Zagazig University

Faculty of computer and informatics

Department of computer science

Graduation Project Documentation

Pharmacy management android application

Supervisor:

Dr.Amr Mohammed AbdEllatif

By assistance of:

Eng.Mahmmoud Mahdi

Team Member:-

1. Saharheshamalialsaka
2. Amira Mohamed shawky
3. AmrAtef Ibrahim Elbaz
4. Ahmed FathyBendary
5. Mahmoud Ahmed Mohamed Shtayeh
6. Ashraf Tarek Ali Ismael Nasr

Abstract:

Mobile phones have been used in all real world fields today.

So we want to provide this android app to Pharmacists to help them manage their pharmacies instead of using desktops or laptops so they only need their mobile phone.

They can deal with companies and employees by our product.

Acknowledgement:-

- **In every case ……. Good worse…….. We must thank the great merciful Allah for all gifts and blessing that we have been given starting with the blessings of Islam.**

- **We would like also to thank our supervisor : Dr.AmrAbdEllatif**

  Who directed us to new color of science, drawing and planning most of all our steps,providing us good resources and references for finishing that work.

- **We would like also to thank : Eng. Mahmmoud Mahdi**

  Who helped us in solving the problems and difficulties that we face and he was step by step with us to finish that work.

- **We would like also to thank everyone helped us through our gradual education through the four years especially the staff members of Computer Science department.**

Content list:

Acknowledgement 4

Chapter 1:

Section 1.1: Introduction. 11

Section 1.2: Methodology:.....12

1.2.1Android devices. 12

1.2.1.1 What is android devise. 12

1.2.1.2 Why android devise. 13

1.2.2 Wireless LAN network. 13

1.2.3 Printer. 13

Chapter 2:

Section 2.1:Mainwork (plane phase): 14

2.1.1 Divide application main work 14

2.1.1.1 Learning android developing basics. .. 14

2.1.1.2 Freeline application 14

Section 2.2: Project requirements 15

2.2.1 Functional requirements for admin. 15

2.2.2 Functional requirements for employee. …….. 15

2.2.3 Non-functional requirements. …….. 15

2.2.4 Security requirements.……. 16

2.2.5 Data requirements.……... 16

2.2.6 User and human factor.……... 16

2.2.7 Domain analysis.……... 16

2.2.8 System developing requirement. ……. 17

Section 2.3: Analysis of the project..... 19

2.3.1 Use case diagram. 19

2.3.2 Sequence diagram. 22

2.3.3 Activity diagram. 25

2.3.4 ER diagram. …… 27

Chapter 3:

Section 3.1: Our work: 28

3.1.1Android code. 28

3.1.2Web service code.....46

Chapter 4:

Section 4.1: How admin usesthis app. 54
Section 4.2:How user employee this app. 73

List of Figures

Chapter 2

Figure 1: Applications languages and API's used (1).....17
Figure 2:Applications languages and API's used (2).....18
Figure 3:Use Case Diagram.....21
Figure 4:Sequence diagram for admin. 23
Figure 5:Sequence diagram for employee. 24
Figure 6: Activity diagram. …….. 26
Figure 7: ER diagram. ...…….... 27

Chapter 3

Figure 8.1:Login Code 1. 28
Figure 8.2:Login Code 2. 29
Figure 8.3:Login Code 3. 29
Figure 8.4:Login Code 4. 30
Figure 8.5:Login Code 5. 31
Figure 8.6:Login Code 6. 31
Figure 9.1:Navigation Code 1.....32

Figure 9.2:Navigation Code 2. 33
Figure 9.3:Navigation Code 3. 34
Figure 9.4:Navigation Code 4. 35
Figure 9.5:Navigation Code 5. 36
Figure 10.1:AddEmployee Code 1. 37
Figure 10.2:AddEmployee Code 2. 37
Figure 10.3:AddEmployee Code 3. 38
Figure 10.4:AddEmployee Code 4. 38
Figure 11.1:UpdateEmployee Code 1. 39
Figure 11.2:UpdateEmployee Code 2. 40
Figure 12:ForgotPassword Code 1. 41
Figure 13.1:SellBill Code 1. 41
Figure 13.2:SellBill Code 2. 42
Figure 13.3:SellBill Code 3. 43
Figure 13.4:SellBill Code 4. 43
Figure 13.5:SellBill Code 5. 44
Figure 13.6:SellBill Code 6. 44
Figure 14:CustomDialog Code 1. 45
Figure 15:constants Code 1. 46
Figure 16:init.php Code 1. 47
Figure 17:CheckLogin.php Code 1. 47
Figure 18:Config.php Code 1. 48
Figure 19:DbConnect.php Code 1. 48
Figure 20.1:AddEmp.php Code 1. 49

Figure 20.2:AddEmp.phpCode 2. 50
Figure 21:FetchEmps.phpCode 1. 50
Figure 22:UpdateEmp.php Code 1. 51
Figure 23:DeleteEmp.phpCode 1. 52
Figure 24:AddBuyBil.php Code 1. 53

Chapter 4

Figure 25:Login Activity. 54
Figure 26:ForgetPassword Activity. 55
Figure 27:Admin Main Activity. 56
Figure 28:AddEmployee Fragment. 57
Figure 29:DeleteEmployee Fragment. 58
Figure 30:UpdateEmployee Fragment. 59
Figure 31:AddDrug Fragment. 60
Figure 32:DeleteDrug Fragment. 61
Figure 33: UpdateDrug Fragment. 62
Figure 34.1: Add Sell Bill Fragment. 63
Figure 34.2: Add Sell Bill Fragment. 64
Figure 34.3:Add Sell Bill Fragment. 65
Figure 34.4:Add Sell Bill Fragment. 66
Figure 35:Add Buy Bill Fragment. 67
Figure 36:Add Insurance Company Fragment. 68
Figure 37:Delete Insurance Company Fragment. 69

Figure 38: Update Insurance Company Fragment. 70
Figure 39:Add Drug Company Fragment. 71
Figure 40:Delete Drug Company Fragment. 72
Figure 41:Update Drug Company Fragment. 73
Figure 42:Employee Main Activity. 74

Chapter 1

Section1.1:-

Introduction

Mobile phones are everywhere and it makes the life easier in many fields especially phones which use android OS. With your mobile you can manage, sell, and buy what you want. They save our time and money because the mobile phone can do many functions instead of many devices. So our graduation project "Pharmacy Management"is Android App

The system's goals :

- **Help pharmacist to manage his pharmacy in terms of :**

1. **Manage employees.**
2. **Sell and buy drugs.**
3. **Interaction with drug companies.**
4. **Connect pharmacies together.**
5. **Minimize the costs he needs to manage the pharmacy.**

Section1.2:-

Methodology

- **We must have these components in our project:**

1. **One terminal android device such as: phones or tablets.**
2. **Good WIFI internet connection.**
3. **Printer.**

1.2.1Android devices:-

What is android device?

Android is designed primarily for touch screen mobile devices such as smart phones and tablet, with variants for televisions (Android TV), cars (Android Auto), and wrists (Android Wear).

Why android devices?

Because it is the world's most popular mobile OS More than a billion phones and tablets around the world run Android. It's customizable, easy to use. Android phones and tablets work perfectly with all your favorite apps.

1.2.2 Wireless LAN network:

A wireless LAN (WLAN) provides network connectivity between devices and server, also known as stations, by using radio as the communication medium.

It is the same as the traditional LAN but they have a wireless interface. WLANs provide high speed data communication in small areas such as a building or an office. It allows users to move around in a confined area while they are still connected to the network...

1.2.3 Printer:

We use itto print bills.

Chapter 2

Section2.1:-

Main work (plan phase)

2.1.1 We can divide our application's main work into:

2.1.1.1 Learning android developing basics:

Before developing our application we needed to read about android because knowledge about android developing was not sufficient enough to start developing the project.

2.1.1.2 Free line application:

We needed free line application to enable the communication between the server and app.

Section 2.2:-

Project requirements

2.2.1 Functional requirements for admin

- **Update the database.**
- **Administering the server.**
- **Manage the employee.**
- **Manage drugs.**
- **Manage bills.**
- **Manage insurance company.**
- **Manage drug company.**

2.2.2 Functional requirements for employee

- **Add sell bill.**
- **Add buy bill.**

2.2.3 Non-functional requirements

- **Any interaction between user and the system should not exceed 2 sec.**
- **System available for use 24 hours per day,365 days per year.**
- **Only direct manager (admin) can see personal records of employees.**

2.2.4 Security requirements

- **Secure server connection.**
- **Secure logging session.**

2.2.5 Datarequirement

- **Data integrity.**

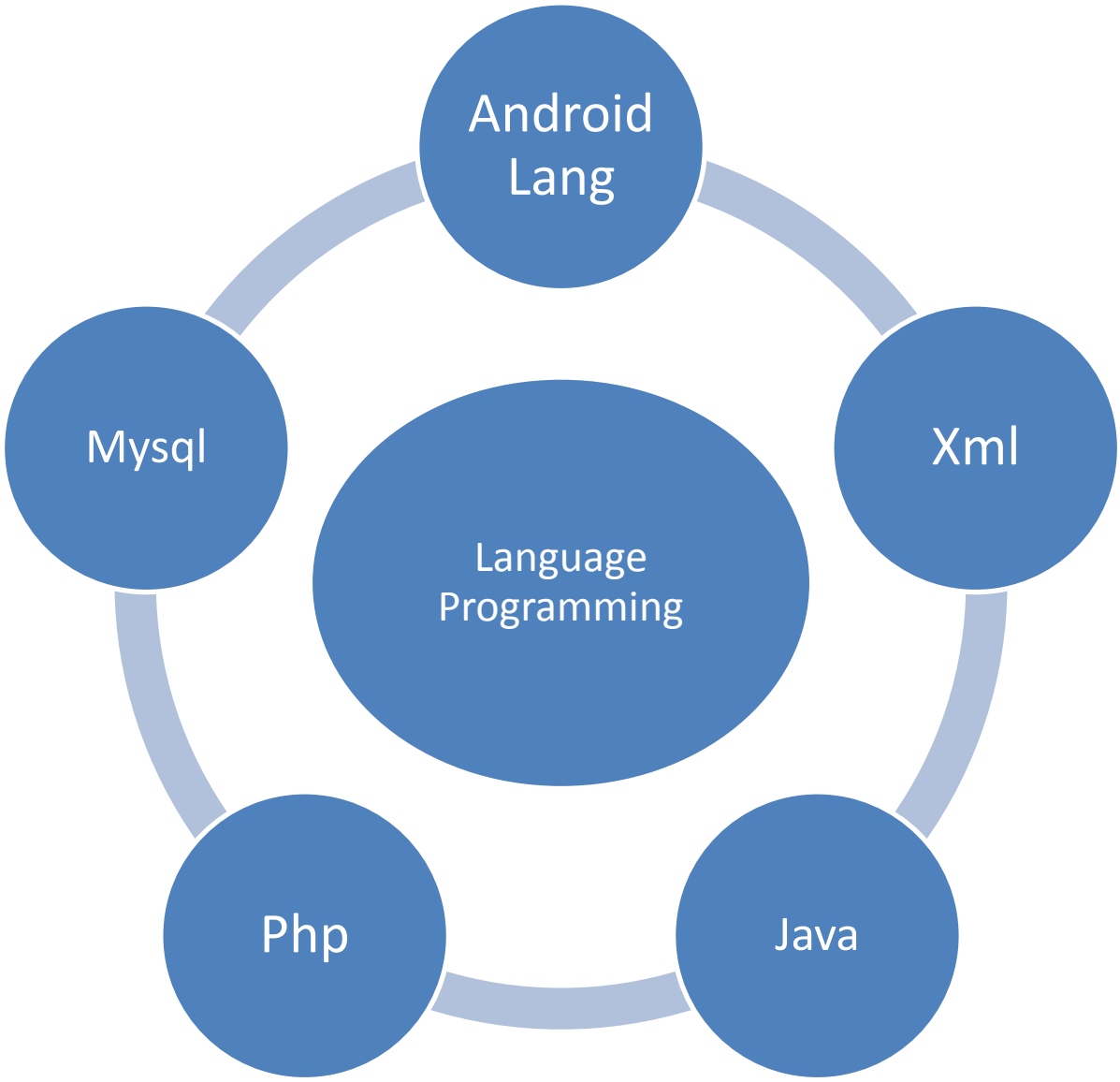
2.2.6 user and human factor

- **Acting with user interface.**

2.2.7 Domain analysis

- **Stakeholder: manager of pharmacy.**
- **Users: manager of pharmacy, employee in pharmacy.**

2.2.8 System developing requirements



(Figure 1) Applications languages and API's used (1).

(Figure 2) Applications languages and API's used (2).

Section 2.3:-

Analysis of the project

♦ **Is the analysis of the project in three phases:-**

1. **Use case diagram.**
2. **Sequence diagram.**
3. **Activity diagram.**
4. **ER diagram.**

2.3.1 Use case diagram:-

Use case diagrams overview the usage requirements for a system. They are useful for presentations to management and/or project stakeholders, but for actual development you will find that use cases provide significantly more value because they describe "the meat" of the actual requirements.

➢ **Use Case.**

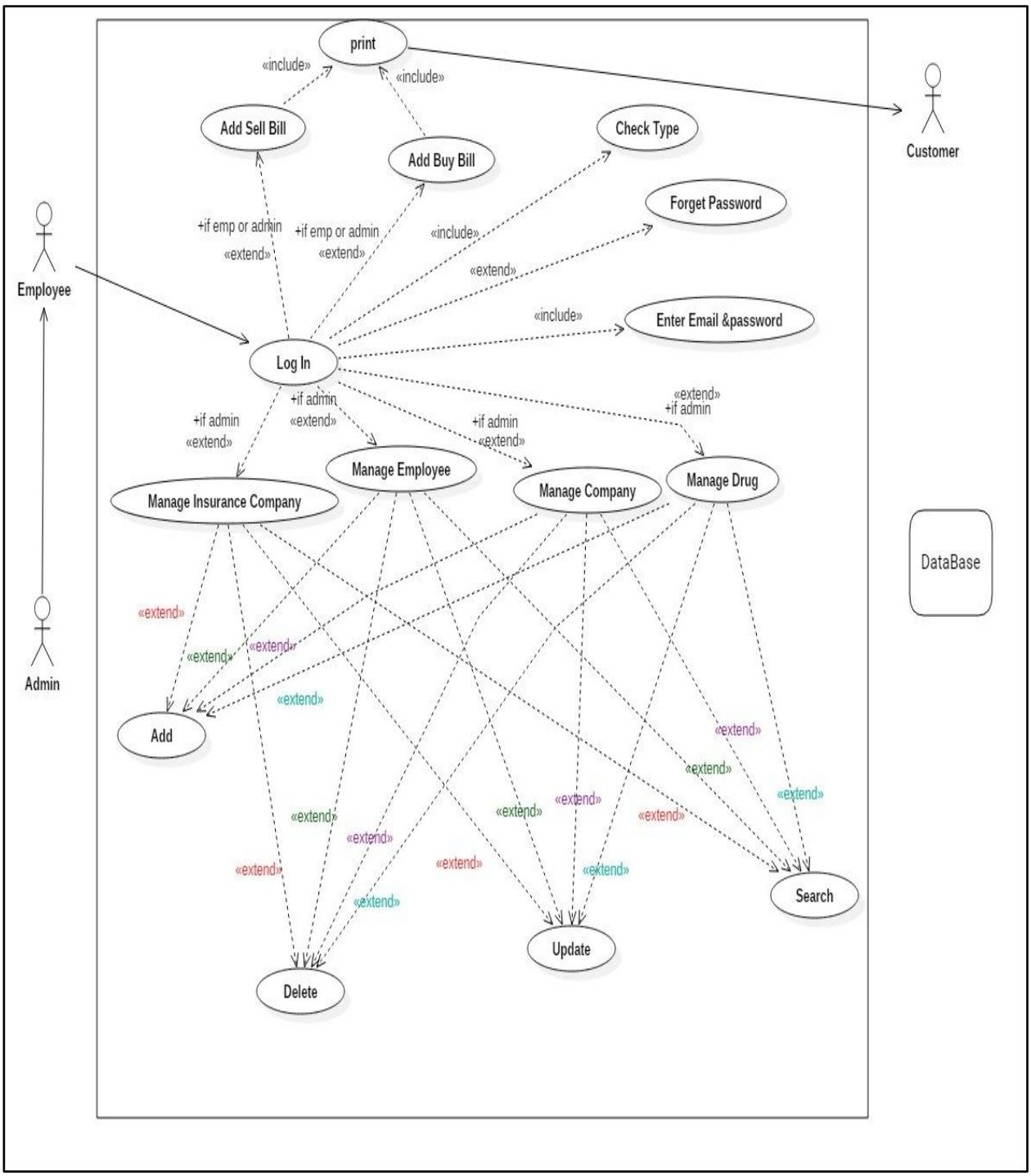
A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

➢ Actors.

An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures.

➢ Association.

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case.



(Figure 3)Use Case Diagram.

2.3.2Sequence diagram:-

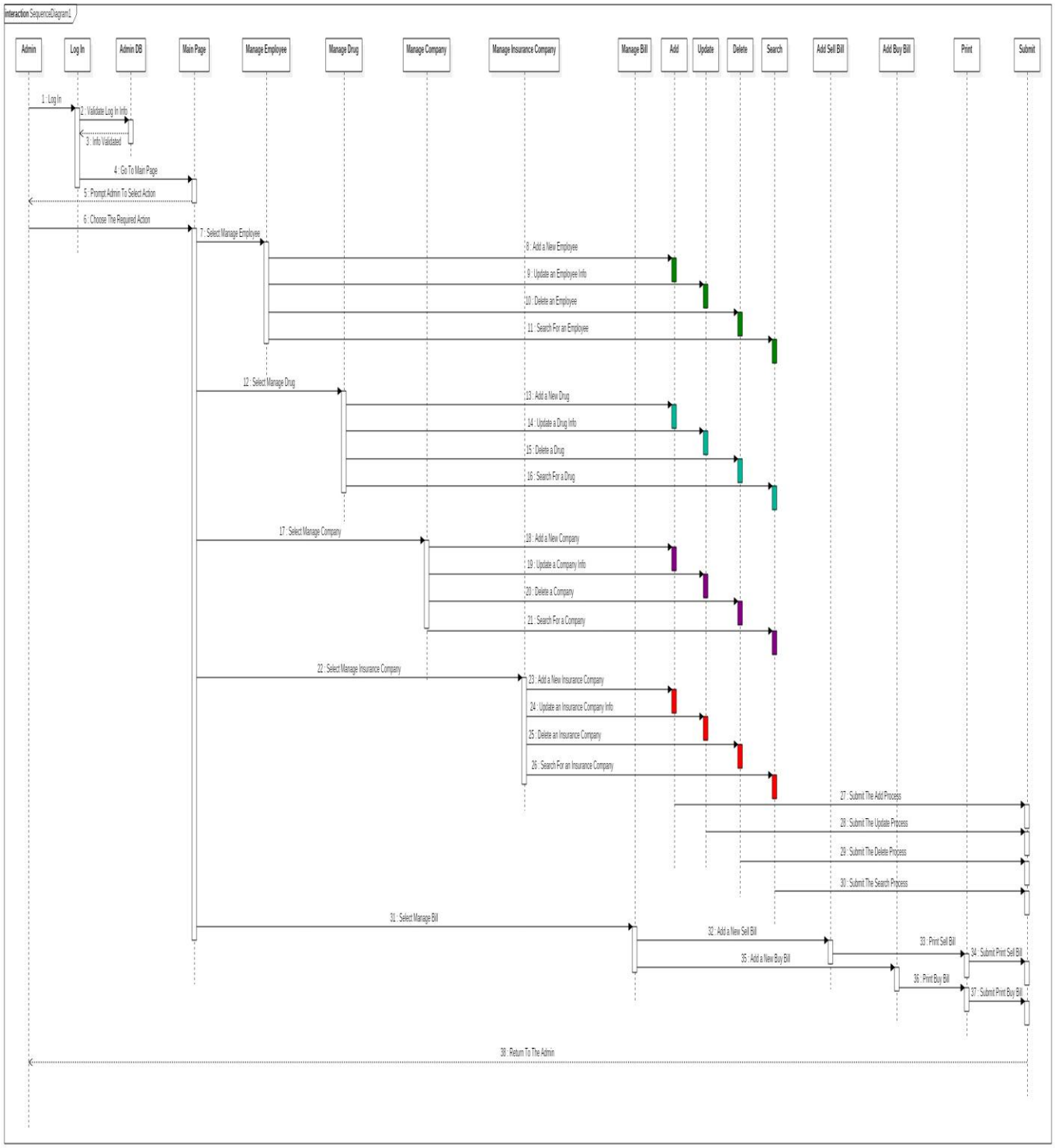
- **Sequence Diagrams are interaction diagrams that detail how operations are carried out.**
- **Interaction diagrams model important runtime interactions between the parts that make up the system.**

♦ What do Sequence Diagrams model?

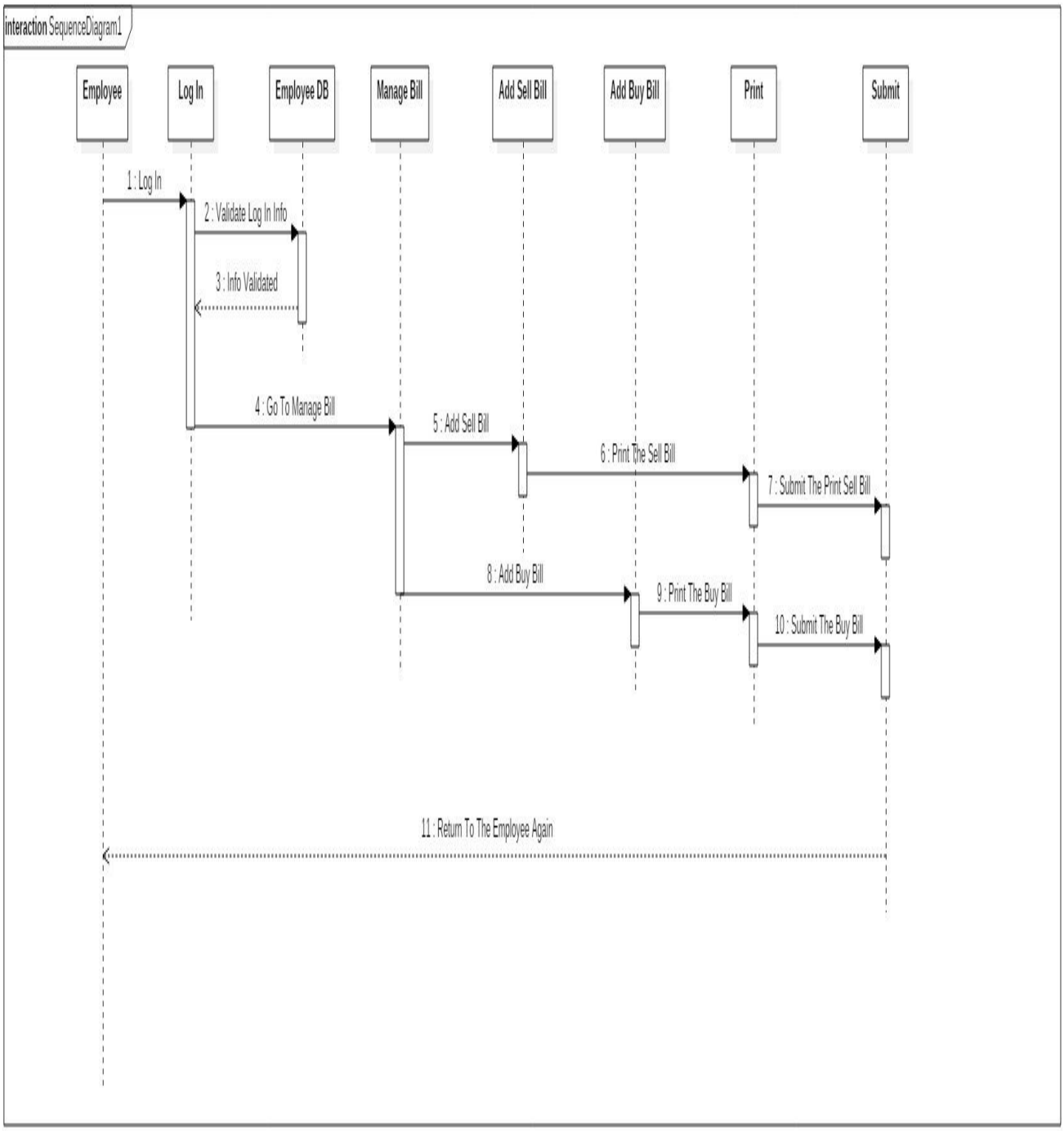
- **Capturethe interaction between objects in the context of a collaboration.**
- **Show the order of the interaction visually by using the vertical axis of the diagram to represent time what messages are sent and when.**
- **Show elements as they interact over time, showing interactions or interaction instances.**

♦ When to Use Sequence Diagrams

- **You should use sequence diagrams when you want to look at the behavior of several objects within a single use case.**
- **Sequence diagrams are good at showing collaborations among the objects.**
- **They are not so good at precise dentition of behavior**



(Figure 4)Sequence diagram for admin.

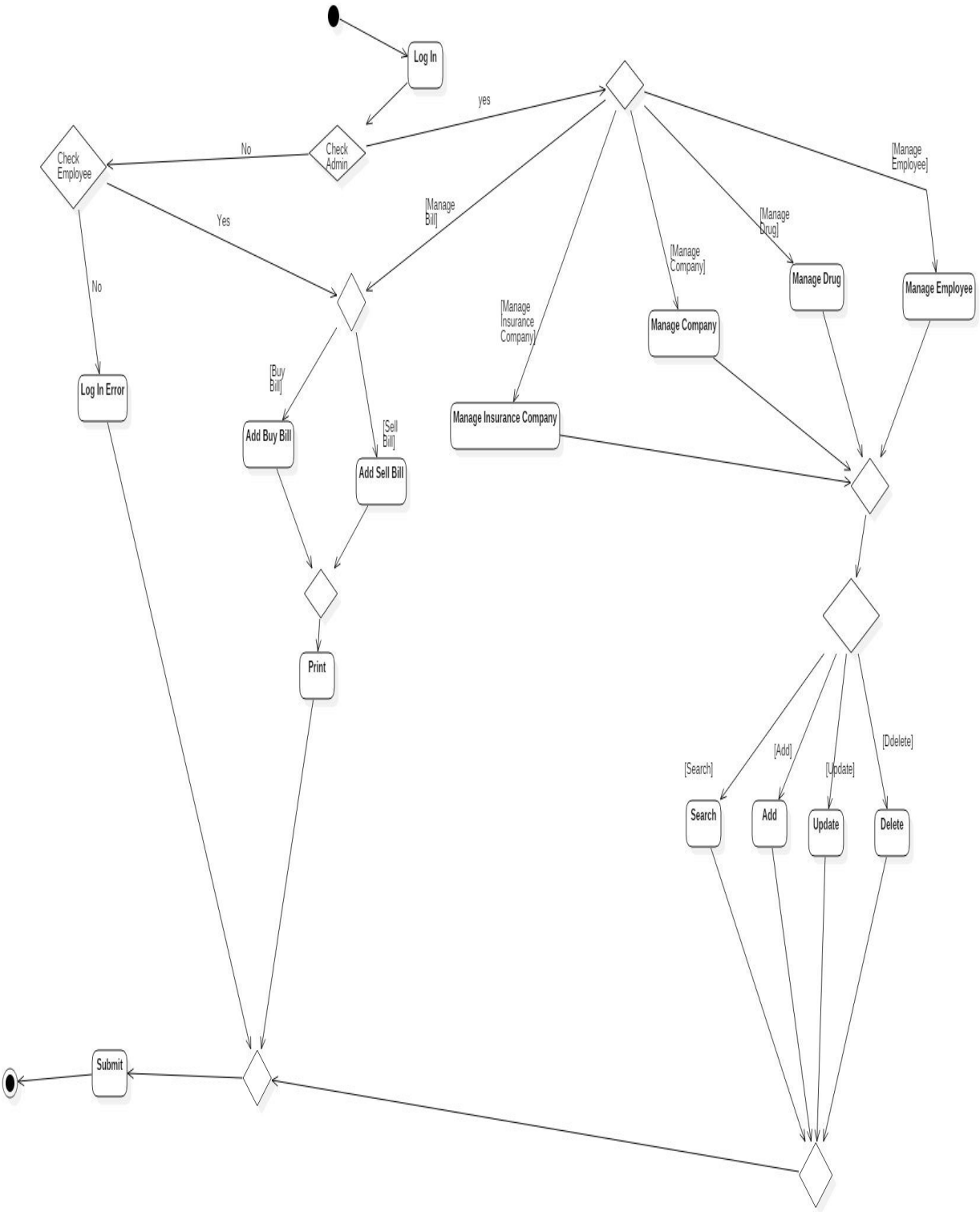


(Figure 5)Sequence diagram for employee.

2.3.3Activity diagram:

- **An Activity diagram is similar to a flowchart. Activity diagrams and State chart diagrams are related. While a State chart diagram focuses attention on an object undergoing a process (or on a process as an object), an Activity diagram focuses on the flow of activities involved in a single process. The Activity diagram shows how these single-process activities depend on one another.**
- **Activity diagrams are helpful in the following phases of a project:**

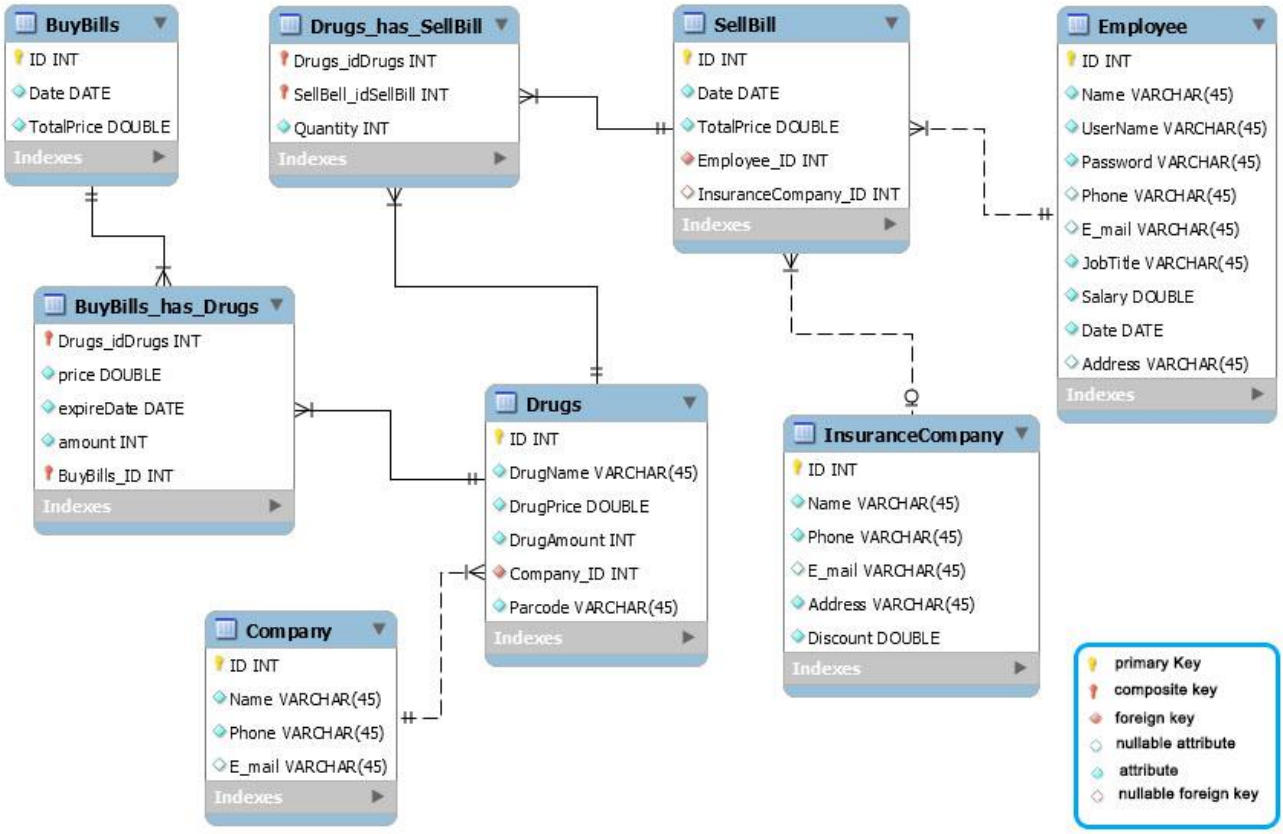
  - **Before starting a project, you can create activity diagrams to model the most important workflows.**
  - **During the requirements phase, you can create activity diagrams to illustrate the flow of events that the use cases describe.**
  - **During the analysis and design phases, you can use activity diagrams to help define the behavior of operations.**



(Figure 6)Activity diagram.

2.3.4 ER diagram:-

- **An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research.**



(Figure 7)ER diagram.

Chapter 3

Section3.1:-

Our work

3.1.1 Android code:

```
if (view.getId() == R.id.login_loginBtn) {
    userName = userNameTxt.getText().toString();
    password = passwordTxt.getText().toString();
    if (!userName.isEmpty() && !password.isEmpty()) {
        checkLogin();
    } else {
        String msg;
        if (userName.isEmpty() && password.isEmpty()) {
            msg = "please enter the user name and password";
        } else if (userName.isEmpty()) {
            msg = "please enter the user name ";
        } else {
            msg = "please enter the password";
        }
        Toast.makeText(this, msg, Toast.LENGTH_LONG).show();
    }
```

(Figure 8.1)Login Code 1.

In (Figure 8.1)we validate the input date from login activity when we press login button, then we check the data in server from method checkLogin().

```
} else if (view.getId() == R.id.login_exitBtn) {
    Intent intent = new Intent(Intent.ACTION_MAIN);
    intent.addCategory(Intent.CATEGORY_HOME);
    intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(intent);
} else if (view.getId() == R.id.login_forgotPassword) {
    startActivity(new Intent(this, ForgetPassword.class));
    finish();
}
```

(Figure 8.2)Login Code 2.

In (Figure 8.2) when we press exit button it exit the login activity, and when we press forgotPassword text it open ForgetPassword activity.

```
private void checkLogin() {
    progress = ProgressDialog.show(MainActivity.this, "Please Wait...", "Check logging...", true);
    StringRequest stringRequest = new StringRequest(
            Request.Method.POST,
            constants.CHECK_LOGIN_URL,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    try {
                        progress.dismiss();
                        Log.i("respo", String.valueOf(response));
                        JSONObject data = new JSONObject(response);
```

(Figure 8.3)Login Code 3.

In (Figure 8.3)this is the checkLogin() method use URL to server to connect with it and send or receive data.
I receive data in onResponse(String response) method in json file.

```
if (!data.getBoolean("error")) {
    name = data.getString("Name");
    empId = data.getString("ID");
    jobTitle = data.getString("JobTitle");
    if (jobTitle.equals("admin") && !jobTitle.isEmpty()) {
        check_login.login = true;
        check_login.type = 0;
        check_login.name = name;
        check_login.userName = MainActivity.this.userName;
        check_login.empPassword = MainActivity.this.password;
        check_login.loggedin("login", "0", MainActivity.this.getApplicationContext().getFilesDir().getPath(),
                userName, name, password);
        startActivity(new Intent(MainActivity.this, NavigationMain.class));
        finish();
    } else if (jobTitle.equals("emp") && !jobTitle.isEmpty()) {
        //  Toast.makeText(MainActivity.this, "after else if", Toast.LENGTH_SHORT).show();
        check_login.login = true;
        check_login.type = 1;
        check_login.name = name;
        check_login.userName = MainActivity.this.userName;
        check_login.empPassword = MainActivity.this.password;
        check_login.loggedin("login", "1", MainActivity.this.getApplicationContext().getFilesDir().getPath(),
                userName, name, password);
        startActivity(new Intent(MainActivity.this, NavigationMain.class));
        finish();
    } else {
        Toast.makeText(getApplicationContext(), "Invalid Username or Password", Toast.LENGTH_LONG).show();
    }
}
```

(Figure 8.4)Login Code 4.

In (Figure 8.4) when I receive data I check the jobTitle if it admin, emp, or the user name or password isn't found in database.

```java
@Override
public void onErrorResponse(VolleyError error) {
    progress.dismiss();
    Toast.makeText(getApplicationContext(), "Low Internet Connection", Toast.LENGTH_LONG).show();

    NetworkResponse networkResponse = error.networkResponse;
    if (networkResponse != null) {

        Log.e("Volley", "Error. HTTP Status Code:" + networkResponse.statusCode);
    }

    if (error instanceof TimeoutError) {
        Log.e("Volley", "TimeoutError");
    } else if (error instanceof NoConnectionError) {
        Log.e("Volley", "NoConnectionError");
    } else if (error instanceof AuthFailureError) {
        Log.e("Volley", "AuthFailureError");
    } else if (error instanceof ServerError) {
        Log.e("Volley", "ServerError");
    } else if (error instanceof NetworkError) {
        Log.e("Volley", "NetworkError");
    } else if (error instanceof ParseError) {
        Log.e("Volley", "ParseError");
    }
}
```

(Figure 8.5)Login Code 5.

In (Figure 8.5)when an error happen in web service we handle any type of error in onErrorResponse() method.

```java
@Override
protected Map<String, String> getParams() throws AuthFailureError {
    Map<String, String> params = new HashMap<>();
    params.put("UserName", userName);
    params.put("Password", password);
    return params;
}
```

(Figure 8.6)Login Code 6.

In (Figure 8.6) we return the data from database.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    try {
        List<Fragment> fragments = this.getSupportFragmentManager().getFragments();
        if (fragments != null) {
            for (Fragment fragment : fragments) {
                fragment.onActivityResult(requestCode, resultCode, data);
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

(Figure 9.1)Navigation Code 1.

In (Figure 9.1) this method is acting as listener and created for the barcode scan result, then resend the result to the fragment that ask to initiate the barcode scan reader library.

```java
public void navigationMain_moreVertImg(View view) {
    PopupMenu popupMenu = new PopupMenu(this, moreVert);
    popupMenu.getMenuInflater().inflate(R.menu.menu, popupMenu.getMenu());

    popupMenu.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
        @Override
        public boolean onMenuItemClick(MenuItem menuItem) {
            if (menuItem.getItemId() == R.id.action_settings) {
                Toast.makeText(NavigationMain.this, "setting", Toast.LENGTH_SHORT).show();
            } else if (menuItem.getItemId() == R.id.action_logout) {
                check_login.loggedout("login", NavigationMain.this.getApplicationContext().getFilesDir().getPath());
                startActivity(new Intent(NavigationMain.this, MainActivity.class));
                finish();
            }
            return true;
        }
    });

    popupMenu.show();
}
```

(Figure 9.2)Navigation Code 2.

In (Figure 9.2)when we press the setting button it opens popup menu with two itemssetting and log out, and when we press setting item it opens report fragment, and when we press the log out item the user is logged out from the system.

```
private void setUpDrawer() {
    mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout);
    //mDrawerLayout.setScrimColor(getResources().getColor(android.R.color.transparent));
    mDrawerLayout.addDrawerListener(mDrawerListener);
    expListView = (ExpandableListView) findViewById(R.id.lvExp);
    prepareListData();
    listAdapter = new ExpandableListAdapter(this, listDataHeader, listDataChild);
    // setting list adapter
    expListView.setAdapter(listAdapter);
    mDrawerLayout.closeDrawer(expListView);
    if (check_login.type == 0) {
        expListView.setOnChildClickListener((parent, v, groupPosition, childPosition, id) -> {
                FragmentManager fragmentManager = getSupportFragmentManager();
                switch (groupPosition) {
                    case 0:
                        switch (childPosition) {
                            case 0:
                                titleText.setText("SearchEmployee");
                                fragmentManager.beginTransaction()
                                        .replace(R.id.content_frame
                                                , new SearchEmployee())
                                        .commit();
                                break;
```

(Figure 9.3)Navigation Code 3.

In (Figure 9.3)this method fills the drawer with collapsed items and set onChildClickListener() for each parent item these parent and child items are distinguished by the jobTitle of the user (if user jobTitleis admin it will show all items, and if it is employee it will show a specific items for the user). We have five parent items (for the admin) and only one parent (for the employee) and every item have unique fragment.

When we press any item it will call that fragment.

```java
private void prepareListData() {
    if (check_login.type == 0) {
        listDataHeader = new ArrayList<String>();
        listDataChild = new HashMap<String, List<String>>();

        // Adding child data
        listDataHeader.add("Manage Employees");
        listDataHeader.add("Manage Drugs");
        listDataHeader.add("Manage Bills");
        listDataHeader.add("Manage Insurance Company");
        listDataHeader.add("Manage Drug Company");

        // Adding child data
        List<String> mangeEmps = new ArrayList<~>();
        mangeEmps.add("Search Employees");
        mangeEmps.add("Add Employee");
        mangeEmps.add("Delete Employees");
        mangeEmps.add("Update Employees");


        List<String> manageDrugs = new ArrayList<~>();
        manageDrugs.add("Search Drugs");
        manageDrugs.add("Add Drugs");
        manageDrugs.add("Delete Drugs");
        manageDrugs.add("Update Drugs");
```

(Figure 9.4)Navigation Code 4.

In (Figure 9.4) this method is used to initialize the parent and their children.

```java
public class ExpandableListAdapter extends BaseExpandableListAdapter {

    private Context _context;
    private List<String> _listDataHeader; // header titles
    // child data in format of header title, child title
    private HashMap<String, List<String>> _listDataChild;

    public ExpandableListAdapter(Context context, List<String> listDataHeader,
                                 HashMap<String, List<String>> listChildData) {
        this._context = context;
        this._listDataHeader = listDataHeader;
        this._listDataChild = listChildData;
    }

    @Override
    public Object getChild(int groupPosition, int childPosititon) {
        return this._listDataChild.get(this._listDataHeader.get(groupPosition))
                .get(childPosititon);
    }

    @Override
    public long getChildId(int groupPosition, int childPosition) { return childPosition; }

    @Override
    public View getChildView(int groupPosition, final int childPosition,
                             boolean isLastChild, View convertView, ViewGroup parent) {
```

(Figure 9.5)Navigation Code 5.

In (Figure 9.5)this class get the parent and children view.

```java
@Override
public void onCheckedChanged(RadioGroup group, int checkedId) {
    if (checkedId == R.id.addEmp_checkAdmin) {
        adminTxt.setEnabled(true);
        adminTxt.setTextColor(ContextCompat.getColor(getContext(), R.color.black));
        editTextEmpJobTitle = "admin";
        empText.setEnabled(false);
        empText.setTextColor(ContextCompat.getColor(getContext(), R.color.aaac));
    } else if (checkedId == R.id.addEmp_checkEmp) {
        empText.setEnabled(true);
        empText.setTextColor(ContextCompat.getColor(getContext(), R.color.black));
        editTextEmpJobTitle = "emp";
        adminTxt.setEnabled(false);
        adminTxt.setTextColor(ContextCompat.getColor(getContext(), R.color.aaac));
    }
}
```

(Figure 10.1)AddEmployee Code 1.

In (Figure 10.1) this method is used when we change radio button checked.

```java
@Override
public void onClick(View v) {
    if (v.getId() == R.id.addEmp_submitBtn) {
        if (!editTextEmpName.getText().toString().isEmpty() && !editTextEmpPassword.getText().toString().isEmpty() &&
                !editTextEmpUserName.getText().toString().isEmpty() && !editTextEmpSalary.getText().toString().isEmpty() &&
                !editTextEmpDate.getText().toString().isEmpty()) {
            if (Integer.parseInt(editTextEmpSalary.getText().toString().trim()) <= 0) {
                Toast.makeText(getContext(), "invalid salary input please insert a positive number ", Toast.LENGTH_LONG).show();
            } else {
                addEmp();
            }

        } else {
            Toast.makeText(getContext(), "the fields with \"*\" must be filled", Toast.LENGTH_LONG).show();
        }
```

(Figure 10.2)AddEmployee Code 2.

In (Figure 10.2) this where we validate input data in AddEmployee fragmentand call addEmp() method to connect to server and make transaction with it.

```
public void addEmp() {
    progress = ProgressDialog.show(AddEmployee.this.getContext(), "Please Wait...", "Adding...", true);
```

(Figure 10.3)AddEmployee Code 3.

In (Figure 10.3) we initialize the progress dialog while the connection with server error (because error or not).

```
@Override
public void onResponse(String response) {
    progress.dismiss();
    Log.i("EmpAdd", response);
    Toast.makeText(getActivity(), "" + response, Toast.LENGTH_SHORT).show();

    editTextEmpName.setText("");
    editTextEmpUserName.setText("");
    editTextEmpPassword.setText("");
    editTextEmpPhone.setText("");
    editTextEmpE_mail.setText("");
    editTextEmpSalary.setText("");
    editTextAddress.setText("");

    FragmentTransaction f = getFragmentManager().beginTransaction();
    f.detach(AddEmployee.this).attach(AddEmployee.this).commit();

}
```

(Figure 10.4)AddEmployee Code 4.

In (Figure 10.4) here we close the progress dialog and reset the variable and reload the fragment.

```java
public void loadEmpNames(final int index) {

    final StringRequest stringRequest = new StringRequest(Request.Method.POST, constants.FETCH_EMP_URL,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {

                    try {

                        JSONObject obj = new JSONObject(response);
                        JSONArray array = obj.getJSONArray("Notifications");
                        Log.i("response", response);
                        for (int i = 0; i < array.length(); i++) {
                            JSONObject o = array.getJSONObject(i);
                            if (index == 1) {
                                listItems.add(o.getString("Name"));
```

(Figure 11.1)UpdateEmployee Code 1.

In (Figure 11.1)here we receive data from employee names from the server on jsonfile and put the data in listItems list and wait the user to write to autocomplete edit text to select the name which he want to update his data.

```
else if (index == 2) {
  String name = o.getString("Name");
  if (name.equals(empName)) {

      nameTxtView.setText(o.getString("Name"));
      emailTxtView.setText(o.getString("E_mail"));
      phoneTxtView.setText(o.getString("Phone"));
      passwordTxtView.setText(o.getString("Password"));
      String ss = o.getString("JobTitle");
      RadioButton r1;
      if (ss.equals("admin")) {
          r1 = (RadioButton) getActivity().findViewById(R.id.updateEmp_checkAdmin);
          r1.setChecked(true);
          adminTxt.setEnabled(true);
          adminTxt.setTextColor(ContextCompat.getColor(getContext(), R.color.black));
          jobTitle = "admin";
          empText.setEnabled(false);
          empText.setTextColor(ContextCompat.getColor(getContext(), R.color.aaac));
      } else if (ss.equals("emp")) {
          r1 = (RadioButton) getActivity().findViewById(R.id.updateEmp_checkEmp);
          r1.setChecked(true);
          empText.setEnabled(true);
          empText.setTextColor(ContextCompat.getColor(getContext(), R.color.black));
          jobTitle = "emp";
          adminTxt.setEnabled(false);
          adminTxt.setTextColor(ContextCompat.getColor(getContext(), R.color.aaac));
      }
      salaryTxtView.setText(o.getString("Salary"));
      userNameTxtView.setText(o.getString("UserName"));
      empId = o.getString("ID");
```

(Figure 11.2)UpdateEmployee Code 2.

In (Figure 11.2) here when we select the employee name the data will appear on the edit text to allow the use to modify it and update the new data on the server.

```
public void sendMail(View v) {
    if (v.getId() == R.id.forgetpassword_sendBtn) {
        ed = (EditText) findViewById(R.id.forgetpassword_emailAddressTxt);
        String mail = ed.getText().toString();
        Log.d("elbaz", mail);
        intent = new Intent(Intent.ACTION_SEND);
        intent.setData(Uri.parse("mailto:"));
        intent.putExtra(Intent.EXTRA_EMAIL, new String[]{mail});
        intent.putExtra(Intent.EXTRA_SUBJECT, "Receive your forgotten password");
        intent.putExtra(Intent.EXTRA_TEXT, password);
        intent.setType("message/rfc822");
        chooser = Intent.createChooser(intent, "send mail");
        startActivity(chooser);
        Log.d("elbaz", "message sent");
    }
}
```

(Figure 12)ForgotPassword Code 1.

In (Figure 12)this method where we send to mail with your password after checking that your E-mail is on the system.

```
list = (RecyclerView) getView().findViewById(R.id.sellBill_listView);

int verticalSpacing = 10;
VerticalSpaceItemDecorator itemDecorator =
        new VerticalSpaceItemDecorator(verticalSpacing);

LinearLayoutManager linearLayoutManager = new LinearLayoutManager(getContext());
linearLayoutManager.setOrientation(LinearLayoutManager.VERTICAL);
list.setLayoutManager(linearLayoutManager);

list.addItemDecoration(itemDecorator);

m = new MyAdapter(getActivity(), drugAmountList, R.layout.single_row);
list.setAdapter(m);
list.setNestedScrollingEnabled(true);
```

(Figure 13.1)SellBill Code 1.

In (Figure 13.1) here we initialize the adapter of the Recycler view.

```
@Override
public void onClick(View view) {
    if (view.getId() == R.id.sellBill_companyCBox) {
        if (c.isChecked()) {
            sp.setVisibility(Spinner.VISIBLE);
            //Toast.makeText(this, "" + formattedDate, Toast.LENGTH_SHORT).show();
            loadInsuranceCompanyNames();
        } else if (!c.isChecked()) {
            sp.setVisibility(Spinner.INVISIBLE);
            listItems.removeAll(listItems);
        }
    } else if (view.getId() == R.id.sellBill_addDrug) {
        IntentIntegrator integrator = new IntentIntegrator(getActivity());
        integrator.setDesiredBarcodeFormats(IntentIntegrator.ALL_CODE_TYPES);
        integrator.setPrompt("Scan");
        integrator.setCameraId(0);
        integrator.setBeepEnabled(true);
        integrator.setOrientationLocked(false);
        integrator.setBarcodeImageEnabled(false);
        integrator.initiateScan();
```

(Figure 13.2)SellBill Code 2.

In (Figure 13.2) here when we press thecompanycboxcheckbox it will show spinner with names of insurance companies for discount, and when you press the addDrug button it will open barcode reader library and send the result to the parent activity in OnActivityResult() method and it resend the result to child fragment and the child fragment receive the result in OnActivityResult() method.

```java
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
    if (result != null) {
        if (result.getContents() == null) {
            Toast.makeText(getActivity(), "you cancelled the scan", Toast.LENGTH_SHORT).show();
            Toast.makeText(getActivity(), "enter the barcode if you want write it(not scan)", Toast.LENGTH_LONG).show();
            dialog = new CustomDialogDiscountAmount(getActivity(), "Enter Barcode manual", "Enter Barcode manual");
            dialog.show();
            dialog.setDialogResult((result, c) -> {
                    parcode = result;
                    check = c;
                    if (check) {
                        dialog = new CustomDialogDiscountAmount(getActivity(), "Enter amount", "Enter amount");
                        dialog.show();
                        dialog.setDialogResult((result, c) -> {
                                drugAmount = Integer.parseInt(result);
                                Toast.makeText(getActivity(), "the drug amount is from barcode " + drugAmount, Toast.LENGTH_SHORT)
                                mh.clear();
                                fetchDrugName();
                        });
                    }
            });
```

(Figure 13.3)SellBill Code 3.

In (Figure 13.3) here we check the result data came from parent activity and if it's equal to null, then we open dialog if he want to enter barcode manually or not

```java
} else {
    barcode = result.getContents();

    CustomDialogDiscountAmount dialog = new CustomDialogDiscountAmount(getActivity(), "Enter amount", "Enter amount");
    dialog.show();
    dialog.setDialogResult((result, c) -> {
            drugAmount = Integer.parseInt(result);
            Toast.makeText(getActivity(), "the drug amount is from barcode " + drugAmount, Toast.LENGTH_SHORT).show();
            mh.clear();
            fetchDrugName();
    });
    //loadDrugCompNames(2);
}
```

(Figure 13.4)SellBill Code 4.

In (Figure 13.4) here we check the result data came from parent activity if it is not equal to null, we open dialog for getting the amount of drug to put row in RecyclerView.

```
public class MyAdapter extends RecyclerView.Adapter<MyHolder> {
    Context context;
    public List<String> drugAmountList;
    int itemResource;

    MyAdapter(Context c, List<String> drugAmountList, int itemResource) {
        this.context = c;
        this.drugAmountList = drugAmountList;
        this.itemResource = itemResource;
    }

    @Override
    public MyHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext())
                .inflate(this.itemResource, parent, false);
        return new MyHolder(this.context, view);
    }

    @Override
    public void onBindViewHolder(MyHolder holder, int position) {
        holder.BindNames(drugNameList.get(position), this.drugAmountList.get(position), drugPriceList.get(position).toString()
    }
```

(Figure 13.5)SellBill Code 5.

In (Figure 13.5) this is the adapter class for RecyclerView.

```
public class MyHolder extends RecyclerView.ViewHolder implements View.OnClickListener {
    Context context;
    TextView drugs, price;
    ImageButton button;
    public int position;
    public TextView drugAmountTxtView;

    public MyHolder(Context context, View itemView) {...}

    public void BindNames(String name, String amount, String price, int position) {
        this.drugs.setText(name);
        this.price.setText(price + " $");
        this.position = position;
        drugAmountTxtView.setText(amount);
        mh.add(this);
    }

    @Override
    public void onClick(View view) {
        if (view.getId() == R.id.singleRow_nameTxt || view.getId() == R.id.singleRow_amountTxt) {
            CustomDialogDiscountAmount dialog = new CustomDialogDiscountAmount(getActivity(), "Enter amount", "Enter amount");
            dialog.show();
            dialog.setDialogResult((result, c) -> {
                    totalPrice -= drugPriceList.get(position) * Integer.parseInt(drugAmountList.get(position));
                    totalPriceTxtView.setText(totalPrice + "");

                    mh.get(position).drugAmountTxtView.setText(result);
                    drugAmountList.set(position, result);
                    totalPrice += drugPriceList.get(position) * Integer.parseInt(drugAmountList.get(position));
```

(Figure 13.6)SellBill Code 6.

In (Figure 13.6) this class of Viewholder that create the row, set data into it, and set OnClick() listener for make update when press the amount text or the name text and delete the row from delete button.

Note:

- **BuyBill code is the same of SellBill with some difference and it's not have barcode reader.**
- **Add, update, delete (drug ,drug company, insurance company, and employee) have the same methods but with difference for handling their case.**

```
@Override
public void onClick(View view) {
    if (view.getId() == R.id.customDialogAmount_submitBtn) {
        if (mDialogResult != null) {
            check = true;
            mDialogResult.finish(String.valueOf(amountEdidtText.getText()), check);
        }
        CustomDialogDiscountAmount.this.dismiss();


    } else if (view.getId() == R.id.customDialogAmount_cancelBtn) {
        CustomDialogDiscountAmount.this.dismiss();
    }
}

public void setDialogResult(OnMyDialogResult dialogResult) { mDialogResult = dialogResult; }


public interface OnMyDialogResult {
    void finish(String result, boolean c);
}
```

(Figure 14)CustomDialog Code 1.

In (Figure 14) this method handle sending data from dialog to fragment that call the dialog.

3.1.2Web service code:

❖ **Operations.php is the backbone of the web service for project, it has the most important queries and validation methods.**

```
package com.example.user.pharmacyproject;

/**
 * Created by User on 6/11/2017.
 */

public class constants {
    private static String ROOT_URL = "https://pharmacy-managememt.000webhostapp.com/";
    public static String ADD_EMP_URL = ROOT_URL+"AddEmp.php";
    public static String FETCH_EMP_URL = ROOT_URL+"FetchEmps.php";
    public static String DELETE_EMP_URL = ROOT_URL+"DeleteEmp.php";
    public static String UPDATE_EMP_URL = ROOT_URL+"UpdateEmp.php";
    public static String ADD_COMPANY_URL = ROOT_URL+"AddCompany.php";
    public static String FETCH_COMPANY_URL = ROOT_URL+"FetchCompany.php";
    public static String DELETE_COMPANY_URL = ROOT_URL+"DeleteComp.php";
    public static String UPDATE_COMPANY_URL = ROOT_URL+"UpdateCompany.php";
    public static String ADD_DRUG = ROOT_URL+"AddDrug.php";
    public static String FETCH_DRUG_URL = ROOT_URL+"FetchDrug.php";
    public static String FETCH_ISURANCE_COMPANY_URL = ROOT_URL+"FetchInsuranceCompany.php";
    public static String ADD_SELL_BILL_URL = ROOT_URL+"AddSellBill.php";
    public static String ADD_BUY_BILL_URL = ROOT_URL+"AddBuyBill.php";
    public static String FETCH_SELL_BILL_MAX_ID_URL = ROOT_URL+"FetchSellBillMaxID.php";
    public static String ADD_DRUG_HAS_SELL_BILL_URL = ROOT_URL+"AddDrugHasSellBill.php";
    public static String CHECK_LOGIN_URL = ROOT_URL+"CheckLogin.php";
    public static String DELETE_DRUG_URL = ROOT_URL + "DeleteDrug.php";
    public static String UPDATE_DRUG_URL = ROOT_URL + "UpdateDrug.php";
    public static String UPDATE_DRUG_AMOUNT_URL = ROOT_URL + "UpdateTheDrugAmount.php";
    public static String FETCH_DRUGS_URL = ROOT_URL + "FetchDrugs.php";
    public static String FETCH_COMP_DISCOUNT_URL = ROOT_URL + "FetchCompDiscount.php";
    public static String ADD_INSURANCE_COMPANY_URL = ROOT_URL + "AddInsuranceCompany.php";
    public static String FETCH_INSURANCE_COMPANIES_URL = ROOT_URL + "FetchInsuranceCompany.php";
    public static String UPDATE_INSURANCE_COMPANIES_URL = ROOT_URL + "UpdateInsuranceCompany.php";
```

(Figure 15)constants Code 1.

In (Figure 15) this class which save all URL for php server files for web service.

```php
<?php
$host = "localhost";
$user = "id1427077_pharmacyapplication";
$password = "01091390584";
$db = "id1427077_pharmacyapplication";

$con = mysqli_connect($host,$user,$password,$db);
if(!$con){
    die("Error in conecction" .mysqli_connect_error());
}else{
  //  echo "<br><h3>connection success ...</h3>";
}
?>
```

(Figure 16)init.php Code 1.

In (Figure 16)this class initialize a connection to database.

```php
<?php
require_once'./Operations.php';
$response = array();
if($_SERVER['REQUEST_METHOD']=='POST'){
    if(isset($_POST['UserName']) and isset($_POST
['Password'])){
        $db = new DbOpertaion();

                $user = $db->userLogin($_POST['UserName'] ,
$_POST['Password']);
            $response['error'] =false;
                $response['ID'] = $user['ID'];
                $response['Name'] = $user['Name'];
            $response['JobTitle'] = $user['JobTitle'];

      }else{
         $response['error']=true;
            $response['message']="Required fields";
    }

      }else{
            $response['error']=true;
            $response['message']="Invalid Request";

      }

      echo json_encode($response);
      ?>
```

(Figure 17)CheckLogin.php Code 1.

In (Figure 17)this class handle the check login query.

```
<?php
 define('DB_USERNAME','id1427077_pharmacyapplication');
 define('DB_PASSWORD','01091390584');
 define('DB_NAME','id1427077_pharmacyapplication');
 define('DB_HOST','localhost');
?>
```

(Figure 18)Config.php Code 1.

In (Figure 18)this class handle the configuration.

```
<?php

class DbConnect{

    private $con;
        function __construct(){}
    function connect (){
        include_once dirname(__FILE__).'/Config.php';
        $this->con = new mysqli(DB_HOST,DB_USERNAME,
DB_PASSWORD, DB_NAME);
         if(mysqli_connect_errno()){
             echo "Failed to connect";

         }
        return $this->con;
    }
}


?>
```

(Figure 19)DbConnect.php Code 1.

In (Figure 19) this class handles the connection by using Config.phpfile.

```php
<?php
require_once'./Operations.php';
$response = array();

if($_SERVER['REQUEST_METHOD']=='POST'){
   if(isset($_POST['Name']) and isset($_POST['UserName']) and
isset($_POST['Password'] ) and isset($_POST['JobTitle'] ) and
isset($_POST['Salary'] )and isset($_POST['Date'] ) ){

        //insert
         $db = new DbOpertaion();
        $result =$db->createUser(
                                 $_POST['Name'] ,
                                  $_POST['UserName'] ,
                                  $_POST['Password'] ,
                                  $_POST['Phone'] ,
                                  $_POST['E_mail'] ,
                                  $_POST['JobTitle'] ,
                                  $_POST['Salary'] ,
                                  $_POST['Date'] ,
                                  $_POST['Address']  );


        if($result == 1 ){

             $response['error']=false;
            $response['message']="User created";
       }elseif($result == 2){

            $response['error']=true;
```

(Figure 20.1)AddEmp.php Code 1.

```
                $response['message']="Errorsssss";
            }elseif($result == 0){
                 $response['error']=true;

                $response['message']="User already Exists ";

            }

        }else{
             $response['error']=true;
        $response['message']="Required fields";
        }

    }else{
        $response['error']=true;
        $response['message']="Invalid Request";

    }
    echo json_encode( $response);

    ?>
```

(Figure 20.2)AddEmp.php Code 2.

In (Figure 20.1,20.2) this class handles adding employee in Data base by using Operations.php file.

```
<?php
require "init.php";

$sql = "select * from Employee";
$result = mysqli_query($con,$sql);
$response = array();
while($row = mysqli_fetch_array($result)) {
array_push($response,array("ID"=>$row[0],"Name"=>$row
[1],"UserName"=>$row[2],"Password"=>$row[3],"Phone"=>$row[4]
,"E_mail"=>$row[5],"JobTitle"=>$row[6],"Salary"=>$row
[7],"Address"=>$row[9],"Date"=>$row[8]));
}
echo json_encode(array("Notifications"=>$response));
mysqli_close($con);

?>
```

(Figure 21)FetchEmps.php Code 1.

In (Figure 21) this class handles the search of employees by using init.php file.

```
<?php
if($_SERVER['REQUEST_METHOD']=='POST'){

require "init.php";

$ID= $_POST['ID'];
$Name=$_POST['Name'];
$UserName=$_POST['UserName'];
$Phone=$_POST['Phone'];
$E_mail=$_POST['E_mail'];
$JobTitle=$_POST['JobTitle'];
$Salary=$_POST['Salary'];
$Password=$_POST['Password'];
$Date=$_POST['Date'];
$Address=$_POST['Address'];

$Sql_Query = " UPDATE Employee SET Name= '$Name', UserName=
'$UserName',
Phone= '$Phone' , E_mail= '$E_mail', JobTitle= '$JobTitle',
Salary= '$Salary', Password= '$Password'
, Date= '$Date', Address= '$Address' WHERE ID= $ID ";

 if(mysqli_query($con,$Sql_Query))
{
 echo 'Record Updated Successfully';
}
else
{
 echo 'Something went wrong';
}}
 mysqli_close($con);
```

(Figure 22)UpdateEmp.php Code 1.

In (Figure 22) this class handles the update employee query by using init.php file.

```
<?php
if($_SERVER['REQUEST_METHOD']=='POST'){

require "init.php";

 $ID= $_POST['ID'];



$Sql_Query = "DELETE FROM Employee WHERE ID= '$ID'";

 if(mysqli_query($con,$Sql_Query))
{
 echo 'Record Updated Successfully';
}
else
{
 echo 'Something went wrong';
 }
 }
 mysqli_close($con);
?>
```

(Figure 23)DeleteEmp.php Code 1.

In (Figure 23) this class handles the delete of an employee by using init.phpfile.

```php
<?php
require_once'./Operations.php';
$response = array();

if($_SERVER['REQUEST_METHOD']=='POST')
{
   if(isset($_POST['Date']) and isset($_POST['TotalPrice']))
   {
        //insert
        $db = new DbOpertaion();
        $result =$db->addBuyBill(
                                 $_POST['Date'] ,
                                 $_POST['TotalPrice']
                                  );
        if($result == 1 )
           {
             $response['error']=false;
             $response['message']="Buy Bill added";
        }else if($result == 2)
           {
            $response['error']=true;
            $response['message']="Errorssssss";
        }
    }else{
         $response['error']=true;
            $response['message']="Required fields";}}
    else{
    $response['error']=true;
    $response['message']="Invalid Request";  }
echo json_encode( $response);
?>
```

(Figure 24)AddBuyBil.php Code 1.

In (Figure 24) this class handles adding buy bills in database.

Note :

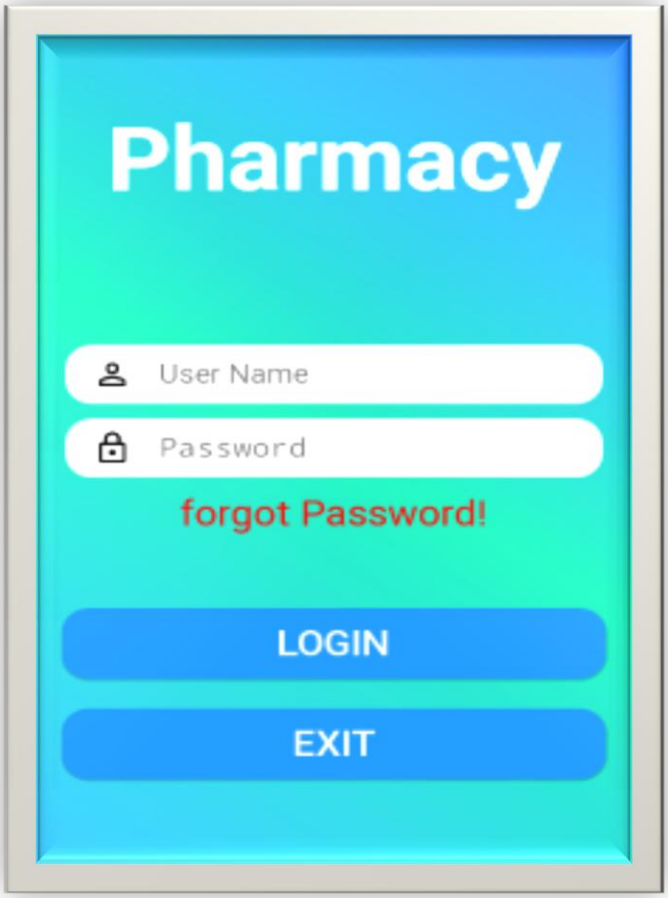
- **Buy bills like Sell bills and have the same class with some difference.**
- **Add, update, delete (drug ,drug company, insurance company, and employee) have the same methods but with difference for handling their case.**

Chapter 4

Section4.1:-

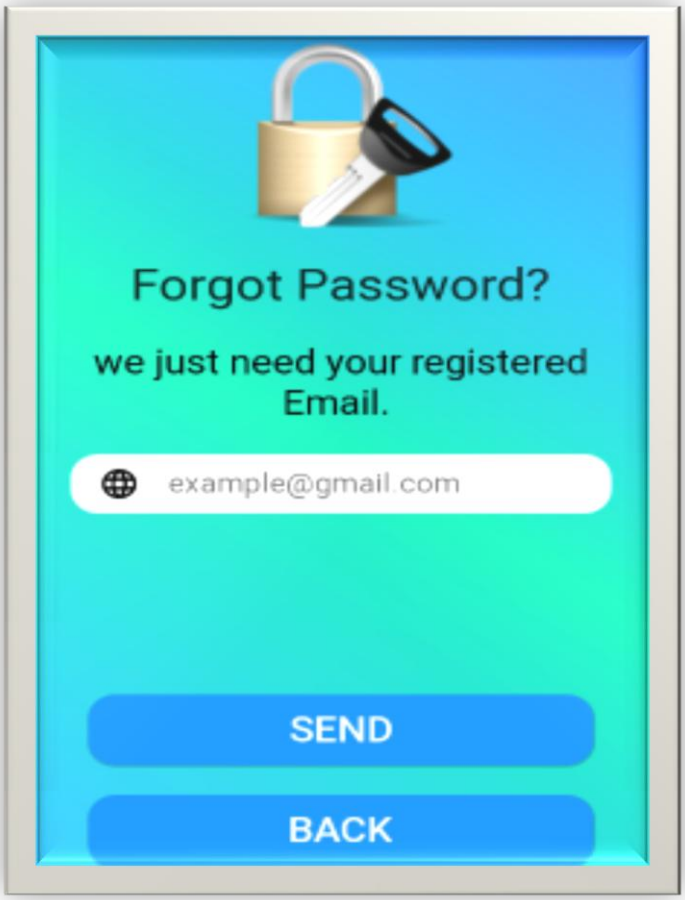
Our work

4.1.1 How admin uses this app:



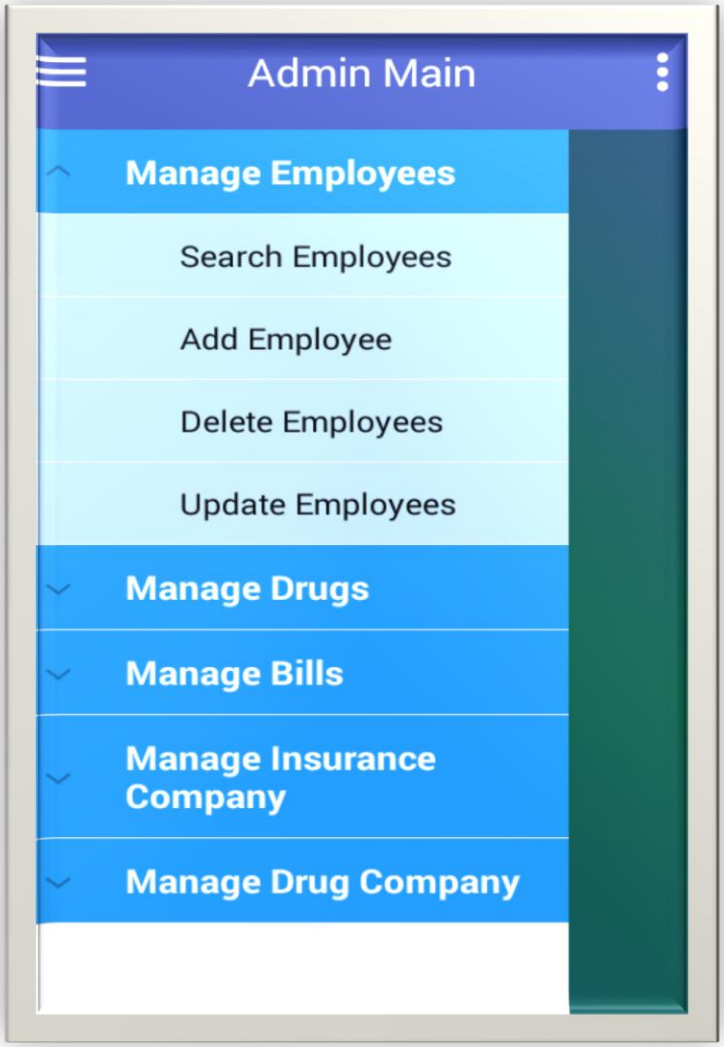
(Figure 25)Login Activity.

In (Figure 25) this is Login activity that appear for both users (admin and employee) if he is not log in of make log out, you must before press login button fill user name and password fields, you can press forgot password Text to go to forget password activity, and you can press exit button to close the program.



(Figure 26)ForgetPassword Activity.

In (Figure 26) in this activity you enter your E-mail that youregistered with it and if it found in database it will send the user name and password to your E-mail.

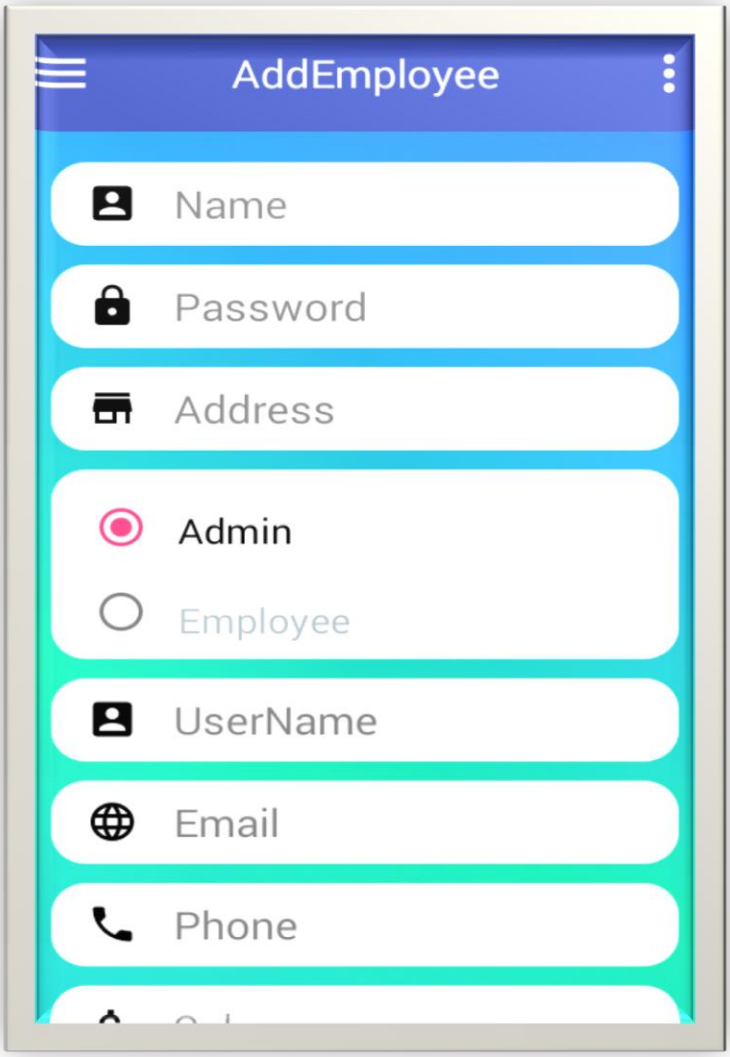


(Figure 27)Admin Main Activity.

In (Figure 27) in thisactivity we have the drawer that has many parent and every parent has many child and every child refer to a fragment.

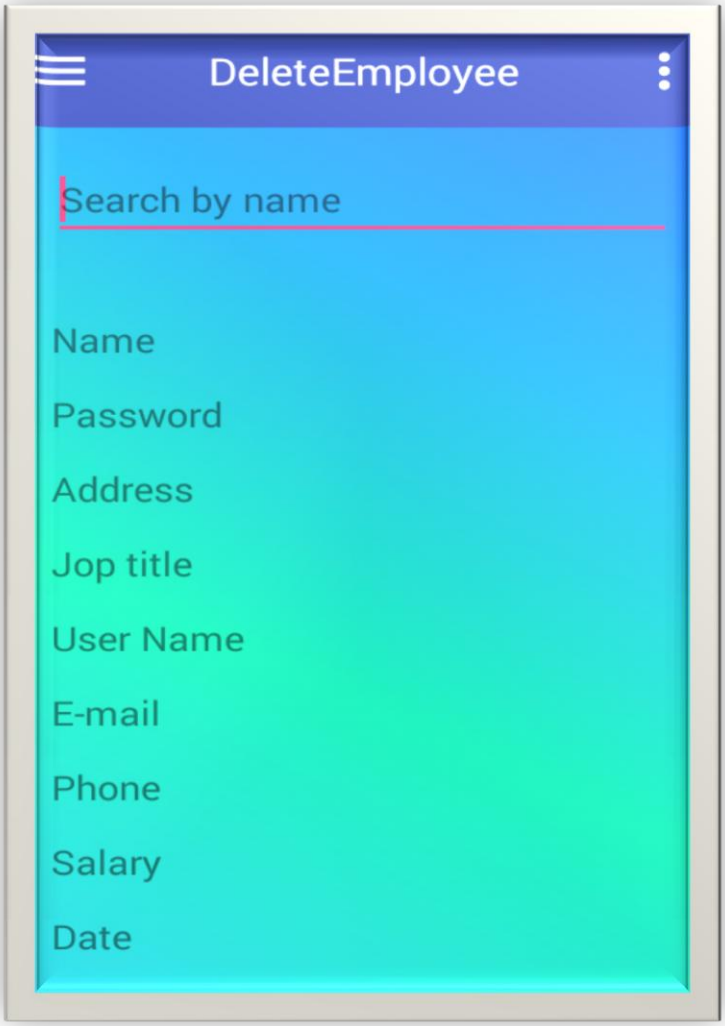
You can get the drawer by press button or drag from left to right.

You can log out by press ⋮ and press log out item.



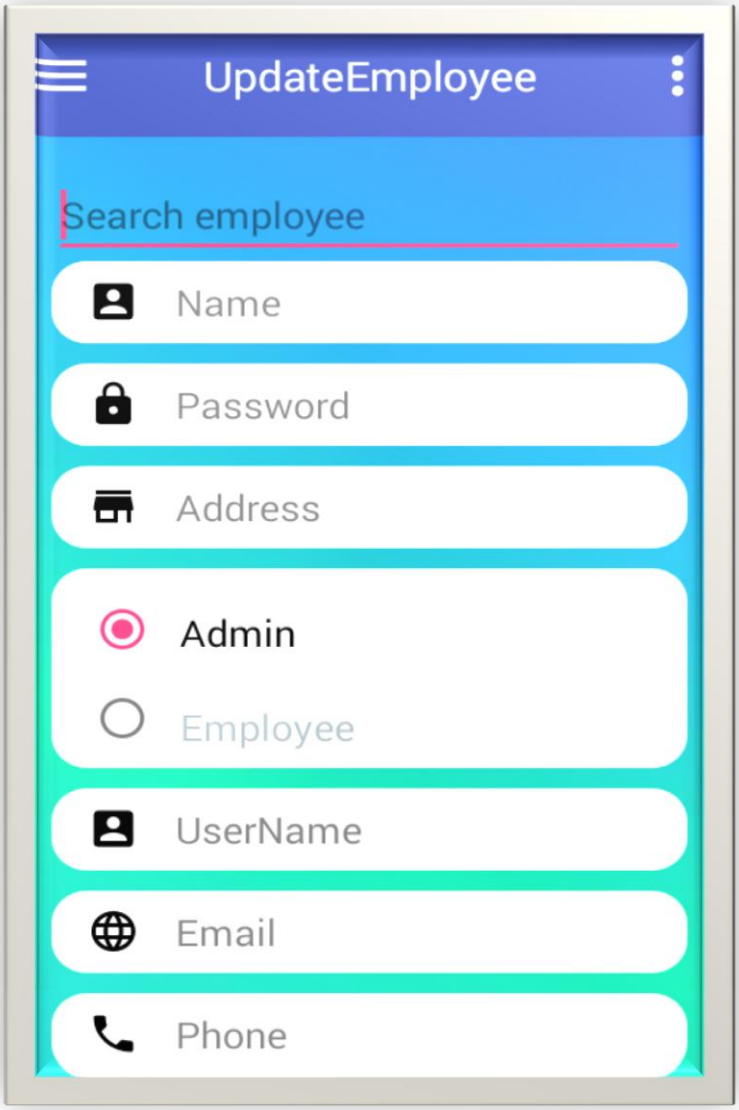
(Figure 28)AddEmployee Fragment.

In (Figure 28) in this fragment you can add employee.



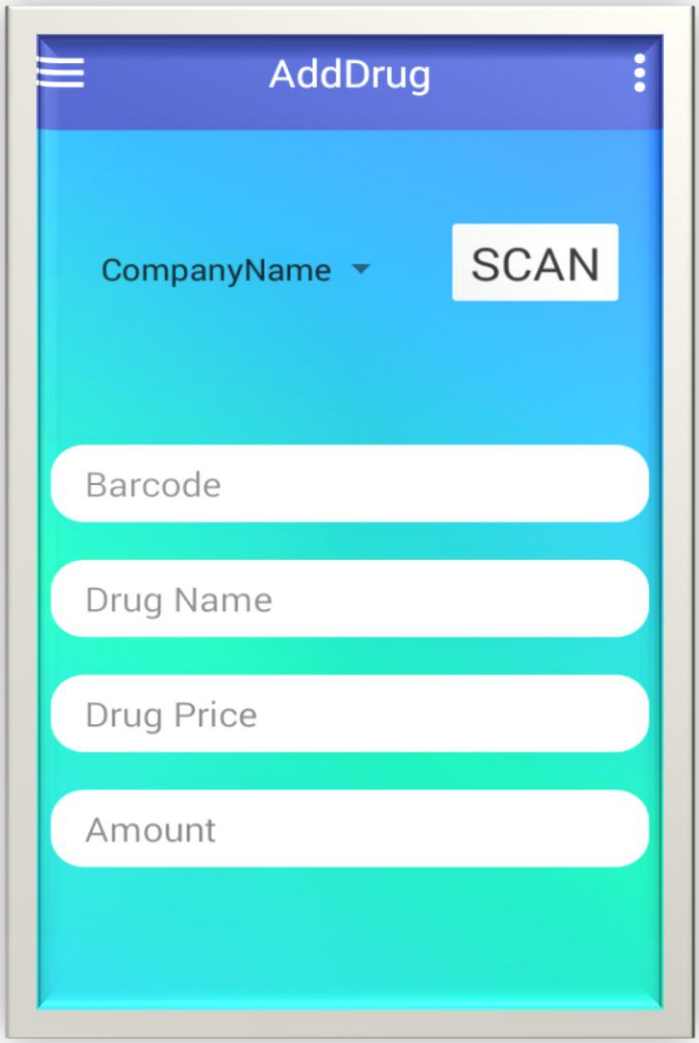
(Figure 29)DeleteEmployee Fragment.

In (Figure 29) in this fragment you can delete employee.



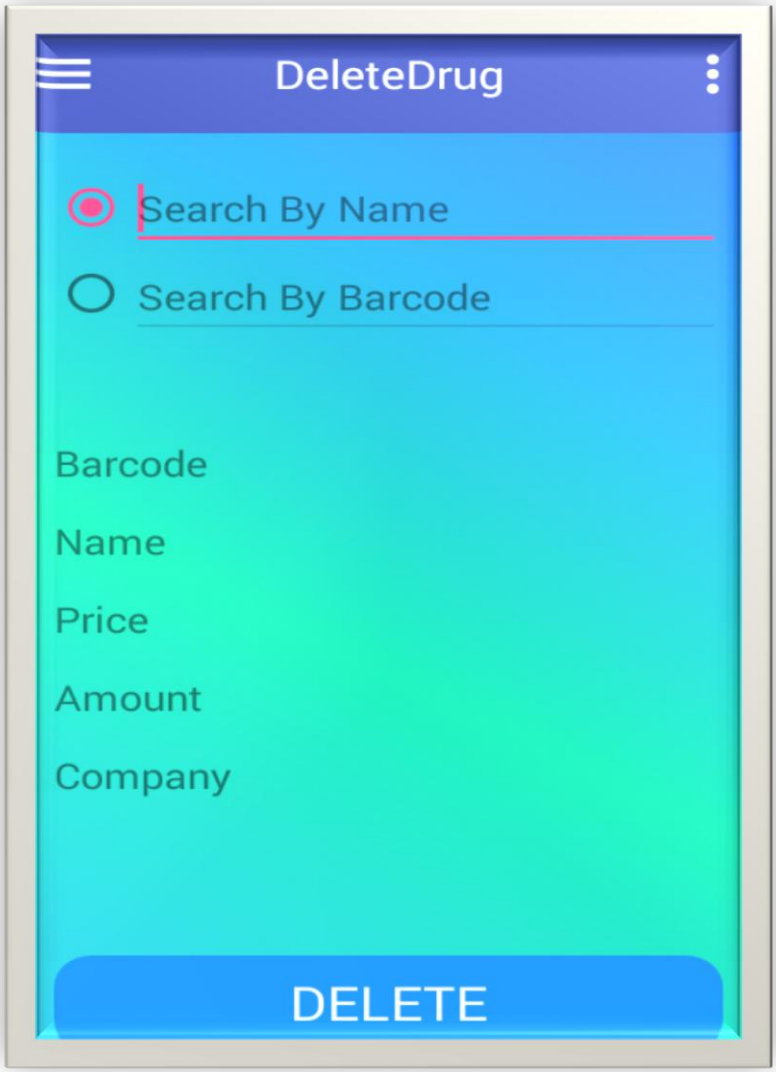
(Figure 30)UpdateEmployee Fragment.

In (Figure 30) in this fragment you can update employee.



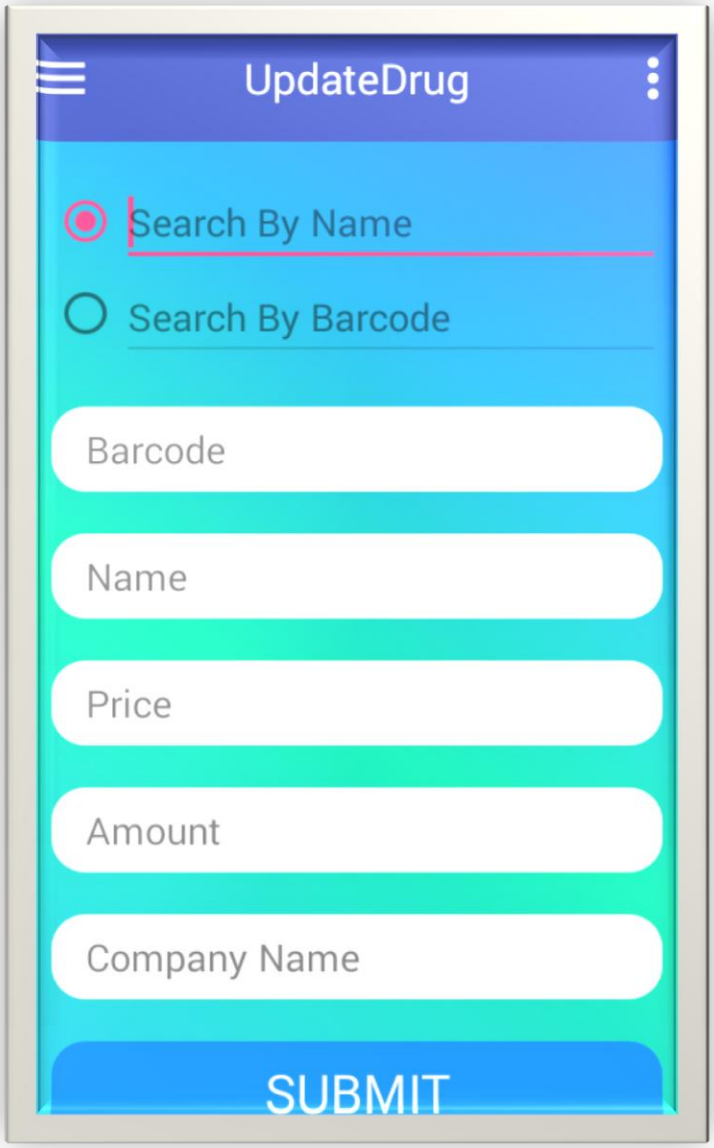
(Figure 31)AddDrug Fragment.

In (Figure 31) in this fragment you can add Drug.



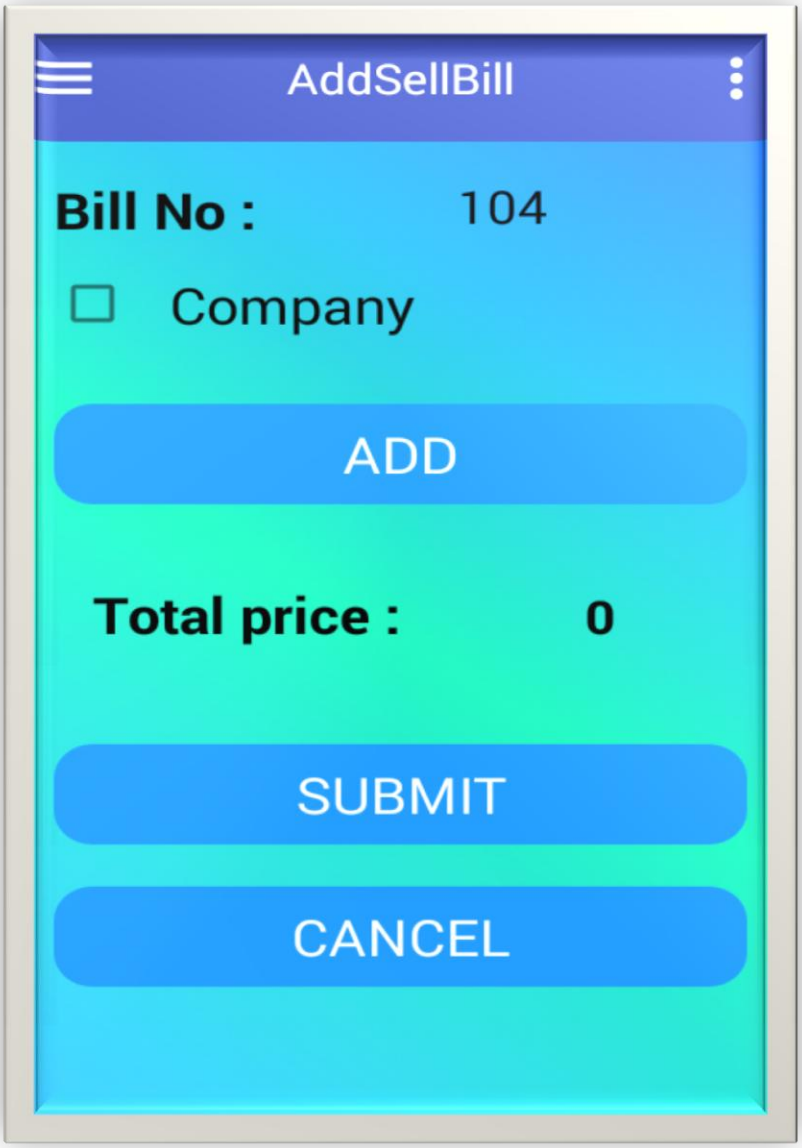
(Figure 32)DeleteDrug Fragment.

In (Figure 32) in this fragment you can delete Drug.



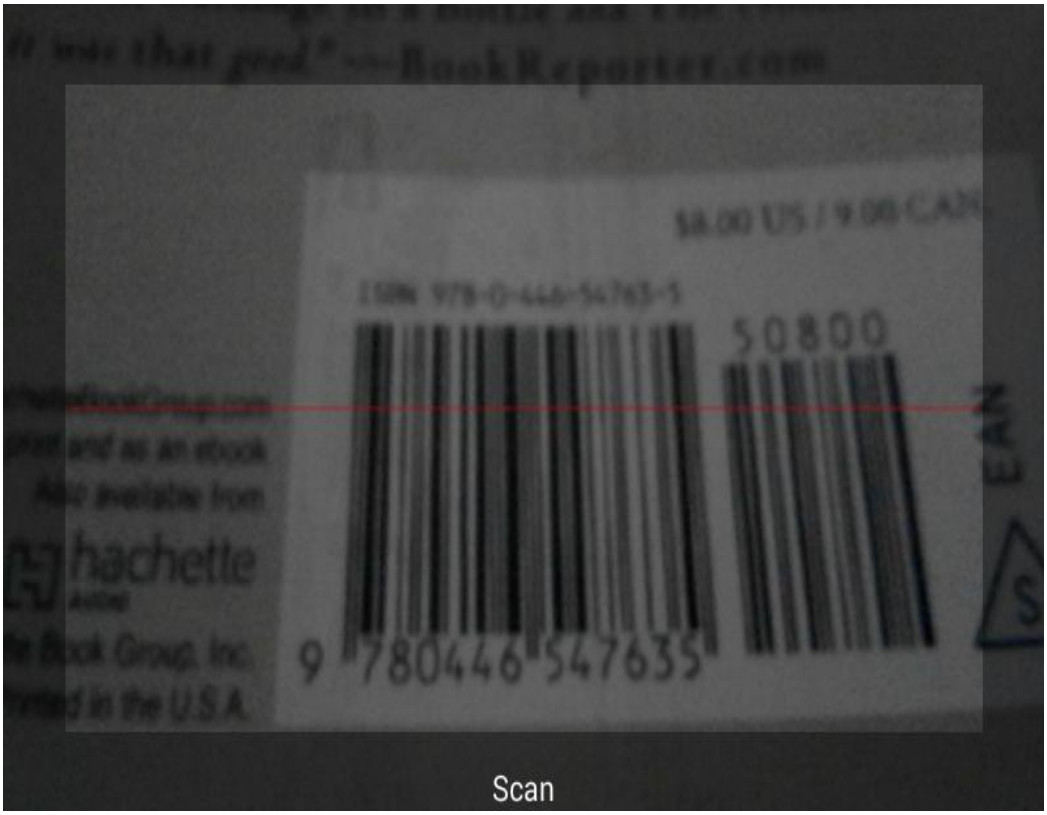
(Figure 33)UpdateDrug Fragment.

In (Figure 33) in this fragment you can update Drug.

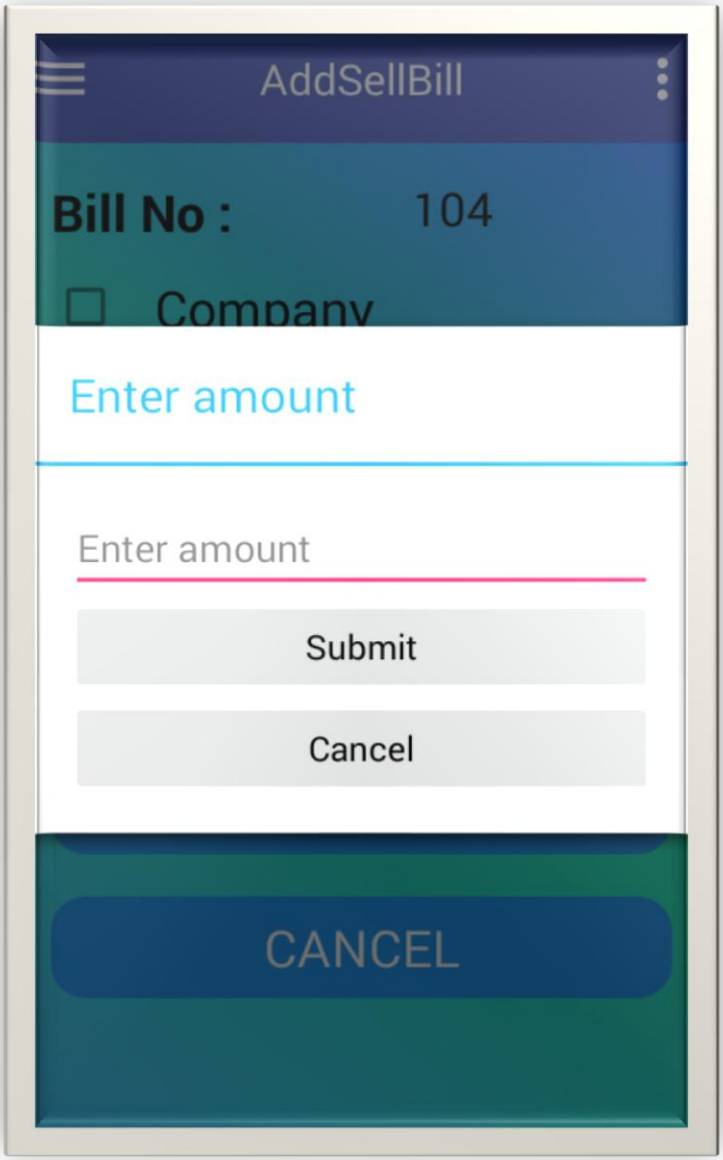


(Figure 34.1)Add Sell Bill Fragment.

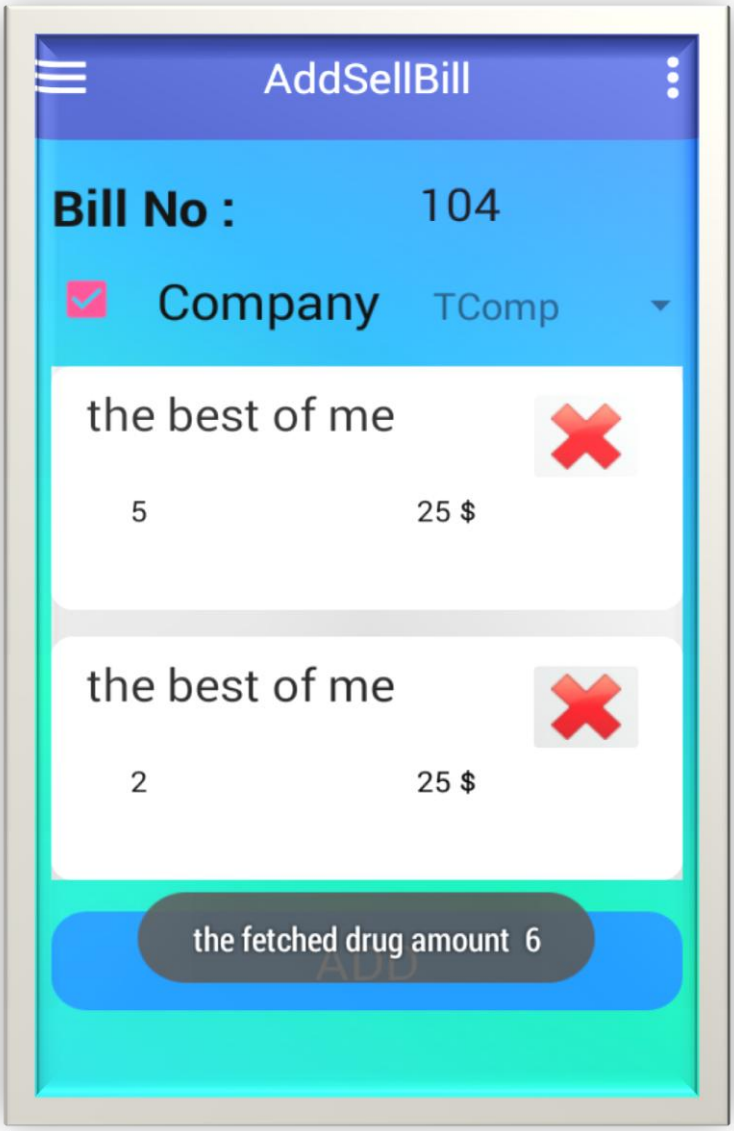
In (Figure 34.1) in this fragment you can add sell bill.



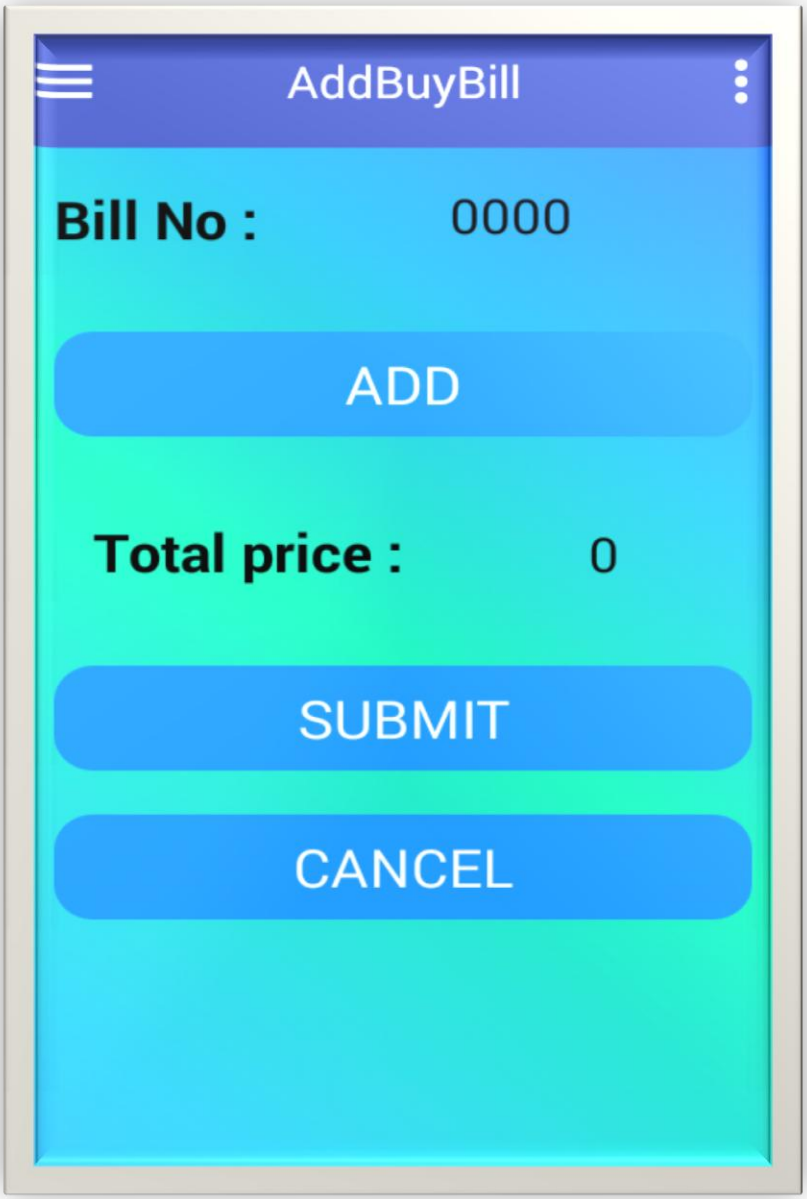
(Figure 34.2)Add Sell Bill Fragment (barcode reader).



(Figure 34.3)Add Sell Bill Fragment.

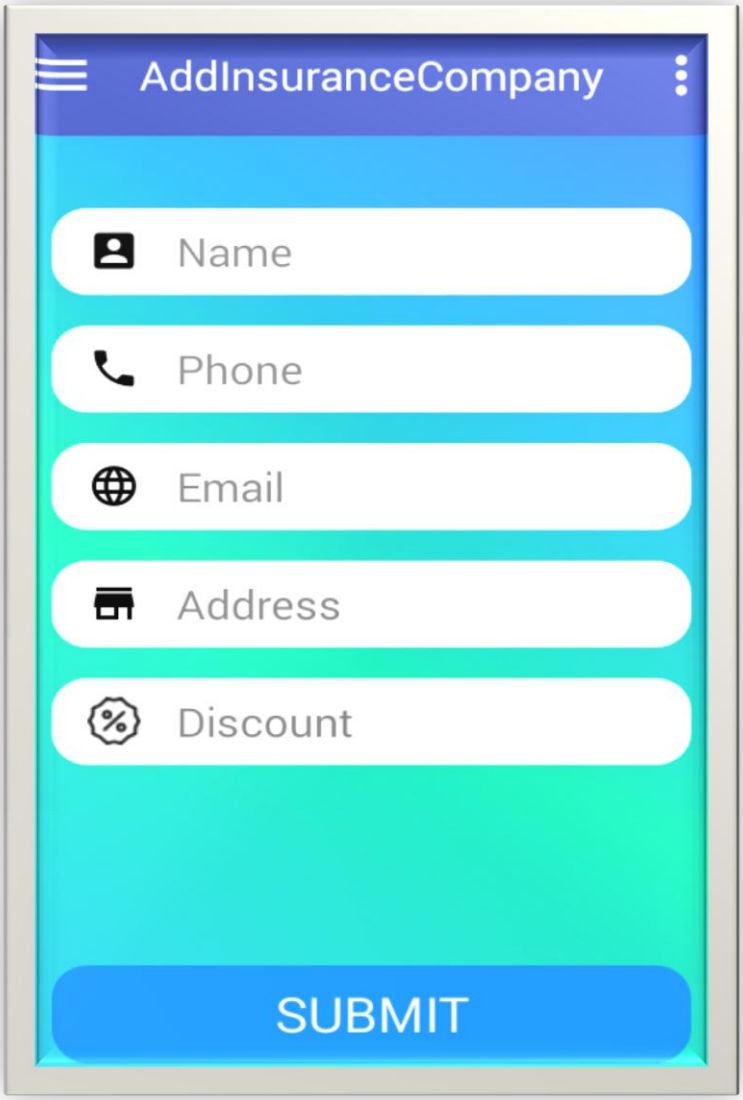


(Figure 34.4)Add Sell Bill Fragment.



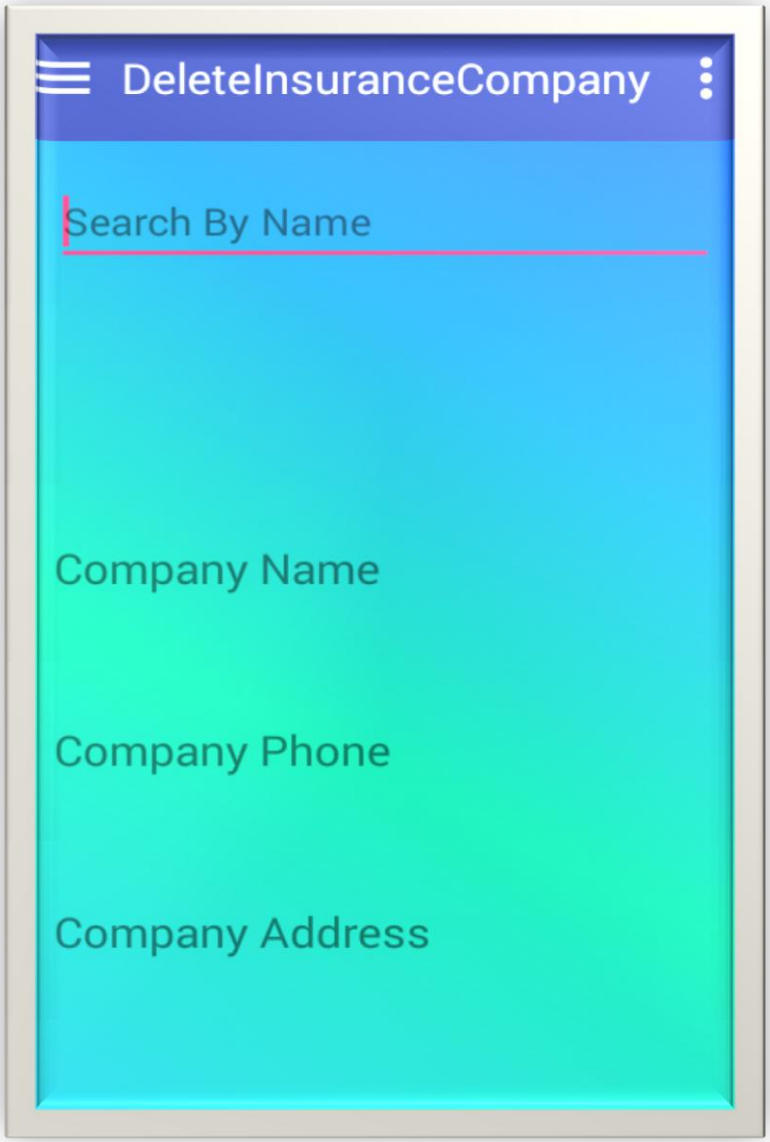
(Figure 35)Add Buy Bill Fragment.

In (Figure 35) in this fragment you can Add buy bill.



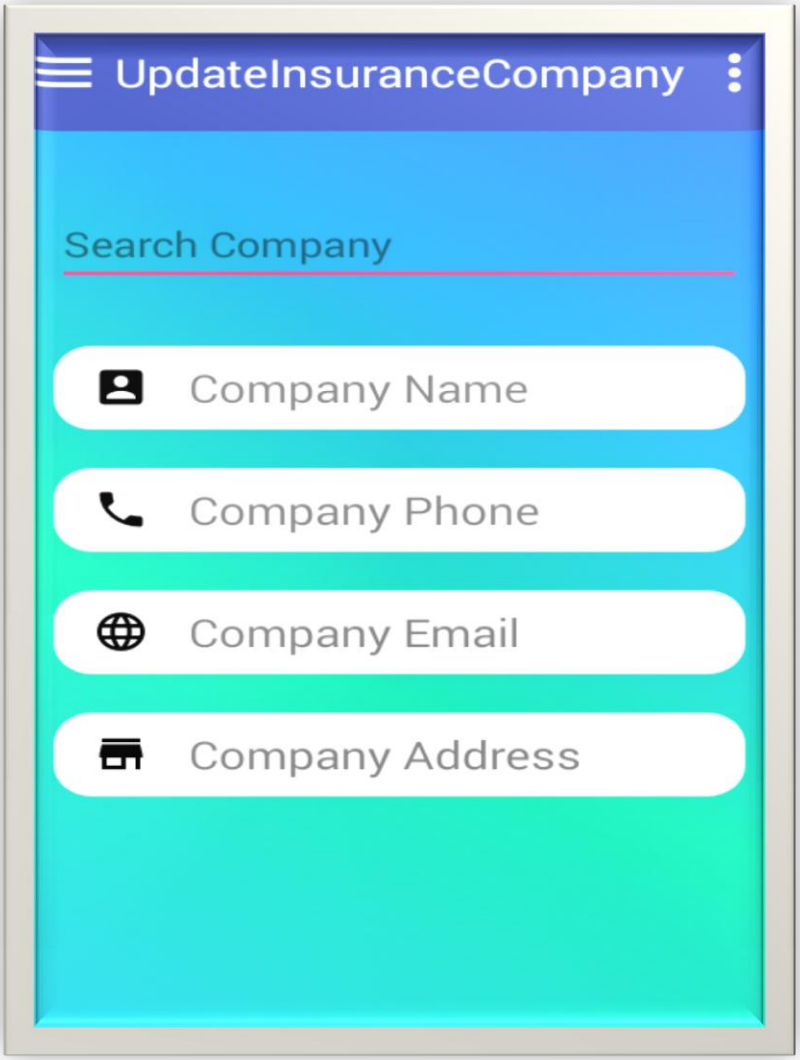
(Figure 36)Add Insurance Company Fragment.

In (Figure 36) in this fragment you can Add insurance company.



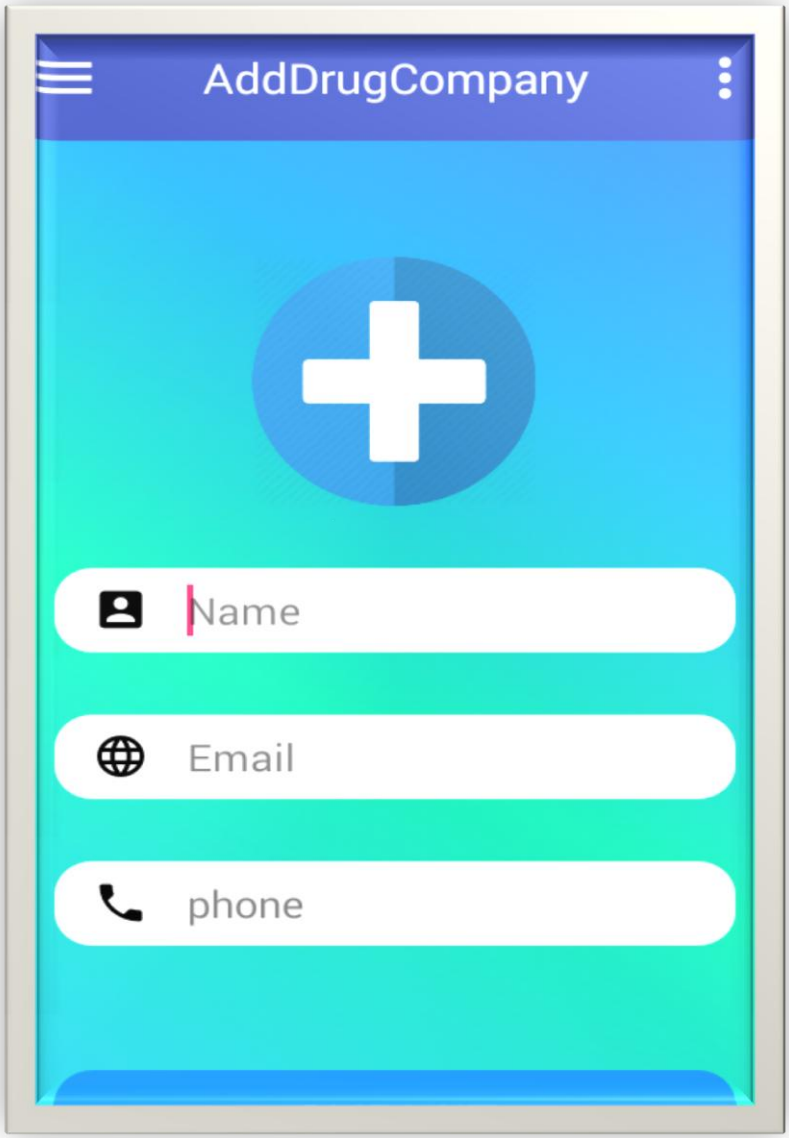
(Figure 37)Delete Insurance Company Fragment.

In (Figure 37) in this fragment you can Delete insurance company.



(Figure 38)Update Insurance Company Fragment.

In (Figure 38) in this fragment you can Update insurance company.

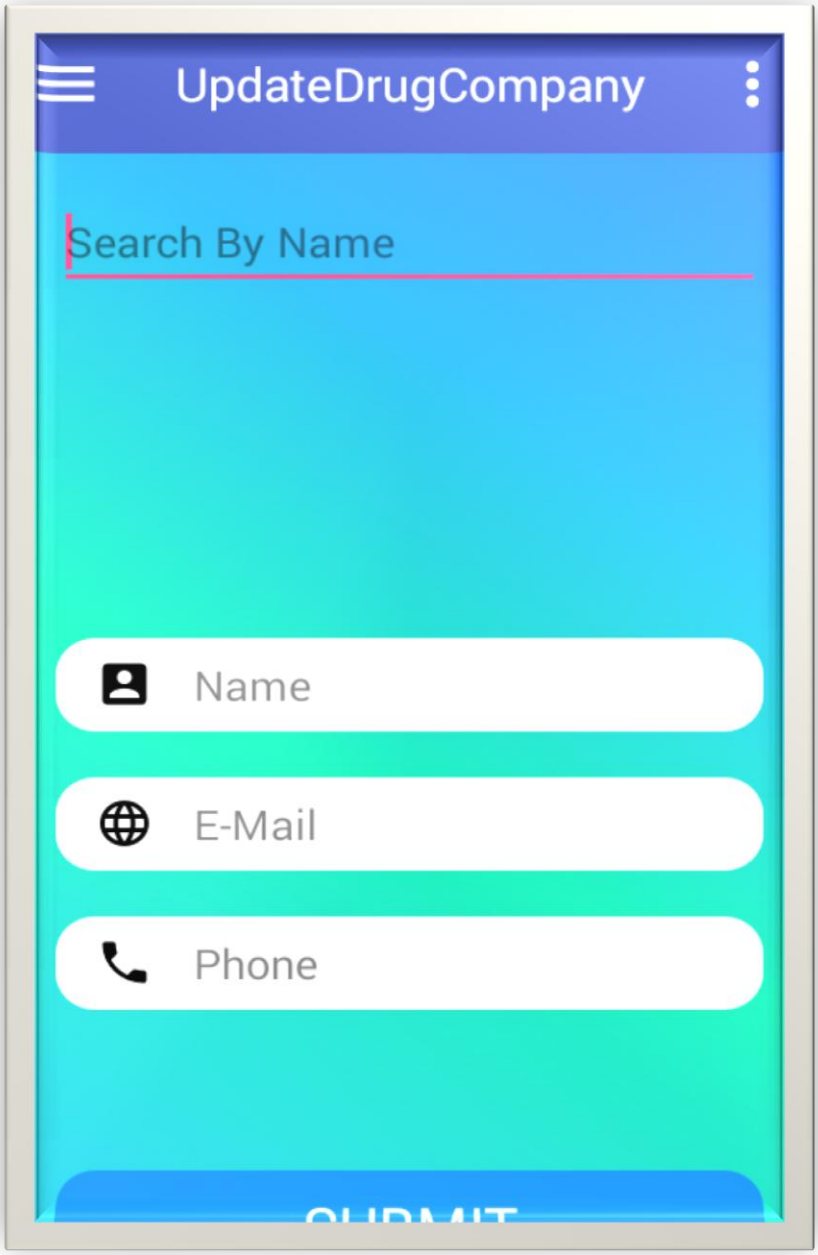


(Figure 39)AddDrug Company Fragment.

In (Figure 39) in this fragment you can Adddrug company.

(Figure 40)Delete Drug Company Fragment.

In (Figure 40) in this fragment you can Delete drug company.

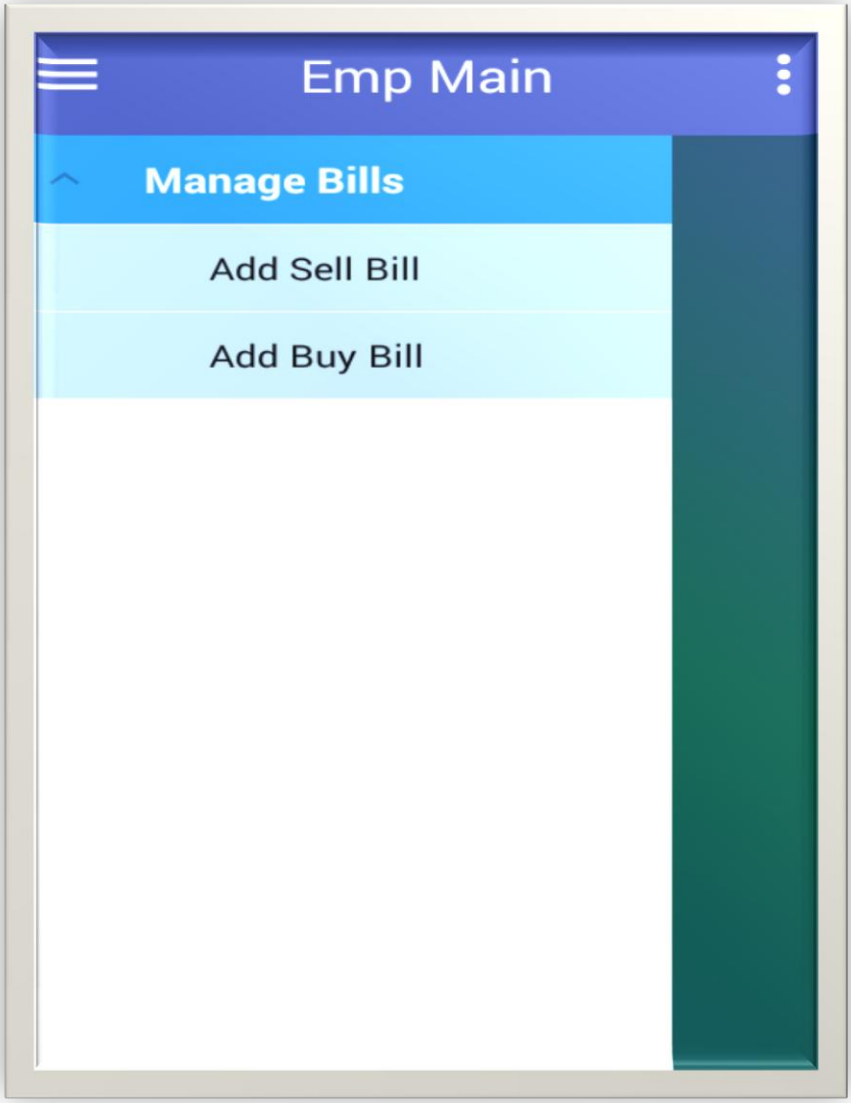


(Figure 41)Update Drug Company Fragment.

In (Figure 41) in this fragment you can Update drug company.

4.1.2Howemployee uses this app:

It's the same in Login and ForgetPassword Activities.



(Figure 42)Employee MainActivity.

And that is the only difference.

The end