CONNECTING WRITERS WITH PUBLISHERS

Submitted by:

PRATEEK SINGH

Github: https://github.com/PrateekSingh29

LinkedIn: https://www.linkedin.com/in/prateek-singh-62065088/

TABLE OF CONTENTS

S No	Topic
3	List of Tables/Figures/Symbols
4	Chapter-1: Introduction
5	Chapter-2: System Analysis
6	Chapter-3: System Design
8	Appendices

LIST OF FIGURES

LIST OF TABLES

METHODOLOGY FOR THE PROJECT WORK

Chapter 1. Systems Introduction

1.1 Brief Description of the System under Study: The first step in system development life cycle is the identification of need of change to improve or enhance an existing system. An initial investigation on existing system was carried out. The present system of connecting writers and publishers is quiet difficult as all work was done offline. Many problems were identified during the initial study of the existing system. To develop this software detailed study is made with local writers and publishers. Based on the information collected it is decided to maintain the basic information about writers and publishers.

1.2 About the proposed System:

AIM: The aim of the study is to connect local publishers and writers and suggest an online connectivity system which will allow writers to cast their writings in a more convenient way, by using available resources which could be beneficial for both writers and publishers.

SCOPE: The scope of the software product developed involves connecting local writers and publishers so that publisher could select best content available and publish the same .It also help writers to publish their writings without any difficulty. It involves a very simple registration procedure to be done by the writers and publishers both.

1.3 Methodology used for Analysis, Design & Development:

SYSTEM DEVELOPMENT LIFECYCLE MODEL:

The **systems development life cycle** (**SDLC**), also referred to as the **application development life-cycle**, is a term used in systems engineering, information systems and software engineering to describe a process for planning, creating, testing, and deploying an information system.

SDLC aims to produce high-quality systems that meet or exceed customer expectations, based on customer requirements, by delivering systems which move through each clearly defined phase, within scheduled time frames and cost estimates. To do this number of SDLC models are developed which are as follows:

- WATERFALL MODEL
1. ITERATIVE ENHANCEMENT MODEL
2. EVOLUTIONARY MODEL
3. SPIRAL MODEL

4. PROTOTYPE MODEL

ITERATIVE ENHANCEMENT MODEL:

- The incremental model (also known as iterative enhancement model) comprises the features of waterfall model in an iterative manner.

- The basic idea of this model is to start the process with requirements and iteratively enhance the requirements until the final software is implemented.

- In addition, as in prototyping, the increment provides feedback from the user specifying the requirements of the software.

- This approach is useful as it simplifies the software development process as implementation of smaller increments is easier than implementing the entire system.

- Each stage of incremental model adds some functionality to the product and passes it on to the next stage.

- The first increment is generally known as a **core product** and is used by the user for a detailed evaluation. This process results in creation of a plan for the next increment. This plan determines the modifications (features or functions) of the product in order to accomplish user requirements.

- The iteration process, which includes the delivery of the increments to the user, continues until the software is completely developed.
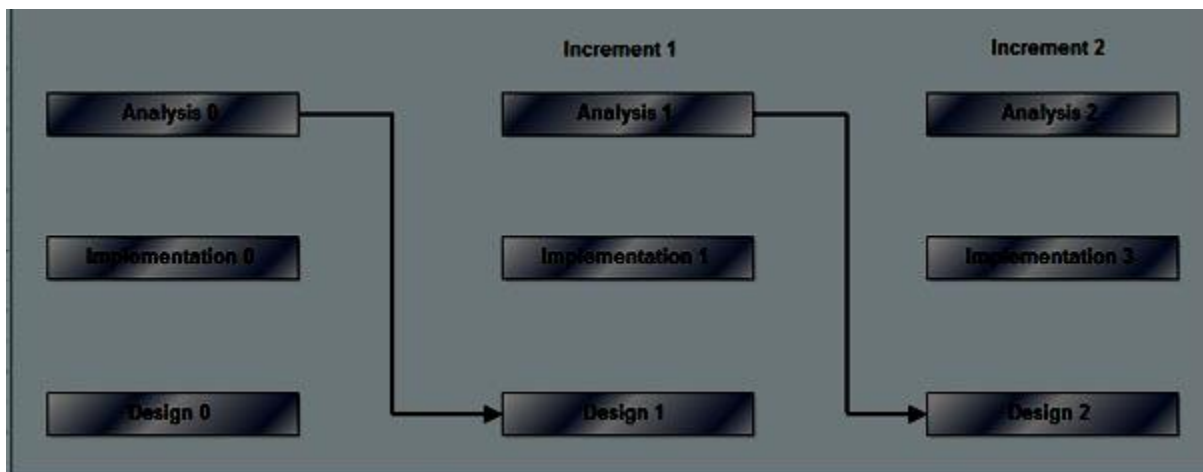
ADVANTAGES:
- Avoids the problems resulting in risk driven approach in the software
1. Understanding increases through successive refinements.

2.	Performs cost-benefit analysis before enhancing software with capabilities.
3.	Incrementally grows in effective solution after every iteration.
4.	Does not involve high complexity rate.
5.	Early feedback is generated because implementation occurs rapidly for a small subset of the software.

1.4 Methodology used for Data Collection:

Data Collection can be defined as the process of collecting information. There are two types of data:
(a) Primary data
(b) Secondary data

Primary data Collection
Primary data means original data that has been collected for the purpose in mind. It means that the data that has been collected from first hand experience is known as primary data.

Secondary Data Collection
Secondary Data Collection is that data which has been already collected by and readily available from other sources in any form. The secondary data are cheaper and more quickly obtainable than primary data.

Sources of secondary data for our project
Web search:
Wikipedia.org.
www.google.com
www.about.com

References: Software Engineering (K. K. Aggarwal)

1.5 System Requirement Tools:

Platform:
Android 2.3.3(Gingerbread)
Hardware Requirements:
4GB RAM
Software Requirements:
1. Android Studio 1.5.1

2. SQL(Structured Query Language)

3. 64 bit Operating System

Chapter 2 System Analysis

Software Requirement Specifications – is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Software requirements specification establishes the basis for an agreement between customers and contractors or suppliers (in market-driven projects, these roles may be played by the marketing and development divisions) on what the software product is to do as well as what it is not expected to do. Software requirements specification permits a rigorous assessment of requirements before design can begin and reduces later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules. Used appropriately, software requirements specifications can help prevent software project failure.

2.1 Introduction

The following subsections of Software Requirement Specifications Document should facilitate in providing the entire overview of the Information system "Connect writers with publishers online" under development. This document aims at defining the overall software requirements for publishers. Efforts have been made to define the requirements of the Information system exhaustively and accurately.

2.1.1   Purpose

The main purpose of Software Requirement Specifications Document is to describe in a precise manner all the capabilities that will be provided by the Software Application "Connecting writers with publishers". It also states the various constraints which the system will be abide to. This document further leads to clear vision of the software requirements, specifications and capabilities. These are to be exposed to the development, testing team and end users of the software

2.1.2   Scope – The scope of the software product developed involves connecting local writers and publishers so that publisher could select best content available and publish the same .It also help writers to publish their writings without any difficulty. It involves a very simple registration
procedure to be done by the writers and publishers both.

2.1.3 Definition, acronyms, abbreviations
   DFD: Data Flow Diagram

   IDE: Integrated Development Environment

   SRS: Software Requirements Specifications

2.1.4 References
- Book:
    I. K.K.Aggarwal,
    II. Rajib Mall
- Websites:
  I. https://www.google.co.in/search?q=google&rlz=1C1CHBD_enIN757IN757&oq=google&aqs=chrome..69i57j69i60j0j69i60l3.1351j0j7&sourceid=chrome&ie=UTF-8
  II. http://hackveda.in/one2one/profile.php
 III. https://developer.android.com/index.html

2.1.5 Overview

Our project mainly focus to connect local publishers and writers and suggest an online connectivity system which will allow writers to cast their writings in a more convenient way, by using available resources which could be beneficial for both writers and publishers. It will be time saving system for both.

The rest of this SRS document describes the various system requirements, interfaces, features and functionality in detail.

2.2 Overall description of proposed system

2.2.1 Product Perspective

The application will be windows-based, self contained and independent software product.
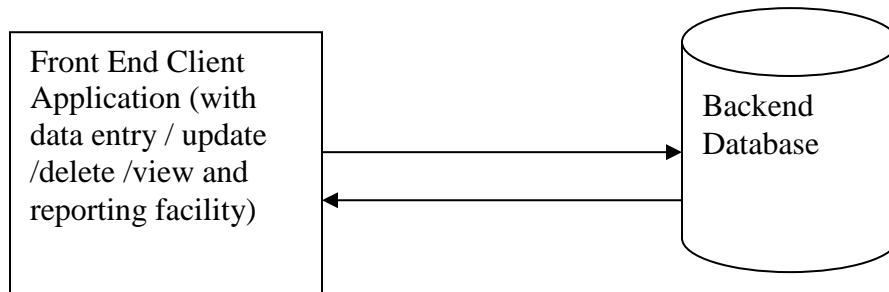
2.2.1.1 System Interfaces

None

2.2.1.2 Interfaces

The application will have a user friendly and menu based interface. Following screens will be provided.
  i. A registration screen will appear for writers & publishers to get themselves register with the proposed system.

ii.   A Login Screen for entering username, password and role (Writers, Publishers) will be provided. Access to different screens will be based upon the role of the user.

iii.   An upload screen will be displayed for writers to upload their content which they want to be displayed to publishers.

2.2.1.3 Hardware Interfaces

There is no special hardware interface requirement.

2.2.1.4 Software Interfaces

PHP MyAdmin

The phpMyAdmin name is obviously a mixture of PHP as the language it uses, MySQL as the database it manages and administration as the activity it handles. Even though the name seems to be quite simple, many people mix it up and they refer to phpMyAdmin under different names, such as myphpadmin, phpadmin, phpmysqladmin (these are the three most frequent Google searches going to this website aside from the correctly spelled variant).
By that time, phpMyAdmin had already become one of the most popular PHP applications and MySQL administration tools, with a large community of users and contributors.
In our respective system we used phpMyadmin for using web hosting services.To create database we had used phpMyAdmin tools.

2.2.1.5 Communication Interfaces

None

2.2.1.6 Memory Constraints

None

2.2.1.7 Operations

This product will not cover any automated housekeeping aspects of database. The DBA at client site will be manually deleting old/ non required data. Database backup and recovery will also have to be handled by DBA.

2.2.1.8 Site Adaptation Requirement

The terminals at client side will have to support the hardware and software interfaces specified.

2.2.2   Product functions

The system will allow access only to authorized users with specific roles (Administrator, Operator). Depending upon the user's role, he/she will be able to access only specific modules of the system.
A summary of the major functions that the software will perform:

i.   A Login facility for enabling only authorized access to the system. Writers & publishers both will login with their respective username & passward to generate a particular id .

So that they will get easily registered to the proposed system.

ii.　　Users will fetch/add/update/delete the stored information and so on

2.2.3　User Characteristics

1. Educational Level: At least graduate and should be comfortable with English language.

2. Technical Expertise: Should be a high or middle level employee of the organization comfortable with using general purpose applications on a computer

2.2.4　Constraints

None

2.2.5　Assumptions and Dependencies

2.2.6　Apportioning Requirement

Not Required

2.3　Specific Requirements

This section contains the software requirements to a level of detail sufficient to enable designers to design the system, and testers to test the system.

2.3.1　External Interfaces

2.3.1.1 User Interfaces

User interface is designed in a user friendly manner and the user, in another end he has to give the order, for that he will interface with keyboard and mouse.

2.3.2　System Features

1. Register

2. Login

3. Upload content

4. Display content

Description

Register module is to register both of the writers & publishers to the proposed system. Then login module is created to login by giving input details of both users. this will enable then to get connected with the application build.

Module of upload content sis designed to upload the writer's book details & study material for publishers to read the content.

10

Validity Checks

Sequencing Information

Error Handling / Response to abnormal situations

2.1.1   Performance Requirements

None

2.1.2   Logical Database Requirements

The proposed information system contains the following data tables in its database collection.

1. Writers
2. Publishers
3. Book

2.1.3   Design Constraints

2.1.3.1 Standard Compliance

None

2.1.4   Software System Attributes

Reliability

This application is a reliable product that produces fast and verified output of all its processes.

Availability

This application will be available to use for end users i.e Publishers and help them to carry out their operations conveniently.

Security

The application will be password protected. User will have to enter correct username, password and role in order to access the application.

Maintainability

The application will be designed in a maintainable manner. It will be easy to to incorporate new requirements in the individual modules.

Portability

The application will be easily portable on any windows-based system that has oracle installed.

2.1.5 Other Requirements

None

3 Methodologies for Data Collection

3.1 Primary Data Collection

Questionnaire to publishers

ABOUT THE PUBLISHER

Please provide the following information for each author or editor. You may fill out a separate sheet for each person.

1. Legal name:
2. Academic degrees and awarding institutions:
3. Current affiliations (title, department or division, institution) and full mailing address:
4. Residential address:
5. Citizenship:
6. Telephone number (office, home, cell):
7. Email address:
8. Preferred mailing address :
9. Membership in professional associations and societies (please indicate any in which you are an officer, officer-elect, or past officer):
10. Any books published, edited, or contributed to (please identify titles, writer):
11. Periodicals that have published:
12. Any specific requirements(from writers):

1.2 Secondary Data Collection

- Book:

    K.K.Aggarwal,

    Rajib Mall

The Internet (Websites):

- https://www.google.co.in/search?q=google&rlz=1C1CHBD_e
  nIN757IN757&oq=google&aqs=chrome..69i57j69i60j0j69i60
  l3.1351j0j7&sourceid=chrome&ie=UTF-8
- http://hackveda.in/one2one/profile.php
- https://developer.android.com/index.html

4. Methodology used for Analysis, Design and Development

4.1 Stages of Model

In this incremental model, the whole requirement is divided into various builds. During each iteration, the development module goes through the requirements, design, implementation and testing phases. Each subsequent release of the module adds function to the previous release. The process continues till the complete system is ready as per the requirement.

A Requirements phase, in which the requirements for the software are gathered and analysed. Iteration should eventually result in a requirements phase that produces a complete and final specification of requirements.

A Design phase, in which a software solution to meet the requirements is designed. This may be a new design, or an extension of an earlier design.

An Implementation and Test phase, when the software is coded, integrated and tested.

A Review phase, in which the software is evaluated, the current requirements are reviewed, and changes and additions to requirements proposed.

4.2 Block Diagram of Model
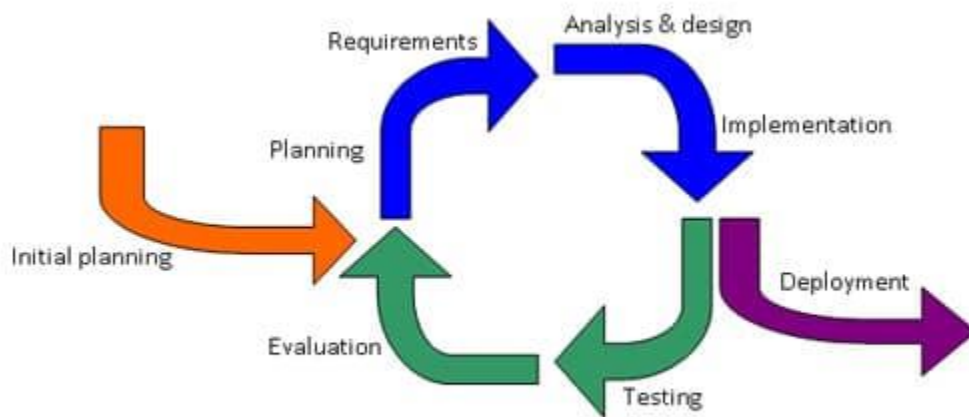
Model 1: Typical iterative development process

4.3 Reasons for choosing Model

Iterative enhancement model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements.

In iterative model we can only create a high-level design of the application before we actually begin to build the product and define the design solution for the entire product.

In iterative model we are building and improving the product step by step. Hence we can track the defects at early stages.

5 Project Planning Gantt chart

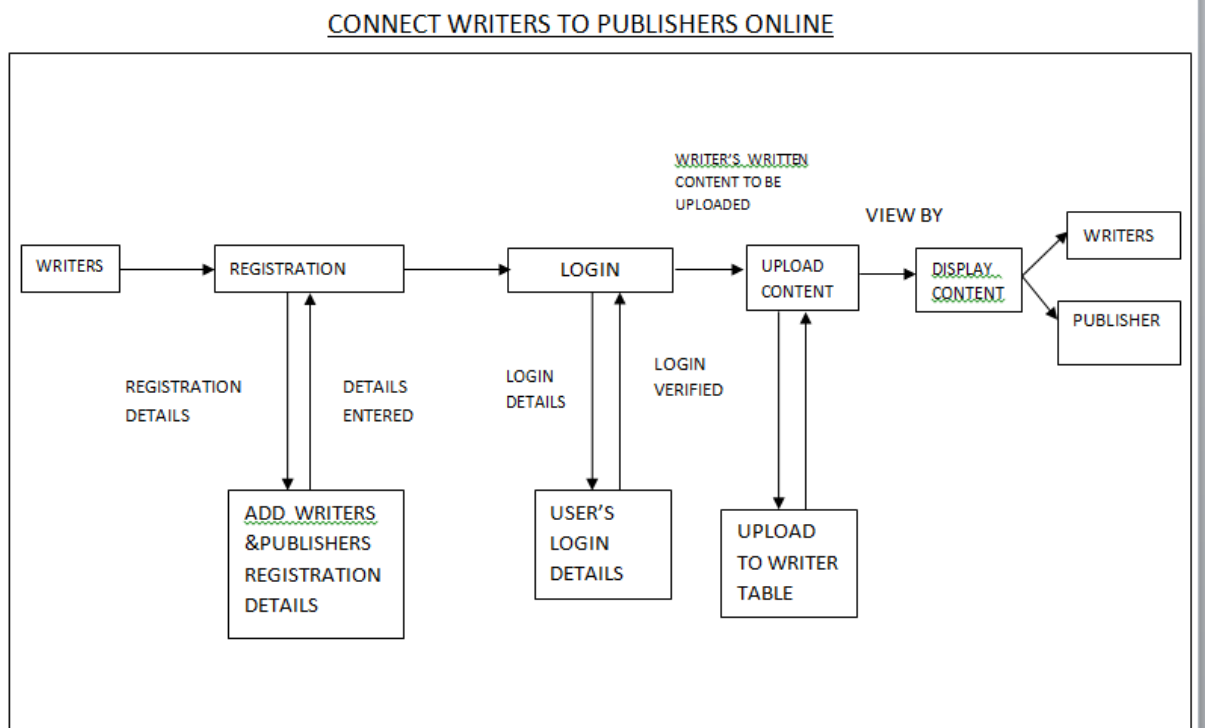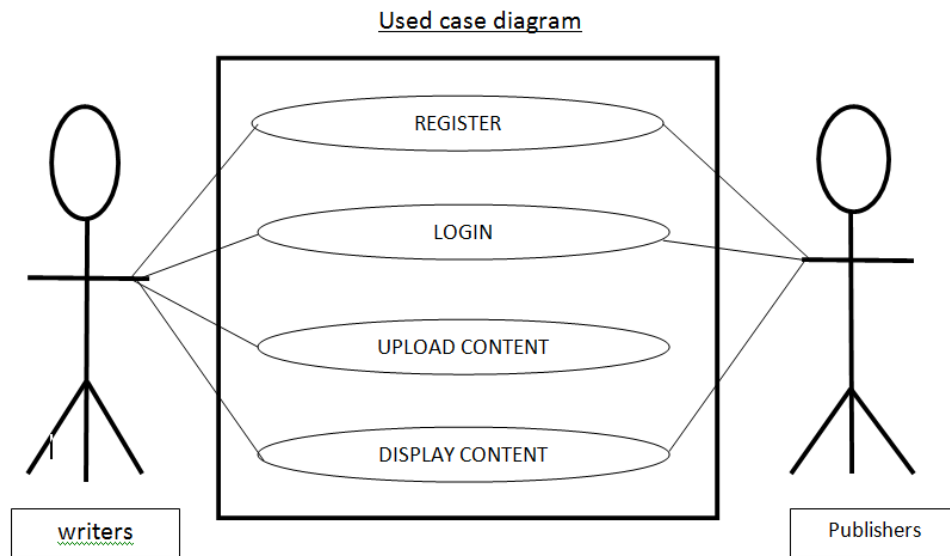|		Task Mode	Task Name	Duration	Start	Finish
1			Requirement Gathering and Analysis	28 days	Wed 7/5/17	Wed 8/9/17
2			Synopsis	3 days	Wed 7/5/17	Fri 7/7/17
3			Chapter-1:Software Project Planning	9 days	Fri 7/7/17	Mon 7/17/17
4			1.1 Description of the Software System under Study	2 days	Fri 7/7/17	Mon 7/10/17
5			1.2 Data Collection	2 days	Tue 7/11/17	Wed 7/12/17
6			1.3 Tools/Platforms	2 days	Thu 7/13/17	Fri 7/14/17
7			1.4 Project Planning	1 day	Sat 7/15/17	Sat 7/15/17
8			1.5 Methodology	2 days	Sun 7/16/17	Mon 7/17/17
9			Chapter-2:Software Requirement Specification(SRS)	6 days	Mon 7/17/17	Mon 7/24/17
10			2.1 Description of Information System(Block Diagram)	2 days	Mon 7/17/17	Tue 7/18/17
11			2.2 Use Case Diagram	2 days	Wed 7/19/17	Thu 7/20/17
12			2.3 Software Product Constraints	2 days	Fri 7/21/17	Mon 7/24/17
13			Chapter-3: Software Project Analysis	6 days	Mon 7/24/17	Mon 7/31/17
14			3.1 DFD	1 day	Mon 7/24/17	Mon 7/24/17
15			3.2 Entity-Relationship Diagram	1 day	Tue 7/25/17	Tue 7/25/17
16			3.3 Database Specifications	2 days	Wed 7/26/17	Thu 7/27/17
17			3.4 Validation Specifications	2 days	Fri 7/28/17	Mon 7/31/17
18			Chapter-4: Software Design	3 days	Mon 7/31/17	Wed 8/2/17
19			4.1 Software Project Flowchart	1 day	Mon 7/31/17	Mon 7/31/17
20			4.2 Layouts of Input and Output design	2 days	Tue 8/1/17	Wed 8/2/17
21			Chapter-5:Implementation of the System	3 days	Wed 8/2/17	Fri 8/4/17
22			Chapter-6: Testing	4 days	Fri 8/4/17	Wed 8/9/17
23			6.1 Functional Testing	2 days	Fri 8/4/17	Mon 8/7/17
24			6.2 Validation Testing	2 days	Tue 8/8/17	Wed 8/9/17

Chapter 3: System Design

1. Block Diagram

CONNECT WRITERS TO PUBLISHERS ONLINE

2.   Use Case

16

Used case diagram


```
writers          REGISTER          Publishers

                  LOGIN

              UPLOAD CONTENT

              DISPLAY CONTENT
```

3. DFD

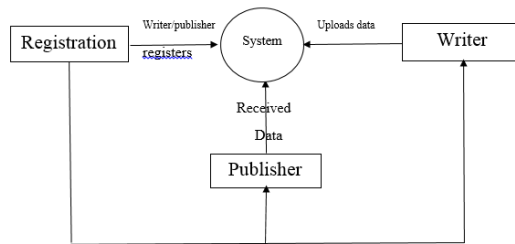Context level DFD

| Writer | Uploads → | System | ← Receiving data | Publisher |

Level -1 DFD

17

Level 1

4. ER Diagram

Writers &publishers are two entities connecting together. Writers table have attributes as w_name, address, id, password, username, contact of writer ,in this table w_name is considered as the primary key ,which makes it unique.

In publishers table have attributes as p_id , username, password, p_name , address,contact of the publisher,here p_id is considered primary key of the table.

Ther is another entity named Book having attribues Book name, author,

Database Design

18
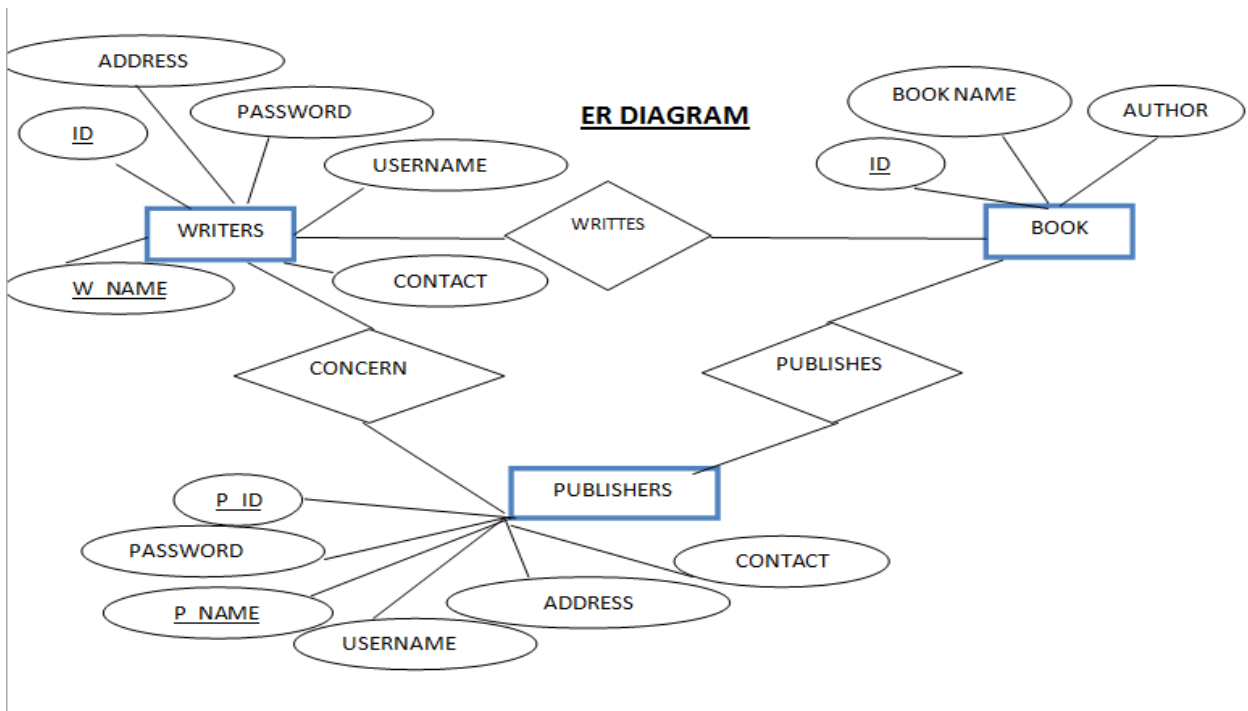
The information system of "CONNECT WRITERS WITH PUBLISHERS ONLINE" performs its function with the help of the data store in certain repositories called Databases of the system. Database design can be represented in the following tabulated form:

User TABLE:

ID	Useraname	Usertype	Password	fathername	firstname	lastname	mobile	emailid	address

BOOKS TABLE

Id	Item_name	Item_price	Item_desc	status	page	author

3. Interface Design

The interface design consists of the input and output source layouts. i.e. the input forms and screens and the report layouts that form as a source of outcome and income in the design and implementation of the information system under study
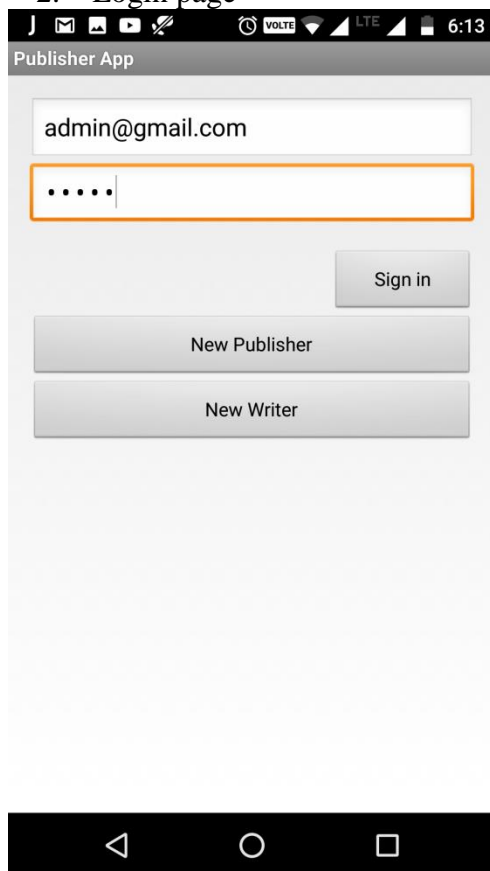
3.1 Input Design

The input specifications of the existing information system include the illustration of the detailed characteristics of contents included in each Input Screen and documents. The description for each graphical user interface has been mentioned.
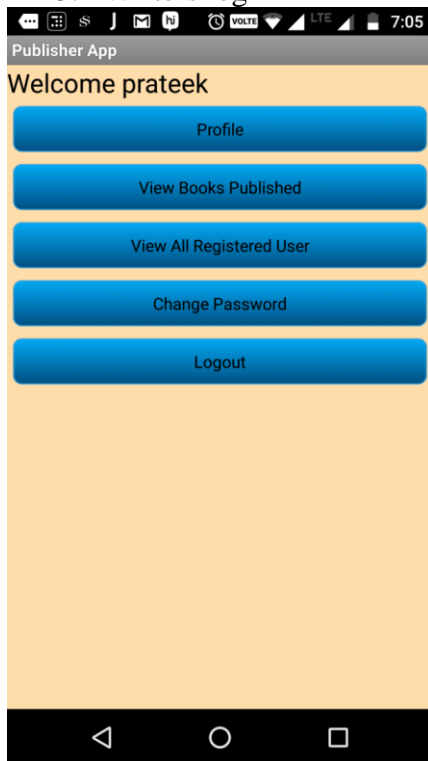
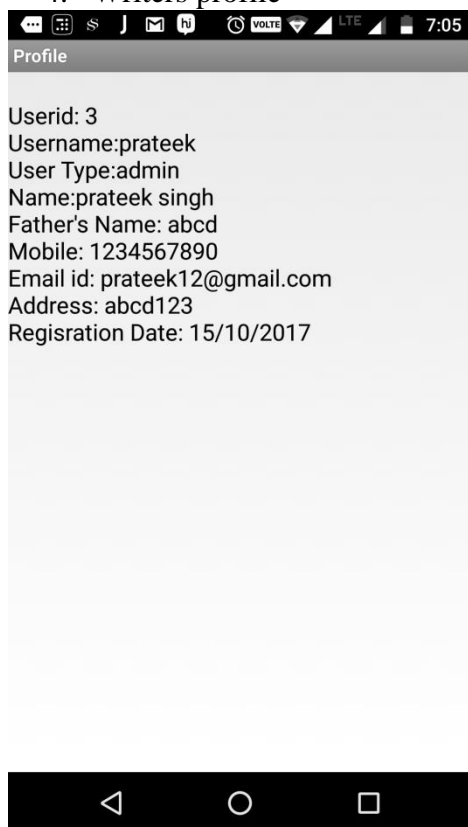EXISTING SYSTEM DESIGN (Graphical User Interface)

1. Splash screen

2. Login page

3. Writers login

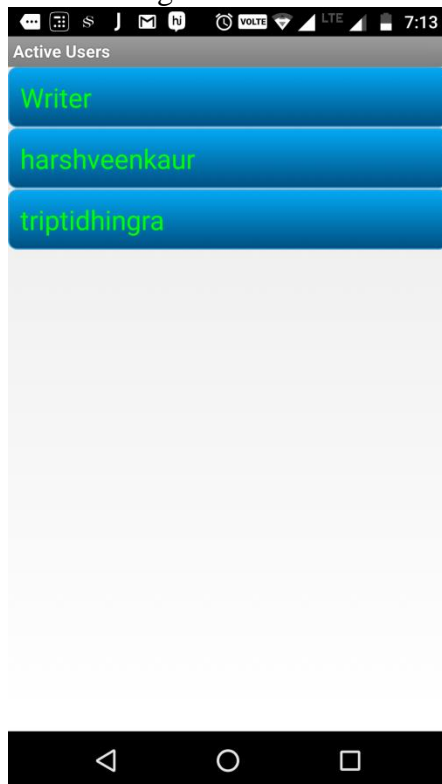4. Writers profile

5. View books published

6. All registered users

7. Change password

8. Publisher login

9. Writer profile

Publisher App

Userid: 4
Username:veene
User Type:user
Name:harshveen kaur
Father's Name: abcdef
Mobile: 1234567800
Email id: veene12@gmail.com
Address: abcd123
Regisration Date: 15/10/2017

10. Post book

Post Books

the immortals of meluha

123456777

500

the immortals of meluha by amish

250

religious thrill

Click Image

Save Item

11. View books

12. Change password

3.2 Source Code

ADMIN PROFILE

```java
package com.example.emp_tracking;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.widget.TextView;


import publisher.writer.com.writerpublisher.R;


public class AdminProfile extends Activity {
    TextView tvusername;

@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.admin_profile);

    tvusername=(TextView) findViewById(R.id.tvusername);
    Intent i=getIntent();
    String strusername=i.getStringExtra("username");

    DBAdapter db=new DBAdapter(this);
    db.open();
    Cursor c=db.getUserByUsername(strusername);
    if(c.moveToNext()){
        final String userdetails = "\nUserid:
"+c.getString(0)+"\nUsername:"+c.getString(1)+"\nUser
Type:"+c.getString(2)+"\nName:"+c.getString(5)
            +" "+c.getString(6)+"\nFather's Name: "+c.getString(4)+"\nMobile:
"+c.getString(7)+"\nEmail id: "+c.getString(8)+"\nAddress:
"+c.getString(9)+"\nRegisration Date: "+c.getString(10);
        tvusername.setText(userdetails);
    }



    db.close();



}




}
```

Change Password-

```java
package com.example.emp_tracking;


import publisher.writer.com.writerpublisher.R;
import android.os.Bundle;
import android.app.Activity;
import android.content.Intent;
import android.view.Menu;
```

```java
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class ChangePassword extends Activity {
    String strCurrentPass = null, strNewPass = null, strConfirmPass = null;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_change_password);
        final EditText etCurrentPass = (EditText)
findViewById(R.id.etCurrentPassword);
        final EditText etNewPass = (EditText) findViewById(R.id.etNewPassword);
        final EditText etconfirmpass = (EditText)
findViewById(R.id.etConfirmPassword);


        Intent i = getIntent();
        final String username = i.getStringExtra("username");

        Button btnUpdatePass = (Button) findViewById(R.id.btnProcced2Pay4Ticket);
        Button btnCancelUpdate = (Button)
findViewById(R.id.btnCancelUpdatePassword);
        btnUpdatePass.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                strCurrentPass = etCurrentPass.getText().toString();
                strConfirmPass = etconfirmpass.getText().toString();
                strNewPass = etNewPass.getText().toString();

System.out.println("curr"+strCurrentPass+"new:"+strNewPass+"confirm:"+strConfir
mPass);
                DBAdapter db = new DBAdapter(getApplication());
                db.open();
                if (db.validateUser(username, strCurrentPass)) {

                    if(strNewPass.equalsIgnoreCase(strConfirmPass)){

                        db.updatePassword(username, strNewPass);
                        Toast.makeText(getApplication(), "Password Updated!",
                                Toast.LENGTH_SHORT).show();
                    }else Toast.makeText(getApplication(), "New Password & Confirm
Password Field do not match.",
                            Toast.LENGTH_SHORT).show();
                } else
                    Toast.makeText(getApplication(), "Wrong Password!",
                            Toast.LENGTH_SHORT).show();

                db.close();
            }
        });
        btnCancelUpdate.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                finish();
            }
        });
```

```
    }


}
```

Database code

```java
package com.example.emp_tracking;

import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import android.annotation.SuppressLint;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteStatement;
import android.util.Log;

@SuppressLint("SimpleDateFormat")
public class DBAdapter {
    public static final String KEY_ROWID = "_id";// to save id //
    public static final String KEY_USERNAME = "username"; //
    public static final String KEY_PASSWORD = "password_"; //
    public static final String KEY_USERTYPE = "usertype"; //
    public static final String KEY_FIRSTNAME = "firstname"; //
    public static final String KEY_LASTNAME = "lastname"; //
    public static final String KEY_FATHERNAME ="fathername";
    public static final String KEY_MOBILE = "mobile"; //
    public static final String KEY_EMAILID = "emailid"; //
    public static final String KEY_ADDRESS = "address"; //
    public static final String KEY_DATE = "date1";// to save date of
registration table
    public static final String KEY_TIME = "time1";// to save time
    public static final String KEY_STATUS = "status";// to check status of task
    private static final String TAG ="db";// for database adapter
    public static final String DATABASE_NAME = "PUBLISHER"; //$NON-NLS-1$
    public static final String DATABASE_TABLE = "REGISTERATION"; //$NON-NLS-1$
    private static final int DATABASE_VERSION = 1;
    private static final String DATABASE_CREATE = "create table REGISTERATION
(_id integer primary key autoincrement, " //$NON-NLS-1$
            + "username text unique not null,usertype text not null, password_
text not null,fathername text not null,firstname text not null,lastname text
not null,mobile text not null,emailid text unique not null,address text not
null, date1 text not null, time1 text not null, status text not null);";
//$NON-NLS-1$
        public static final String KEY_ITEM_NAME = "item_name";
    public static final String KEY_ITEM_PRICE = "item_price";
    public static final String KEY_ITEM_ISDN = "item_ISDN";
    public static final String KEY_ITEM_DONATE = "item_donate";
    public static final String KEY_ITEM_DESC = "item_desc";
    public static final String KEY_ITEM_page = "page";
    public static final String KEY_title = "title";
    public static final String KEY_author = "author";
    public static final String KEY_offeredPrice = "offeredPrice";
    public static final String KEY_status = "status";
    public static final String KEY_IMG = "item_img";
    public static final String DATABASE_TABLE_ITEMS = "books"; // to

    private static final String DATABASE_CREATE_ITEM = "create table
```

```java
        "+DATABASE_TABLE_ITEMS+" (_id integer primary key autoincrement, " //$NON-NLS-
1$
            + "item_name text not null,item_ISDN text unique not null,"
            + "item_price text not null,item_donate text not null,"
            + "item_desc text not null, item_img text not null,"
            + "status text not null,userid text not null,"
            + "page text not null,title text not null,"
            + "author text not null,offeredPrice text not null);"; //$NON-NLS-1$
    private final Context context;
    private DatabaseHelper DBHelper;
    private static SQLiteDatabase db;

    public DBAdapter(Context ctx) {
        // super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
        this.context = ctx;
        DBHelper = new DatabaseHelper(context);
    }

    private static class DatabaseHelper extends SQLiteOpenHelper {
        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }

        @Override
        public void onCreate(SQLiteDatabase db) {
            try {
                db.execSQL(DATABASE_CREATE);
                db.execSQL(DATABASE_CREATE_ITEM);
                saveInitialValues(db);
                Log.w(TAG, "database created "); //$NON-NLS-1$
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }

        private static void save(SQLiteDatabase db, String username,
                String usertype, String password, String firstname,
                String lastname, String fathername, String mobile,
                String emailid, String address, String date, String time,
                String status) {
            ContentValues initialValues = new ContentValues();
            initialValues.put(KEY_USERNAME, username);
            initialValues.put(KEY_PASSWORD, password);
            initialValues.put(KEY_USERTYPE, usertype);
            initialValues.put(KEY_FIRSTNAME, firstname);
            initialValues.put(KEY_LASTNAME, lastname);
            initialValues.put(KEY_FATHERNAME, fathername);
            initialValues.put(KEY_MOBILE, mobile);
            initialValues.put(KEY_EMAILID, emailid);
            initialValues.put(KEY_ADDRESS, address);
            initialValues.put(KEY_DATE, date);
            initialValues.put(KEY_TIME, time);
            initialValues.put(KEY_STATUS, status);

            db.insert(DATABASE_TABLE, null, initialValues);
        }

        private static void saveInitialValues(SQLiteDatabase db) {

            save(db,
                    "Publisher", "Publishertest", "asdfgh", "Publisher", "",
"ABCDE", "123456789", "writer@gmail.com", "Delhi", "21/10/2013", "18:44:25",
"True"); //$NON-NLS-1$ //$NON-NLS-2$ //$NON-NLS-3$ //$NON-NLS-4$ //$NON-NLS-5$
```

29

```java
//$NON-NLS-6$ //$NON-NLS-7$ //$NON-NLS-8$ //$NON-NLS-9$ //$NON-NLS-10$ //$NON-
NLS-11$ //$NON-NLS-12$
        save(db,
                "Writer", "Writertest", "asdfgh", "Writer", "", "ABCDE",
"989898998", "user@gmail.com", "Delhi", "21/10/2013", "18:44:25", "True");
//$NON-NLS-1$ //$NON-NLS-2$ //$NON-NLS-3$ //$NON-NLS-4$ //$NON-NLS-5$ //$NON-
NLS-6$ //$NON-NLS-7$ //$NON-NLS-8$ //$NON-NLS-9$ //$NON-NLS-10$ //$NON-NLS-11$
//$NON-NLS-12$


    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
{
        Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
//$NON-NLS-1$ //$NON-NLS-2$
                + newVersion + ", which will destroy all old data"); //$NON-NLS-
1$
        db.execSQL("DROP TABLE IF EXISTS tasks"); //$NON-NLS-1$
        onCreate(db);// to create db
    }
}
    public long saveItems(String lname,
                    String ldesc, String img,String status,String userid,String
isdn,String donate,String price, String page, String title, String author,
String offeredPrice) {
        ContentValues initialValues = new ContentValues();

        initialValues.put(KEY_ITEM_NAME, lname);
        initialValues.put(KEY_ITEM_PRICE, price);
        initialValues.put(KEY_ITEM_ISDN, isdn);
        initialValues.put(KEY_ITEM_DONATE, donate);
        initialValues.put(KEY_ITEM_DESC, ldesc);

        initialValues.put(KEY_IMG, img);
        initialValues.put(KEY_status, status);
        initialValues.put("userid", userid);
        initialValues.put(KEY_ITEM_page, page);
        initialValues.put(KEY_title, title);
        initialValues.put(KEY_author, author);
        initialValues.put(KEY_offeredPrice, offeredPrice);
        return db.insert(DATABASE_TABLE_ITEMS, null, initialValues);

    }

    public int activateUser(String userna) {

        ContentValues args = new ContentValues();

        args.put(KEY_STATUS, "True");

        return db.update(DATABASE_TABLE, args, KEY_USERNAME
                + "=" + "'" + userna + "'", null); //$NON-NLS-1$

    }

    // ---opens the database---
    public DBAdapter open() throws SQLException {
        db = DBHelper.getWritableDatabase();
        return this;
    }

    // ---closes the database---
```

```java
    public void close() {
        DBHelper.close();
    }

    /*
     * private static final String DATABASE_CREATE_APPOINT = "create table " +
     * DATABASE_TABLE_APPOINT + " (_id integer primary key autoincrement, "
     * //$NON-NLS-1$ + "" + KEY_f_userid + " text not null," + KEY_USERID +
     * " text not null," + KEY_APPOINTMENT_TIME +
     * " text not null,"+KEY_APPOINT_DATE
     * +" text not null,"+KEY_APPOINTMENT_TIME+" text null,status text null);";
     * //$NON-NLS-1$
     */

    public String getDate() {
        DateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");
        // DateFormat dateFormat1 = new SimpleDateFormat("HH:mm:ss");

        Calendar cal = Calendar.getInstance();

        String reg_date = dateFormat.format(cal.getTime());// "11/03/14
12:33:43";
        // String reg_time = dateFormat1.format(cal.getTime());
        return reg_date;

    }

    public Cursor getItemsByKey(String key) {
        String sql;
if(key==null){
    sql="select * from "+DATABASE_TABLE_ITEMS;
}else
        sql="select * from "+DATABASE_TABLE_ITEMS+" where "+KEY_ITEM_NAME+" like
'%"+key+"%'";
        Log.i(AppConstant.tag, sql);
        Cursor c=db.rawQuery(sql, null);

        return c;

    }
    // ---register a user into the database---
    public long registerUser(String username, String usertype, String password,
            String firstname, String lastname, String fathername,
            String mobile, String emailid, String address, String date,
            String time, String status) {
        ContentValues initialValues = new ContentValues();
        initialValues.put(KEY_USERNAME, username);
        initialValues.put(KEY_PASSWORD, password);
        initialValues.put(KEY_USERTYPE, usertype);
        initialValues.put(KEY_FIRSTNAME, firstname);
        initialValues.put(KEY_LASTNAME, lastname);
        initialValues.put(KEY_FATHERNAME, fathername);
        initialValues.put(KEY_MOBILE, mobile);
        initialValues.put(KEY_EMAILID, emailid);
        initialValues.put(KEY_ADDRESS, address);
        initialValues.put(KEY_DATE, date);
        initialValues.put(KEY_TIME, time);
        initialValues.put(KEY_STATUS, status);

        return db.insert(DATABASE_TABLE, null, initialValues);

    }
```

```java
    // ---deletes a particular contact---
    public boolean deleteContact(long rowId) {
        return db.delete(DATABASE_TABLE, KEY_ROWID + "=" + rowId, null) > 0;
//$NON-NLS-1$
    }

    public int getRows() {
        String sql = "SELECT COUNT(*) FROM " + DATABASE_TABLE; //$NON-NLS-1$

        SQLiteStatement statement = db.compileStatement(sql);
        long count = statement.simpleQueryForLong();
        return (int) count;

    }

    // ---retrieves all the contacts---
    public Cursor getAllRegisteredActiveUser() {

        return db.query(DATABASE_TABLE, new String[] { KEY_ROWID, KEY_USERNAME,
                KEY_USERTYPE, KEY_FIRSTNAME, KEY_LASTNAME, KEY_FATHERNAME,
                KEY_MOBILE, KEY_ADDRESS, KEY_EMAILID, KEY_DATE, KEY_TIME,
                KEY_STATUS },
                KEY_STATUS + "=" + "'True'", null, null, null, null); //$NON-NLS-1$
//$NON-NLS-2$

    }

    public Cursor getAllRegisteredUser() {

        return db.query(DATABASE_TABLE, new String[] { KEY_ROWID, KEY_USERNAME,
                KEY_USERTYPE, KEY_FIRSTNAME, KEY_LASTNAME, KEY_FATHERNAME,
                KEY_MOBILE, KEY_ADDRESS, KEY_EMAILID, KEY_DATE, KEY_TIME,
                KEY_STATUS }, null, null, null, null, KEY_ROWID + " desc");

    }

    public Cursor getAllRegisteredUser(String type) {

        return db.rawQuery("select * from " + DATABASE_TABLE
                + " where usertype='" + type + "'", null);

    }



    public Cursor getValidRegisteredUser() {

        return db.query(DATABASE_TABLE, new String[] { KEY_ROWID, KEY_USERNAME,
                KEY_USERTYPE, KEY_FIRSTNAME, KEY_LASTNAME, KEY_FATHERNAME,
                KEY_MOBILE, KEY_ADDRESS, KEY_EMAILID, KEY_DATE, KEY_TIME,
                KEY_STATUS }, KEY_STATUS + "=" + "'True'", null, null, null,
                null);

    }

    public Cursor getValidRegisteredUserExceptSelf(String username) {

        return db.query(DATABASE_TABLE, new String[] { KEY_ROWID, KEY_USERNAME,
                KEY_USERTYPE, KEY_FIRSTNAME, KEY_LASTNAME, KEY_FATHERNAME,
                KEY_MOBILE, KEY_ADDRESS, KEY_EMAILID, KEY_DATE, KEY_TIME,
                KEY_STATUS }, KEY_STATUS + "=" + "'True'" + " and "
                + KEY_USERNAME + "!=" + "'" + username + "'", null, null, null,
                null);
```

```java
    }

    public int updatePassword(String username, String password) {
        ContentValues args = new ContentValues();

        args.put(KEY_PASSWORD, password);

        return db.update(DATABASE_TABLE, args, KEY_USERNAME
                + "=" + "'" + username + "'", null); //$NON-NLS-1$

    }

    public Cursor getAllRegisteredUserWithPassword() {

        return db.query(DATABASE_TABLE, new String[] { KEY_ROWID, KEY_USERNAME,
                KEY_USERTYPE, KEY_PASSWORD, KEY_FIRSTNAME, KEY_EMAILID,
                KEY_STATUS }, null, null, null, null, null);

    }

    // ---retrieves a particular contact---
    public Cursor getUserById(long rowId) throws SQLException {
        Cursor mCursor = db.query(DATABASE_TABLE, new String[] { "*" },
                KEY_ROWID + "=" + rowId, null, null, null, null); //$NON-NLS-1$

        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;
    }

    public Cursor getUserMobileByEmail(String email) throws SQLException {
        Cursor mCursor = db.query(DATABASE_TABLE, new String[] { KEY_MOBILE,
                KEY_PASSWORD },
                KEY_EMAILID + "=" + "'" + email + "'", null, null, null, null);
//$NON-NLS-1$

        if (mCursor != null) {
            mCursor.moveToFirst();
        }
        return mCursor;
    }

    public Cursor getUserByUsername(String username) throws SQLException {
String sql="select * from "+DATABASE_TABLE+" where
"+KEY_USERNAME+"='"+username+"'";
        Log.i(TAG, "getUserByUsername: "+sql
        );
        Cursor mCursor=db.rawQuery(sql,null);
        System.out.println(KEY_USERTYPE + "=" + "'" + username + "'");

        return mCursor;
    }

    public Cursor getUserByUserID(String username) throws SQLException {
        Cursor mCursor = db
                .query(DATABASE_TABLE,
                        new String[] { "*" },
                        KEY_ROWID
                                + "=" + "'" + username + "' and usertype!='admin' ||
usertype!='user'", null, null, null, null); //$NON-NLS-1$
```

```java
        System.out.println(KEY_USERTYPE + "=" + "'" + username + "'");
        if (mCursor != null) {
            return mCursor;
        }
        return mCursor;
    }

    public Boolean validateUser(String username, String password)
            throws SQLException {
        Cursor mCursor = db
                .query(DATABASE_TABLE,
                        new String[] { KEY_USERNAME, KEY_STATUS },
                        KEY_USERNAME
                                + "=" + "'" + username + "'" + " and " + KEY_PASSWORD +
"=" + "'" + password + "'", null, null, null, null); //$NON-NLS-1$ //$NON-NLS-
2$ //$NON-NLS-3$

        if (mCursor != null) {
            return true;
        }
        return false;
    }

    // ---updates a contact---
    public boolean updateContact(String username, String password,
            String firstname, String lastname, String mobile, String emailid,
            String address) {
        ContentValues args = new ContentValues();
        args.put(KEY_FIRSTNAME, firstname);
        args.put(KEY_LASTNAME, lastname);
        args.put(KEY_PASSWORD, password);

        args.put(KEY_MOBILE, mobile);
        args.put(KEY_EMAILID, emailid);
        args.put(KEY_ADDRESS, address);

        return db.update(DATABASE_TABLE, args,
                KEY_USERNAME + "=" + username, null) > 0; //$NON-NLS-1$
    }

    // ---updates a contact---
    public boolean deactivateUser(String username) {
        ContentValues args = new ContentValues();

        args.put(KEY_STATUS, "False");

        return db.update(DATABASE_TABLE, args, KEY_USERNAME
                + "=" + "'" + username + "'", null) > 0; //$NON-NLS-1$
    }

}
```

Forget Password

```java
package com.example.emp_tracking;


import publisher.writer.com.writerpublisher.R;

import android.app.Activity;
import android.database.Cursor;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
```

```java
import android.widget.EditText;
import android.widget.Toast;

public class ForgetPassword extends Activity{
    EditText etForget=null;
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.forget_password);
    Button btnSubmit=(Button)findViewById(R.id.btnSumbitForget);
    etForget=(EditText) findViewById(R.id.etforgetPass);
    btnSubmit.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            String email=etForget.getText().toString();
            String mobile="";
            String pass="";
            try{
            DBAdapter db=new DBAdapter(getApplicationContext());
            db.open();
            Cursor c= db.getUserMobileByEmail(email);
            mobile=c.getString(0);
             pass=c.getString(1);
            System.out.print("mob="+mobile+"pass"+pass);

            db.close();
            }catch (Exception e) {
                // TODO: handle exception
                e.printStackTrace();
            }
            if(!mobile.equalsIgnoreCase("")){
                try{
                    SmsManager smsManager = SmsManager.getDefault();

                        smsManager.sendTextMessage(mobile, null,"Your Password is:
"+pass ,
                            null, null);
                    Toast.makeText(getApplicationContext(),
                            "Password has been sent on your mobile",
Toast.LENGTH_SHORT)
                            .show();
                }catch (Exception e) {
                    // TODO: handle exception
                    Toast.makeText(getApplicationContext(),
                            "An erro occured", Toast.LENGTH_SHORT).show();
                }
            }else Toast.makeText(getApplicationContext(),
                "Emailid not valid.", Toast.LENGTH_SHORT).show();


        }
    });

}
}
```

List of all active users

```java
package com.example.emp_tracking;

import java.util.ArrayList;
import android.app.Activity;
```

```java
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import publisher.writer.com.writerpublisher.R;


public class ListAllActiveUsers extends Activity{


String username;
    ArrayList<String>lstUserid=new ArrayList<String>();
    ArrayList<String>lstDetails=new ArrayList<String>();
    ListView list;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.list_offers);
        username= getIntent().getStringExtra("username");
        list=(ListView) findViewById(R.id.list);
        DBAdapter db=new DBAdapter(getBaseContext());
        db.open();
        Cursor c=db.getAllRegisteredUser("user");
        while(c.moveToNext()){
            lstUserid.add(c.getString(c.getColumnIndex("username")));

lstDetails.add(c.getString(c.getColumnIndex("firstname"))+c.getString(c.getColu
mnIndex("lastname")));
        }
        c.close();
        db.close();
        Adapter1 adapter=new Adapter1(getBaseContext(), R.layout.custum_list,
lstDetails);
        list.setAdapter(adapter);
        list.setOnItemClickListener(new OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> parent, View view,
                    int position, long id) {
                // TODO Auto-generated method stub
                Intent i=new Intent(getBaseContext(),UserProfile.class);
                i.putExtra("username",lstUserid.get(position));
                startActivity(i);
            }
        });

    }


}
```

Login activity

```java
package com.example.emp_tracking;


import android.animation.Animator;
import android.animation.AnimatorListenerAdapter;
import android.annotation.TargetApi;
```

```java
import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.AsyncTask;
import android.os.Build;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.inputmethod.EditorInfo;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import com.example.emp_tracking.admin.AdminPanel;
import com.example.emp_tracking.user.UserPanel;
import publisher.writer.com.writerpublisher.R;
/**
 * Activity which displays a login screen to the user, offering registration as
 * well.
 */
public class LoginActivity extends Activity {
    /**
     * The default email to populate the email field with.
     */
    public static final String EXTRA_EMAIL =
"com.example.android.authenticatordemo.extra.EMAIL";

    /**
     * Keep track of the login task to ensure we can cancel it if requested.
     */
    private UserLoginTask mAuthTask = null;

    // Values for email and password at the time of the login attempt.
    private String mEmail;
    private String mPassword;

    // UI references.
    private EditText mEmailView;
    private EditText mPasswordView;
    private View mLoginFormView;
    private View mLoginStatusView;
    private TextView mLoginStatusMessageView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_login);

        // Set up the login form.
        mEmail = getIntent().getStringExtra(EXTRA_EMAIL);
        mEmailView = (EditText) findViewById(R.id.email);
        mEmailView.setText(mEmail);

        mPasswordView = (EditText) findViewById(R.id.password);
        mPasswordView
                .setOnEditorActionListener(new TextView.OnEditorActionListener() {
                    @Override
                    public boolean onEditorAction(TextView textView, int id,
                            KeyEvent keyEvent) {
```

```java
            if (id == R.id.login || id == EditorInfo.IME_NULL) {
                attemptLogin();
                return true;
            }
            return false;
        }
    });

    mLoginFormView = findViewById(R.id.login_form);
    mLoginStatusView = findViewById(R.id.login_status);
    mLoginStatusMessageView = (TextView)
findViewById(R.id.login_status_message);

    findViewById(R.id.sign_in_button).setOnClickListener(
        new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                attemptLogin();

            }
        });
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    getMenuInflater().inflate(R.menu.login, menu);
    return true;
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    // TODO Auto-generated method stub
    switch (item.getItemId()) {
    case R.id.action_forgot_password:
        Intent i = new Intent(getBaseContext(), ForgetPassword.class);
        i.putExtra("username", username);
        startActivity(i);
        finish();
        break;

    default:
        break;
    }
    return super.onOptionsItemSelected(item);

}
/**
 * Attempts to sign in or register the account specified by the login form.
 * If there are form errors (invalid email, missing fields, etc.), the
 * errors are presented and no actual login attempt is made.
 */
public void attemptLogin() {
    if (mAuthTask != null) {
        return;
    }

    // Reset errors.
    mEmailView.setError(null);
    mPasswordView.setError(null);

    // Store values at the time of the login attempt.
    mEmail = mEmailView.getText().toString();
    mPassword = mPasswordView.getText().toString();
```

38

```java
        boolean cancel = false;
        View focusView = null;

        // Check for a valid password.
        if (TextUtils.isEmpty(mPassword)) {
            mPasswordView.setError(getString(R.string.error_field_required));
            focusView = mPasswordView;
            cancel = true;
        } else if (mPassword.length() < 4) {
            mPasswordView.setError(getString(R.string.error_invalid_password));
            focusView = mPasswordView;
            cancel = true;
        }

        // Check for a valid email address.
        if (TextUtils.isEmpty(mEmail)) {
            mEmailView.setError(getString(R.string.error_field_required));
            focusView = mEmailView;
            cancel = true;
        } else if (!mEmail.contains("@")) {
            mEmailView.setError(getString(R.string.error_invalid_email));
            focusView = mEmailView;
            cancel = true;
        }

        if (cancel) {
            // There was an error; don't attempt login and focus the first
            // form field with an error.
            focusView.requestFocus();
        } else {
            // Show a progress spinner, and kick off a background task to
            // perform the user login attempt.
            mLoginStatusMessageView.setText(R.string.login_progress_signing_in);
            showProgress(true);
            mAuthTask = new UserLoginTask();
            mAuthTask.execute((Void) null);
        }


    }
}

/**
 * Shows the progress UI and hides the login form.
 */
@TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
private void showProgress(final boolean show) {
    // On Honeycomb MR2 we have the ViewPropertyAnimator APIs, which allow
    // for very easy animations. If available, use these APIs to fade-in
    // the progress spinner.
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB_MR2) {
        int shortAnimTime = getResources().getInteger(
                android.R.integer.config_shortAnimTime);

        mLoginStatusView.setVisibility(View.VISIBLE);
        mLoginStatusView.animate().setDuration(shortAnimTime)
                .alpha(show ? 1 : 0)
                .setListener(new AnimatorListenerAdapter() {
                    @Override
                    public void onAnimationEnd(Animator animation) {
                        mLoginStatusView.setVisibility(show ? View.VISIBLE
                                : View.GONE);
                    }
```

39

```java
                });

        mLoginFormView.setVisibility(View.VISIBLE);
        mLoginFormView.animate().setDuration(shortAnimTime)
                .alpha(show ? 0 : 1)
                .setListener(new AnimatorListenerAdapter() {
                    @Override
                    public void onAnimationEnd(Animator animation) {
                        mLoginFormView.setVisibility(show ? View.GONE
                                : View.VISIBLE);
                    }
                });
    } else {
        // The ViewPropertyAnimator APIs are not available, so simply show
        // and hide the relevant UI components.
        mLoginStatusView.setVisibility(show ? View.VISIBLE : View.GONE);
        mLoginFormView.setVisibility(show ? View.GONE : View.VISIBLE);
    }
}

/**
 * Represents an asynchronous login/registration task used to authenticate
 * the user.
 */
String username="";
String usertype="";
public class UserLoginTask extends AsyncTask<Void, Void, Boolean> {
    private String firstname;
    private String password;
    private String status="";

    @Override
    protected Boolean doInBackground(Void... params) {
        // TODO: attempt authentication against a network service.
        Boolean check = false;
        try {
            // Simulate network access.
            Thread.sleep(2000);

        } catch (InterruptedException e) {
            return check;
        }

        DBAdapter db = new DBAdapter(getBaseContext());
        db.open();
        Cursor cursor = db.getAllRegisteredUserWithPassword();
        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {


            if
(mEmailView.getText().toString().equalsIgnoreCase(cursor.getString(5)) &
mPasswordView.getText().toString()
                    .equalsIgnoreCase(cursor.getString(3))) {
                username=cursor.getString(1);
                firstname=cursor.getString(4);
                usertype=cursor.getString(2);
                password=cursor.getString(3);
                status=cursor.getString(6);
                check = true;


                break;
```

40

```java
            }
            cursor.moveToNext();
        }
        db.close();
        if(status.equalsIgnoreCase("False")){
            if(usertype.equalsIgnoreCase("instructor")){
                Toast.makeText(getBaseContext(), "You cannot activate
yourself.\nPlease contact admin", Toast.LENGTH_LONG).show();
                check=false;
            }

        }


        return check;
    }

    @Override
    protected void onPostExecute(final Boolean success) {
        mAuthTask = null;
        showProgress(false);

        if (success) {
            if(usertype.equalsIgnoreCase("admin")){
                Intent i = new Intent(getBaseContext(), AdminPanel.class);
                i.putExtra("username", username);
                i.putExtra("password", password);
                i.putExtra("firstname", firstname);
                startActivity(i);
            }else if(usertype.equalsIgnoreCase("user")){
                Intent i = new Intent(getBaseContext(), UserPanel.class);
                i.putExtra("username", username);
                i.putExtra("firstname", firstname);
                i.putExtra("password", password);
                System.out.println("here");
                startActivity(i);
            }


        } else{
          mPasswordView
                .setError(getString(R.string.error_incorrect_password));
          mPasswordView.requestFocus();
        }


    }
    @Override
    protected void onCancelled() {
        mAuthTask = null;
        showProgress(false);
    }
}
    @Override
    public void onBackPressed() {
        // TODO Auto-generated method stub

        finish();
    }
    public void registerPub(View v){
        Intent i=new Intent(getBaseContext(),Registeration.class);
        i.putExtra("title", "Register Self");
        i.putExtra("usertype", "admin");
```

```java
        startActivity(i);
        /*Intent i=new Intent(getBaseContext(),PlumberRegisteration.class);
             i.putExtra("usertype", "plumber");
             i.putExtra("title", "New Plumber");
             startActivity(i);

        * */
    }
    public void registerWri(View v){
        Intent i=new Intent(getBaseContext(),Registeration.class);
        i.putExtra("title", "Register Self");
        i.putExtra("usertype", "user");
        startActivity(i);
        /*Intent i=new Intent(getBaseContext(),PlumberRegisteration.class);
             i.putExtra("usertype", "plumber");
             i.putExtra("title", "New Plumber");
             startActivity(i);

        * */
    }
}
```

Post item'

```java
package com.example.emp_tracking;

import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;




import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Intent;
import android.graphics.Bitmap;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;
import publisher.writer.com.writerpublisher.R;
public class PostItem extends Activity{
    EditText
txtName,txtDesc,txtISDN,txtPrice,txtPage,txtTitle,txtAuthor,txtOfferedPrice;
    CheckBox cbxDonate;
    //TextView tvImg;
    String userid;
    ImageView imgDisplay;
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.create_item);
    userid=getIntent().getStringExtra("username");
    txtName=(EditText) findViewById(R.id.txtLocName);
    txtDesc=(EditText) findViewById(R.id.txtAddress);
    txtISDN=(EditText) findViewById(R.id.txtISDN);
    txtPrice=(EditText) findViewById(R.id.txtPrice);
    txtPage=(EditText) findViewById(R.id.txtPage);
```

```java
        txtTitle=(EditText) findViewById(R.id.txtTitle);
        txtAuthor=(EditText) findViewById(R.id.txtAuthor);
        txtOfferedPrice=(EditText) findViewById(R.id.txtOfferedPrice);
        cbxDonate=(CheckBox) findViewById(R.id.cbxDonate);
        imgDisplay=(ImageView) findViewById(R.id.imgDisplay);
        File f=new File(AppConstant.sdcard);

        if(!f.exists()){
           f.mkdir();
        }
    }
    private static final int CAMERA_REQUEST = 0;
    public void clickImage(View v){
         Intent cameraIntent = new
Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
          startActivityForResult(cameraIntent, CAMERA_REQUEST);
    }
    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == CAMERA_REQUEST && resultCode == RESULT_OK) {
                Bitmap photo = (Bitmap) data.getExtras().get("data");
                imgDisplay.setImageBitmap(photo);
                fileName=AppConstant.sdcard+getDate()+".png";
                FileOutputStream out = null;
                try {
                    out = new FileOutputStream(fileName);
                    photo.compress(Bitmap.CompressFormat.PNG, 100, out); // bmp is
your Bitmap instance
                    // PNG is a lossless format, the compression factor (100) is
ignored
                } catch (Exception e) {
                    e.printStackTrace();
                } finally {
                    try {
                        if (out != null) {
                            out.close();
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }

        }
    }
    /*
     * Cuts selected file name from real path to show in screen.
     */
    private String getStringNameFromRealPath(final String bucketName) {
        return bucketName.lastIndexOf('/') > 0 ? bucketName
                .substring(bucketName.lastIndexOf('/') + 1) : bucketName;
    }
    @SuppressLint("SimpleDateFormat")
    public String getDate() {
        DateFormat dateFormat = new SimpleDateFormat("yyyy_MM_dd_hh_mm_ss");
        // DateFormat dateFormat1 = new SimpleDateFormat("HH:mm:ss");

        Calendar cal = Calendar.getInstance();

        String reg_date = dateFormat.format(cal.getTime());// "11/03/14 12:33:43";
        // String reg_time = dateFormat1.format(cal.getTime());
```

```java
    return reg_date;

}
String fileName="hh";

public void saveLocation(View v){
    String lname=txtName.getText().toString();
    String ldesc=txtDesc.getText().toString();
    String isdn=txtISDN.getText().toString();
    String price=txtPrice.getText().toString();
    String page=txtPage.getText().toString();
    String title=txtTitle.getText().toString();
    String author=txtAuthor.getText().toString();
    String offeredPrice=txtOfferedPrice.getText().toString();
    File f=new File(fileName);
    if(f.exists()){

if(lname.length()>0&&ldesc.length()>0&&page.length()>0&&title.length()>0&&price
.length()>0){
        if(isdn.length()>0){
            if(price.length()>0){
            String donate="false";
            if(    cbxDonate.isChecked()){
                donate="true";
            }
                DBAdapter db=new DBAdapter(getBaseContext());
                db.open();

                long l=db.saveItems(lname, ldesc, fileName,
"true",userid,isdn,donate,price,page,title,author,offeredPrice);
                db.close();
                if(l>0){
                    Toast.makeText(getBaseContext(), "Book saved successfully!!",
Toast.LENGTH_SHORT).show();
                }else{
                    Toast.makeText(getBaseContext(), "Failed to save item!!",
Toast.LENGTH_SHORT).show();
                }
            }else {
                Toast.makeText(getBaseContext(), "Please enter book price",
Toast.LENGTH_SHORT).show();
            }
        }else{
            Toast.makeText(getBaseContext(), "Please enter book isdn no.",
Toast.LENGTH_SHORT).show();
        }

    }else{
        Toast.makeText(getBaseContext(), "Please check all the fields",
Toast.LENGTH_SHORT).show();
    }
    }else{
        Toast.makeText(getBaseContext(), "Please select an Image",
Toast.LENGTH_SHORT).show();
    }



}
}
```

Registration

```java
package com.example.emp_tracking;
```

```java
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import publisher.writer.com.writerpublisher.R;
public class Registeration extends Activity {

    String strFirstname = "";
    String strLastname = "";
    String strusername = "";
    String strpassword = "";
    String strFathername = "";
    String strMobile = "";
    String strEmailid = "";
    String strAddress = "";
    String strUsertype = "";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.registeration);
        TextView tvtitle = (TextView) findViewById(R.id.tvUser);
        Intent i = getIntent();
        String title = i.getStringExtra("title");
        tvtitle.setText(title);
        strUsertype = i.getStringExtra("usertype");
        System.out.println("usertype=" + strUsertype);
        final EditText etFirstname = (EditText) findViewById(R.id.etFirstName);
        final EditText etLastname = (EditText) findViewById(R.id.etLastname);
        final EditText etusername = (EditText) findViewById(R.id.etusername1);
        final EditText etpassword = (EditText) findViewById(R.id.etpassword);
        final EditText etFathername = (EditText) findViewById(R.id.etfathername);
        final EditText etMobile = (EditText) findViewById(R.id.etmobile);
        final EditText etEmailid = (EditText) findViewById(R.id.etemailid);
        final EditText etAddress = (EditText) findViewById(R.id.etaddress);
        Button btnSubmit = (Button) findViewById(R.id.btnSubmit);

        btnSubmit.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                try {
                    strFirstname = etFirstname.getText().toString();
                    strLastname = etLastname.getText().toString();

                    strusername = etusername.getText().toString();
                    strpassword = etpassword.getText().toString();
                    strFathername = etFathername.getText().toString();
```

```java
                strMobile = etMobile.getText().toString();
                strEmailid = etEmailid.getText().toString();
                strAddress = etAddress.getText().toString();
            } catch (Exception e) {
                e.printStackTrace();
            }

            System.out.println("name=" + strFirstname + ":" + strLastname
                + ":" + strusername + ":" + strUsertype);
            Context context = Registeration.this;
            String title = "Warning!!";
            String message = "Save Details";
            String button1String = "Save";
            String button2String = "Cancel";
            AlertDialog.Builder ad = new AlertDialog.Builder(context);
            ad.setTitle(title);
            ad.setMessage(message);
            ad.setPositiveButton(button1String, new OnClickListener() {

                @Override
                public void onClick(DialogInterface arg0, int arg1) {
                    // TODO Auto-generated method stub
                    DateFormat dateFormat = new SimpleDateFormat(
                        "dd/MM/yyyy");
                    DateFormat dateFormat1 = new SimpleDateFormat(
                        "HH:mm:ss");

                    Calendar cal = Calendar.getInstance();

                    String reg_date = dateFormat.format(cal.getTime());//
"11/03/14 12:33:43";
                    String reg_time = dateFormat1.format(cal.getTime());
                    if (verifyLastName(strFirstname)) {
                        if(strEmailid.contains("@")&&strEmailid.contains(".")){
                            DBAdapter db1 = new DBAdapter(getBaseContext());
                            db1.open();

                            long check = db1.registerUser(strusername,
                                strUsertype, strpassword, strFirstname,
                                strLastname, strFathername, strMobile,
                                strEmailid, strAddress, reg_date, reg_time,
                                "True");
                            db1.close();
                            if (check == -1) {
                                Toast.makeText(getBaseContext(),
                                        "An error occured!!",
                                        Toast.LENGTH_SHORT).show();
                            } else if (check > 0) {

                                Toast.makeText(getBaseContext(), "Saved!!",
                                        Toast.LENGTH_SHORT).show();
                                finish();
                            }
                        }else{
                            Toast.makeText(getBaseContext(), "Please enter a valid
emailid",

                                    Toast.LENGTH_SHORT).show();
                        }


                    }else{
                        Toast.makeText(getBaseContext(), "Name should be
alphabetical only",
```

```java
                    Toast.LENGTH_SHORT).show();

                }

            }
        });
        ad.setNegativeButton(button2String, new OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int which) {
                // TODO Auto-generated method stub
                Toast.makeText(getBaseContext(), "Canceled!",
                        Toast.LENGTH_SHORT).show();
                finish();
            }
        });

        ad.show();

        }
    });

}

private boolean verifyLastName(String lname) {
    lname = lname.trim();

    if (lname == null || lname.equals(""))
        return false;

    if (!lname.matches("[a-zA-Z]*"))
        return false;

    return true;
}
}
```

Splash screen

```java
package com.example.emp_tracking;

import publisher.writer.com.writerpublisher.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.animation.Animation;
import android.view.animation.AnimationUtils;
import android.widget.TextView;
import publisher.writer.com.writerpublisher.R;
public class SplashActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.splash);
        TextView imageView = (TextView) findViewById(R.id.tvtimer);
        Animation animation = AnimationUtils.loadAnimation(
                getApplicationContext(), R.anim.abc);
        imageView.startAnimation(animation);

        Thread thread = new Thread() {
            public void run() {
                try {
                    sleep(2000);
                } catch (InterruptedException e) {
```

```java
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            SplashActivity.this.finish();

            /*Intent i = new Intent(SplashActivity.this, MainActivity.class);
            startActivity(i);*/
            Intent i = new Intent(SplashActivity.this, LoginActivity.class);

            startActivity(i);

        }

    };

    thread.start();
    }
}
```

User Profile

```java
package com.example.emp_tracking;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;
import publisher.writer.com.writerpublisher.R;

public class UserProfile extends Activity {
    TextView tvusername;
    String username;
@Override
protected void onCreate(Bundle savedInstanceState) {
    // TODO Auto-generated method stub
    super.onCreate(savedInstanceState);
    setContentView(R.layout.profile);
    tvusername=(TextView) findViewById(R.id.tvusername);
    Intent i=getIntent();
    username=i.getStringExtra("username");

    DBAdapter db=new DBAdapter(this);
    db.open();
    Cursor c=db.getUserByUsername(username);
    if(c.moveToNext()){
        username=c.getString(c.getColumnIndex("username"));
        final String userdetails = "\nUserid:
"+c.getString(0)+"\nUsername:"+c.getString(1)+"\nUser
Type:"+c.getString(2)+"\nName:"+c.getString(5)
        +" "+c.getString(6)+"\nFather's Name: "+c.getString(4)+"\nMobile:
"+c.getString(7)+"\nEmail id: "+c.getString(8)+"\nAddress:
"+c.getString(9)+"\nRegisration Date: "+c.getString(10);
        tvusername.setText(userdetails);
    }



    db.close();


}
```

```
}
```

View item

```java
package com.example.emp_tracking;

import java.io.File;
import java.util.ArrayList;



import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;



import publisher.writer.com.writerpublisher.R;

public class ViewItems extends Activity {

    ArrayList<String> lstItems = new ArrayList<String>();
    ArrayList<String> lstUserDetails = new ArrayList<String>();
    ArrayList<String> lstMobile = new ArrayList<String>();
    ArrayList<String> lstItemid = new ArrayList<String>();
    ArrayList<String> lstImg = new ArrayList<String>();
    ArrayAdapter<String> adapter;
    ImageView imgRoute;
    String phone_no;
    Button btnCall;
    ListView listItems;
    TextView tvDetails;
    EditText txtSearch;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // TODO Auto-generated method stub
        super.onCreate(savedInstanceState);
        setContentView(R.layout.view_route);
        listItems = (ListView) findViewById(R.id.listitems);
        tvDetails = (TextView) findViewById(R.id.tvDetails);
        imgRoute = (ImageView) findViewById(R.id.imgROute);
        txtSearch = (EditText) findViewById(R.id.txtSearch);


tvDetails.setText("");
    }

    public void getDetails(View v) {
```

```java
        String search = txtSearch.getText().toString();

        DBAdapter db = new DBAdapter(getBaseContext());
        lstImg.clear();
        lstItemid.clear();
        lstItems.clear();
        lstMobile.clear();
        lstUserDetails.clear();

        tvDetails.setText("");

    try{
        db.open();
        Cursor c = db.getItemsByKey(search);


        while (c.moveToNext()) {


            lstImg.add(c.getString(c.getColumnIndex("item_img")));
            lstItemid.add(c.getString(c.getColumnIndex("_id")));
            Cursor c1 = db.getUserByUsername((c.getString(c
                    .getColumnIndex("userid"))));
            if (c1.moveToNext()) {
                lstItems.add("ISDN NO.: "+ c.getString(c.getColumnIndex("item_ISDN"))+
                        "\nBook Name: "+ c.getString(c.getColumnIndex("item_name"))
                        +"\nBook Price: "+ c.getString(c.getColumnIndex("item_price"))
                        +"\nPrice: "+ c.getString(c.getColumnIndex("page"))
                        +"\nBook Title: "+ c.getString(c.getColumnIndex("title"))
                        +"\nBook Author: "+ c1.getString(c1.getColumnIndex("firstname"))
                        //+"\nOffered Price: "+
c.getString(c.getColumnIndex("offeredPrice"))
                        + "\nItem Description: "+
c.getString(c.getColumnIndex("item_desc")));
                lstUserDetails.add("Userid: "
                        + c.getString(c.getColumnIndex("userid")) + "\nName: "
                        + c1.getString(c1.getColumnIndex("firstname")));
                lstMobile.add(c1.getString(c1.getColumnIndex("mobile")));
            }

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

        if (lstImg.size() > 0) {
            Adapter1 adapter = new Adapter1(ViewItems.this,
                    R.layout.custum_list, lstItems);
            listItems.setAdapter(adapter);
            listItems.setOnItemClickListener(new OnItemClickListener() {

                @Override
                public void onItemClick(AdapterView<?> parent, View view,
                        int position, long id) {
                    // TODO Auto-generated method stub
    try
    {
        final File targetLocation = new File(lstImg.get(position));
        Log.i(AppConstant.tag, targetLocation.getPath());
        tvDetails.setText(lstUserDetails.get(position));
        phone_no = lstMobile.get(position);
        imgRoute.setImageBitmap(BitmapFactory
                .decodeFile(targetLocation.getPath()));
```

```java
        imgRoute.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {
                // TODO Auto-generated method stub
                Intent intent = new Intent();
                intent.setAction(Intent.ACTION_VIEW);
                intent.setDataAndType(
                        Uri.parse("file://"
                                + targetLocation.getPath()),
                        "image/*");
                startActivity(intent);
            }
        });
        if (targetLocation.exists()) {
            Toast.makeText(getBaseContext(),
                    "Click on Image to zoom!!", Toast.LENGTH_SHORT)
                    .show();
        } else {
            Toast.makeText(getBaseContext(), "No Image Found!!",
                    Toast.LENGTH_SHORT).show();
        }
}catch (Exception e){e.printStackTrace();}


            }
        });
    }else{
        Toast.makeText(getBaseContext(), "No books Found",
Toast.LENGTH_SHORT).show();
    }


    }

}
```

Annexure 1

Title of The Project Report
(Font size = 18)

*Submitted in partial fulfillment of the requirements
for the award of the degree of*

Bachelor of Computer Applications

To

Guru Gobind Singh Indraprastha University, Delhi

Guide: Submitted by:
(Guide Name)   1. (Student name Roll No)

IITM

Nurturing Excellence

Institute of Innovation In Technology & Management, New Delhi – 110058

Batch (2015-2018)

Annexure II

Certificate

We, 1. (Name & Roll No) & 2. (Name & Roll No) certify that the Summer Training Project Report (BCA-355) entitled "_____" is done by us and it is an authentic work carried out by us at _____ (Name of the organisation or of the Institute). The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

1. Signature of the Student   2. Signature of the Student

Date:

Certified   that   the   Project   Report   (BCA-355) entitled"_____"

done by the above students is completed under my guidance.

Signature of the Guide
Date:
Name of the Guide:
ASSISTANT   PROFESSOR
(IT)

Countersigned

Director

FORMAT FOR TABLE OF CONTENTS

TABLE OF CONTENTS

S No	Topic	Page No
1	Certificate	-
2	Acknowledgements	-
3	List of Tables/Figures/Symbols	-
4	Chapter-1: Introduction	
5	Chapter-2: System Analysis	
6	Chapter-3: System Design	
8	Appendices	

FORMAT FOR LIST OF TABLES/FIGURES/ SYMBOLS

LIST OF TABLES

Table No	Title	Page No
1	File Design for Employee Record	
2	File Design for Personal Details	

LIST OF FIGURES

Figure No	Title	Page No
1	Data Flow Diagram	
2	Input Screen for Data Entry	

LIST OF SYMBOLS

S No	Symbol	Nomenclature & Meaning
1	Σ	Sigma (Summation)
2	**kbps**	Kilo bits per second

Appendices

7.   The appendices are to be attached at the end of the report and to be numbered as <u>Appendix-A</u>, <u>Appendix-B</u> etc right justified at the top of the page. Below the word Appendix write in parenthesis "Refer Para No__". The para number is to be the number in the body of text where the reference of appendix is given. An appendix may have annexure (s). If there are annexure, these are to be attached immediately after the said appendix. The annexures are to be numbered as <u>Annexure-I</u>, <u>Annexure-II</u> etc.

Annexure 1

Title of the project

)

*Submitted in partial fulfillment of the requirements
for the award of the degree of*

Bachelor of Computer Applications

To

Guru Gobind Singh Indraprastha University, Delhi

Guide: Submitted by:
                                                         GURPREET SINGH GALSIAN
                                                         41124402015

Institute of Innovation In Technology & Management,
New Delhi - 110058
Batch (2015-2018)

Annexure II

Certificate

I GURPREET SINGH GALSIAN & 41124402015 certify that the Summer Training Project Report (BCA-355) entitled "BLOOD BANK " is done by us and it is an authentic work carried out by us at DUCAT matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

Signature of the Student
Date:

Certified that the Project Report (BCA-355) entitled"BLOOD BANK "

done by the above students is completed under my guidance.

Countersigned   Signature of the Guide
Director   Date:
                                                              Name of the Guide:Mr.PROMOD SONI
                                                              Designation:ASSISTANT PROFESSOR

FORMAT FOR TABLE OF CONTENTS

TABLE OF CONTENTS

S No	Topic	Page No
1	Certificate	-
2	Acknowledgements	-
3	List of Tables/Figures/Symbols	-
4	Chapter-1: Introduction	
5	Chapter-2: System Analysis	
6	Chapter-3: System Design	
8	Appendices	

FORMAT FOR LIST OF TABLES/FIGURES/ SYMBOLS

LIST OF TABLES

Table No	Title	Page No
1	File Design for Employee Record	
2	File Design for Personal Details	

LIST OF FIGURES

Figure No	Title	Page No
1	Data Flow Diagram	
2	Input Screen for Data Entry	

LIST OF SYMBOLS

S No	Symbol	Nomenclature & Meaning
1	Σ	Sigma (Summation)
2	**kbps**	Kilo bits per second