**Chapter 4. IMPLEMENTATION**

**4.1 Waterfall Model**

The waterfall model is a sequential (non-iterative) design process, used in software development processes, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception, initiation, analysis, design, construction, testing, production/implementation and maintenance.

First introduced by Dr. Winston W. Royce in a paper published in 1970, the waterfall model is a software development process. The waterfall model emphasizes that a logical progression of steps be taken throughout the software development life cycle (SDLC), much like the cascading steps down an incremental waterfall. While the popularity of the waterfall model has waned over recent years in favour of more agile methodologies, the logical nature of the sequential process used in the waterfall method cannot be denied, and it remains a common design process in the industry.

Actually implementing a waterfall model within a new software project is a rather straightforward process, thanks in large part due to the step-by-step nature of the method itself. There are minor differences in the numbers and descriptions of the steps involved in a waterfall method, depending on the developer you ask (and even the year during which you ask him or her). Regardless, the concepts are all the same and encompass the broad scope of what it takes to start with an idea and develop a full-scale, live application.

Definition: The waterfall model is a classical model used in system development life cycle to create a system with a linear and sequential approach. It is termed as waterfall because the model develops systematically from one phase to another in a downward fashion. This model is divided into different phases and the output of one phase is used as the input of the next phase. Every phase has to be completed before the next phase starts and there is no overlapping of the phases.

The waterfall development model originates in the manufacturing and construction industries: highly structured physical environments in which after-the-fact changes are prohibitively costly, if not impossible.

Because it was created in a time when no formal software development methodologies existed, this hardware-oriented model was simply adapted for software development.

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The Waterfall model is the earliest SDLC approach that was used for software development.

The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

**Waterfall Model - Design**

Waterfall approach was first SDLC Model to be used widely in Software Engineering to ensure success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In this Waterfall model, typically, the outcome of one phase acts as the input for the next phase sequentially.

SDLC Waterfall Model

The sequential phases in Waterfall model are −

**Requirement Gathering and analysis**− All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

**System Design**−The requirement specifications from first phase are studied in this phase 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** −With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

**Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.

**Deployment of system**− Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

**Maintenance**− There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in which progress is seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for previous phase and it is signed off, so the name "Waterfall Model". In this model, phases do not overlap.

**Waterfall Model - Application**

Every software developed is different and requires a suitable SDLC approach to be followed based on the internal and external factors. Some situations where the use of Waterfall model is most appropriate are −

Requirements are very well documented, clear and fixed.

Product definition is stable.

Technology is understood and is not dynamic.

There are no ambiguous requirements.

Ample resources with required expertise are available to support the product.

The project is short.

Waterfall Model - Advantages

The advantages of waterfall development are that it allows for departmentalization and control. A schedule can be set with deadlines for each stage of development and a product can proceed through the development process model phases one by one.

Development moves from concept, through design, implementation, testing, installation, troubleshooting, and ends up at operation and maintenance. Each phase of development proceeds in strict order.

While the waterfall model has seen as low phasing out in recent years in favour of more agile methods, it can still provide a number of benefits, particularly for larger projects and organizations that require the stringent stage sand deadlines available within these cool, cascading waters.

Some of the major advantages of the Waterfall Model are as follows −

Simple and easy to understand and use

Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.

Phases are processed and completed one at a time.

Works well for smaller projects where requirements are very well understood.

Clearly defined stages.

Well understood milestones.

Easy to arrange tasks.

Process and results are well documented.

Waterfall Model - Disadvantages

The disadvantage of waterfall development is that it does not allow much reflection or revision. Once an application is in the testing stage, it is very difficult to go back and change something that was not well-documented or thought upon in the concept stage.

While some things in software development never really change, many others often fall by the way side. While Dr. Royce’s initial proposal of what is now known as the waterfall model was ground breaking when first published back in 1970, over four decades later, a number of cracks are showing in the arm or of this once heralded model.

The major disadvantages of the Waterfall Model are as follows −

No working software is produced until late during the life cycle.

High amounts of risk and uncertainty.

Not a good model for complex and object-oriented projects.

Poor model for long and ongoing projects.

Not suitable for the projects where requirements are at a moderate to high risk of changing. So, risk and uncertainty is high with this process model.

It is difficult to measure progress within stages.

Cannot accommodate changing requirements.

Adjusting scope during the life cycle can end a project.

Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

**4.2 Dataflow Diagram**

# 

Analyzes the feedback

Installation Feedback

**Administrator**

**Customer**

Retrieve Purchase Details and Desired Product Information

Sales

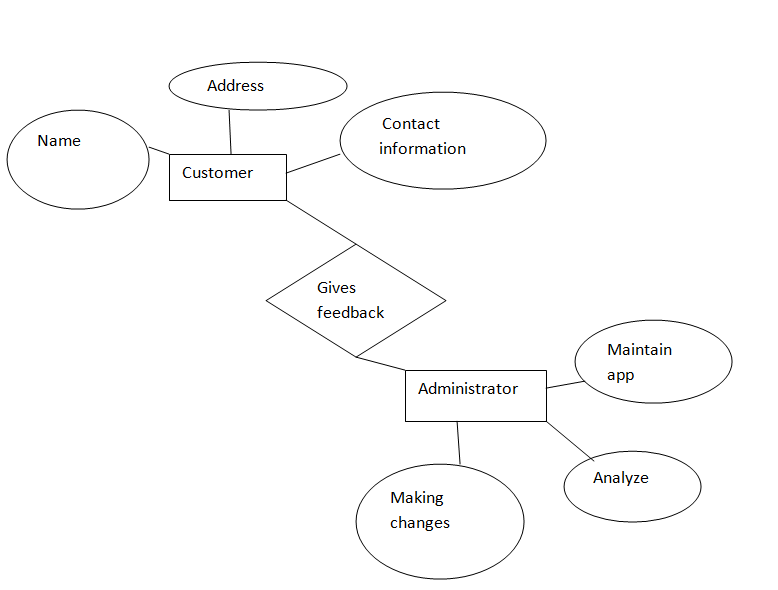
Application activity

Application management

**Application Administrator**

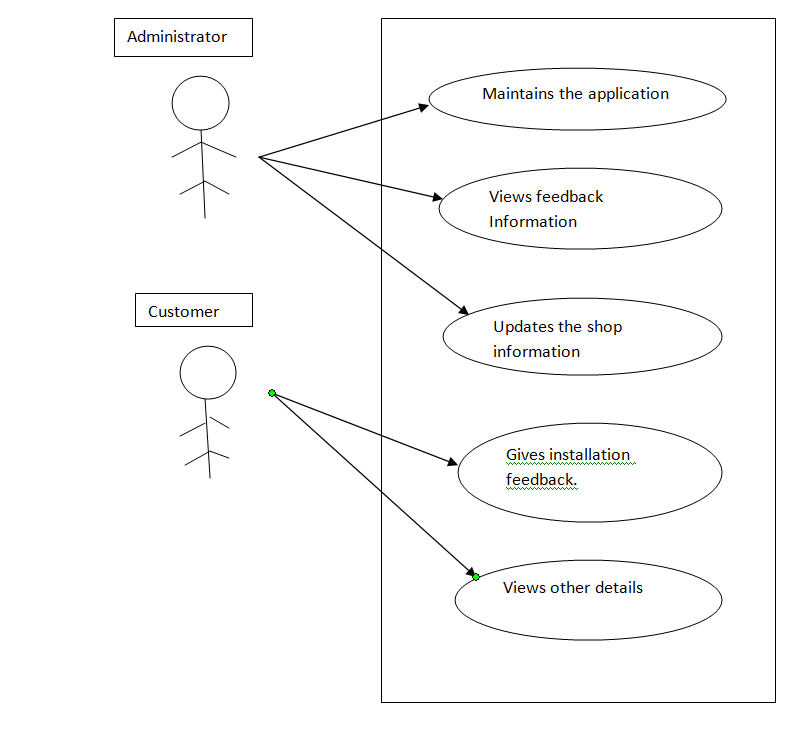
**Figure 4.2.1**

**4.3 ER Diagram**

****

**Figure 4.3.1**

**4.4 Use Case Diagram**

****

**Figure 4.4.1**

**4.5 User Interface Design**

**1. First Activity (Splash Activity)**

****

**Screen 4.5.1 Splash Activity snapshot**

This above snapshot 4.5.1 explains about the source code of splash activity which is generally designed to give user general idea about the application and shop.

**SOURCE CODE**

<?xml version="1.0" encoding="utf-8"?>

<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="com.example.administrator.naviationactivity.splash">

<ImageView

android:id="@+id/imageView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginBottom="42dp"

android:layout\_marginTop="42dp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:srcCompat="@drawable/ac" />

<TextView

android:id="@+id/textView15"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginBottom="8dp"

android:layout\_marginEnd="8dp"

android:layout\_marginStart="8dp"

android:layout\_marginTop="8dp"

android:fontFamily="cursive"

android:text="Air Conditioning Store"

android:textColor="@android:color/holo\_blue\_light"

android:textSize="30sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.953" />

<TextView

android:id="@+id/textView16"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginBottom="8dp"

android:layout\_marginEnd="8dp"

android:layout\_marginStart="8dp"

android:layout\_marginTop="8dp"

android:fontFamily="cursive"

android:text="Trust the experts."

android:textColor="@android:color/holo\_blue\_light"

android:textSize="30sp"

android:textStyle="bold"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintEnd\_toEndOf="parent"

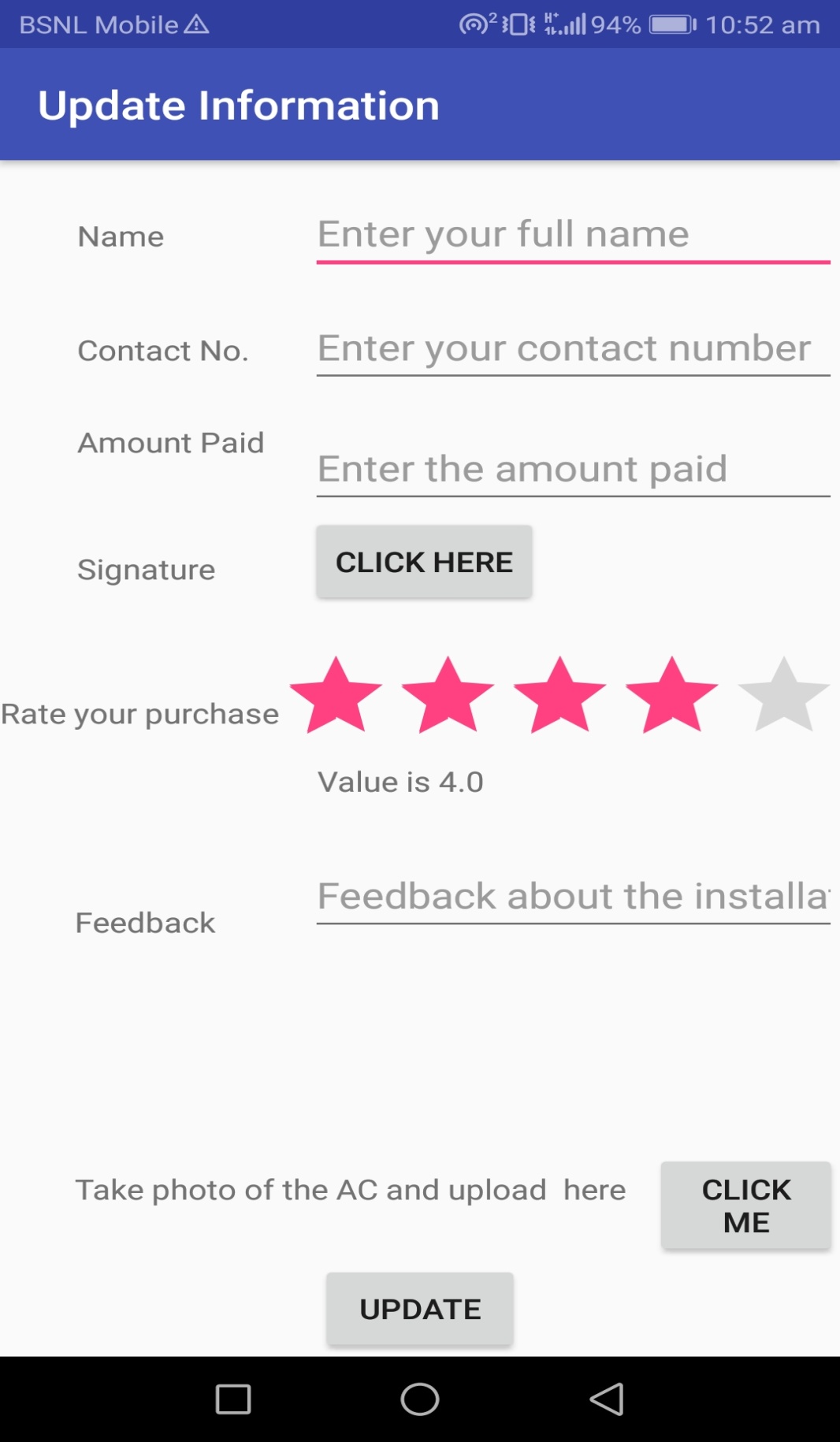
app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintVertical\_bias="0.041" />

</android.support.constraint.ConstraintLayout>

**2. Update Information Activity**



**Screen 4.5.2 Snap shot for feedback activity**

This snapshot explains about the functionality of how feedback is taken from the customer. It contains all the details about the information about the installation.

**SOURCE CODE**

package com.example.administrator.naviationactivity;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.app.FragmentManager;

import android.support.design.widget.FloatingActionButton;

import android.support.design.widget.Snackbar;

import android.support.design.widget.NavigationView;

import android.support.v4.view.GravityCompat;

import android.support.v4.widget.DrawerLayout;

import android.support.v7.app.ActionBarDrawerToggle;

import android.support.v7.widget.Toolbar;

import android.view.Menu;

import android.view.MenuItem;

import android.widget.TextView;

import android.content.Intent;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageView;

import android.widget.RatingBar;

import android.widget.TextView;

import android.widget.Toast;

import com.basgeekball.awesomevalidation.AwesomeValidation;

import com.basgeekball.awesomevalidation.ValidationStyle;

import com.basgeekball.awesomevalidation.utility.RegexTemplate;

public class UpdateInformation extends AppCompatActivity {

ImageView imageView;

RatingBar rb;

TextView value;

EditText fname,cnumber,ampaid,feed;

Button submit;

AwesomeValidation awesomeValidation;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_update\_information);

Button Clickhere = (Button) findViewById(R.id.oswald);

Clickhere.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view6) {

Intent i8 = new Intent(UpdateInformation.this, signaturepad.class);

startActivity(i8);

}

});

Button Clickme = (Button) findViewById(R.id.noddy);

Clickme.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view7) {

Intent i6 = new Intent(UpdateInformation.this, camera.class);

startActivity(i6);

}

});

awesomeValidation = new AwesomeValidation(ValidationStyle.BASIC);

updateUI();

rb = (RatingBar) findViewById(R.id.ratingBar);

value = (TextView) findViewById(R.id.value);

rb.setOnRatingBarChangeListener(new RatingBar.OnRatingBarChangeListener() {

@Override

public void onRatingChanged(RatingBar ratingBar, float rating, boolean fromUser) {

value.setText("Value is " + rating);

}

}); }

private void updateUI() {

fname = (EditText) findViewById(R.id.fname);

cnumber = (EditText) findViewById(R.id.cnumber);

ampaid = (EditText) findViewById(R.id.ampaid);

feed = (EditText) findViewById(R.id.feed);

submit =(Button) findViewById(R.id.submit);

awesomeValidation.addValidation(UpdateInformation.this,R.id.fname, "[a-zA-Z\\s]+", R.string.fnamer);

awesomeValidation.addValidation(UpdateInformation.this,R.id.cnumber, RegexTemplate.TELEPHONE,R.string.cnumberr);

awesomeValidation.addValidation(UpdateInformation.this,R.id.ampaid, RegexTemplate.TELEPHONE,R.string.ampaidr);

awesomeValidation.addValidation(UpdateInformation.this,R.id.feed,"[a-zA-Z\\s]+", R.string.feedr );

submit.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

if(awesomeValidation.validate()){

Toast.makeText(UpdateInformation.this, "Data received successfully" , Toast.LENGTH\_SHORT);

}

else

{

Toast.makeText(UpdateInformation.this, "Error" , Toast.LENGTH\_SHORT);

}

}

});

}

}