



**Department of Mathematics & Computing**  
**Banasthali Vidyapith, Banasthali - 304022**  
**Session: 2017-18**

## **A PROJECT REPORT**

**ON**

## **RectiFYX APP**



**SUBMITTED FOR THE PARTIAL FULFILLMENT FOR THE AWARD OF THE DEGREE**  
**OF**  
**BTECH CSE (SEM VI)**

### **SUBMITTED BY**

S. RAMYA	9161
SHIVANGI SHARMA	9182
SHIVI MALHOTRA	9189
SONALI ASWAL	9216

**Under the supervision of**

### **MENTOR:**

Mrs. MANISHA AGARWAL

### **PROJECT GUIDE:**

Mr. VIVEK PUROHIT

Mrs. NEELAM SHARMA



## Certificate

Certified that **S. Ramya , Shivangi Sharma, Shivi Malhotra, Sonali Aswal** has carried out the project work titled “**RectiFYX**” from **26 December, 2017** to **13 April, 2018** for the award of the **B.Tech (CS) 3<sup>rd</sup> year** from **Banasthali Vidyapith** under my supervision. The thesis embodies result of original work and studies carried out by Student herself and the contents of the thesis do not form the basis for the award of any other degree to the candidate or to anybody else.

Dr. C.K. Jha

Head of Department

AIM & ACT

Banasthali Vidyapith

# TABLE OF CONTENTS

<b>S. NO.</b>	<b>CONTENTS</b>	<b>PAGE NO.</b>
<b>1</b>	<b>OBJECTIVE</b>	<b>4</b>
<b>2</b>	<b>REQUIREMENT ANALYSIS (SRS)</b>	<b>5</b>
	2.1 REQUIREMENT SPECIFICATION	5
	2.2 H/W AND S/W REQUIREMENTS	11
	2.3 FEASIBILITY STUDY	12
	2.4 PRODUCT FUNCTIONS	15
	2.5 USE-CASE DIAGRAMS	17
<b>3</b>	<b>SYSTEM DESIGN (SDS)</b>	<b>18</b>
	3.1 HIGH- LEVEL DESIGN	18
	3.2 CLASS DIAGRAM	19
	3.3 DATABASE DESIGN	20
	3.4 ACTIVITY DIAGRAMS	21
	3.5 SEQUENCE DIAGRAM	22
<b>4</b>	<b>CODING</b>	<b>24</b>
<b>5</b>	<b>TESTING</b>	<b>31</b>
<b>6</b>	<b>USER INTERFACES</b>	<b>34</b>
<b>7</b>	<b>REFERENCES</b>	<b>41</b>

# 1. OBJECTIVE

The application rectiFYX is created with the motivation of providing a user friendly interface for splitting of bills and settling of dues. The rectiFYX system is composed of two main components: a client-side application which will run on Android handsets, and a server-side application which will support and interact with various client-side features. The system is designed to facilitate the process of settling shared expenses. Potential scenarios include splitting a bill at dinner, sharing travel expenses, etc.

Focus of this android application revolves around the following:

- This android application provides the feature of rectifying the total amount spent in an event by individuals.
- Now, the total amount is divided equally among all the individuals of the group with a feature of sending notifications to the contacts to communicate regarding the splitting of bills.

With rectiFYX being ported solely for the Android platform, this software application has the advantage of being portable and convenient to use whenever and wherever. Overall, the app balances both the ease of use and the ease of learning. The layout and UI of the app will be simple enough that users will take no time to learn its features and navigate through it with little difficulty. Animations have been used in the application to provide it an attractive look and feel.

The graphical user interface of rectiFYX is designed with usability as the first priority. The app is presented and organized in a manner that is both visually appealing and easy for the user to navigate. There will be notifications to inform users regarding their pending dues.

## 2. REQUIREMENT ANALYSIS (SRS)

### 2.1 REQUIREMENT SPECIFICATION

#### 2.1.1 FUNCTIONAL REQUIREMENTS

#### **USE CASES**

Use case 1- Login

Use Case No.	1
Use Case Name	Login
Actors	Users
Description	Login
Necessary Condition	The users must be registered.
Normal Course Event	<ul style="list-style-type: none"> <li>• Actors enter their registered email id</li> <li>• Actors enter their password</li> <li>• Actors click login button</li> <li>• System connects to database</li> <li>• Homepage displayed</li> </ul>
Alternate Course Event	<ul style="list-style-type: none"> <li>• Actors can enter their email id and password incorrectly</li> <li>• Error message appears</li> <li>• Continue with step 1 in the normal course events</li> <li>• An error may occur during the database operation</li> <li>• System show error message</li> </ul>

## Use case 2- Sign Up

Use Case No.	2
Use Case Name	Sign Up
Actors	Users
Description	User`s Signup
Necessary Condition	New user should give valid information while registration.
Normal Course Event	<ul style="list-style-type: none"> <li>• Actors enter their email id</li> <li>• Actors enter their password</li> <li>• Actor tap Signup button</li> <li>• System connects to database</li> <li>• A message appears which shows that they are registered.</li> </ul>
Alternate Course Event	<ul style="list-style-type: none"> <li>• Email id should be unique</li> <li>• Error message will appear</li> <li>• Continue with step1 in normal course events</li> <li>• An error may occur during database operation</li> <li>• System show error messages</li> </ul>

### Use case 3- Logout

Use Case No.	3
Use Case Name	Logout
Actors	Users
Descriptions	It Logs out the actor from his account.
Necessary conditions	The user must be logged in.
Normal Course Events	<ul style="list-style-type: none"> <li>• The application users click to Logout button.</li> <li>• DB connection terminated.</li> <li>• The users logout successfully.</li> </ul>
Output	Application homepage will be displayed.

### Use case 4 - Forget Password

Use Case No	4
Use Case Name	Forget Password
Actors	Users
Descriptions	This module helps the staff members, students, etc., to remind their forgotten password.
Necessary conditions	The user must be a member of the system.
Input	Username.
Alternative Courses	<ul style="list-style-type: none"> <li>• Wrong email-address is entered.</li> <li>• Error message appears.</li> </ul>

### Use case 5 – Rectify

Use Case No	5
Use Case Name	Rectify
Actors	Users
Descriptions	This module helps the user to rectify the total amount fed by the user.
Necessary conditions	The user must be logged on to the system.
Input	Total amount.
Normal Course Events	<ul style="list-style-type: none"> <li>• The user must be logged on to the system.</li> <li>• The user taps on the rectify button.</li> <li>• The system receives user`s information from database.</li> </ul>
Output	Total amount to be divided is balanced.

### Use case 6-Notify

Use Case No	6
Use Case Name	Notify
Actors:	User
Descriptions	This module helps user to notify the other users regarding pending dues.
Necessary conditions	The users must be a member of the system.



### 2.2.2 NON-FUNCTIONAL REQUIREMENTS

#### ➤ **Availability**

The availability of this android application is up to the Internet connection and need of the client. User should have an account to enter the system.

#### ➤ **Security**

Security is one of the crucial things of this project. Only the valid user must be allowed to access the database. The software is secure as it is not allowing users to upload any document from their side.

#### ➤ **Reliability**

The app made is be reliable i.e. program should not crash and other reliability feature like proper exception handling, maintenance of integrity and consistencies of database is taken care of.

#### ➤ **Portability**

Updated support for Android based mobile phones.

#### ➤ **Maintainability**

The project is made in a simple and lucid style so that future maintenance of the project is easy. Different module for each different function is made to enhance the readability of the code.

### 2.2.3 OTHER REQUIREMENTS

#### ➤ **Performance Requirements**

Performance should not be an issue because all of our server queries involve small pieces of data. Changing screens will require very little computation and thus will occur very quickly. Server updates should only take a few seconds as long as the phone can maintain a steady signal. The cost-division algorithms used by in application will be highly efficient, taking only a fraction of a second to compute.

### ➤ **Software Quality Attributes**

The graphical user interface of rectiFYX is to be designed with usability as the first priority. The app will be presented and organized in a manner that is both visually appealing and easy for the user to navigate. There will be feedbacks and visual cues such as notifications to inform users of updates and pop-ups to provide users with instructions. To ensure reliability and correctness, there will be zero tolerance for errors in the algorithm that computes and splits expenses between the members.

Furthermore, the group leader also has the option to add members who do not own an Android phone and add transactions on their behalf. With rectiFYX being ported solely for the Android platform, this software application has the advantage of being portable and convenient to use whenever and wherever. Overall, the app balances both the ease of use and the ease of learning. The layout and UI of the app will be simple enough that users will take no time to learn its features and navigate through it with little difficulty.

## **2.2 HARDWARE AND SOFTWARE REQUIREMENTS**

### **2.2.1 Hardware Requirements**

#### **Server Side**

- 64 BIT Processor and above.
- RAM – 4 GB and above.
- Up to 20 GB of internal storage.
- Network Interface Card.

### **2.2.2 Software Requirements**

#### **Server Side**

- Microsoft WINDOWS 10
- Android Studio 2.3.3
- Firebase 10.0.1

### **2.2.3 Hardware and Software Requirements**

#### **Client Side**

- Smart Phone
- Android OS (2.3.3 or above)
- Internet Connection

## **2.3 FEASIBILITY STUDY**

### **DESCRIPTION**

The rectiFYX system is composed of two main components: a client-side application which will run on Android handsets, and a server-side application which will support and interact with various client-side features. The system is designed to facilitate the process of settling shared expenses.

Potential scenarios include splitting a bill at dinner, sharing travel expenses, etc. Hence, we can summarize the purpose of this android app as:

- The project is a mobile application and can be used by anyone.
- The users can include any number of people and the amount can be rectified easily.
- This app helps to balance the amount.
- It saves time for the user, who can access it from anywhere anytime, with just a click.

### **PURPOSE OF FEASIBILITY STUDY**

The purpose of this feasibility study is to determine the market and technical feasibility of creating a bill splitting application which splits the amount among any number of people or among two users and also send notifications for the same.

### **PROBLEM STATEMENT**

The organization of trips to various places, dining out, gathering and spending at functions or sharing workspace or living space with more than one person is quite frequent of an event these days. The problem arises where an interconnected pile of dues appear to be settled in different ways among different people. The application provides a user friendly and easy way to settle expenses among any number of users. The settling of expenses will be a click away.

### **FEASIBILITY RESEARCH QUESTIONS**

- What is the current market potential of the application?
- What is the future market potential of the application?
- What are the probable competitions to the application in the current scenario?

- What are the technical aspects and potentials of the application?

## MARKET FEASIBILITY

- The application defines its utility in the sphere of bill splitting and settling of dues among any number of users. The handling of due amount and settling of bills finds its use in several instances such as vacation trips, among different friend circles, a group of people living together or sharing same working space and several other cases.
- Current market potential:
  - Splitwise:
    - An application to simplify the entire process of splitting bills for those who constantly share expenses on a monthly basis.
    - A ‘recent activity’ tab lets you review each entry and changes made, in case you want to double check.
  - Settle up:
    - It lets you create a new group of people and add payment details – say for a dinner at a restaurant – and then key in who paid and who owes how much (by adjusting the ratios) to figure who owes what.
    - If your friends are on the app as well, they’ll get notifications and can instantly pay up, if they have a PayPal account, or can later update payments made.
- Future market potential:
  - But despite the glut of bill splitting apps out there, there is still a void that is only slowly getting filled.
  - At the Disrupt NY Hackathon, a developer showcased an app called Split that would allow users to click a picture of the final bill and flick various items on it to various friends, thus ending the infamous “who paid for what” awkward conversation.
  - Although Split is still a hack, and not yet a fully functioning app.
  - Another app that’s currently only in the US is Plates, created by the makers of Splitwise. Designed specifically to split restaurant checks, the app promises to help you split shared costs like appetizers and tax and tip, with up to 10 people.
- The competition to the app rectiFYX is assumed to be of the applications currently trending and well marketed. Once appropriately and properly marketed, the application will be at par with the current applications in the market and even better.
- The prospective users of the application will include college students sharing same living space or friend circle, colleagues sharing same work environment or a group of people sharing expenses like on a trip or a lunch outing etc.

**TECHNICAL FEASIBILITY**

- The exclusive points of the application include:
  - The option to send notifications regarding the splitting of bill to the users.
  - Send application invites to register.
  - Authenticated login using Google accounts.
- The working of application requires a smart phone with an appreciable data speed and android 2.3.3.

## **2.4 PRODUCT FUNCTIONS**

The following list offers a brief outline and description of the main features and functionalities of the rectiFYX system. The features are split into two major categories: core features and additional features. Core features are essential to the application's operation, whereas additional features simply add new functionalities. The latter features will only be implemented as time permits.

### **2.4.1 CORE FEATURES**

#### **1. Welcome Page**

- Allows the user to Sign up or Login with the rectiFYX server.
- Enables the user to customize his/her account settings and preferences.

#### **2. Rectify**

- Stores and monitors the bill amount, the number of individuals involved, etc.
- Efficiently distributes debt amongst the individuals responsible for the bill.

#### **3. Final Debt Resolution**

- Calculates the most efficient method of sorting out debts
- Notifies users of unresolved debts, credits, etc.

#### **4. Push Notifications**

- Remind users of unresolved debts.

### **2.4.2 USER CHARACTERISTICS**

- The user must be familiar to work with GUI components.
- The user should have knowledge of Basic English.

- The user should know how to use a smart phone (Android).

#### 2.4.3 GENERAL CONSTRAINTS

- Our project is GUI based application.
- Login password is used for the identification of the authorized user.
- Valid information should be filled during login.
- Unique ID must be created for all the customers.

#### 2.4.4 ASSUMPTIONS AND DEPENDENCIES

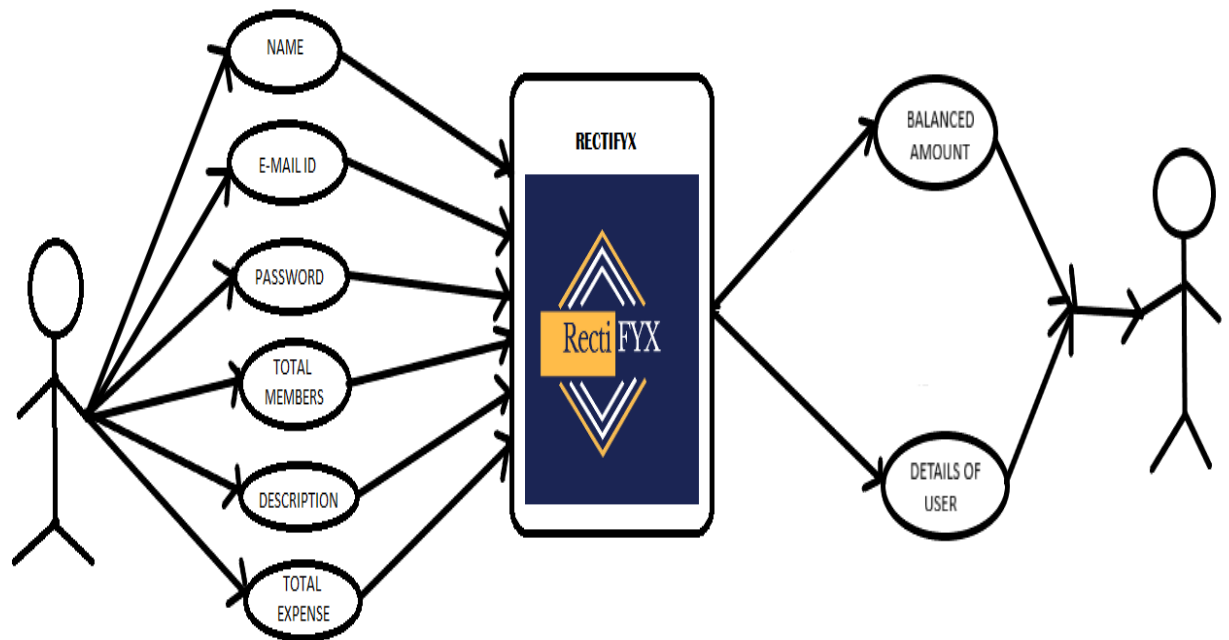
- Users can login with their email ID's.
- Valid information should be filled during login.
- Internet facility is available all the time.
- System date and time is correct.

#### 2.4.5 TECHNOLOGIES USED

- **Front End:** Android Studio
- **Back End:** Java, firebase
- **Design Tool:** Android Studio

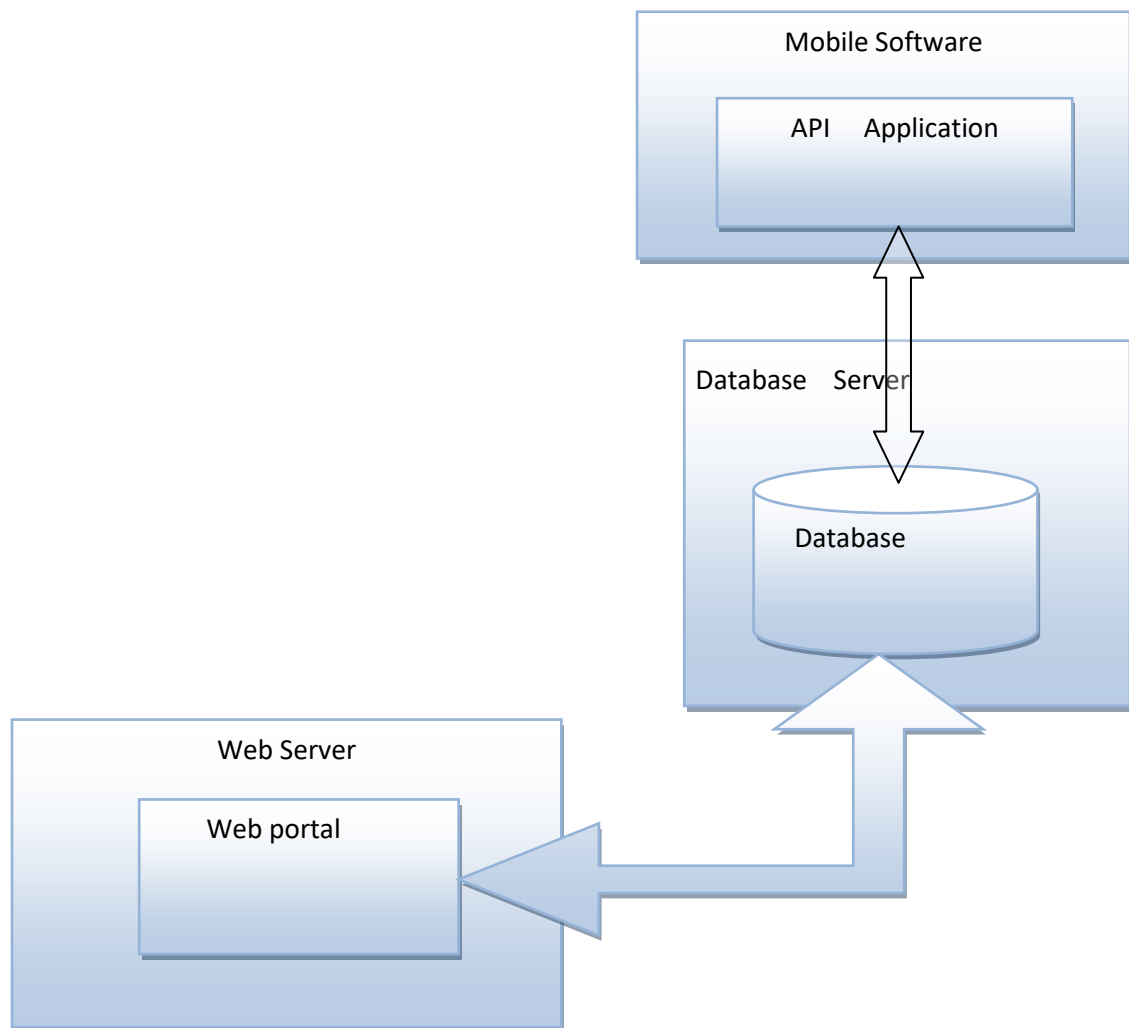


## 2.5 USE CASE DIAGRAM

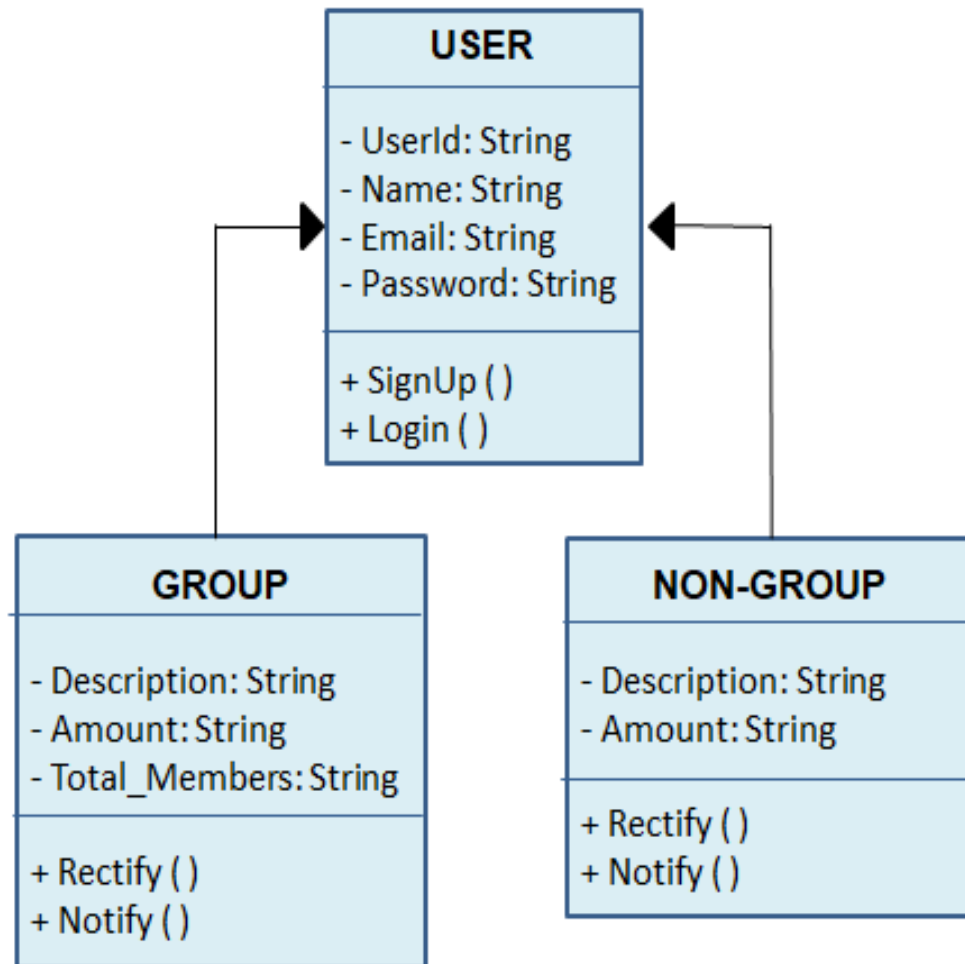


### 3. SYSTEM DESIGN (SDS)

#### 3.1 ARCHITECHTURE DESIGN



### 3.2 CLASS DIAGRAM



### 3.3 DATABASE DESIGN

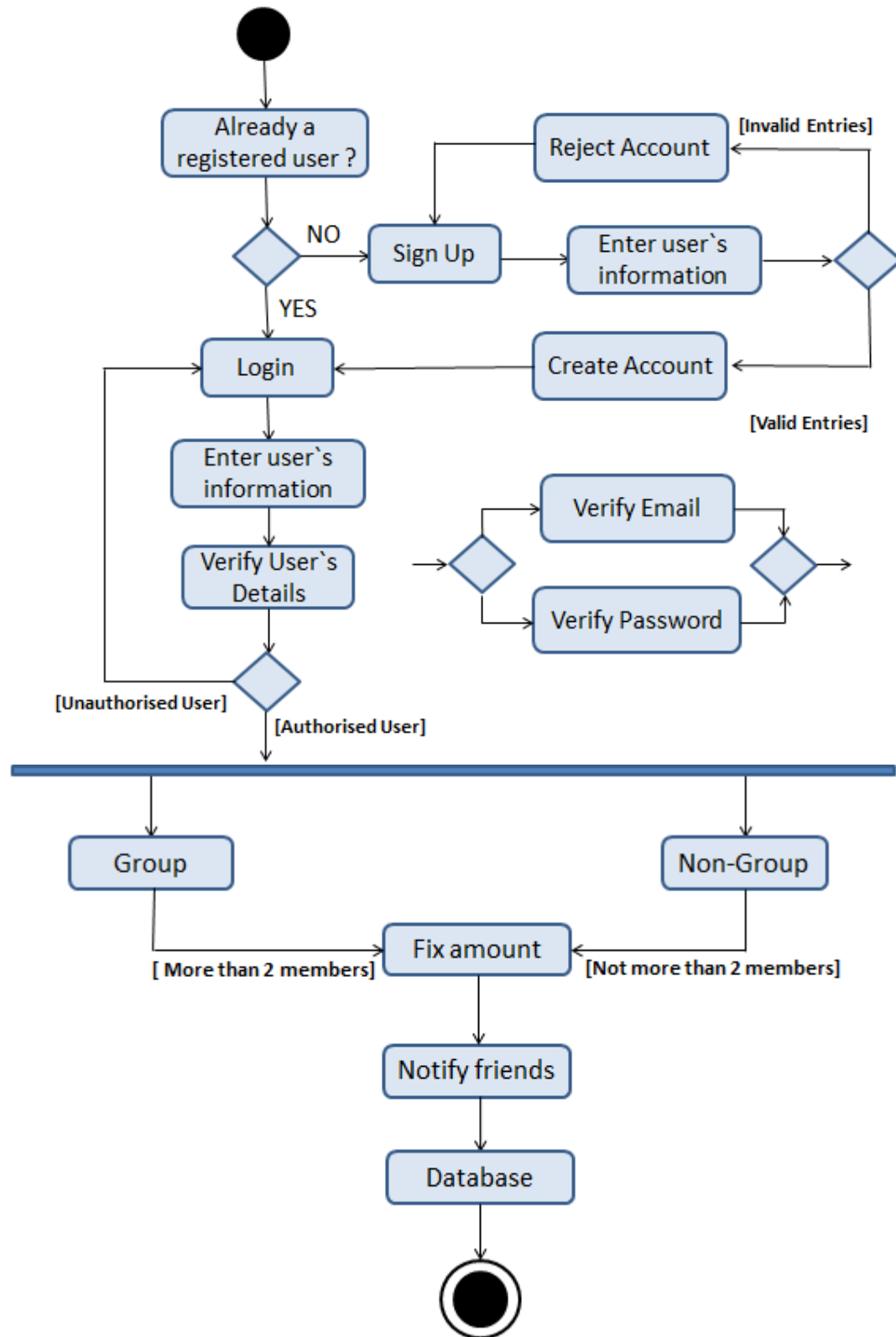
#### 3.3.1 – User Table

FIELD	TYPE	CONSTRAINTS	DESCRIPTION
UserId	AUTONUMBER	Unique	Auto generated User's Id
Name	TEXT	Not Null	Name of the User
Email	TEXT	Not Null, Primary Key	Email Id of the User
Password	TEXT	Not Null	Password of the User

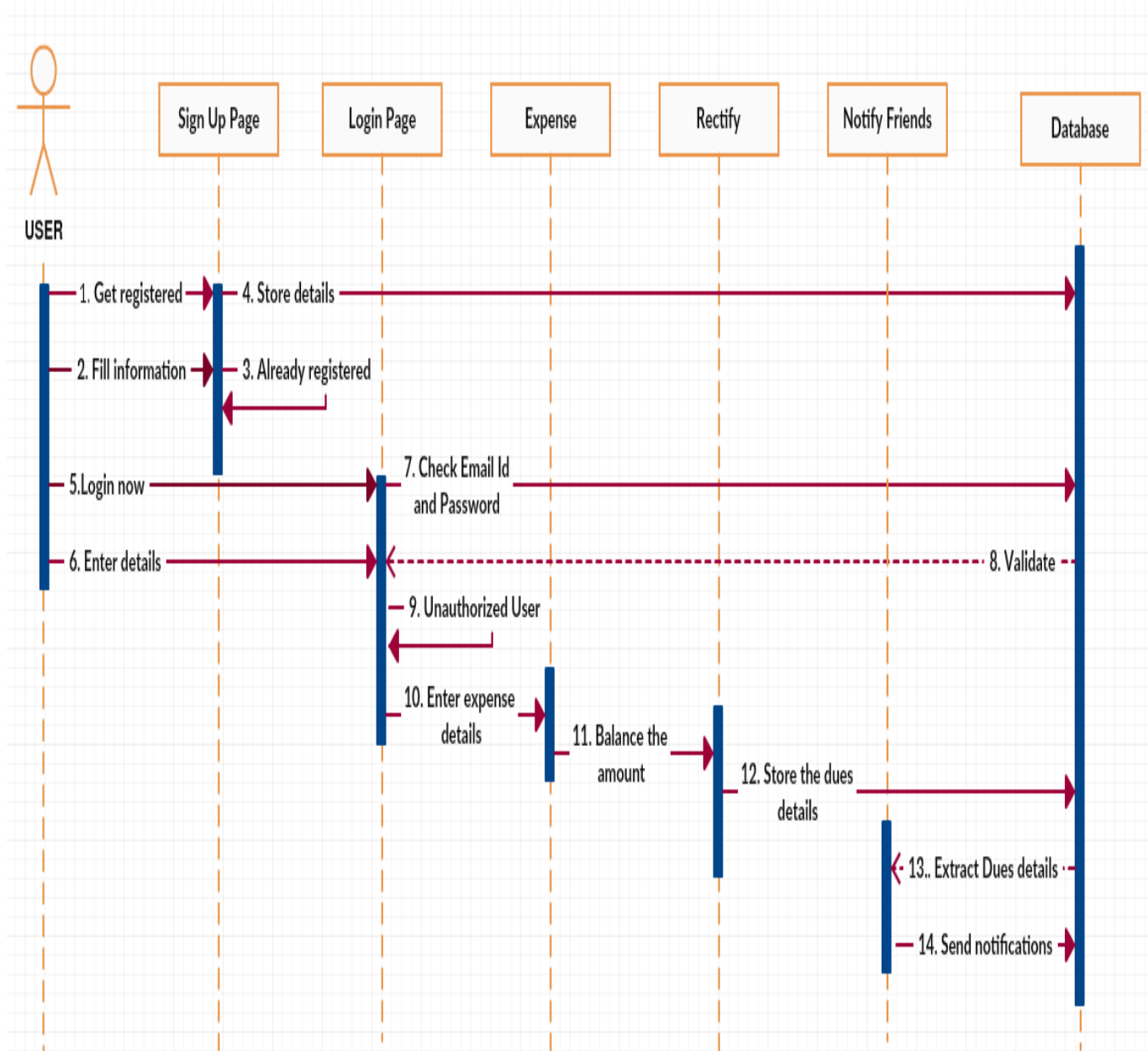
#### 3.3.2 – Expense Table

FIELD	TYPE	CONSTRAINTS	DESCRIPTION
UserId	AUTONUMBER	Unique, Foreign Key	Auto generated User's Id
Amount	REAL	Not Null	Total amount to be divided
Description	TEXT		Description of the event on which the amount is spend
No_of_Members	INTEGER	Not Null	Total number of users among which the amount is to be divided

### 3.4 ACTIVITY DIAGRAMS



### 3.5 SEQUENCE DIAGRAM



## 4. CODING

### 4.1 Code Of Rectify

```
package com.trial.rectifyapp;

/**
 * Created by admin on 11-04-2018.
 */

public class fix {

    String Description;
    int Amount,Members;

    public fix(){

    }

    public fix(String description, int amount, int members) {
        Description = description;
        Amount = amount;
        Members = members;
    }
    public String getDescription()
    {
        return Description;
    }
    public int getAmount()
    {
        return Amount;
    }
    public int getMembers()
    {
        return Members;
    }

    public void setDescription(String description) {
        Description = description;
    }

    public void setAmount(int amount) {
        Amount = amount;
    }

    public void setMembers(int members) {
        Members = members;
    }
}
```

```

    }
}

```

## 4.2 Code of Group and Non Group Expense

```

package com.trial.rectifyapp;

import android.content.DialogInterface;
import android.content.Intent;
import android.graphics.drawable.AnimationDrawable;
import android.net.Uri;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.TextUtils;
import android.view.View;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RelativeLayout;
import android.widget.Spinner;
import android.widget.Toast;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;

import static android.R.attr.button;

public class group1 extends AppCompatActivity {

    private EditText Amount, Description, Members;

    private Button Back, rectify, Invite, ok;
    DatabaseReference databaseFix;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_group1);
    }
}

```



```

        RelativeLayout mylayout;
        AnimationDrawable animationDrawable;

        mylayout = (RelativeLayout)
findViewById(R.id.mylayout1);
        animationDrawable = (AnimationDrawable)
mylayout.getBackground();
        animationDrawable.setEnterFadeDuration(500);
        animationDrawable.setExitFadeDuration(500);
        animationDrawable.start();

        databaseFix =
FirebaseDatabase.getInstance().getReference("fixusers");
        Amount = (EditText) findViewById(R.id.editText99);
        Description = (EditText) findViewById(R.id.editText);
        Members = (EditText) findViewById(R.id.editText96);

        //Button Fix= (Button) findViewById(R.id.button57);
        Back = (Button) findViewById(R.id.button2);
        rectify = (Button) findViewById(R.id.button5);
        Invite= (Button) findViewById(R.id.button10);

        Back.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(group1.this,
g_ng.class);
                startActivity(intent);
            }
        });

        rectify.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                int amount =
Integer.parseInt(Amount.getText().toString());
                int members =
Integer.parseInt(Members.getText().toString());
                String description =
Description.getText().toString().trim();
                int ppamt = (amount / members);
                int fixamt = (amount - ppamt);

                if (TextUtils.isEmpty(description)) {
                    Toast.makeText(getApplicationContext(),
"Please enter the description", Toast.LENGTH_SHORT).show();
                    return;
                }
            }
        });

```

```

else
{
String id = databaseFix.push().getKey();
fix Fix = new fix(description, amount,
members);

databaseFix.child(id).setValue(Fix);
Toast.makeText(getApplicationContext(),
"Amount added to the database", Toast.LENGTH_SHORT).show();
Toast.makeText(getApplicationContext(),
"Amount added="+ppamt, Toast.LENGTH_SHORT).show();
//Intent intent = new Intent(group1.this,
group_creation.class);
//startActivity(intent);
}

Toast.makeText(getApplicationContext(), "others
owe" + fixamt, Toast.LENGTH_LONG).show();

AlertDialog.Builder builder=new
AlertDialog.Builder(group1.this);
builder.setTitle("Details");
builder.setMessage("Amount per person:
"+ppamt+"\n Amount recieved should be: "+fixamt);
builder.setCancelable(false);
builder.setPositiveButton("Yes", new
DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog,
int which) {
Toast.makeText(getApplicationContext(),
"Amount Added.", Toast.LENGTH_LONG).show();
}
});
builder.setNegativeButton("No", new
DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog,
int which) {

dialog.cancel();
}
});
AlertDialog alertDialog=builder.create();
alertDialog.show();

}
});
Invite.setOnClickListener(new View.OnClickListener() {
@Override

```

```

        public void onClick(View v) {
            int amount =
Integer.parseInt(Amount.getText().toString());
            int members =
Integer.parseInt(Members.getText().toString());
            String description =
Description.getText().toString().trim();
            int ppamt = (amount / members);
            int fixamt = (amount - ppamt);

            Intent ss = new Intent(Intent.ACTION_VIEW);
            String msg = "Hi, Please install RectiFYX App
from PlayStore to know your owes/dues. You're succesfully added
to the group! You owe Rs";
            ss.setData(Uri.parse("sms://"));
            ss.putExtra("sms_body", msg+""+ppamt);
            ss.setType("vnd.android-dir/mms-sms");
            startActivity(ss);

        }
    });
}
}

```

### 4.3 Code to select an activity

```

package com.trial.rectifyapp;

import android.content.Intent;
import android.graphics.drawable.AnimationDrawable;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.text.format.DateFormat;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.ListView;
import android.widget.RelativeLayout;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.appinvite.AppInviteInvitation;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.FirebaseDatabase;

public class g_ng extends AppCompatActivity {

    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_g_ng);

        RelativeLayout mylayout;
        AnimationDrawable animationDrawable;

        mylayout = (RelativeLayout) findViewById(R.id.mylayout1);
        animationDrawable = (AnimationDrawable)
mylayout.setBackground();
        animationDrawable.setEnterFadeDuration(500);
        animationDrawable.setExitFadeDuration(500);
        animationDrawable.start();

        // Button NonGroup = (Button) findViewById(R.id.button5);
        mAuth=FirebaseAuth.getInstance();
        Button Groups = (Button) findViewById(R.id.button4);
        Button invite2 = (Button) findViewById(R.id.invite);
        Button logout= (Button) findViewById(R.id.button14);
        Button nongroups = (Button) findViewById(R.id.button13);
    }
}

```

```

        Groups.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(g_ng.this,
group1.class);
                startActivity(intent);
            }

        });

        invite2.setOnClickListener(new View.OnClickListener() {
            public static final String TAG = "g_ng" ;
            public static final int REQUEST_INVITE = 100;

            @Override
            public void onClick(View
v) {

                Intent intent = new
AppInviteInvitation.IntentBuilder("invitation_title")

                .setMessage("Hey, join RectiFYX to enjoy the ease of settling up
dues/owes.")

                .setDeepLink(Uri.parse("http://google.com"))

                .setCallToActionText("Invitation CTA")

                .build();

                startActivityForResult(intent, REQUEST_INVITE);

            }

            /* @Override
            protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
                super.onActivityResult(requestCode, resultCode,
data);

                Log.d(TAG, "onActivityResult: requestCode=" +
requestCode + ", resultCode=" + resultCode);

                if (requestCode == REQUEST_INVITE) {
                    if (resultCode == RESULT_OK) {
                        // Get the invitation IDs of all sent
messages

                        String[] ids =
AppInviteInvitation.getInvitationIds(resultCode, data);


```

```

        for (String id : ids) {
            Log.d(TAG, "onActivityResult: sent
invitation " + id);
        }
    } else {
        // Sending failed or it was canceled,
show failure message to the user
        // ...
    }
}
}*/

});

```

```

logout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        mAuth.signOut();
        finish();
        Intent intent = new Intent(g_ng.this,
welcome3.class);
        startActivity(intent);

    }

});

nongroups.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Toast.makeText(getApplicationContext(), "Note:
Only 2 members are allowed.", Toast.LENGTH_LONG).show();
        Intent intent = new Intent(g_ng.this,
group1.class);
        startActivity(intent);

    }

});
}

}

```

## 5. TESTING

Testing is the process of evaluation of a software item to detect differences between given input and expected output. We also conducted various testing on the rectify app to assess its features. The testing assesses the quality of the product. Software testing was done during the development process. A successful verification and validation process was conducted.

### **Verification**

Verification is the process to make sure the product satisfies the conditions imposed at the start of the development phase. In other words, to make sure the product behaves the way we want it to. The application successfully achieved the aim of splitting of bills with the required accuracy.

### **Validation**

Validation is the process to make sure the product satisfies the specified requirements at the end of the development phase. In other words, to make sure the product is built as per customer requirements. The product satisfies to all customer constraints, i.e. sending notifications regarding dues, splitting the amount with desired accuracy and a personal chat box.

### **BASICS OF SOFTWARE TESTING:**

There are two basics of software testing: blackbox testing and whitebox testing.

#### **Blackbox Testing**

Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

#### **Whitebox Testing**

White box testing is a testing technique that takes into account the internal mechanism of a system. It is also called structural testing and glass box testing.

Black box testing is often used for validation and white box testing is often used for verification.

## Types of testing

There are many types of testing like

- Unit Testing
- Integration Testing
- Functional Testing
- System Testing
- Stress Testing
- Performance Testing
- Usability Testing
- Acceptance Testing
- Regression Testing

### **Unit Testing**

Unit testing is the testing of an individual unit or group of related units. It falls under the class of white box testing. It was done by us test to that the unit implemented is producing expected output against given input.

### **Integration Testing**

Integration testing is testing in which a group of components are combined to produce output. Also, the interaction between software and hardware was tested by us to check if the software and hardware components have any relation. It may fall under both white box testing and black box testing.

### **Functional Testing**

Functional testing was conducted to ensure that the specified functionality required in the system requirements works. It falls under the class of black box testing.

### **System Testing**

System testing was conducted to ensure that by putting the software in different environments it still works. System testing is done with full system implementation and environment. It falls under the class of black box testing.



**Stress Testing**

Stress testing was conducted to evaluate how system behaves under unfavorable conditions. All possible inputs and constraints were cross-checked. Testing was conducted at beyond limits of the specifications. It falls under the class of black box testing.

**Performance Testing**

Performance testing was performed to assess the speed and effectiveness of the system and to make sure it is generating results within a specified time as in performance requirements. It falls under the class of black box testing.

**Usability Testing**

Usability testing was performed to the perspective of the client, to evaluate how the GUI is user-friendly? How easily can the client learn? After learning how to use, how proficiently can the client perform? How pleasing is it to use its design? This falls under the class of black box testing.

**Acceptance Testing**

Acceptance testing was done by the customer point of view to ensure that the delivered product meets the requirements and works as the customer expected. It falls under the class of black box testing.

**Regression Testing**

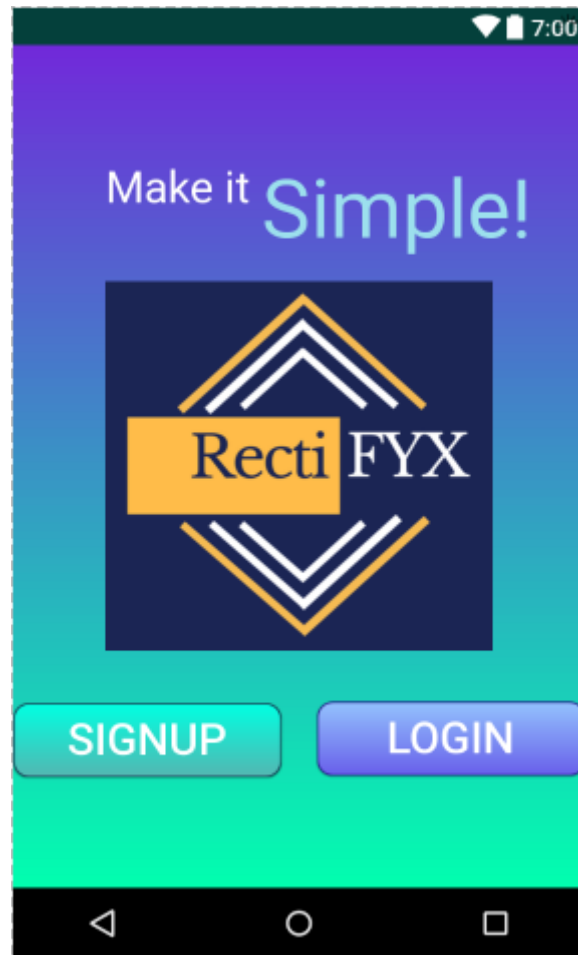
Regression testing was performed after modification of each system, component, or a group of related units to ensure that the modification is working correctly and is not damaging or imposing other modules to produce unexpected results. It falls under the class of black box testing.

## 6. USER INTERFACES

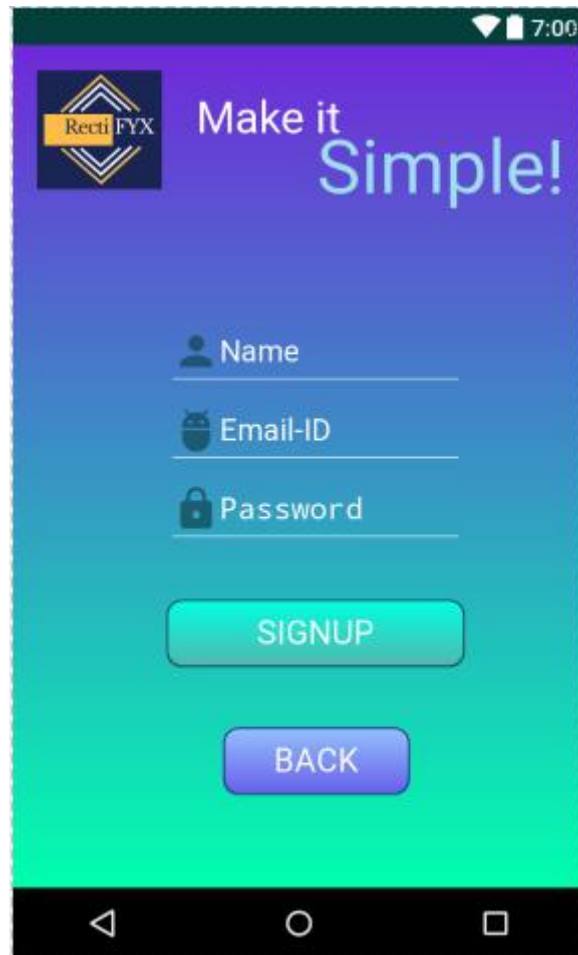
### 1. Let's Get Started



## 2. Welcome Page



### 3. Sign Up Page



RectiFYX

Make it Simple!

Name

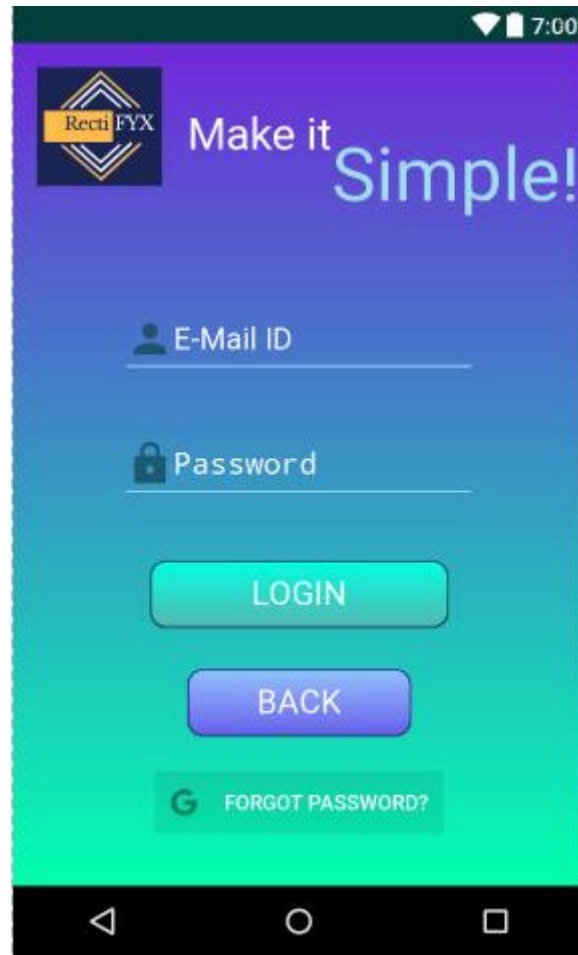
Email-ID

Password

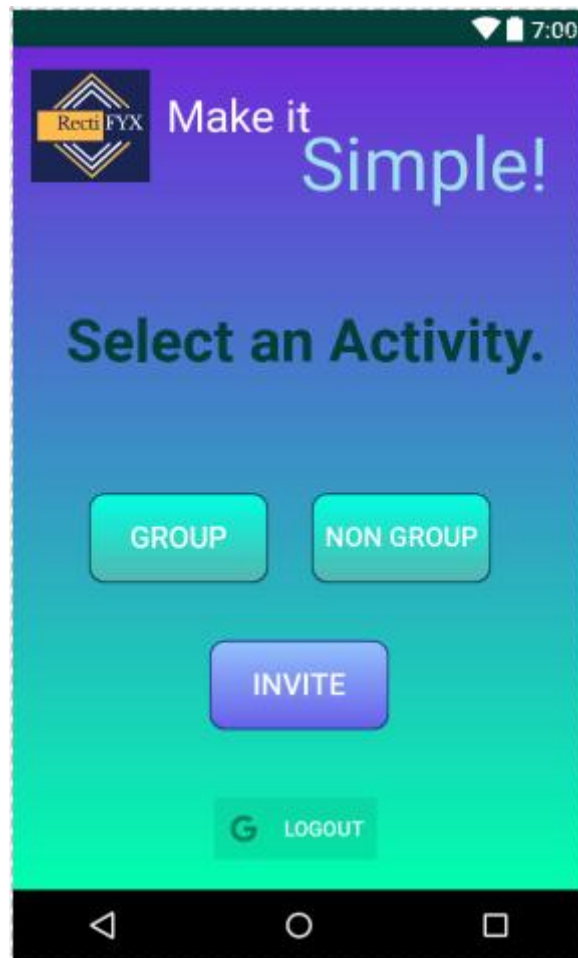
SIGNUP

BACK

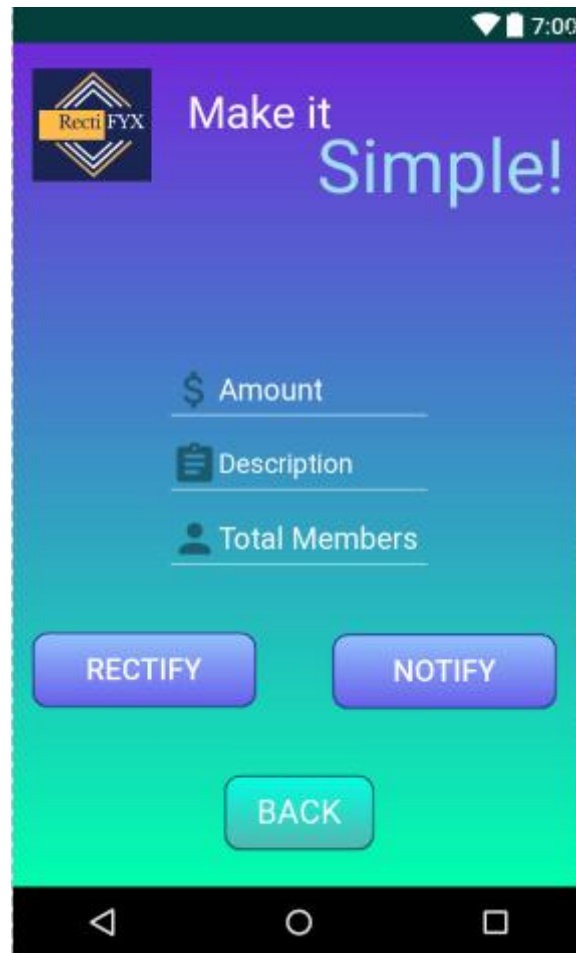
#### 4. Login Page



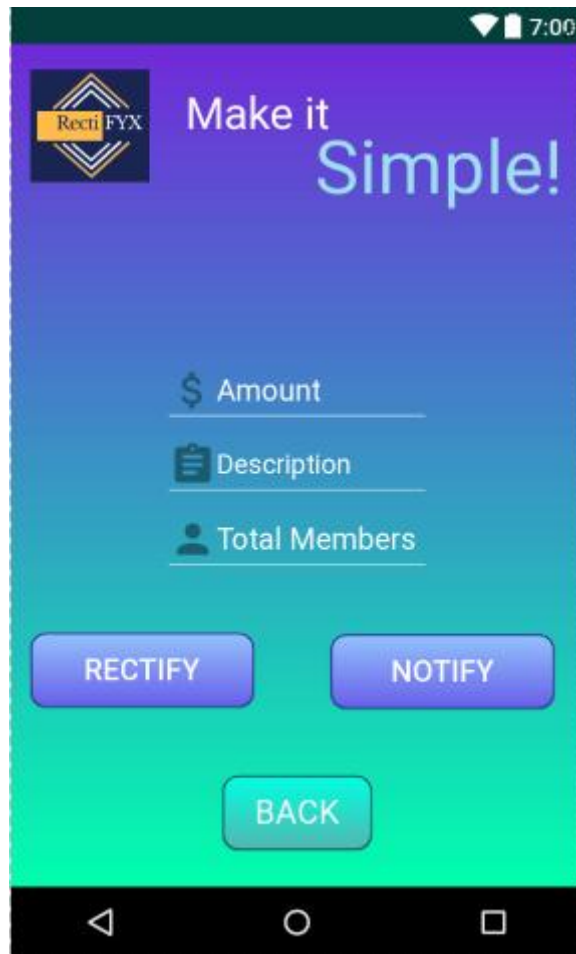
## 5. Activity Page



## 6. Group page



## 7. Non Group Expense Page



The screenshot shows a mobile application interface for a "Non Group Expense Page". At the top, there is a dark green status bar with a Wi-Fi icon, a battery icon, and the time "7:00". Below this is a purple header area. On the left of the header is a logo consisting of a diamond shape with the text "Recti FYX" inside. To the right of the logo, the text "Make it Simple!" is displayed in white. The main content area has a blue-to-green gradient background. It contains three input fields, each with a small icon to its left: a dollar sign icon for "Amount", a clipboard icon for "Description", and a person icon for "Total Members". Below these fields are three buttons: "RECTIFY" and "NOTIFY" are purple with white text, and "BACK" is a light blue button with white text. At the very bottom is a black Android navigation bar with three white icons: a back arrow, a circle, and a square.



## 7. REFERENCES

- <http://androiddeveloper.galileo.edu/2017/02/08/the-top-10-books-android-programming-2017/>
- [https://www.ieee.org/publications\\_standards/publications/books/index.html](https://www.ieee.org/publications_standards/publications/books/index.html)
- [ieeexplore.ieee.org/abstract/document/972716/](http://ieeexplore.ieee.org/abstract/document/972716/)
- <https://www.gitbook.com/book/google-developer-training/android-developer.../details>