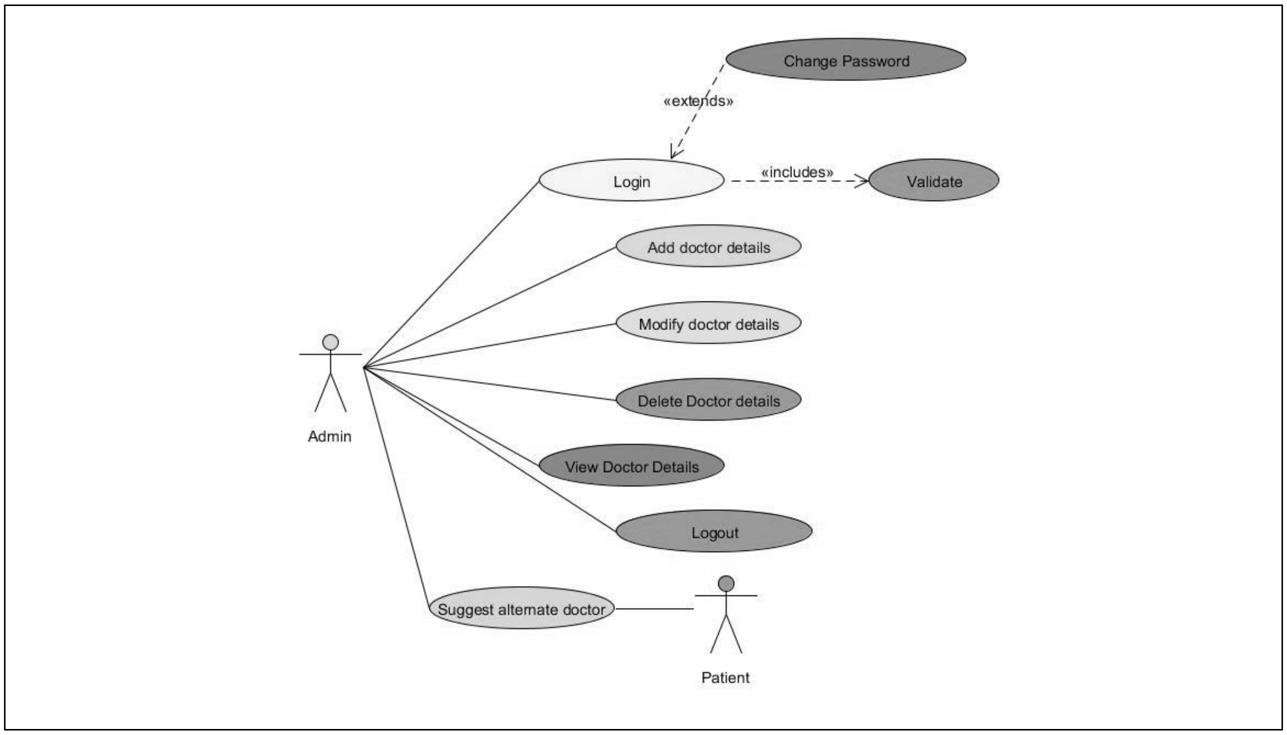
3. High Level Design

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.

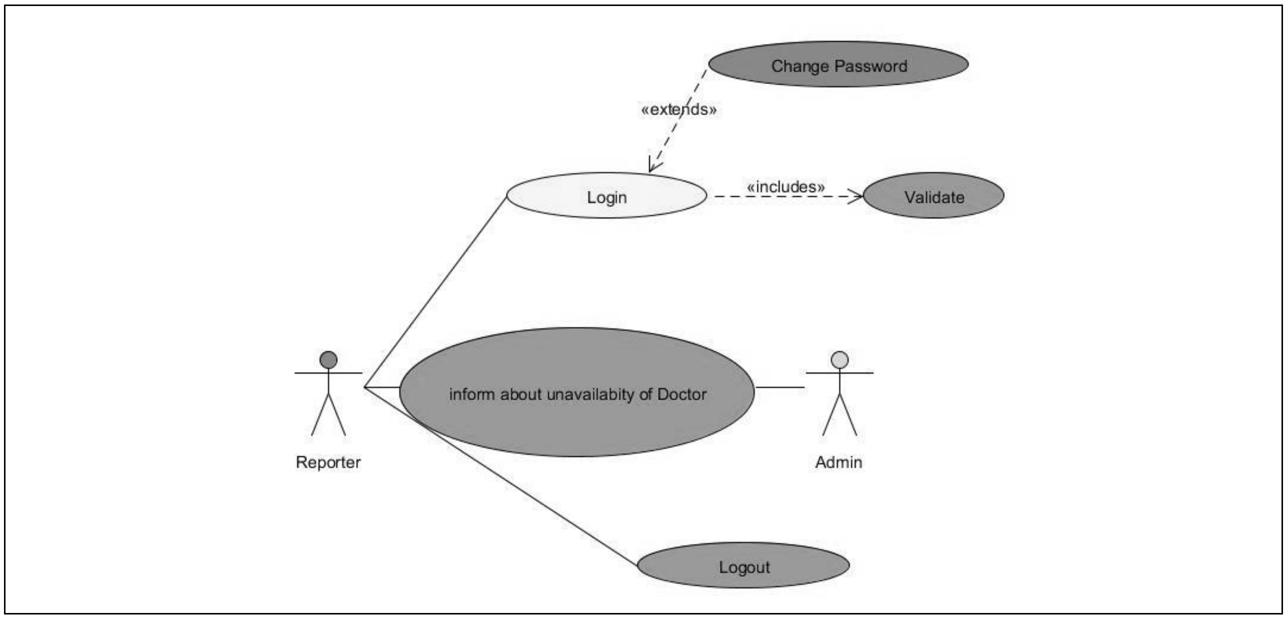
3.1 Use Case Diagrams

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

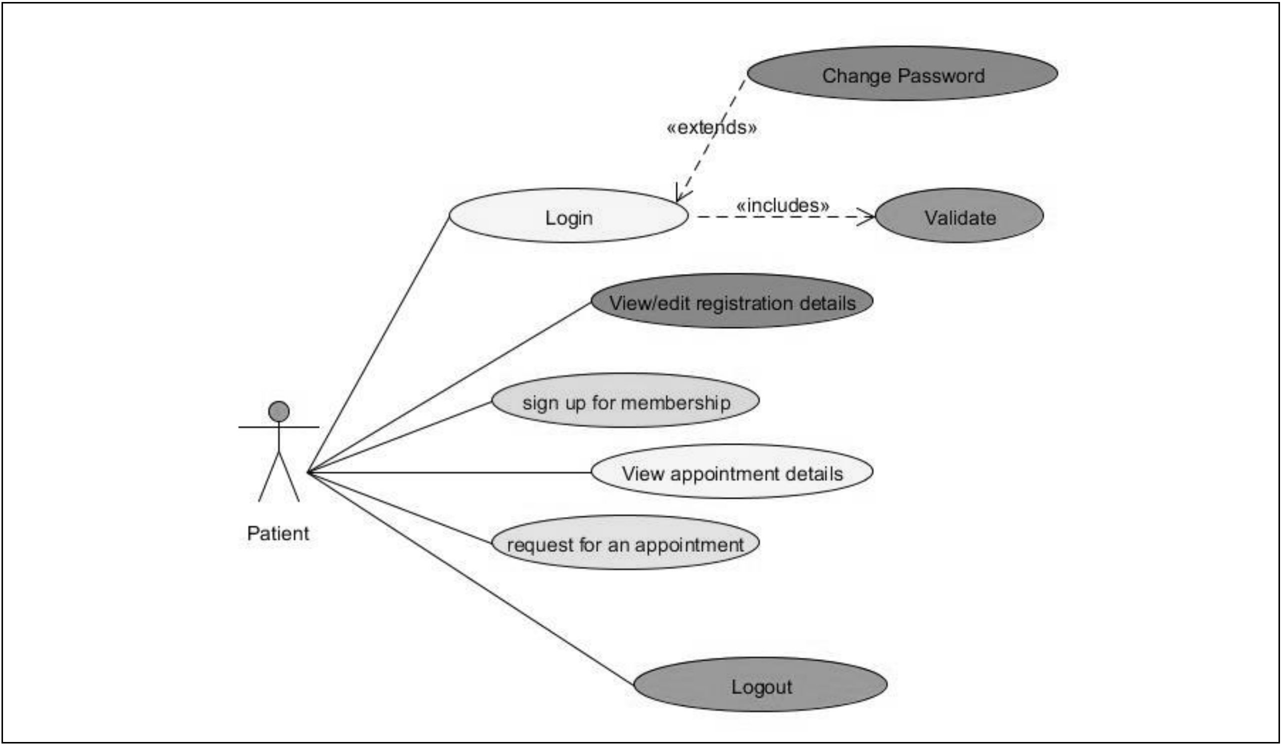
3.1.1 Use Case Diagram for Admin



3.1.2 Use Case Diagram for Reporter



3.1.3 Use Case Diagram for Patient



3.2 Use case Definition

Generally, in a design document, Use case definitions should be written for all the *Requirements* of the system.

Note: Participants are expected to document use case definitions for all requirements. However, for few requirements documented below for reference.

Below table explains ‘Use Case’ definition for requirement “AA-001” - Login operation for all users.

3.2.1 Login

USE CASE #	AA-001 *Login*	
Goal	*All users logging into the system should be authenticated using a unique login-id and password (operations to be supported based on type of user)*	
Preconditions	If the user type is 'Admin' or 'Reporter', credential details should exist. If the user type is 'Patient', he/she should be registered.	
Success End Condition	If the user type is 'Admin', then redirect to the Admin page. If the user type is 'Reporter', then redirect to the Reporter page. If the user type is 'Patient', then redirect to the Patient page.	
Failed End Condition	*The end user is redirected to an Error Page, and/or is asked to re-enter login credentials.*	
Primary, Secondary Actors	Admin, Reporter, Patient	
Trigger	*Login button*	
DESCRIPTION	**Step**	**Action**
	1	*Enter Login credentials (id & password)*
	2	*Click on Login button*
	3	*If id & password is Success, then identify user type* *Display appropriate(Admin/Patient) home page*
	Step	**Branching Action**
	1	*If 'id' is not existing then return with requesting for registration*
	2	*If password is not matching return with suitable error message say 'reenter id & password'*
Related Information/Use cases	*Not Applicable*	
Priority	*P1*	
Performance	*Approx. in seconds*	
Frequency	*Approx. few users per minute*	
Assumptions	Admin login credentials are available in the database and others are already registered with their credentials	

Below table explains ‘Use Case’ definition for requirement “AD-001” – ADD Doctor Details operation for Admin user only.

3.2.2 ADD Doctor Details

USE CASE #	*AD-001: Add Doctor Details*	
Goal	*“Admin” can add new Doctor.*	
Preconditions	*“Doctor” should not exist.*	
Success End Condition	*Doctor details get added. An appropriate success message is displayed. (“Success: successfully added”)*	
Failed End Condition	*Can’t be completed (“Failure: Null or wrong datatype”)*	
Primary, Secondary Actors	*Primary- Admin*	
Trigger	*Whenever Admin need to add a new doctor details.*	
DESCRIPTION	***Step***	***Action***
	1	*The Admin selects Add doctor option*
	2	*Admin enters new doctor details.*
	3	*System accepts the doctor details and adds it into the database and displays doctor_ID.*
	Step	***Branching Action***
	1	*If he enters wrong input or if it is null. Displays error message.*
	2	*If he enters correct input it returns success.*
Related Information/Use cases	*3.1.1*	
Priority	*High*	
Performance	*1.5 minutes (assuming time taken for Admin input).*	
Frequency	*As per requirement.*	
Assumptions	*No field is left blank.*	

3.3 Class Diagram

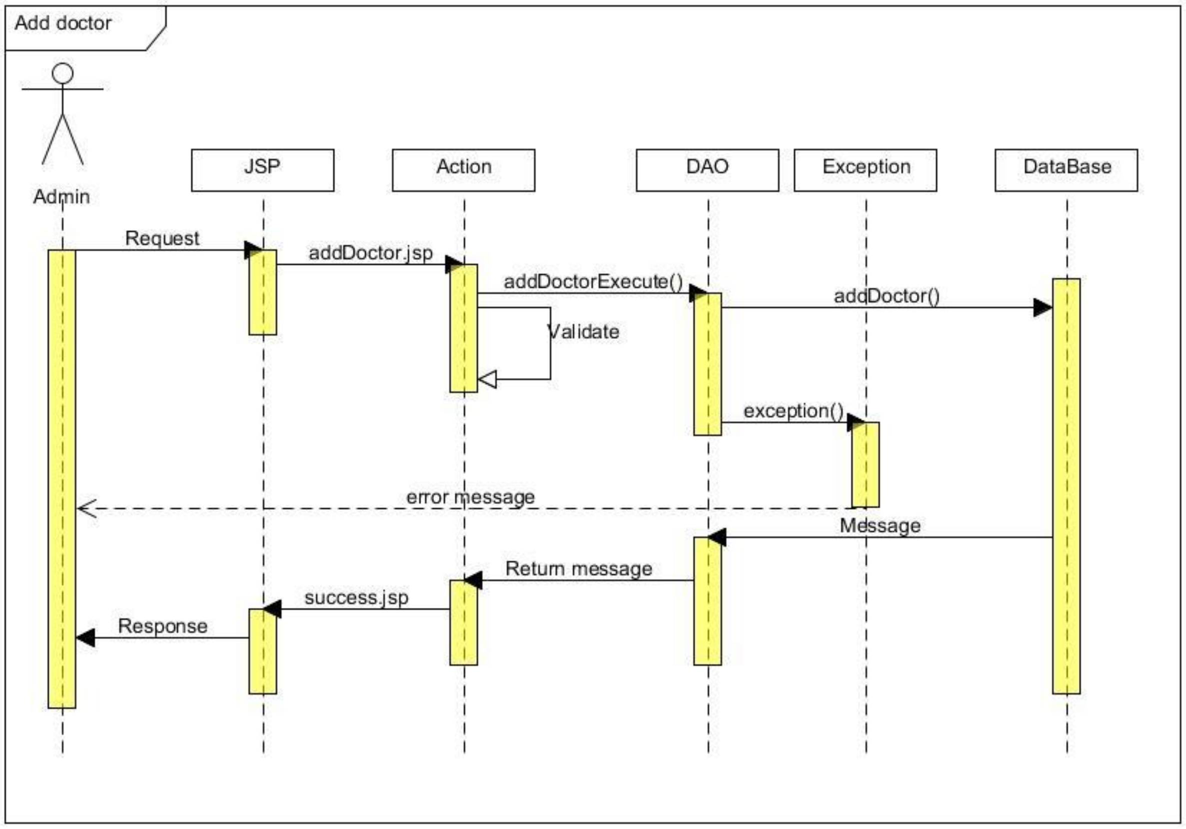
The class diagram is a very basic concept in object-oriented world. Class diagrams demonstrate a model, describing what attributes and behavior it has rather than describing the methods for accomplishing operations. Class diagrams are very useful in representing relationships between classes and interfaces.

<class diagram to be drawn by RLL participants>

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 servlet/jsp.

3.4.1 Add Doctor details:



3.5 Packages / Classes / Interface

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 GUI and Web Application.

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

This package is used only for a GUI application.

com.wipro.ocs.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.ocs.listener **or** **com.wipro.ocs.servlet** **or** **com.wiro.ocs.action** **or** **com.wiro.ocs.controller**	This package contains all the controller classes **listener - core java** **servlet - Web Applications** **action - Struts** **controller - Spring**

Package com.wipro.ocs.bean

Class Name	Attributes	Data Type
CredentialsBean	userID	String
	password	String
	userType	String
	loginStatus	int
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
DoctorBean	doctorID	String
	doctorName	String
	dateOfBirth	Date
	dateOfJoining	Date
	gender	String
	qualification	String
	specialization	String
	yearsOfExperience	int
	street	String
	location	String
	city	String
	state	String
	pincode	String
	contactNumber	String
	emailID	String

LeaveBean	doctorID	String
	leaveFrom	Date
	leaveTo	Date
	reason	String
	status	int

PatientBean	patientID	String
	userID	String
	appointmentDate	Date
	ailmentType	String
	ailmentDetails	String
	diagnosisHistory	String

AppointmentBean	appointmentID	String
	doctorID	String
	patientID	String
	appointmentDate	Date
	appointmentTime	String

ScheduleBean	scheduleID	String
	doctorID	String
	availableDays	String
	slots	String

Package com.wipro.ocs.service

Interface Summary	
Interface	**Description**
Administrator	Entity class for Administrator dealing with the admin process functionalities **Method Summary** String **addDoctor**(DoctorBean doctoerBean) Return value must be either: "SUCCESS", "FAIL" Boolean **modifyDoctor**(DoctorBean doctorBean) ArrayList<DoctorBean> **viewAllDoctors**() int **removeDoctor**(String doctorID) ArrayList<DoctorBean> **suggestDoctors(**String patientId, Date date) Map <PatientBean, AppointmentBean> **viewPatientsByDate**(Date appointmentDate) Note: Suitable data to be returned from Patient and Appointment tables
Reporter	Entity class for Reporter for dealing with the Reporter process functionalities **Method Summary** ArrayList<DoctorBean> **viewAllAvailableDoctors**(Date date) ArrayList<DoctorBean> **intimateAdmin**(Date date, String status)
Patient	Entity class for Patient dealing with the patient process functionalities **Method Summary** String **addAilmentDetails**(PatientBean patientBean) Return value must be either: "SUCCESS", "FAIL" boolean **modifyAilmentDetails**(PatientBean patientBean) ArrayList<PatientBean> **viewAilmentDetails**(String patientID) ArrayList<DoctorBean> **viewListOfDoctors**(String specialization, Date date) String **requestforAppointment**(String doctorID, Date appointmentDate) Return value must be either: "CONFIRMED", "PENDING', "FAIL" Map<AppointmentBean, PatientBean> **viewAppointmentDetails**(String patientID, Date date)

Package com.wipro.ocs.dao

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

Interface Name	Description
xyzDAO	DAO interface/class 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.ocs.util

Interface Summary	
Interface	**Description**
Authentication	This class is responsible for performing the Authentication and Authorization process.
	Methods
	boolean **authenticate**(**CredentialsBean** user)
	String **authorize**(String userId)
	boolean **changeLoginStatus**(**CredentialsBean** user, int loginStatus)
DBUtil	This class is responsible for the Database connection establishment.
	Methods
	static Connection **getDBConnection**(String driverType)
User	Entity class for handing different types of users
	Method Summary
	String **login**(**CredentialsBean** credentialsBean) Return value must be either: "A", "P","R", "FAIL", "INVALID" A->Admin, R->Reporter , P-> Patient **Wrong username/password should return INVALID.**
	boolean **logout**(String userId)
	String **changePassword**(**CredentialsBean**, String) 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

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

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, Address, 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 Districts in a drop down from places table.
Automation	Label Text Field [Read Only]	Data's 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].

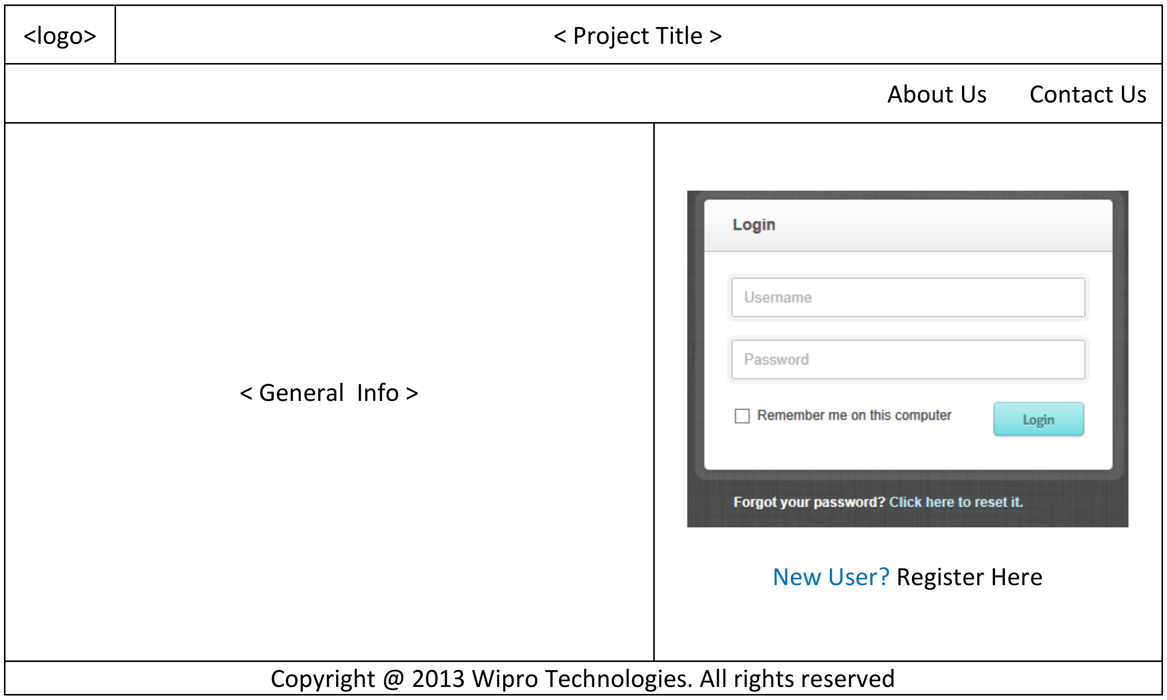
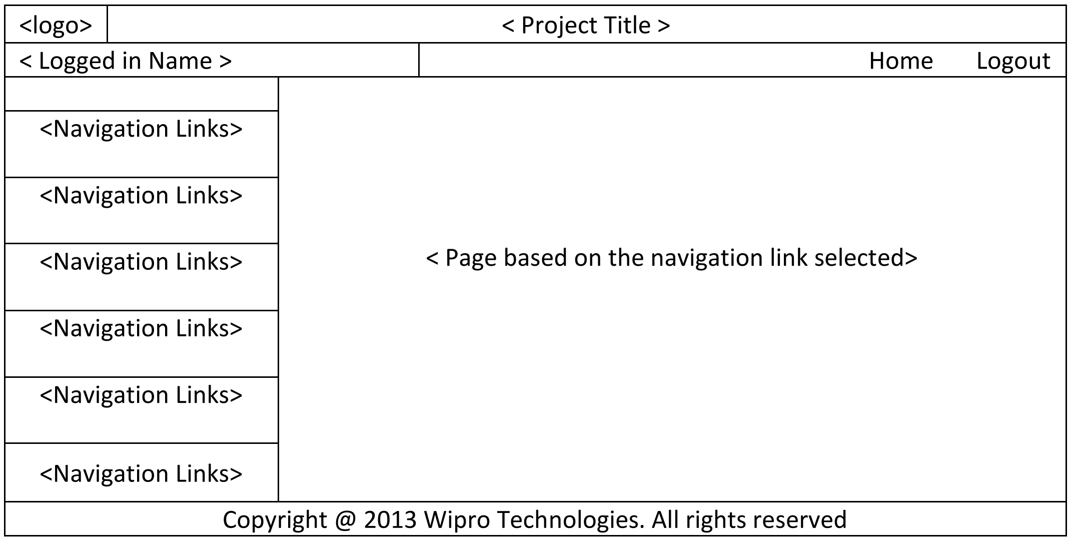

<logo>
< Project Title >
About Us
Contact Us
< General Info >
Login
Username
Password
Remember me on this computer
Login
Forgot your password? Click here to reset it.
New User? Register Here
Copyright @ 2013 Wipro Technologies. All rights reserved

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



<logo> < Project Title >

< Logged in Name > Home Logout

<Navigation Links>

<Navigation Links>

<Navigation Links>

<Navigation Links>

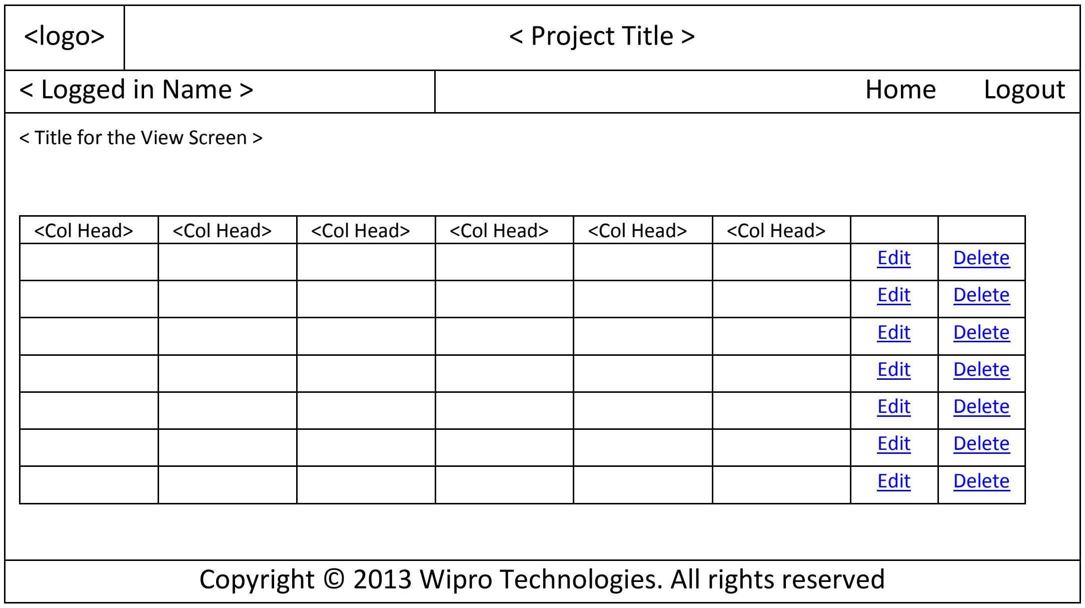
<Navigation Links>

<Navigation Links>

< Page based on the navigation link selected>

Copyright @ 2013 Wipro Technologies. All rights reserved

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 clinic.
- Each patient will hold only one Appointment ID for one successful appointment.
- The reporters are assumed to be already registered with the system.
- Doctor is fixed by the admin based on the availability of a doctor. Every appointment should be of 15mins duration.
- Doctors may attend patients from 9am to 12pm and from 2pm to 5pm.

6. APPENDIX

1. Table: OCS_TBL_User_Credentials

This table contains Authentication Information for Administrator, Reporter & Patient

Field Name	Data Type	Description
USERID	VARCHAR2(6)	Primary Key*
PASSWORD	VARCHAR2(20)	Not Null
USERTYPE	VARCHAR2(1)	Either ['A','P','R']
LOGINSTATUS	NUMBER(1)	Either [1 ,0]

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

2. Table: OCS_TBL_User_Profile

This table contains User specific details entered during User Registration.

Field Name	Data Type	Description
USERID	VARCHAR2(6)	Primary Key*
FIRSTNAME	VARCHAR2(15)	Not Null
LASTNAME	VARCHAR2(15)	Not Null
DATEOFBIRTH	DATE	Not Null
GENDER	VARCHAR2(7)	Not Null
STREET	VARCHAR2(30)	Not Null
LOCATION	VARCHAR2(15)	Not Null
CITY	VARCHAR2(15)	Not Null
STATE	VARCHAR2(15)	Not Null
PINCODE	VARCHAR2(10)	Not Null
MOBILENO	VARCHAR(10)	Exact 10 digit only
EMAILID	VARCHAR2(30)	

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

3. **Table: OCS_TBL_Doctor**

This table contains Doctor specific information, as necessary and applicable.

Field Name	Data Type	Description
DOCTORID	VARCHAR2(6)	Auto Generated Primary Key
DOCTORNAME	VARCHAR2(25)	Not Null
DATEOFBIRTH	DATE	Not Null
DATEOFJOINING	DATE	Not Null
GENDER	VARCHAR2(15)	Not Null
QUALIFICATION	VARCHAR2(25)	Not Null
SPECIALIZATION	VARCHAR2(25)	Not Null
YEARSOFEXPERIENCE	NUMBER	Not Null
STREET	VARCHAR2(25)	
LOCATION	VARCHAR2(25)	Not Null
CITY	VARCHAR2(25)	Not Null
STATE	VARCHAR2(25)	Not Null
PINCODE	VARCHAR2(6)	Not Null
CONTACTNUMBER	VARCHAR2(10)	Exact 10 digit only
EMAILID	VARCHAR2(30)	Not Null

4. **Table: OCS_TBL_Leave**

This table contains Doctor Leave details to be referred by Reporter.

Field Name	Data Type	Description
LEAVEID	VARCHAR2(4)	Auto Generated Primary Key
DOCTORID	VARCHAR2(6)	Foreign Key
LEAVE_FROM	DATE	Not Null
LEAVE_TO	DATE	Not Null
REASON	VARCHAR2(20)	Not Null
STATUS	NUMBER(1)	1 -> Has Appointments 0 -> No Appointments

5. **Table: OCS_TBL_Patient**
   This table contains details of patients registered.

Field Name	Data Type	Description
PATIENTID	VARCHAR2(6)	Auto Generated Primary Key
USERID	VARCHAR2(6)	Foreign Key
APPOINTMENT_DATE	DATE	Not Null
AILMENT_TYPE	VARCHAR2(30)	Not Null
AILMENT_DETAILS	VARCHAR2(50)	Not Null
DIAGNOSIS_HISTORY	VARCHAR2(200)	

6. **Table: OCS_TBL_Appointments**
   This table contains details about various appointments fixed by patients.

Field Name	Data Type	Description
APPOINTMENTID	VARCHAR2(8)	Auto Generated Primary Key
DOCTORID	VARCHAR2(6)	Foreign Key
PATIENTID	VARCHAR2(6)	Foreign Key
APPOINTMENT_DATE	DATE	Not Null
APPOINTMENT_TIME	VARCHAR2(10)	Not Null

- *Appointment ID should be created as MMDD<4digit sequence Number>*

7. **Table: OCS_TBL_Schedules**
   This table contains the details about the available days/slots for each doctors.

Field Name	Data Type	Description
SCHEDULEID	VARCHAR2(4)	Auto Generated Primary Key
DOCTORID	VARCHAR2(8)	Foreign Key
AVAILABLE_DAYS	VARCHAR2(50)	Not Null
SLOTS	VARCHAR2(50)	Not Null

8. **Table:OCS_TBL_Slots**
   This table contains the details about the different slots available for appointments.

Field Name	Data Type	Description
SLOTNUMBER	NUMBER	Auto Generated Primary Key
DURATION	VARCHAR2(30)	Not Null

Database Sequences

Sequence Name	Purpose	Start With
OCS_SEQ_USERID	User ID	1000
OCS_SEQ_DOCTORID	Doctor ID	1000
OCS_SEQ_LEAVEID	Reporter ID	1000
OCS_SEQ_PATIENTID	Patient ID	1000
OCS_SEQ_APPOINTMENTID	Appointment ID	1000
OCS_SEQ_SCHEDULEID	Schedule ID	1000