Student ID: 15042984

Student Name: Amrit Shrestha

Assessment Component	Full Marks	Marks Obtained	Remarks
Planning, requirements modelling and analysis	**40**		
Gantt Chart	5		
Use Case Model	5		
Use Case Descriptions	5		
Communication Diagram Formation Logic	5		
Communication Diagram	5		
Class Diagram Formation Logic	5		
Class Diagram	10		
Design	***10***		
Design Stage Justification	10		
Implementation	***50***		
Android App OR iOS App	25		
Web App	25		

Islington College

<u>Software Engineering</u>

<u>CS5052</u>

Individual Assignment

Submitted By:

Amrit Shrestha

15042984

L2/C3

Date: 3th May, 2017

Submitted To:

Mr. Mukesh Regmie

Lecturer, IT Faculty.

Contents

Table of Figure

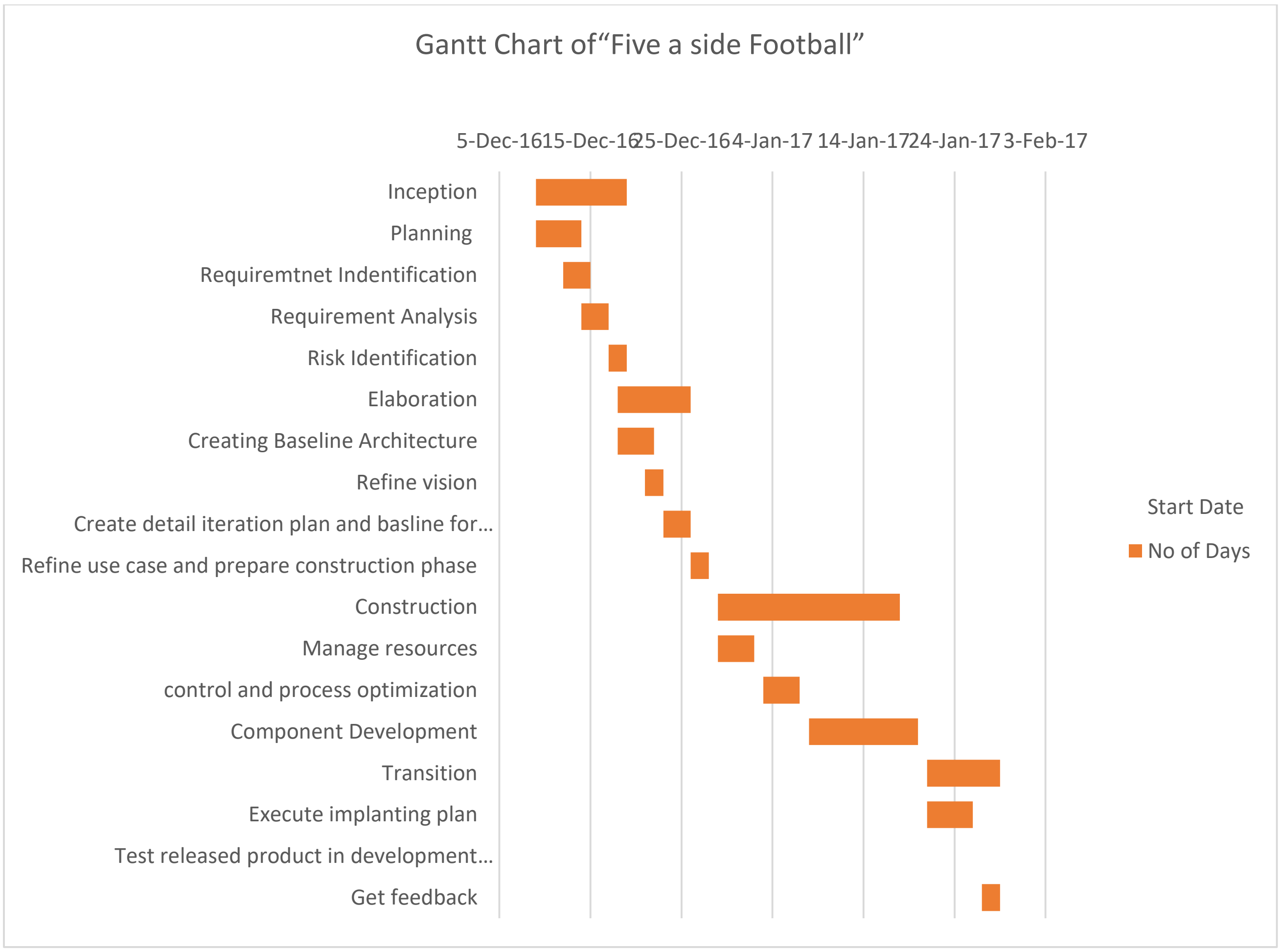
Part -1(Theory)

Planning, requirements modelling and analysis **(40%)**

Produce a Gantt chart indicating how you might schedule the work of developing the system. Your schedule should clearly be related to delivering the requirements of "Five a side Football"

Plc. and should reflect RUP/USDP practice.   (5 marks)

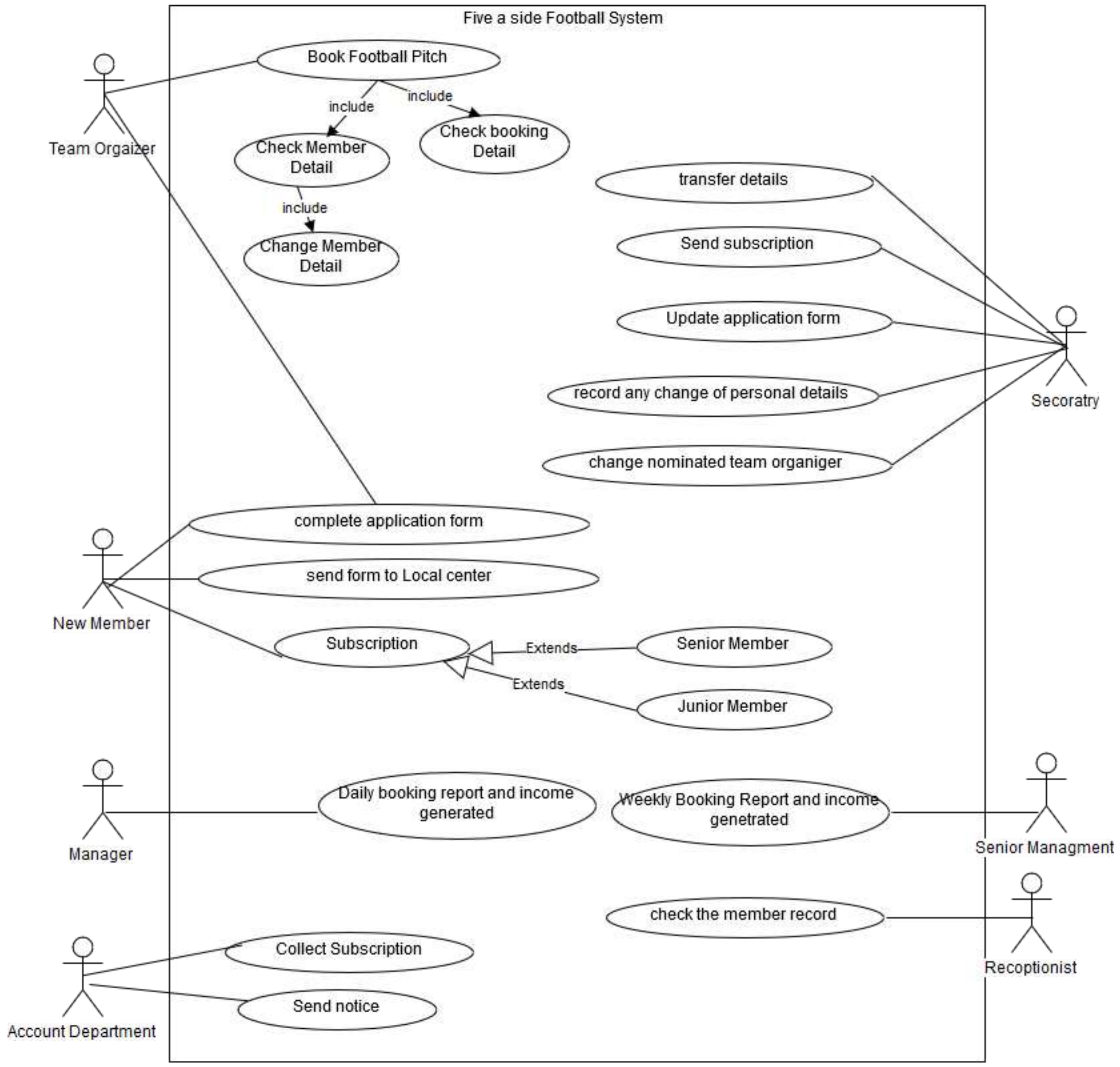
Answer:



2. Produce a Use Case Model for the required system – including a Use Case diagram, and High Level Use Case Descriptions for each Use Case. For two of the Use Cases, produce example Expanded Use Case Descriptions. **(10 marks).**

Choose one of these expanded Use Case Descriptions to model both the Collaboration and Sequence diagram, in 3, below.

Answer:



High Level Description (Book Football Pitch)

Name: Book Football Pitch

Actor: Team Organizer

Description:

A team organiser is the nominated team member who makes the football pitch bookings.

If the team organizer has his/her membership card available, a booking may be made at once

High Level Description (transfer details)

Name: transfer details

Actor: Secretory

Description: The “Five a side Football” secretary transfers the details to a membership record form which is then marked with the joining date and the subscription fee paid. The membership record form is placed in a filing cabinet at the centre, which is organised by team, then in membership number order.

High Level Description (send the subscription)

Name: send the subscription

Actor: Secretory

Description: Subscriptions are due each year on the anniversary of the member's date of joining. The company reviews subscription levels at the beginning of each calendar year.

High Level Description (update the application form)

Name: update the application form

Actor: Secretory

Description: The Accounts Department at Head Office collect subscriptions (sending out reminders as necessary) – and send a notice to the relevant Sports Centre secretary that the subscription for the year has been paid. The secretary then updates the application forms in the filing cabinet to show this information and sends out a new membership card for the member/team organizer.

High Level Description (record any change of personal details)

Name: record any change of personal details

Actor: Secretory

Description: The sports center secretary records any changes of personal details informed by the member

High Level Description (change nominated team organizer)

Name: change nominated team organizer

Actor: Secretory

Description:

A team organiser is the nominated team member who makes the football pitch bookings.

The centre secretary can also change the nominated team organizer, for example if the current organizer is ill, on holiday or has left the team, to another team member who will take over the role of Team organizer in order to book a pitch.

High Level Description (Complete the application form)

Name: Complete the application form

Actor: New Member and team organizer

Description:

A new member, who may or may not be the team organiser of a specific 5 a side football team, completes an application form and takes or sends it to their local centre, together with the first year's subscription.

High Level Description (Subscription)

Name: Subscription

Actor: New Member

Description:

There are various levels of subscription depending on the type of membership. For example, junior members'/ team organisers (all football team members must be under 18 at the start of the calendar year) are charged less than senior members.

High Level Description (Daily booking report and income generated)

Name: Daily booking report and income generated

Actor: Manager

Description:

Bookings report for each sports centre (daily) showing the total number of hours of bookings for each pitch within the centre and the income generated that day.

High Level Description (Weekly booking report and income generated)

Name: Weekly booking report and income generated

Actor: Senior Managment

Description:

Bookings report for the whole company (weekly) showing the number of hours of pitch bookings over the week at each centre, plus the income generated.

High Level Description (Check the member record)

Name: Check the member record

Actor: Receptionist

Description:

When a team organizer phones to book a pitch, the receptionist checks the member record of the team organizer in the filing cabinet and, if his/her subscriptions are up-to-date (or no more than one month in arrears), records the booking in the register.

High Level Description (Collect Subscription)

Name: Collect Subscription

Actor: Account Department

Description: Account Department at head office collect subscription (sending out reminders as necessary)

High Level Description (send notice)

Name: send notice

Actor: Receptionist

Description: Account Department at head office send a notice to the relevant Sports Centre secretary that the subscription for the year has been paid.

High Level Description (Check Member Detail)

Name: Check Member Detail

Actor: Team Organizer

Description:

A team check the member details

High Level Description (Check Booking Detail)

Name: Check Booking Detail

Actor: Team Organizer

Description:

A team organizer check booking details

High Level Description (Change Member detail)

Name: Change Member detail

Actor: Team Organizer

Description:

A team organizer change member details

Expanded Level Description

- Name: Book Football Pitch
- Actor(s): Team Organizer
- Purpose: Booking Football Pitch
- Overview: A team organiser is the nominated team member who makes the football pitch bookings. A team organizer must pay for the pitch before the game at reception. Team organizer collects money from the individual team members before the booking and pays at reception
- Action Steps:

Actor Action	System Response
1. Team organizer collects money from the individual team members before the booking.	
2. When a team organizer phones to book a pitch, the receptionist checks the member record of the team organizer in the filing cabinet and, if his/her subscriptions are up-to-date (or no more than one month in	

arrears), records the booking in the register	
	3. Store/ Update Team organizer and member details
4. Team Organizer get the notification after booking	

Expanded Level Description

- Name: Record any change in personal details
- Actor(s): Secretory
- Purpose: Record any change in personal details as informed by member
- Overview: A team organiser is the nominated team member who makes the football pitch bookings. A team organizer must pay for the pitch before the game at reception. Team organizer collects money from the individual team members before the booking and pays at reception
- Action Steps:

Actor Action	System Response
1. Member inform the change in personal details	
2. Secretory change the personal details	
	3. Change personal details
4. Old details is changed and updated	

3. Explain in words how you would go about producing the Communication diagrams, that is a Collaboration diagram and a Sequence diagram, from the Use Case model. **(5 marks)**

Produce a Collaboration diagram and then produce a Sequence diagram for the same Use Case**. (5 marks)**

Answer:

To Produce the Communication Diagram I would go through Use Case Description. I will be producing Communication diagram of Book Football Pitch. At first I will find the objects of communication diagram. Then I will be adding control object (Book football) pitch to manage the collaboration of the object. Boundary object will be added which will have same name as control object with the addition of UI. Then adding the actor I will be connecting each object and actor by drawing the line. Again I will be adding message to the line.

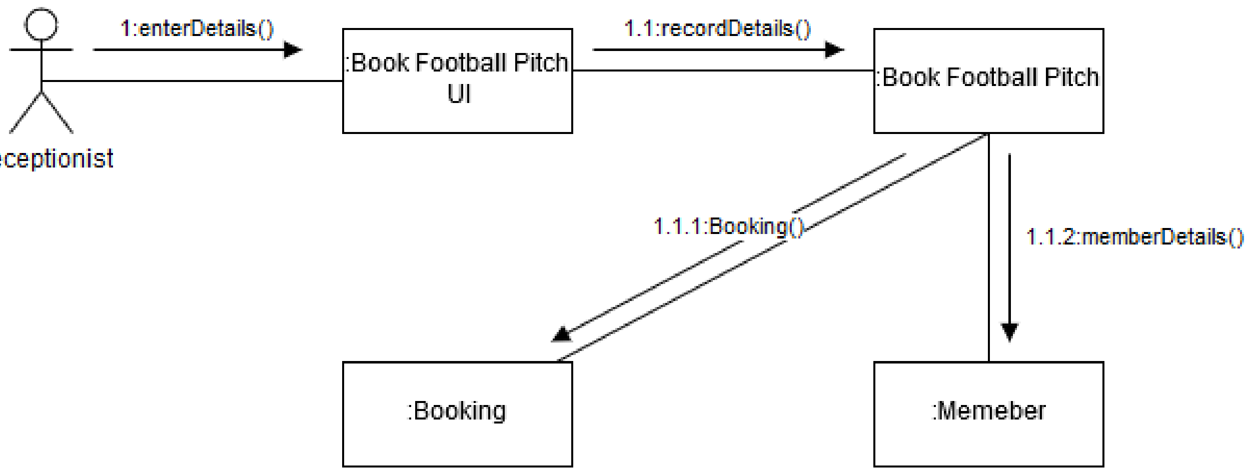


Figure 1: Communication Diagram of Booking Football Pitch

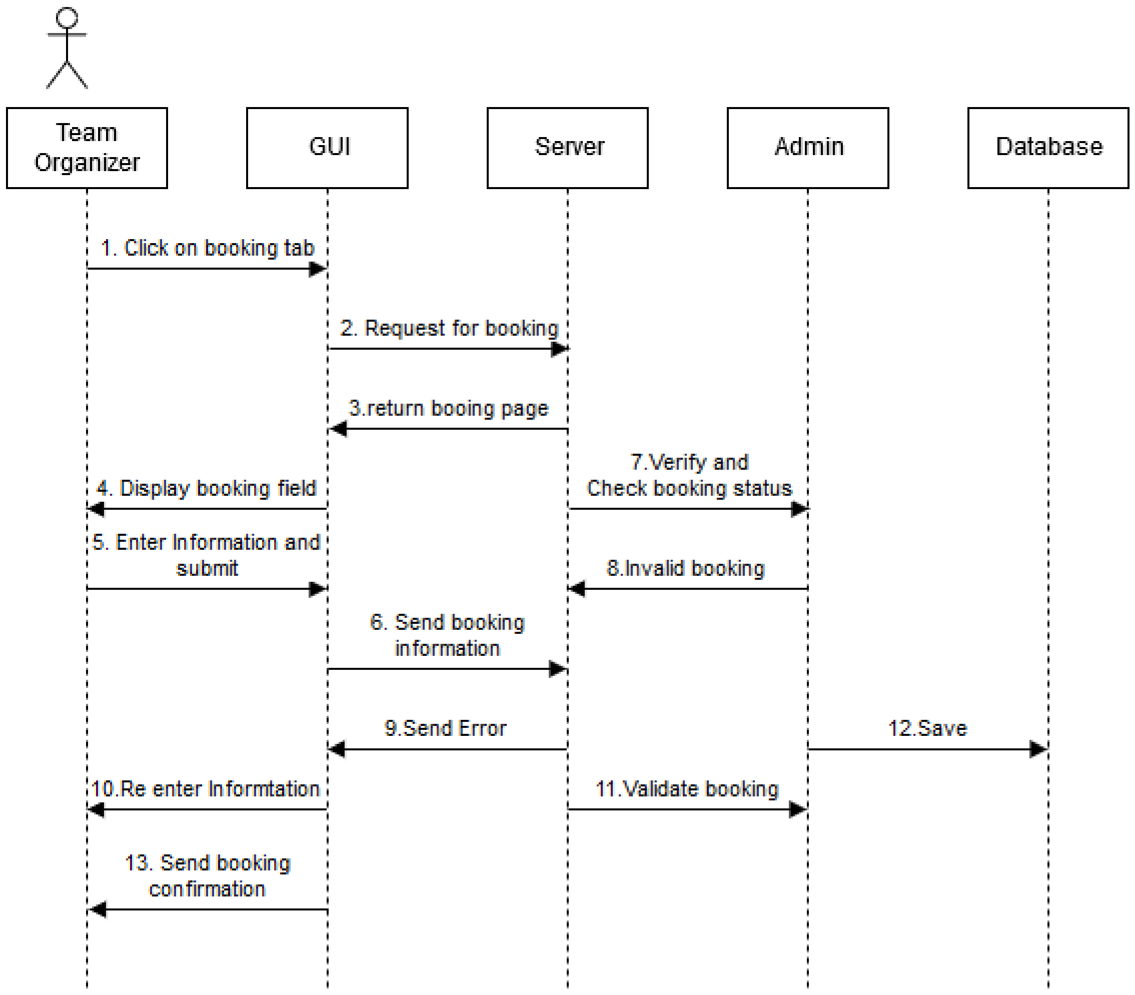


Figure 2: Sequence Diagram

4. Explain in words how you would produce the Class Diagram. **(5 marks)**

Answer:

The class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing and documenting different aspects of a system but also for constructing executable code of the software application.

At first I would identify the class to describe the aspect of the system and give it meaning full name. Classes are represented in the boxes that contain three compartments. In first compartment I would specify the name of class and make it bold and centred. The first letter is capitalized. In second compartment I would list the attribute of the class. They are left-aligned and the first letter is lowercase. And the last compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses. Finally I would specify the relationship between the class.

Produce an Analysis Class diagram showing the domain classes. **(10 Marks)**

Answer:

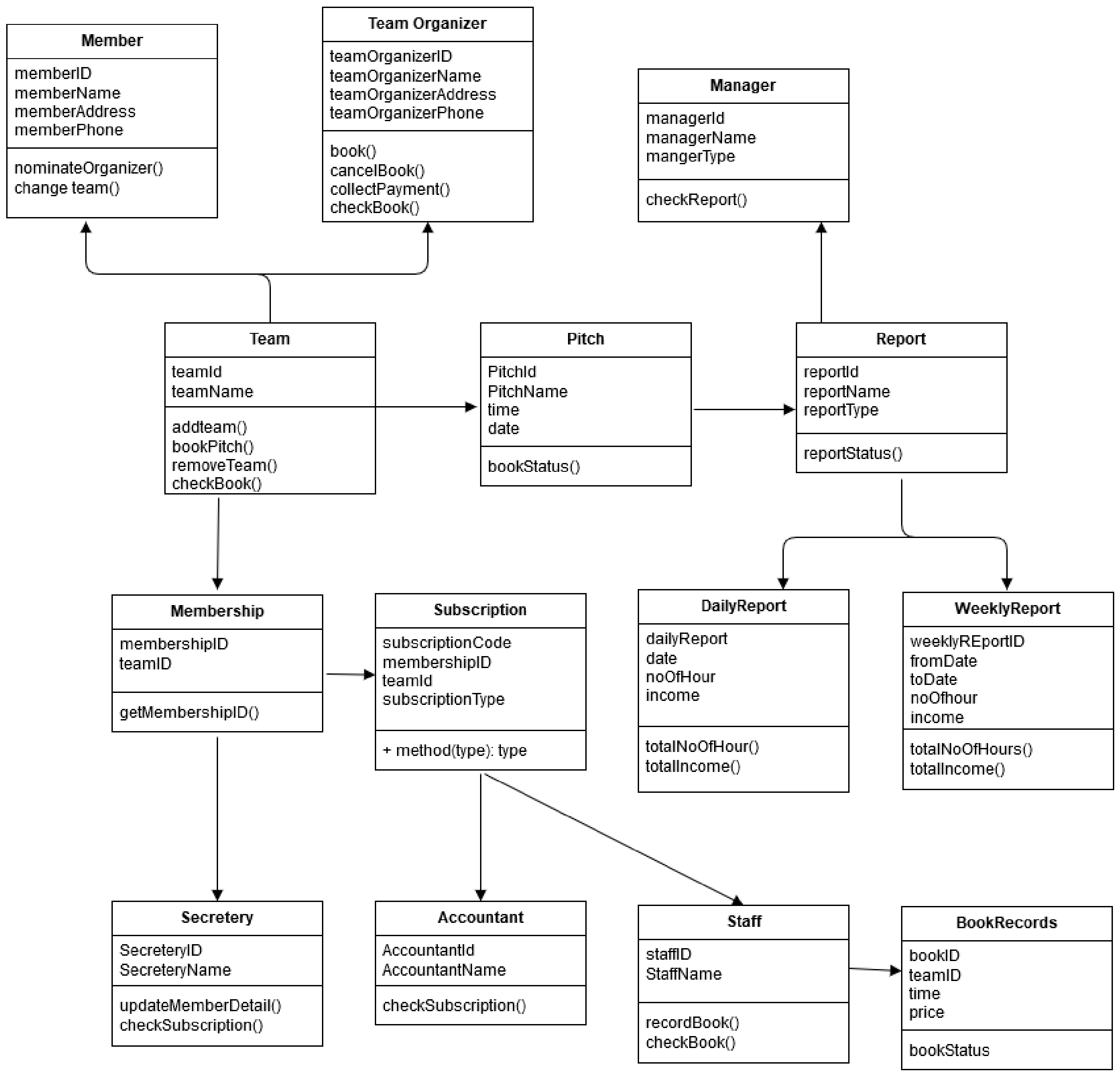


Figure 3: Class Diagram

5. Explain how you would move forward from this part of the development process. What techniques/products might be produced for the design? Explain briefly the purpose of each**. (10 marks)**

Answer:

To move forward from this part of the development, I would follow different steps. I would use Waterfall module for the development process which is divided into separate phases. In the first phase, Requirement gathering and analysis all the possible requirement of the system to be developed are captured and documented in a requirement specification document. The next will be system Design phase, where requirement specification form first phase are studied and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture. Implementation is done with the help of system design. Each unit is developed and tested for its functionality. Then testing will be performed. Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market. Finally, maintenance is done if there is some issues.

Part -2(Android App Development)

Source Code File Descriptions

SN	Source Code File Name	Description	Location
1	Contactdisplay.java	It display the view of contact	MyApplication2\app\src\main\java\com\example\user\myapplication
2	ContactDisplayAdapter.java	It set adapter to the view of contact display	
3	ContactDisplayView.java	It display the expandable list view	
4	DatabaseInseter.java	It conduct query transaction from and into the data base	
6	activity_contact_display_view.xml	It display the view of contact.	MyApplication2\app\src\main\res\layout
7	Activity_contactdisplay.xml	It display the contact.	

Application Workflow Steps and Description

1. Server Side Code responsible for fetching data from mysql

```
<?php
        $master_array=array();//all the database result is inside $result
        require "conn.php"; //calls the line:
$con=mysqli_connect($server_name,$mysql_username,$mysql_password,$db_
name); from conn.php
        $sql="select * from emp_contact1";
        $result=mysqli_query($conn,$sql); //all the database result is inside
$result
                while($row=mysqli_fetch_array($result)) //it will fetch all the result
from the variable (row) in the table by using this built in function of array
                {
                        $tmp_array=array("empId"=>$row[0],
"empName"=>$row[1],"mobileNo"=>$row[2], "officeNo"=>$row[3],
"homeNo"=>$row[4], "email"=>$row[5]);
                        array_push($master_array,$tmp_array); //creating response
array and added one record which will push first ID of the employee from the
database
                }
        $json_data=array("server_response"=>$master_array);
        echo json_encode($json_data);
        mysqli_close($conn);
?>
```

2. Android Code Responsible for fetching data from web server

```
protected String doInBackground(Void... params) {
    String json_string;


    try {
        StringBuilder stringBuilder = new StringBuilder();
        URL url = new URL(service_url);
        HttpURLConnection httpURLConnection = (HttpURLConnection)
url.openConnection();
        httpURLConnection.setConnectTimeout(7000);
        httpURLConnection.setReadTimeout(7000);
        httpURLConnection.connect();
        if (httpURLConnection.getResponseCode() == 200) {
            InputStream inputStream= httpURLConnection.getInputStream();
            BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));
            while ((json_string = bufferedReader.readLine()) != null) {

                stringBuilder.append(json_string + "\n");

            }

            bufferedReader.close();
            inputStream.close();
            httpURLConnection.disconnect();

            result = stringBuilder.toString().trim();

            Log.i("Contacts are", result);

        }
        else
        {
            result = null;
            Log.i("ContactDisplay", "Data Cannot Be Fetched");
        }
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    return result;
}
```

3. Android Code Responsible for inserting or syncing fetched data from web server into local SQLite Database

```
protected void onPostExecute(String result)
{
    json_str = result;
    Thread thd = new Thread()
    {
        @Override
        public void run()
        {
            try
            {
                if(json_str != null)
                {
                    int c = 0;
                    String empId, empName, mobileNo, officeNo,homeNo,email;
                    jsonObject = new JSONObject(json_str);
                    jsonArray = jsonObject.getJSONArray("server_response");
                    inserter = new DatabaseInseter(contactdisplay.this);
                    db= inserter.getWritableDatabase();
                    db.execSQL("Delete from emp_contact");
                    db.beginTransaction();

                    while(c<jsonArray.length())
                    {
                        JSONObject row = jsonArray.getJSONObject(c);
                        empId = row.getString("empId");
                        empName = row.getString("empName");
                        mobileNo = row.getString("mobileNo");
                        homeNo = row.getString("homeNo");
                        officeNo = row.getString("officeNo");
                        email = row.getString("email");

DatabaseInseter.insertData(db,empId,empName,mobileNo,officeNo,homeNo,email);
                        c++;
                    }
                    db.setTransactionSuccessful();
                    Log.i("Status:","Inserted Sucessfully");

                }
            }
            catch (Exception e)
            {
                e.printStackTrace();
            }
            finally {
                {
                    db.endTransaction();
                    db.close();
                }
            }
        }


    };
    thd.start();

    Intent intent=new Intent(getApplicationContext(),ContactDisplayView.class);
    startActivity(intent);
}
```

4. Android Code Responsible for fecthing data from local SQLiteDatabase and Displaying it on ExpandableListView

```
try
    {
        String empId,empName,mobileNo,officeNo, homeNo,email;
        final List<String> Headings=new ArrayList<String>();
        final HashMap<String,List<String>> childList=new
HashMap<String,List<String>>();
        List<String> L3;
        Inseter =new DatabaseInseter(ContactDisplayView.this);
        SQLiteDatabase db=Inseter.getWritableDatabase();
        Cursor res=Inseter.getAllData(db);
        int count =0;
        while(res.moveToNext())
        {
            L3=new ArrayList<String>();
            empName=res.getString(1);
            mobileNo=res.getString(2);
            homeNo=res.getString(3);
            officeNo=res.getString(4);
            email=res.getString(5);
            Headings.add(empName);
            L3.add("MobileNo:"+mobileNo);
            L3.add("HomeNo:"+homeNo);
            L3.add("OfficeNo:"+officeNo);
            L3.add("Email:"+email);
            childList.put(empName,L3);
            count++;
        }
         ContactDisplayAdapter= new
ContactDisplayAdapter(this,Headings,childList);
        expandableListView.setAdapter(ContactDisplayAdapter);


    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}
```

5. Android Code for searching item in ExpandableListView

```
public void filterdata(String query) {
    query = query.toLowerCase();
    header_titles.clear();

    if (query.isEmpty()) {
        header_titles.addAll(search_title);
    } else {
        for (String title: search_title) {
            List<String> childs = child_title.get(title);
            for (String child: childs) {
                if (child.toLowerCase().contains(query)
                        || title.toLowerCase().contains(query)) {
```

```
                    if (!header_titles.contains(title)) {
                        header_titles.add(title);
                    }
                }
            }
        }
    }
    notifyDataSetChanged();
}
```

6. Android Code for making a call

```
    Button callBtn_mobile=(Button) convertView.findViewById(R.id.callBtn_mobile);
    callBtn_mobile.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            String mobile_no=(String)mobile_item.getText();

    mobile_no=mobile_no.substring(mobile_no.indexOf(":")+1,mobile_no.length());
            Intent phoneCallIntent = new Intent(Intent.ACTION_CALL);
            phoneCallIntent.setData(Uri.parse("tel:"+mobile_no));

    if(ActivityCompat.checkSelfPermission(ctx,android.Manifest.permission.CALL_PHON
    E)!= PackageManager.PERMISSION_GRANTED)
            {
                return;
            }

          ctx.startActivity(phoneCallIntent);
        }
    });
```

7. Android code for making SMS

```
Button callBtn_sms=(Button) convertView.findViewById(R.id.smsBtn_mobile);
callBtn_sms.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String mobile_no=(String)mobile_item.getText();
        mobile_no=mobile_no.substring(mobile_no.indexOf(":"+1,mobile_no.length()));

       // Intent sendIntent= new intent(sendIntent.ACTION_VIEW);
        //sendIntent.putExtra("sms_body","default")

        Intent intent=new Intent(Intent.ACTION_SENDTO);
        intent.setData(Uri.parse("smsto:" + Uri.encode(mobile_no)));
        ctx.startActivity(intent);
    }
});
```

Part -3(Spring App Development)

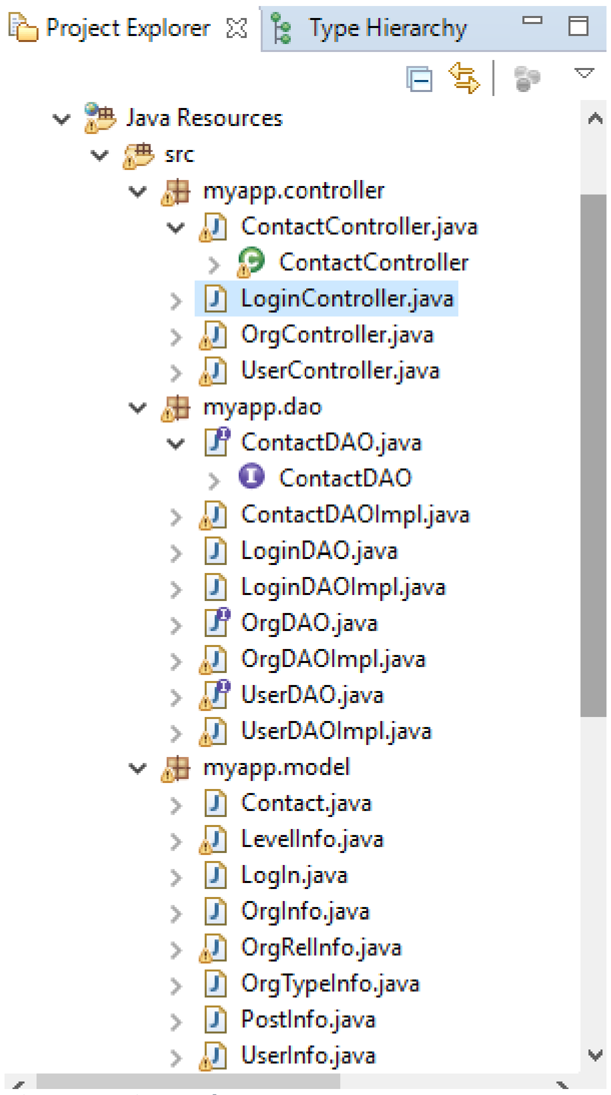
Source Code File Descriptions

SN	Source Code File Name	Description	Location
1	ContactController.java	This class provide contact view and control contact model.	NepalGovContact\src\myapp\controller
2	LoginController.java	This class provide Login view and control login model.	
3	OrgController.java	This class provide Org view and control org model.	
4	UserController.java	This class provide User view and control user model.	
5	ContactDAOImpl.java	This class handle data model of all contact	NepalGovContact\src\myapp\dao
6	OrgDAOImpl.java	This class handle data model of all orgs	
7	UserDAo.java	This class handle data model of all user.	
8	DBConnector.java	This class opens, closes and perform JDBC operation	NepalGovContact\src\myapp\util
9	Status.java	This class controls the status in website	
10	Contact.hbm.xml	This xml file maps database with data model classes	NepalGovContact\src
11	Hibernate.cfg.xml	This xml file configures database connection with the system	
12	spring-servelet.xml	This xml file configures database connection with web content	NepalGovContact\WebContent\WEB-INF
13	.jsp *	All JSP file work as website design provider	NepalGovContact\WebContent\WEB-INF\views

Product Feature Check List

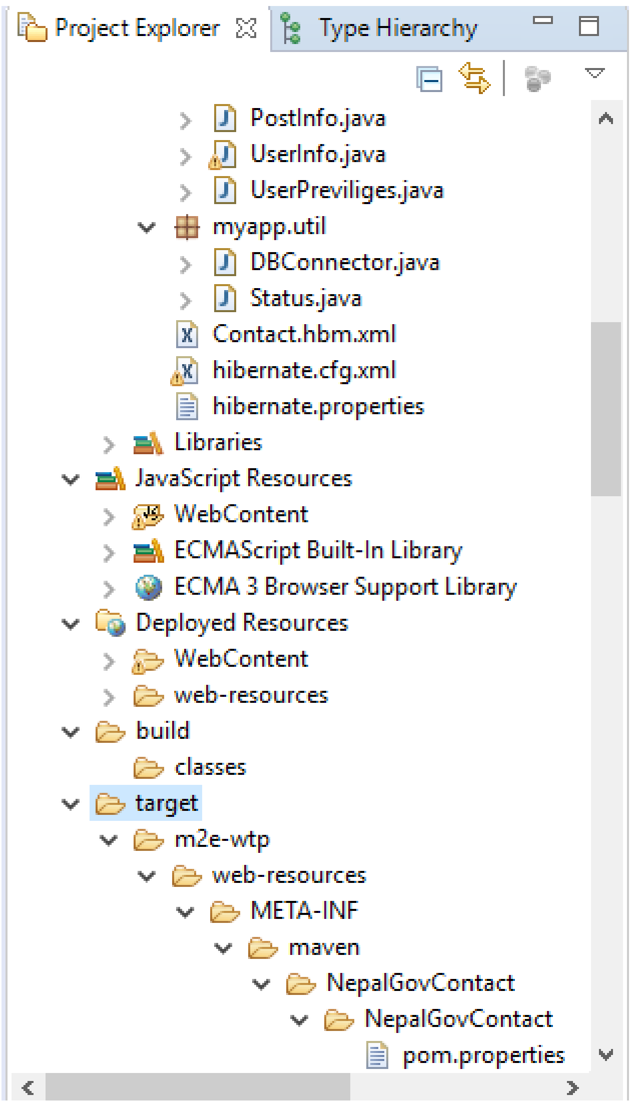
SN	Feature Description	Implemented(Yes/No)
1	**Product has capability to record/view/update/delete ministry**	Yes
2	**Product has capability to record/view/update/delete departments and divisions under ministry**	Yes
3	**Product has capability to record/view/update/delete sections under departments**	Yes
4	**Product has capability to record/view/update/delete units under sections**	Yes
5	Product has capability to records/updates/delete each employees (id, name, address, telephone mobile number, home phone number, email and fax) and his office	Yes
6	There exists admin user in each Ministry/Department/Division/Section (Local Admin) which has authority to view/add/edit/delete employee's contact belonging to his organization only.	Yes
7	Super Administrator user has rights to view/add/edit/delete employees of all offices of Government of Nepal.	Yes
8	Employees are frequently transferred in different offices. Either Super Admin or Local Admin has authority to transfer employees.	Yes
Note: Feature no 1 to 5 is available to super administrator user only		

Complete Screen Shot of Project Ex


Project Explorer   Type Hierarchy

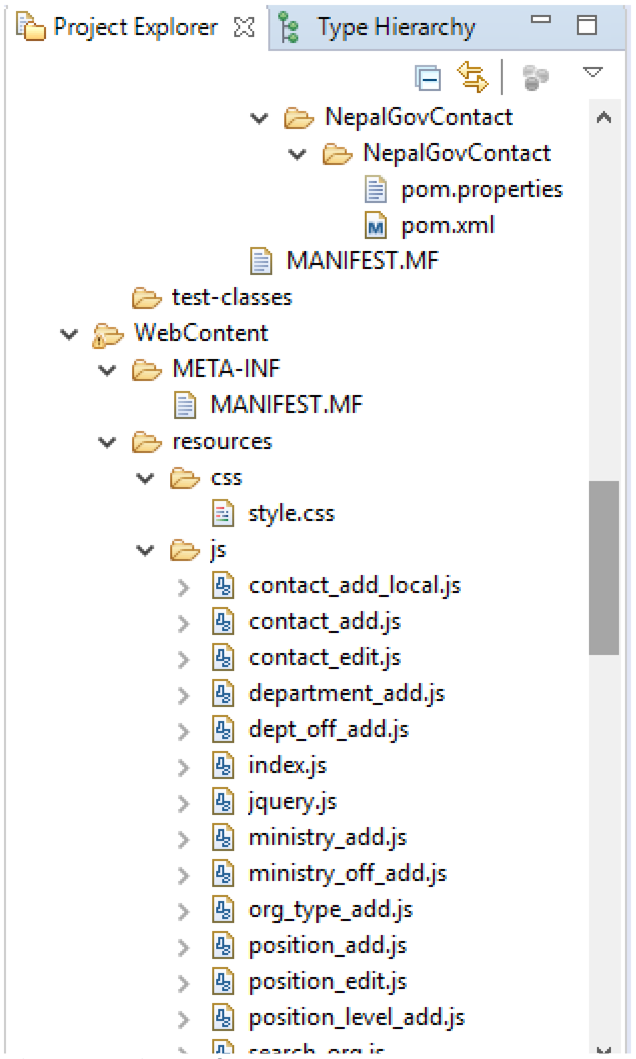
- Java Resources
  - src
    - myapp.controller
      - ContactController.java
        - ContactController
      - LoginController.java
      - OrgController.java
      - UserController.java
    - myapp.dao
      - ContactDAO.java
        - ContactDAO
      - ContactDAOImpl.java
      - LoginDAO.java
      - LoginDAOImpl.java
      - OrgDAO.java
      - OrgDAOImpl.java
      - UserDAO.java
      - UserDAOImpl.java
    - myapp.model
      - Contact.java
      - LevelInfo.java
      - LogIn.java
      - OrgInfo.java
      - OrgRelInfo.java
      - OrgTypeInfo.java
      - PostInfo.java
      - UserInfo.java

Figure 5:Project Explorer1


Project Explorer   Type Hierarchy

- PostInfo.java
- UserInfo.java
- UserPreviliges.java
- myapp.util
  - DBConnector.java
  - Status.java
- Contact.hbm.xml
- hibernate.cfg.xml
- hibernate.properties
- Libraries
- JavaScript Resources
  - WebContent
  - ECMAScript Built-In Library
  - ECMA 3 Browser Support Library
- Deployed Resources
  - WebContent
  - web-resources
- build
  - classes
- target
  - m2e-wtp
    - web-resources
      - META-INF
        - maven
          - NepalGovContact
            - NepalGovContact
              - pom.properties

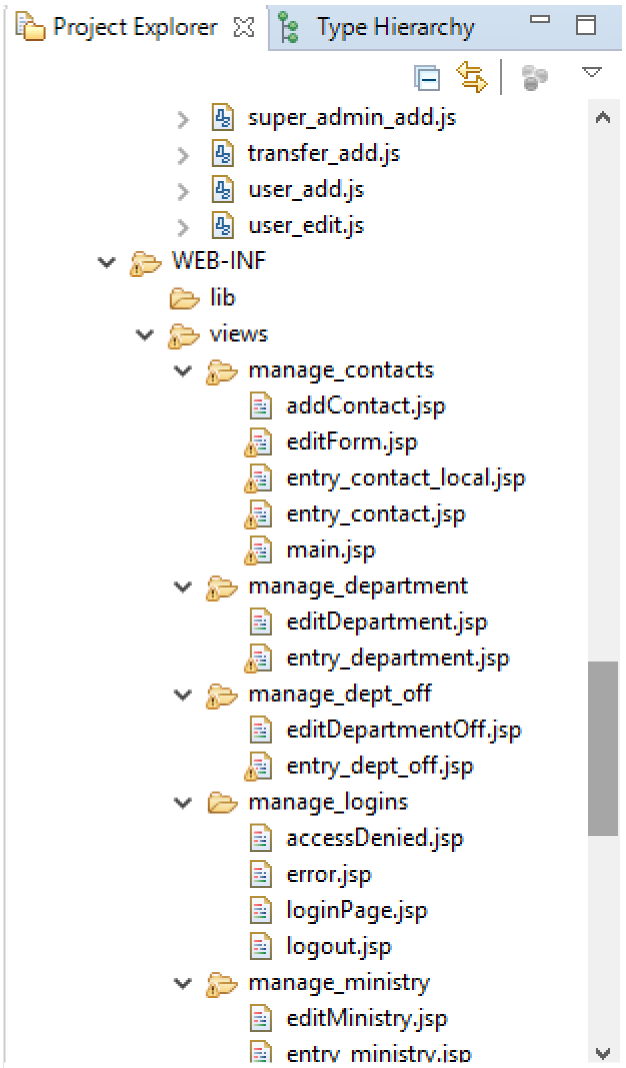
Figure 4:Project Explorer2



```
Project Explorer    Type Hierarchy

        NepalGovContact
            NepalGovContact
                pom.properties
                pom.xml
        MANIFEST.MF
    test-classes
  WebContent
    META-INF
      MANIFEST.MF
    resources
      css
        style.css
      js
        contact_add_local.js
        contact_add.js
        contact_edit.js
        department_add.js
        dept_off_add.js
        index.js
        jquery.js
        ministry_add.js
        ministry_off_add.js
        org_type_add.js
        position_add.js
        position_edit.js
        position_level_add.js
        search_org.js
```

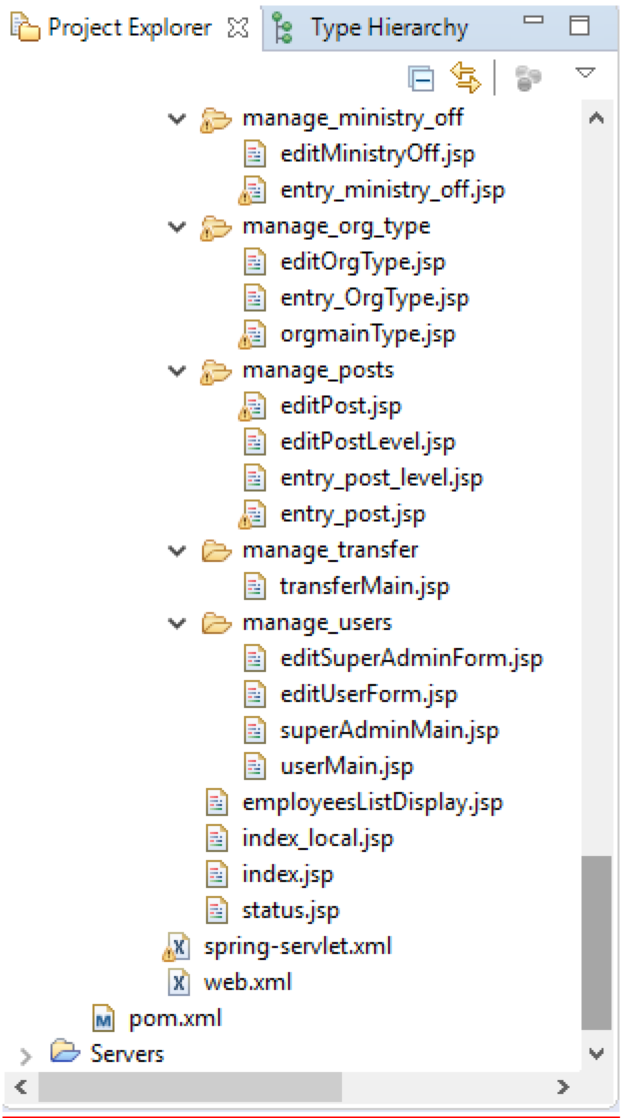
Figure 7: Project Explorer3



```
Project Explorer    Type Hierarchy

        super_admin_add.js
        transfer_add.js
        user_add.js
        user_edit.js
    WEB-INF
      lib
      views
        manage_contacts
          addContact.jsp
          editForm.jsp
          entry_contact_local.jsp
          entry_contact.jsp
          main.jsp
        manage_department
          editDepartment.jsp
          entry_department.jsp
        manage_dept_off
          editDepartmentOff.jsp
          entry_dept_off.jsp
        manage_logins
          accessDenied.jsp
          error.jsp
          loginPage.jsp
          logout.jsp
        manage_ministry
          editMinistry.jsp
          entry_ministry.jsp
```

Figure 6: Project Explorer4



```
manage_ministry_off
    editMinistryOff.jsp
    entry_ministry_off.jsp
manage_org_type
    editOrgType.jsp
    entry_OrgType.jsp
    orgmainType.jsp
manage_posts
    editPost.jsp
    editPostLevel.jsp
    entry_post_level.jsp
    entry_post.jsp
manage_transfer
    transferMain.jsp
manage_users
    editSuperAdminForm.jsp
    editUserForm.jsp
    superAdminMain.jsp
    userMain.jsp
employeesListDisplay.jsp
index_local.jsp
index.jsp
status.jsp
spring-servlet.xml
web.xml
pom.xml
Servers
```

Figure 8: Project Explorer5

Working Mechanism of Hibernate

Hibernate framework simplifies the development of java application to interact with the database. Hibernate is an open source, lightweight, ORM (Object Relational Mapping) tool. The Hibernate architecture includes many objects persistent object, session factory, transaction factory, connection factory, session, transaction etc. There are 4 layers in hibernate architecture java application layer, hibernate framework layer, backhand api layer and database layer

Controller, DAO and View Interaction Descriptions

The View is the information that is being presented to a user. A View will normally be a web page.

The Controller serves as an intermediary between the Model, the View, and any other resources needed to process the HTTP request and generate a web page.

The Model represents your data structures. Typically your model classes will contain functions that help you retrieve, insert, and update information in your database.

Data Access Objects (or DAOs for short) are used as a direct line of connection and communication with our database. DAOs are used when the actual CRUD (CRUD = Create, Read, Update, Delete) operations are needed and invoked in our Java code. These data access objects also represent the "data layer" of our application. These objects are still just plain old Java objects that incorporate the use of some Hibernate annotations to give them the functionality we need from them. Again, that functionality being the communication with the database.

Controller Method Descriptions

editContactRecord(HttpServletRequest request)

- This method is used to edit employee contact record.

displayTable(HttpServletRequest request,HttpServletResponse response)

- This method is used to display employee record on the table

displayLocalTable(HttpServletRequest request,HttpServletResponse response)

- This method is used to display employee record on local table

editContactRecord1(HttpServletRequest request,HttpServletResponse response)

- This method is used to edit the contact record

transferEmployee(HttpServletRequest request,HttpServletResponse response)

- This method is used to transfer the Employee

entry_transfer()

- This method is used to return manage_transfer/ entry_contact.

entry_contact()

- This method is used to return manage_contacts/entry_contact.

entry_contact_local()

- This method is used to return manage_contacts/entry_contact_local.

indexpage()

- This method is used to return to the index page

mainpage()

- This method id used to return to the main page.

addRecord(@ModelAttribute("objContact")Contact contact)

- This method is used to add record.

editContact(HttpServletRequest request)

- This method is used to edit contact.

deleteContact(HttpServletRequest request)

- This method is used to delete contact.

displayMainPage(ModelAndView model)

- ➢ This method display the main page

display_login_page()

- ➢ This method is used to display login page

logout()

- ➢ This method is used for logout

error()

- ➢ This method return error.

deleteMinistry(HttpServletRequest request)

- ➢ This method delete the Ministy.

deletePost(HttpServletRequest request)

- ➢ This method delete Post.

deleteLevelRecord(HttpServletRequest request)

- ➢ This method delete level Record.

deleteOrgTypeRecord(HttpServletRequest request)

- ➢ This method delete Org type record.

editMinistryRecord(HttpServletRequest request)

- ➢ This method edit Ministry Record.

editPostRecord(HttpServletRequest request)

- ➢ This method edit post record.

editLevelRecord(HttpServletRequest request)

- ➢ This method edit Level Record.

editOrgTypeRecord(HttpServletRequest request)

- ➢ This method edit Org Type Record.

editDepartment(HttpServletRequest request)

- ➢ This method edit Department.

editDepartmentOff(HttpServletRequest request)

- ➢ This method is used to add office record.

editMinistry(HttpServletRequest request)

- This method is used to edit Ministry Record.

editPost(HttpServletRequest request)

- This method is used to edit post.

editLevel(HttpServletRequest request)

- This method is used to edit Level.

editMinistryType(HttpServletRequest request)

- This method is used to edit Ministry type

search_org()

- This method is used to search_org

displayMinTable(HttpServletRequest request,HttpServletResponse response)

- This method display ministrie table

displayPostTable(HttpServletRequest request,HttpServletResponse response)

- This method display post table

displayOfficeName(HttpServletRequest request,HttpServletResponse response)

- This method display office name.

displayCurrentEmpInfos(HttpServletRequest request,HttpServletResponse response)

- This method display Current Employee Info

displayLevelTable(HttpServletRequest request,HttpServletResponse response)

- This method display level table

displaySearchTable(HttpServletRequest request,HttpServletResponse response)

- This method display search table.

displayOrgCombo(HttpServletRequest request,HttpServletResponse response)

- This method display OrgCombo.

displayLevelCombo(HttpServletRequest request,HttpServletResponse response)

- This method display Level combo

displayDeptCombo(HttpServletRequest request,HttpServletResponse response)

- This method display Department Combo.

displayDeptTable(HttpServletRequest request,HttpServletResponse response)

- This method display Department table.

displayDeptOffTable(HttpServletRequest request,HttpServletResponse response)

- This method display department and office table

displayOrgTypeTable(HttpServletRequest request,HttpServletResponse response)

- This method is used to display Organization table.

addMinRecord(HttpServletRequest request,HttpServletResponse response)

- This method add ministry record.

addPostRecord(HttpServletRequest request,HttpServletResponse response)

- This method add Post record

addLevelRecord(HttpServletRequest request,HttpServletResponse response)

- This method add Level Record

addDeptRecord(HttpServletRequest request,HttpServletResponse response)

- This method add Department record.

addDeptOffRecord(HttpServletRequest request,HttpServletResponse response)

- This method add department office record.

addOrgTypeRecord(HttpServletRequest request,HttpServletResponse response)

- This method add Organization type record.

addUserRecord(HttpServletRequest request,HttpServletResponse response)

- This method user add user record.

addSuperAdminUserRecord(HttpServletRequest request,HttpServletResponse response)

- This method add super admin user record.

displayTable(HttpServletRequest request,HttpServletResponse response)

- This method display the table.

displaySuperAdminDashboard(HttpServletRequest request,HttpServletResponse response)

- This method display super admin.
-

editUser(HttpServletRequest request)

- This method is used to edit the user.

editSuperAdminUser(HttpServletRequest request)

- This method is used to edit super admin.

editUserRecord(HttpServletRequest request,HttpServletResponse response)

- This method is used to edit user record.

editSuperAdminRecord(HttpServletRequest request,HttpServletResponse response)

- This method edit super admin record.

deleteContact(HttpServletRequest request)

- This method delete contact.

deleteSuperAdmin(HttpServletRequest request)

- This method delete super admin.

checkUserType(HttpServletRequest request)

- this method check user type.

DAO Method Description

getJdbcTemplate()

- This method return java database connectivity templete

setJdbcTemplate(JdbcTemplate jdbcTemplate)

- This method set the value of Java Database connectivity Templete

retAllContacts()

- This return all contact

retLocalContacts(String org_code)

- This method return Local Contact

insert(Contact contact)

- This method insert the contact

transferEmp(String empID,String office_code)

- This method is used to transfer Employee.

displayAllContacts()

- This method display all contact

deleteContact(String id)

- This method delete the contact

getContact(String id)

- This method return contact

updateContact(String id,String empName, String mobileNo,String officeNo,String homeNo,String email,String office_code,String post_code)

- This method is used to update the contact information such as ID, Employe Name, mobile number, office number, home number, email, office code and post code.

getJdbcTemplate()

- This method return JdbcTemplete

setJdbcTemplate(JdbcTemplate jdbcTemplate)

- This method set the value for set JdbcTemplete

insert_org_type(OrgTypeInfo org_type_info)

- This method is used to insert organization type

insert_dept(OrgInfo org_info, OrgRelInfo org_rel_info)

- This method id used to insert department

retAllOrgTypes()

- This method return organization type

retAllOrgs()

- This method return all organiztion

getOrgType(String org_type_code)

- This method return org type

updateOrgRecord(String org_code,String org_name,String org_type)

- This method update org record

deleteOrgRecord(String org_code)

- This method is used to delete the organization record

deleteOrgTypeRecord(String org_type_code)

- This method is used to delete org_type_record

updateOrgType(String org_type_code, String org_type)

- This method update Org type

retAllMins_combo(String org_type_code)

- ➢ This method return all min combo

retEmpInfos(String empID)

- ➢ This method return employee info

searchAllOrgs(String searchParam, String searchType)

- ➢ This method is used to search orgs

insert_level(LevelInfo level_info)

- ➢ This method insert level

retAllLevels()

- ➢ This method return all level

getLevel(String level_id)

- ➢ This method return level

updateLevelRecord(String level_id, String level_desc)

- ➢ This method update level record.

deleteLevelRecord(String level_id)

- ➢ This method delete level record.

retAllLevel_combo()

- ➢ This method return all level combo.

insert_post(PostInfo post_info)

- ➢ This method insert the post.

retAllPosts()

- ➢ This method return all posts.

getPost(String post_id)

- ➢ This method return the post

updatePostRecord(String post_id, String post_name, String level_id)

- ➢ This method update post record

deletePostRecord(String post_id)

- ➢ This method delete post record.

getOrgName(String org_code)

- ➢ This method return organization name.

insert(UserInfo user, UserPreviliges user_prevs)

- ➢ This method insert user.

retAllUsers()

- ➢ This method return all the user

getUser(String user_code)

- ➢ This method return the user.

updateUser(UserInfo user_info,UserPreviliges user_prevs)

- ➢ This method update the user info

deleteUser(String id)

- ➢ This method is used to delete the user.

insert_super_admin(UserInfo user)

- ➢ This method insert super admin

retAllSuperAdminUsers()

- ➢ This method return all the super admin user

getSuperAdminUser(String user_code)

- ➢ This method return super admin

updateSuperAdminUser(UserInfo user_info)

- ➢ This method update super admin

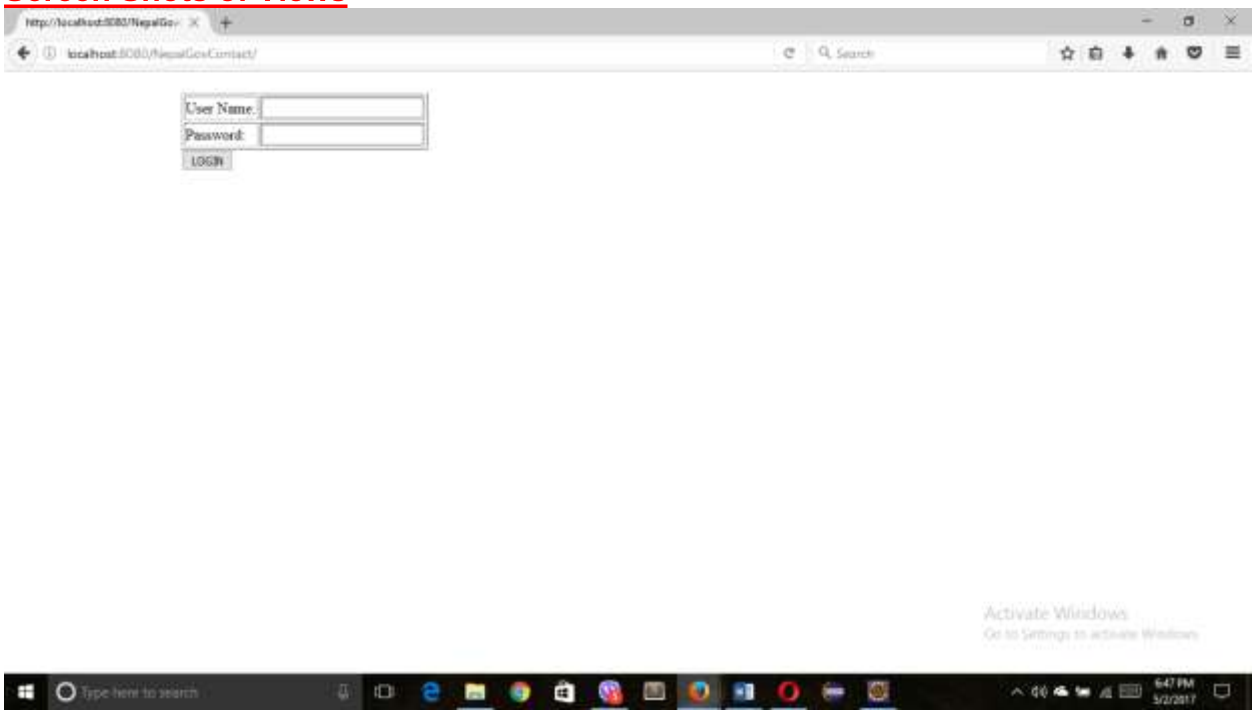
deleteSuperAdmin(String id)

- ➢ This method delete the super admin.

verifyUser(String user_name, String password)

- ➢ This method verify user name and password of the user.

Screen Shots of Views



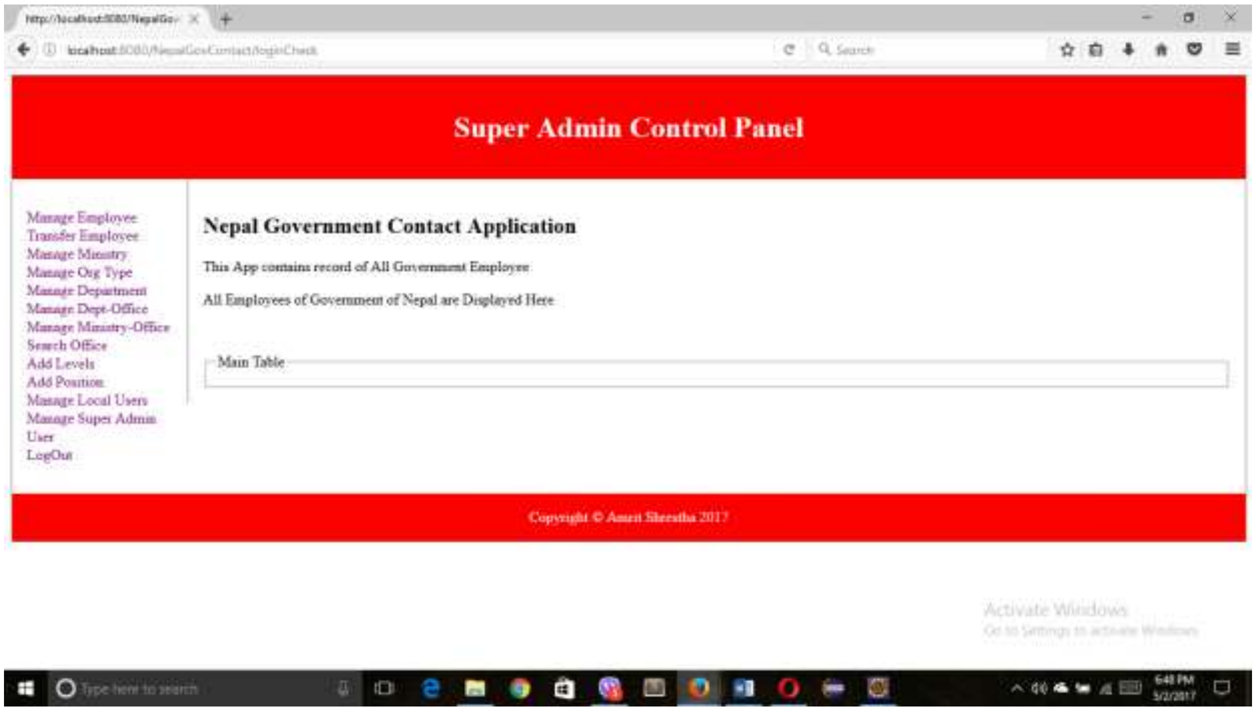
localhost:8080/NepalGovContact/

User Name:

Password:

LOGIN

Figure 9: View of Login page



localhost:8080/NepalGovContact/loginCheck

Super Admin Control Panel

Manage Employee
Transfer Employee
Manage Ministry
Manage Org Type
Manage Department
Manage Dept-Office
Manage Ministry-Office
Search Office
Add Levels
Add Position
Manage Local Users
Manage Super Admin
User
LogOut

Nepal Government Contact Application

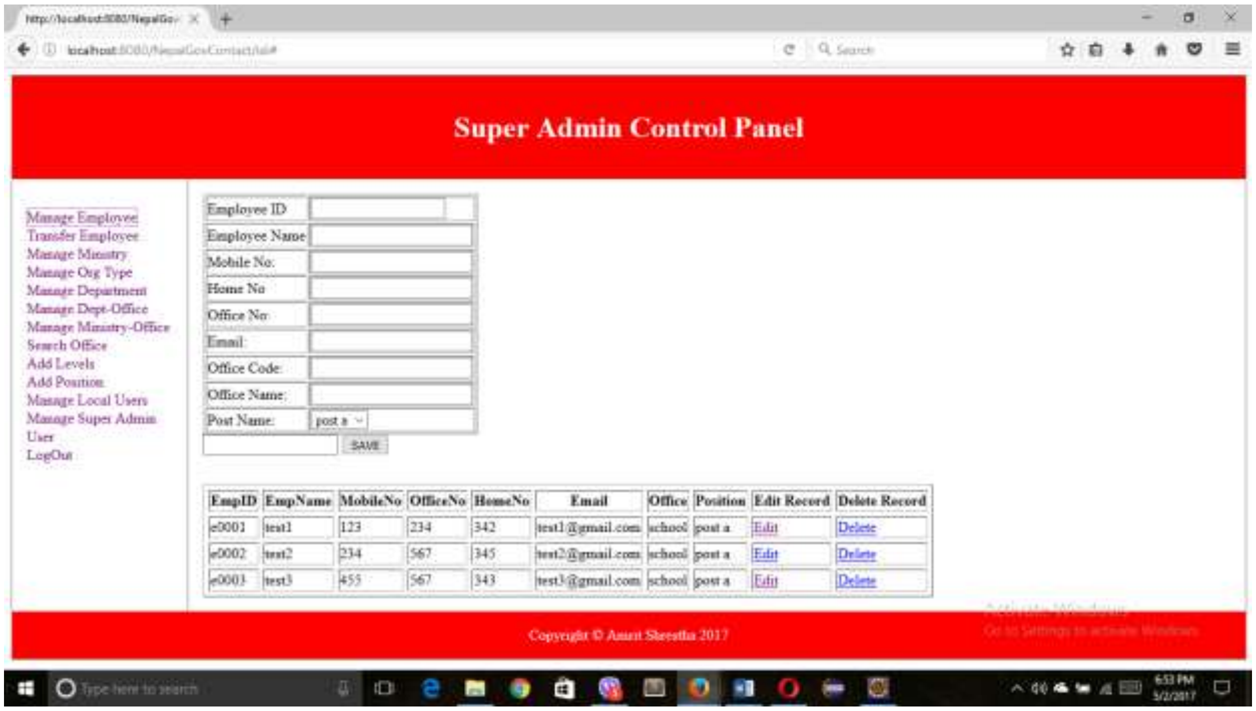
This App contains record of All Government Employee

All Employees of Government of Nepal are Displayed Here

Main Table

Copyright © Amrit Shrestha 2017

Figure 10:View of Index Page



Super Admin Control Panel

Manage Employee
Transfer Employee
Manage Ministry
Manage Org Type
Manage Department
Manage Dept-Office
Manage Ministry-Office
Search Office
Add Levels
Add Position
Manage Local Users
Manage Super Admin
User
LogOut

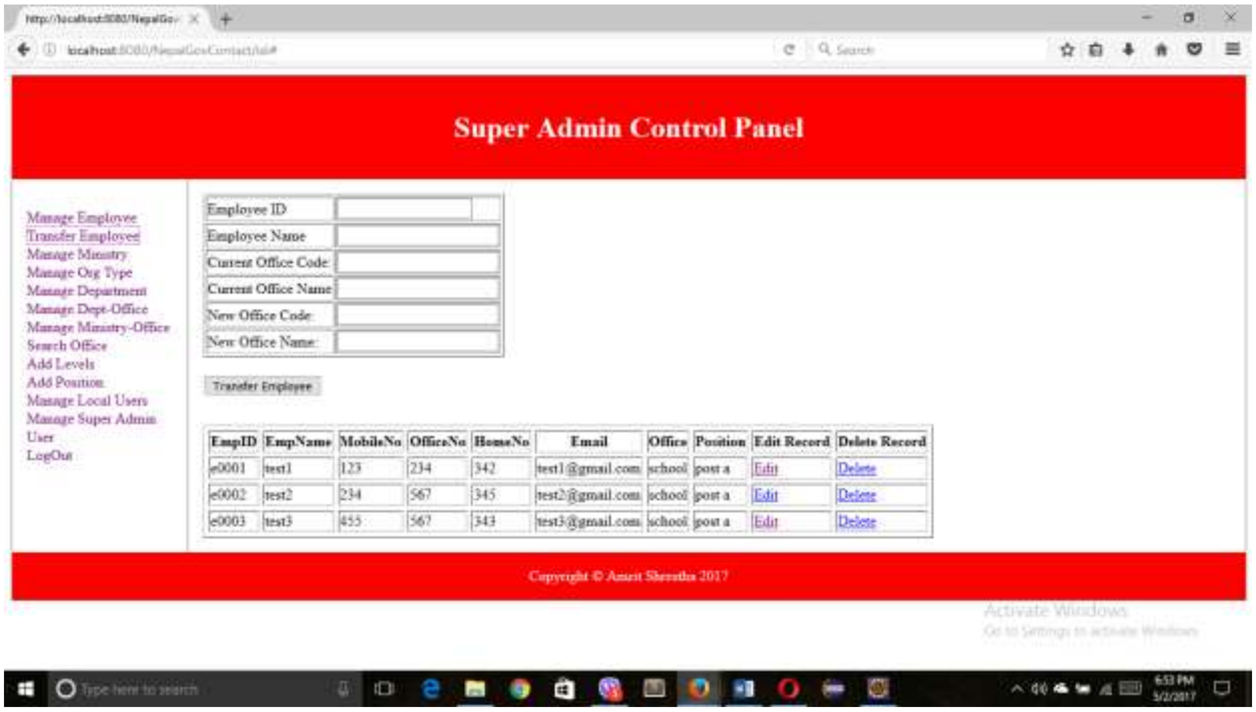
Employee ID	
Employee Name	
Mobile No:	
Home No	
Office No:	
Email:	
Office Code:	
Office Name:	
Post Name:	post a

SAVE

EmpID	EmpName	MobileNo	OfficeNo	HomeNo	Email	Office	Position	Edit Record	Delete Record
e0001	test1	123	234	342	test1@gmail.com	school	post a	Edit	Delete
e0002	test2	234	567	345	test2@gmail.com	school	post a	Edit	Delete
e0003	test3	455	567	343	test3@gmail.com	school	post a	Edit	Delete

Copyright © Amrit Shrestha 2017

Figure 11: Manage Employee view



Super Admin Control Panel

Manage Employee
Transfer Employee
Manage Ministry
Manage Org Type
Manage Department
Manage Dept-Office
Manage Ministry-Office
Search Office
Add Levels
Add Position
Manage Local Users
Manage Super Admin
User
LogOut

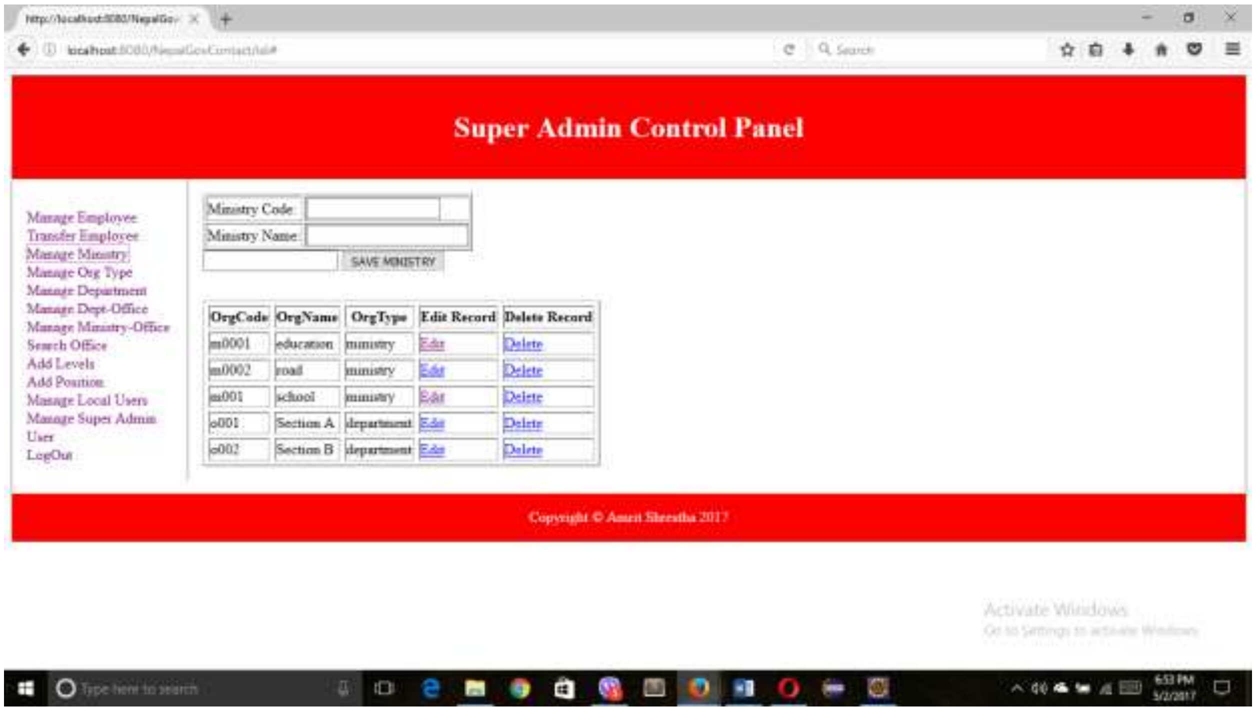
Employee ID	
Employee Name	
Current Office Code:	
Current Office Name:	
New Office Code:	
New Office Name:	

Transfer Employee

EmpID	EmpName	MobileNo	OfficeNo	HomeNo	Email	Office	Position	Edit Record	Delete Record
e0001	test1	123	234	342	test1@gmail.com	school	post a	Edit	Delete
e0002	test2	234	567	345	test2@gmail.com	school	post a	Edit	Delete
e0003	test3	455	567	343	test3@gmail.com	school	post a	Edit	Delete

Copyright © Amrit Shrestha 2017

Figure 12: Transfer Employee View



Super Admin Control Panel

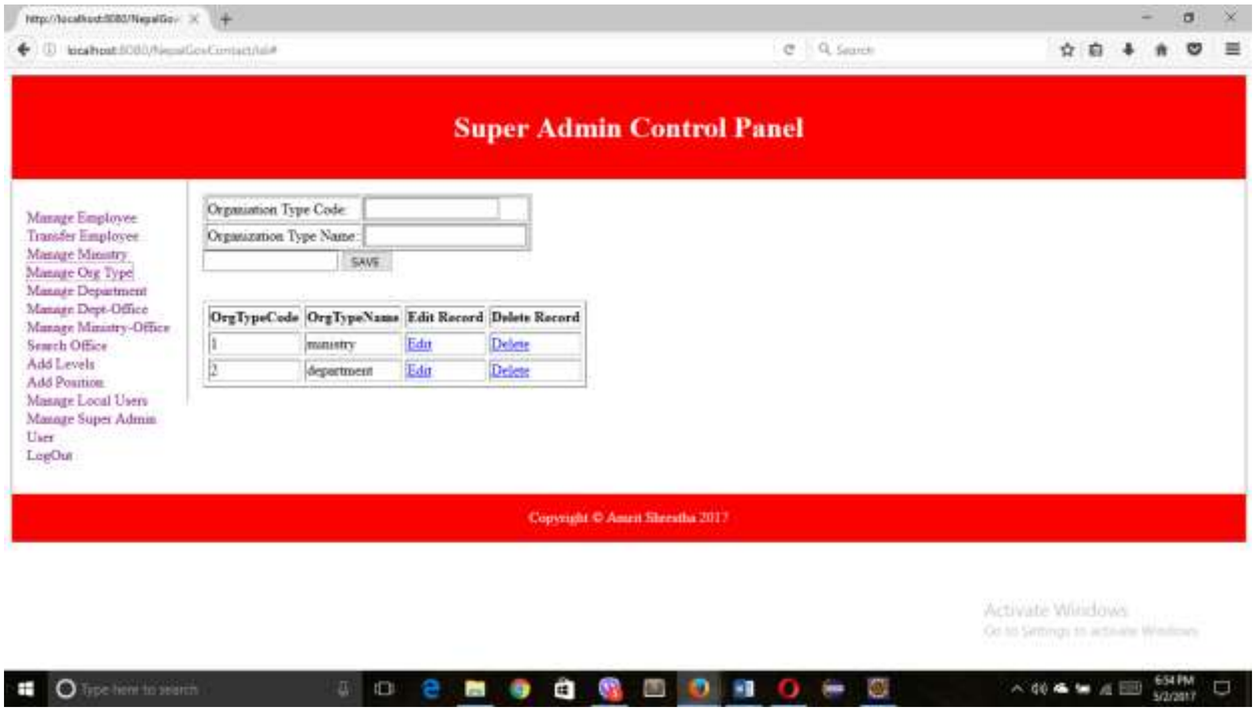
Manage Employee
Transfer Employee
Manage Ministry
Manage Org Type
Manage Department
Manage Dept-Office
Manage Ministry-Office
Search Office
Add Levels
Add Position
Manage Local Users
Manage Super Admin User
LogOut

Ministry Code:
Ministry Name:
SAVE MINISTRY

OrgCode	OrgName	OrgType	Edit Record	Delete Record
m0001	education	ministry	Edit	Delete
m0002	road	ministry	Edit	Delete
m001	school	ministry	Edit	Delete
o001	Section A	department	Edit	Delete
o002	Section B	department	Edit	Delete

Copyright © Amrit Shrestha 2017

Figure 13: Manage Ministry View



Super Admin Control Panel

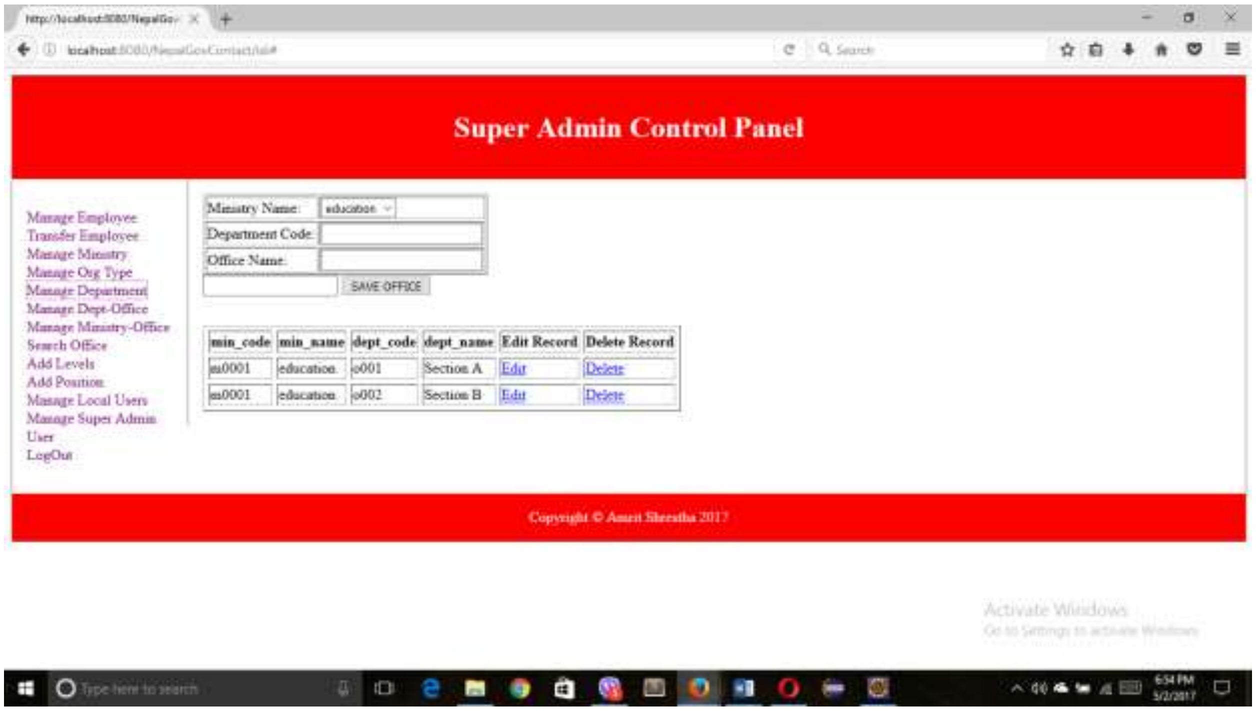
Manage Employee
Transfer Employee
Manage Ministry
Manage Org Type
Manage Department
Manage Dept-Office
Manage Ministry-Office
Search Office
Add Levels
Add Position
Manage Local Users
Manage Super Admin User
LogOut

Organization Type Code:
Organization Type Name:
SAVE

OrgTypeCode	OrgTypeName	Edit Record	Delete Record
1	ministry	Edit	Delete
2	department	Edit	Delete

Copyright © Amrit Shrestha 2017

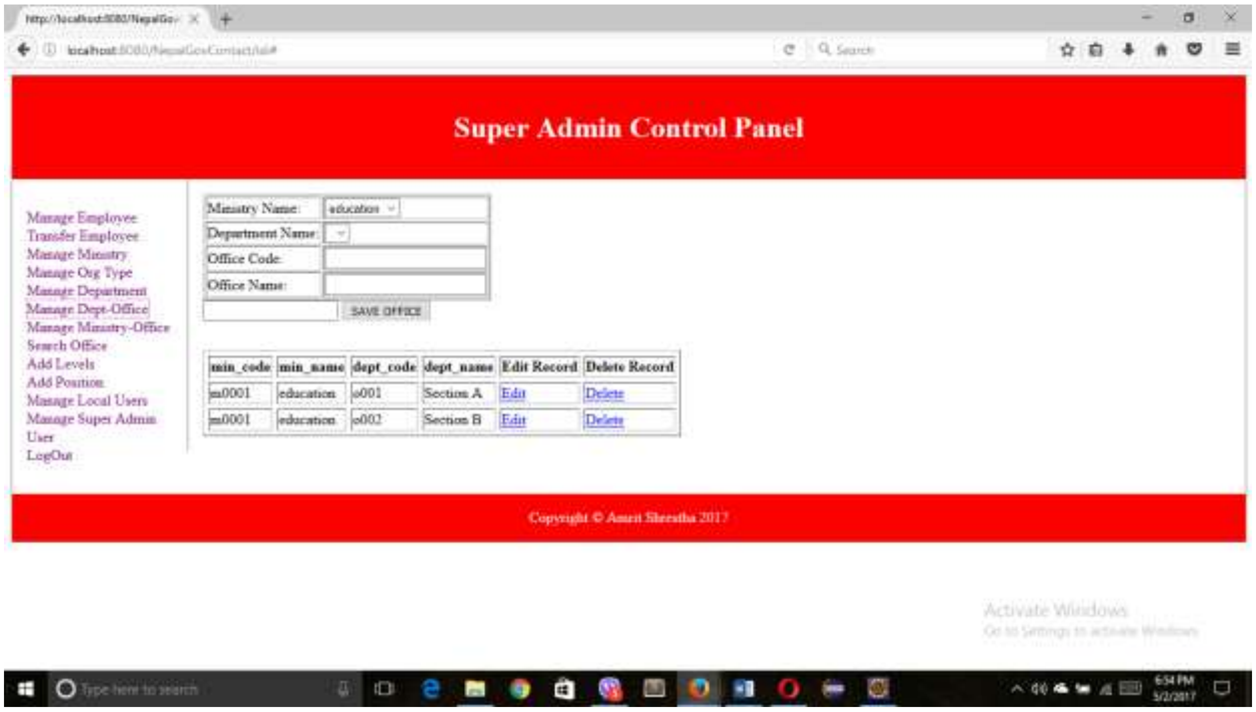
Figure 14: Manage Org Type View



Ministry Name:	education
Department Code:	
Office Name:	

min_code	min_name	dept_code	dept_name	Edit Record	Delete Record
m0001	education	o001	Section A	Edit	Delete
m0001	education	o002	Section B	Edit	Delete

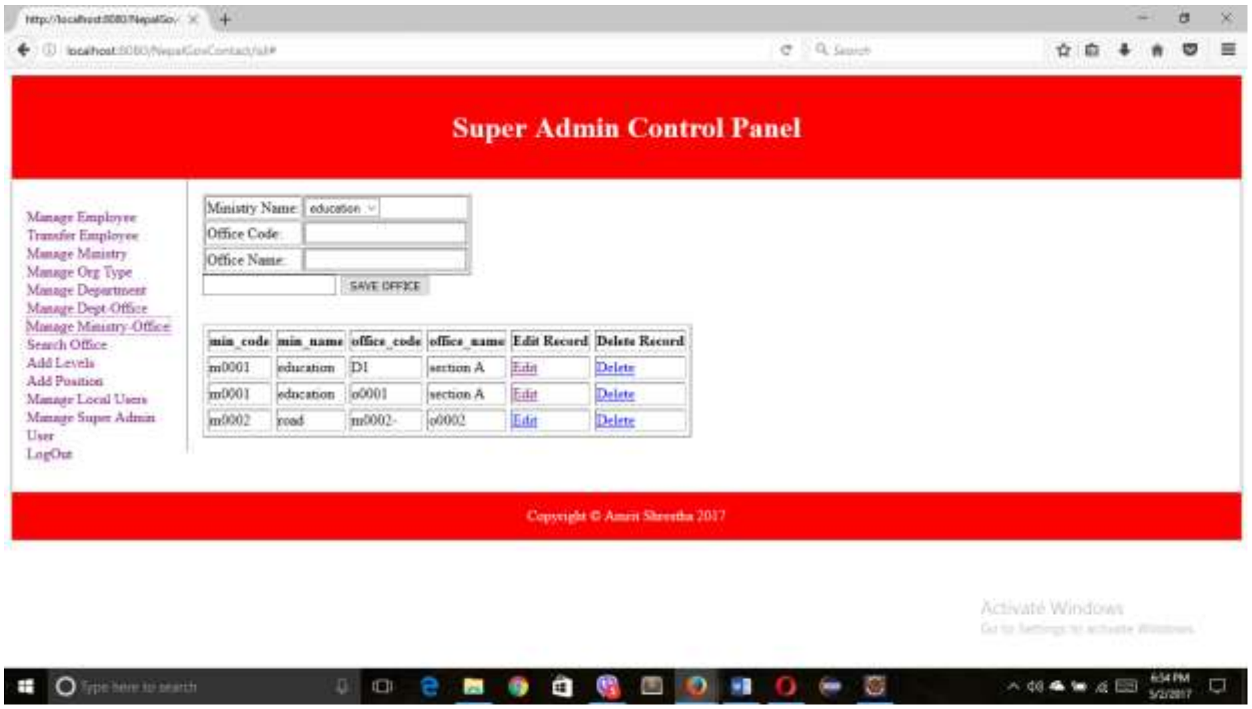
Figure 15: Manage Department View



Ministry Name:	education
Department Name:	
Office Code:	
Office Name:	

min_code	min_name	dept_code	dept_name	Edit Record	Delete Record
m0001	education	o001	Section A	Edit	Delete
m0001	education	o002	Section B	Edit	Delete

Figure 16: Manage Dept Office View



Super Admin Control Panel

- Manage Employee
- Transfer Employee
- Manage Ministry
- Manage Org Type
- Manage Department
- Manage Dept-Office
- Manage Ministry-Office
- Search Office
- Add Levels
- Add Position
- Manage Local Users
- Manage Super Admin User
- LogOut

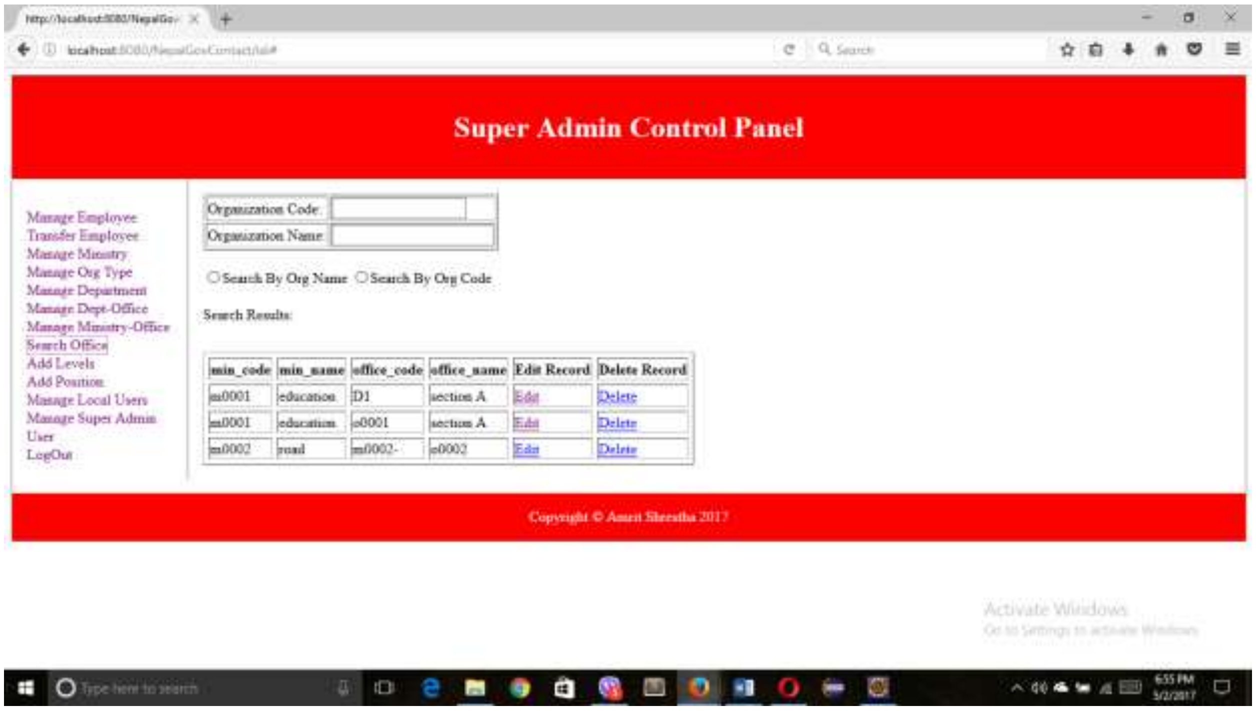
Ministry Name:	education
Office Code:	
Office Name:	

SAVE OFFICE

min_code	min_name	office_code	office_name	Edit Record	Delete Record
m0001	education	D1	section A	Edit	Delete
m0001	education	o0001	section A	Edit	Delete
m0002	road	m0002-	o0002	Edit	Delete

Copyright © Amrit Shrestha 2017

Figure 17: Manage Ministry office view



Super Admin Control Panel

- Manage Employee
- Transfer Employee
- Manage Ministry
- Manage Org Type
- Manage Department
- Manage Dept-Office
- Manage Ministry-Office
- Search Office
- Add Levels
- Add Position
- Manage Local Users
- Manage Super Admin User
- LogOut

Organization Code:	
Organization Name:	

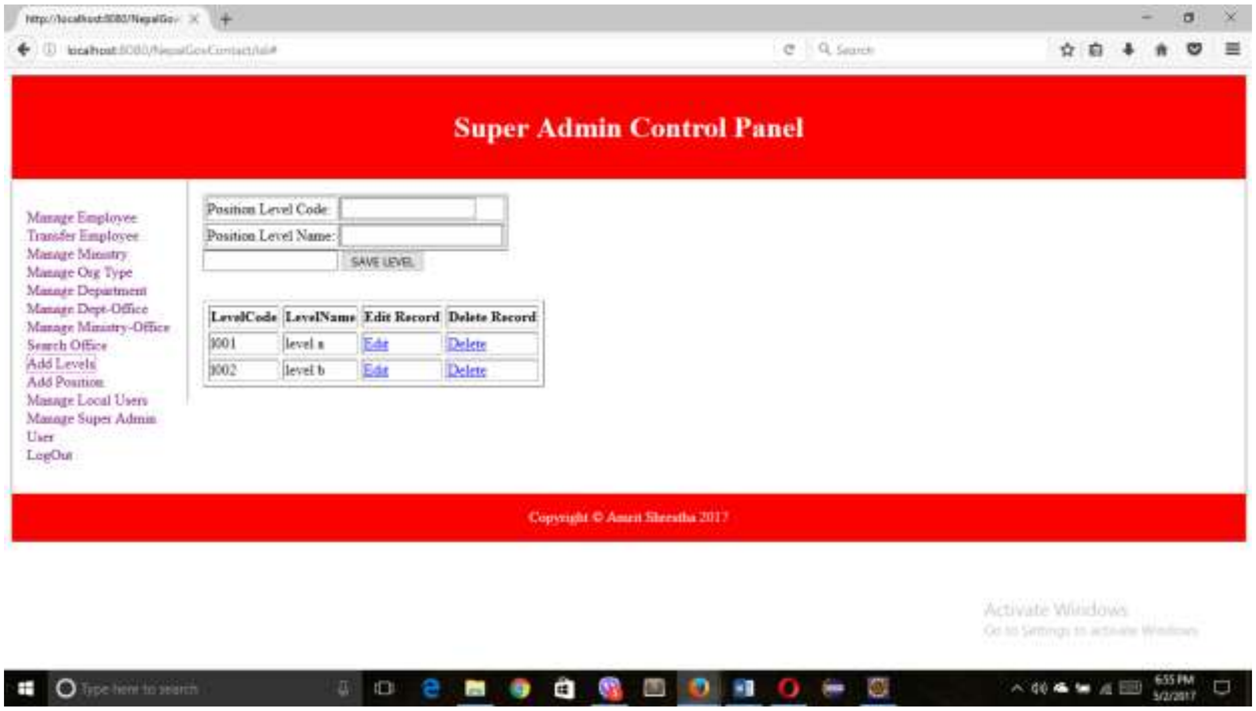
○ Search By Org Name ○ Search By Org Code

Search Results:

min_code	min_name	office_code	office_name	Edit Record	Delete Record
m0001	education	D1	section A	Edit	Delete
m0001	education	o0001	section A	Edit	Delete
m0002	road	m0002-	o0002	Edit	Delete

Copyright © Amrit Shrestha 2017

Figure 18: Search Office View



Super Admin Control Panel

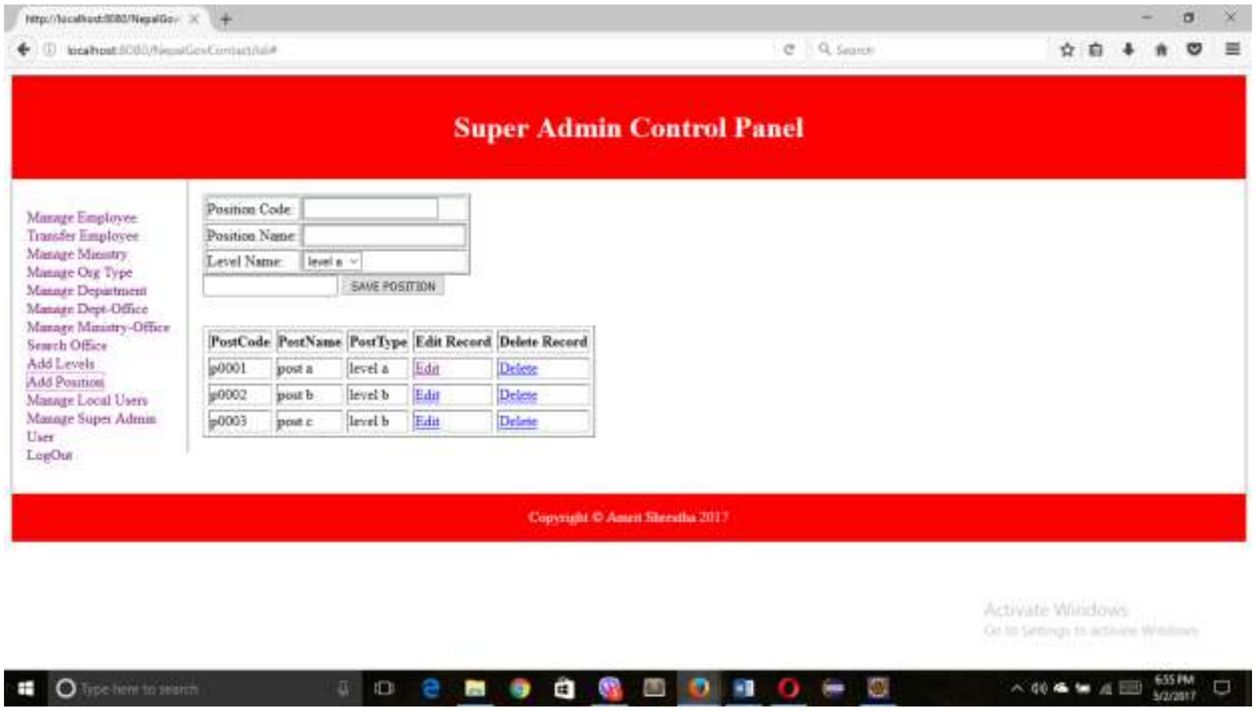
Manage Employee
Transfer Employee
Manage Ministry
Manage Org Type
Manage Department
Manage Dept-Office
Manage Ministry-Office
Search Office
Add Levels
Add Position
Manage Local Users
Manage Super Admin
User
LogOut

Position Level Code:
Position Level Name:

LevelCode	LevelName	Edit Record	Delete Record
l001	level a	Edit	Delete
l002	level b	Edit	Delete

Copyright © Amrit Shrestha 2017

Figure 19: Add Level View



Super Admin Control Panel

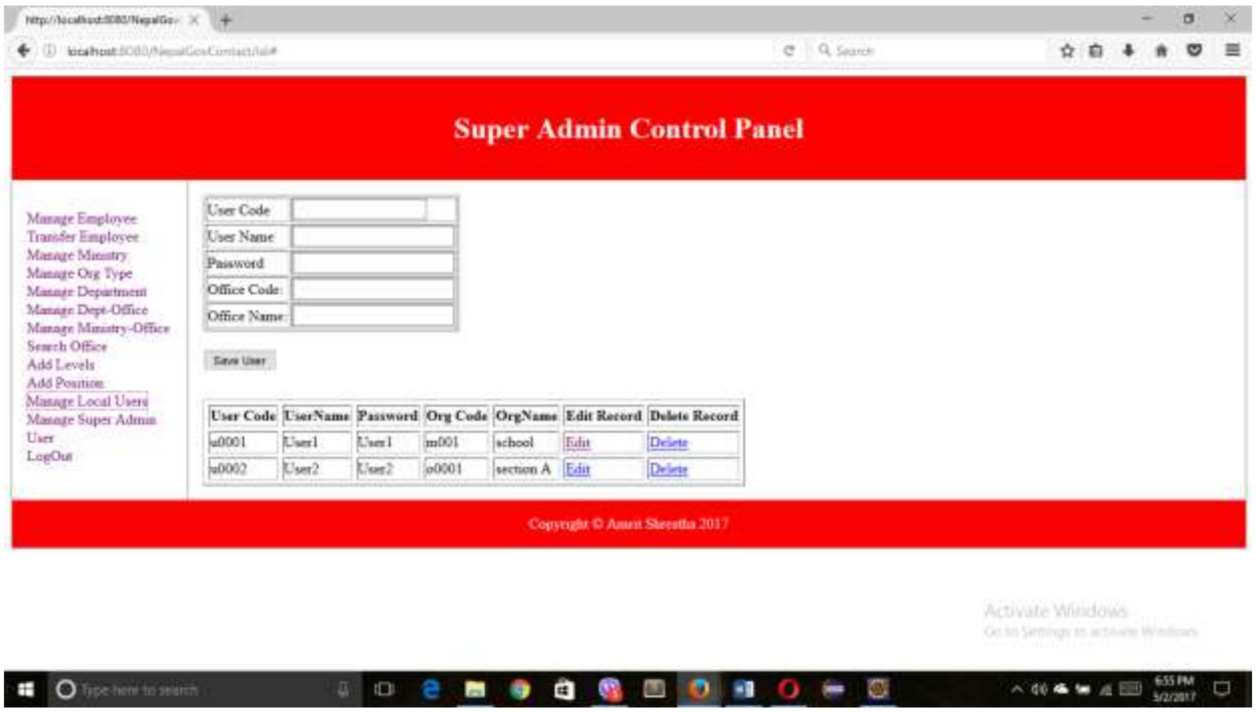
Manage Employee
Transfer Employee
Manage Ministry
Manage Org Type
Manage Department
Manage Dept-Office
Manage Ministry-Office
Search Office
Add Levels
Add Position
Manage Local Users
Manage Super Admin
User
LogOut

Position Code:
Position Name:
Level Name: level a

PostCode	PostName	PostType	Edit Record	Delete Record
p0001	post a	level a	Edit	Delete
p0002	post b	level b	Edit	Delete
p0003	post c	level b	Edit	Delete

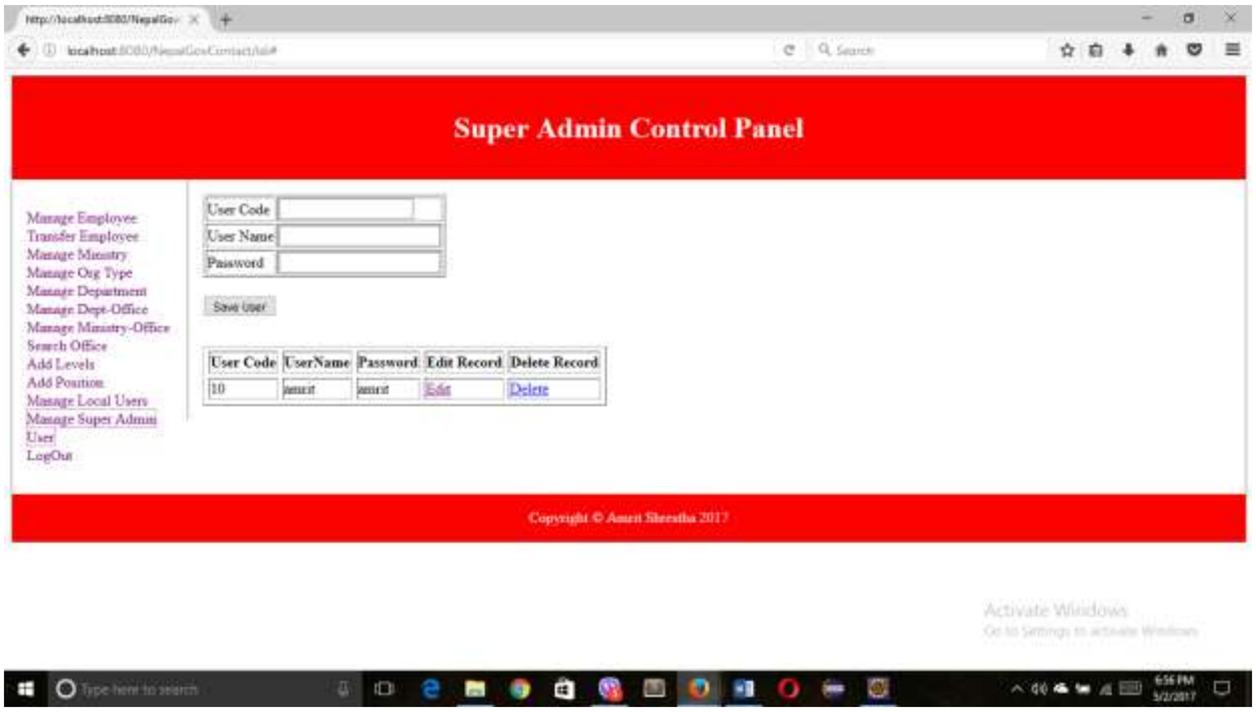
Copyright © Amrit Shrestha 2017

Figure 20: Add Position View



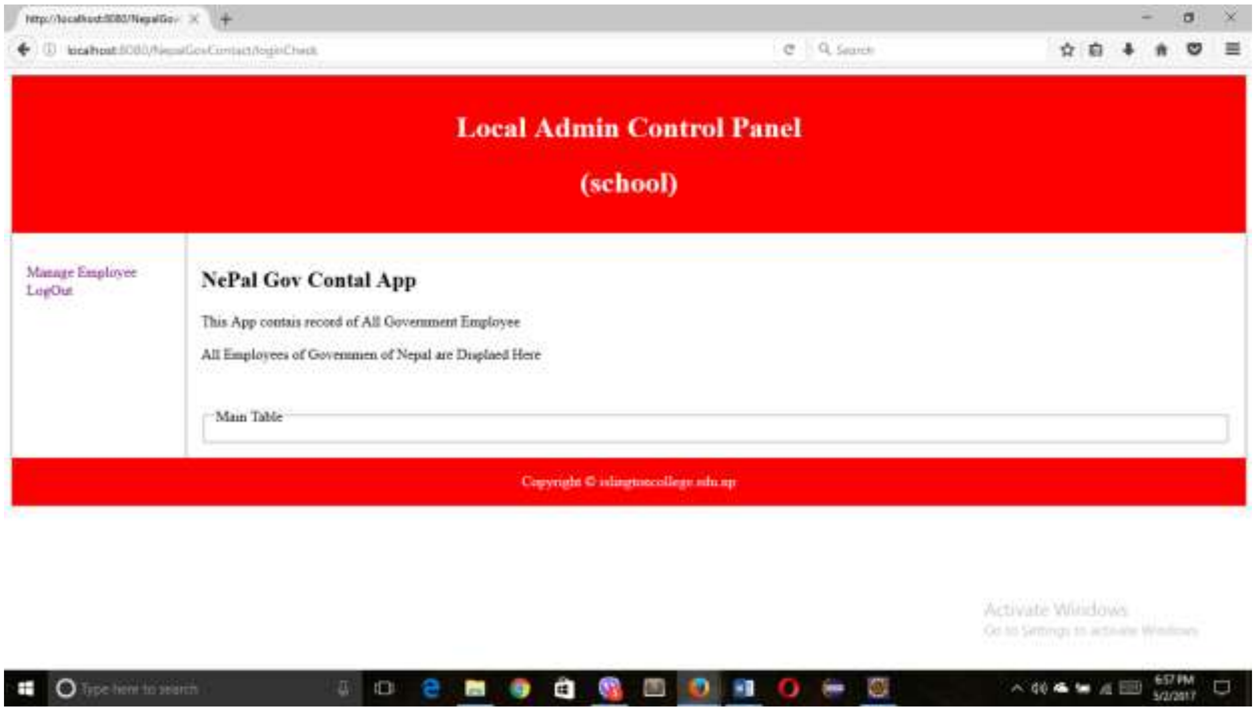
User Code	UserName	Password	Org Code	OrgName	Edit Record	Delete Record
u0001	User1	User1	m001	school	Edit	Delete
u0002	User2	User2	o0001	section A	Edit	Delete

Figure 21: Manage Local User View



User Code	UserName	Password	Edit Record	Delete Record
10	amrit	amrit	Edit	Delete

Figure 22: Manage Super Admin View


Local Admin Control Panel

(school)

Manage Employee
LogOut

NePal Gov Contal App

This App contais record of All Government Employee

All Employees of Governmen of Nepal are Displaed Here

Main Table

Copyright © islingtoncollege.edu.np

Figure 23: View of Local Admin