

**Dharmsinh Desai University, Nadiad**

**Faculty of Technology**

**Department of Computer Engineering**

**B. Tech. CE Semester – VI**

**Subject: System Design Practice**

**Project Title: Billify**

**By:**

**1) Italiya Deval, Roll no : CE043, ID : 17CEUOS070**

**2) Italiya Kaushal, Roll no : CE044, ID : 17CEUOS072**

**3) Khokhani Prijen, Roll no : CE060, ID : 17CEUOG010**

**Guided By : Prof. Jigar M Pandya**



**Dharmsinh Desai University, Nadiad**

Faculty of Technology, Department of Computer Engineering

**CERTIFICATE**

This is to certify that System Design Practice entitled “Billify” is the bonafied report of work carried out by

1) Italiya Deval (17CEUOS070)

2) Khokhani Prijen (17CEUOG010)

3) Italiya Kaushal (17CEUOS072)

Of Department of Computer Engineering ,Semester VI , academic year 2019-2020, under our supervision and guidance.

|  |  |  |
| --- | --- | --- |
| Guide  **Prof. Jigar M Pandya**  Department of Computer Engineering,  Faculty of Technology,  **Dharmsinh Desai University ,**  **Nadiad - 387001**  **Date : 24/04/2020** |  | HOD  **Dr. C. K. Bhensdadia**  Department of Computer Engineering,  Faculty of Technology,  **Dharmsinh Desai University,**  **Nadiad – 387001**  **Date : 24/4/2020** |

**Contents:**

**1. Abstract 5**

**2. Introduction 6**

**1. Brief Introduction 6**

**2. Technology/Platform/Tools used 6**

**3. Software Requirement Specifications 7**

**4. Design 10**

**I. Use case diagram 10**

**II. Class diagram. 11**

**III. Sequence diagram 12**

**IV. Activity diagram 13**

**V. State diagram 14**

**Vl. ER diagram 15**

**5.Data Dictionary 16**

**6.Implementation Detail 18**

**I. Modules created and brief description of each modules. 18**

**Il. Function prototypes which implements major functionality. 20**

**7.Testing 28**

**8.Screen-Shots 32**

**9. Miscellaneous 42**

1. **Deployment Details. 42**
2. **Folder structure of the project. 45**
3. **Owner Ship of module 51**

**11.Conclusion 51**

**12.Limitation and Future extension 53**

**13.Bibliography 54**

**Abstract**

Toady we all know that people are used to go out for dinner,trip,some personal work,professional work with their friends, family etc.So, in such kind situation it may happen that any one or more person may pay some payment.So,then in such kind of situation people have remember all things that he/she has pay something for some else and owes from him/her or they themselves owe for someone.So, for this kind of confusion we come with one wonderful solutions.

Billify is solution for such scenario.Just add all your friends in this app.You may create group for common scenarion if it is.Then just mention that people and add expenses.They will be automatically notified by this app and not get confused for anykind such situation. Billify can handle a variety of payment scenarios: When one person pays or multiple people have paid, it can split payments evenly based on the amounts or allow you to select individual amounts for each person to pay.

**Introduction**

The app tracks borrowing and lending over time. And, until the balance is settled up, each person’s outstanding balance may go up and down. Billify sends reminders at the end of the month so everyone can pay what they owe and go into a new month with a fresh start. On the homepage, you’ll find a tally of the amount you owe and you can settle your balance at any time – you don’t have to wait until the end of the month. Payments can be settled with Paypal or Paytm directly from the app, or if the person who owes you pays in cash, you can record a cash payment to clear up the outstanding balance.

We’ve all been to group dinners and the server brings out just one check – or the restaurant has a policy of not splitting up checks for large groups. Tab solves the headache of figuring out what each person owes. Each person in the group can pay for the items they ordered without having to send the server back to separate the ticket.

**Tools and Technologies**

**Tools**

-Android Studio

**Technologies**

-Firebase Database

-Firebase Cloud Function

-Firestore

-Google Api

-Facebook Api

-Github Repository

**Software Requirement Specifications**

R.1.1 Authenticate user

Description:User can Sign up,login.

R.1.1.1 Sign out

Input:email,password,confirm password,username

Output:Send verification email

R.1.1.2 Verify user

Input:verify link

Output:Generate unique token

R.1.1.3 Login

Input:email,password

Output:Login successfully

R.1.1.4 Forgot password

Input:email

Output:Sent link successfully

R.1.1.5 Reset password

Input:paasword,confirm password

Output:Reset successfully

R.1.2 Manage friends

R.1.2.1 Request friends

Input:friends number or email

Output:sent request

R.1.2.2 Delete friends

Input:friends number or email

Output:delete friends

R.1.2.3 Acctept request

Input:command

Output:accept succefully

R.1.3 Manage group

R.1.3.1 Create group

Input:group name,friends number

Output:created successfully

R.1.3.2 Edit group

Description: Change group name,add new friends and delete friends

R.1.3.2.1 Change group name

Input: Enter new name

Output: Named changed successfully

R.1.3.2.2 Add friends

Input: Enter the number of friends

Output: Friends added successfully

R.1.3.2.3 Remove friends

Input: Enter the number of friend

Output: Removed successfully

R.1.3.3 Delete group

Input: Select group

Output: Deleted successfully

R.1.4 Manage expenses

Descritpion: Manage the expenses of friends and groups

R.1.4.1 Manage Friends Expenses

R.1.4.1.1 Add expense

Description: User can add equally or unequally expense also it can select option of who owns.

Input: Expense amount,Friends number,Enter the message

Output: expense added successfully

R.1.4.1.2 Edit expense

Input: Enter new amount,Edit friends

Output: Edited successfully

R.1.4.1.3 Delete expense

Input: Select expense

Output: Deleted successfully

R.1.4.2 Manage group expenses

R.1.4.2.1 Add expense

Description: User can add equally or unequally expense also it can select option of who owns.

Input: Expense amount and enter the message

Output: expense added successfully

R.1.4.2.2 Edit expense

Input: Enter new amount,Edit friends amount

Output: Edited successfully

R.1.4.2.3 Delete expense

Input: Select expense

Output: Deleted successfully

R.1.5 Settle Up

Description: User can settle up indivisually

Input: Click on settle up and write message

Output: Sette up successfully

R.1.6 Back up

Description:User can take backup his/her data as per requirement

Input:select back up

Output:backup data succefully

R.1.7 Expense history

R.1.7.1 view history

Input:select view

Output:Display history

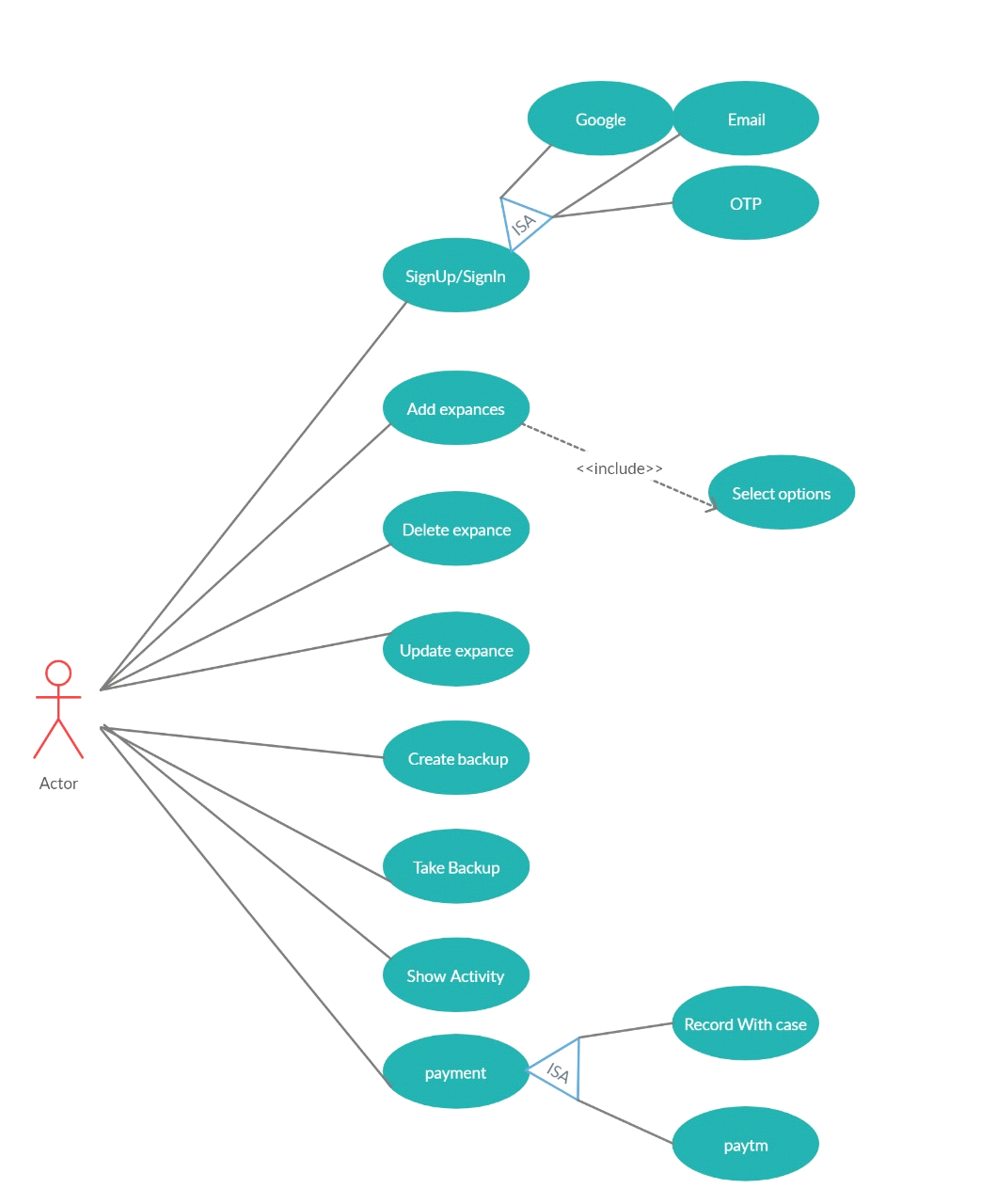
R.1.7.2 Search history

Input:search record

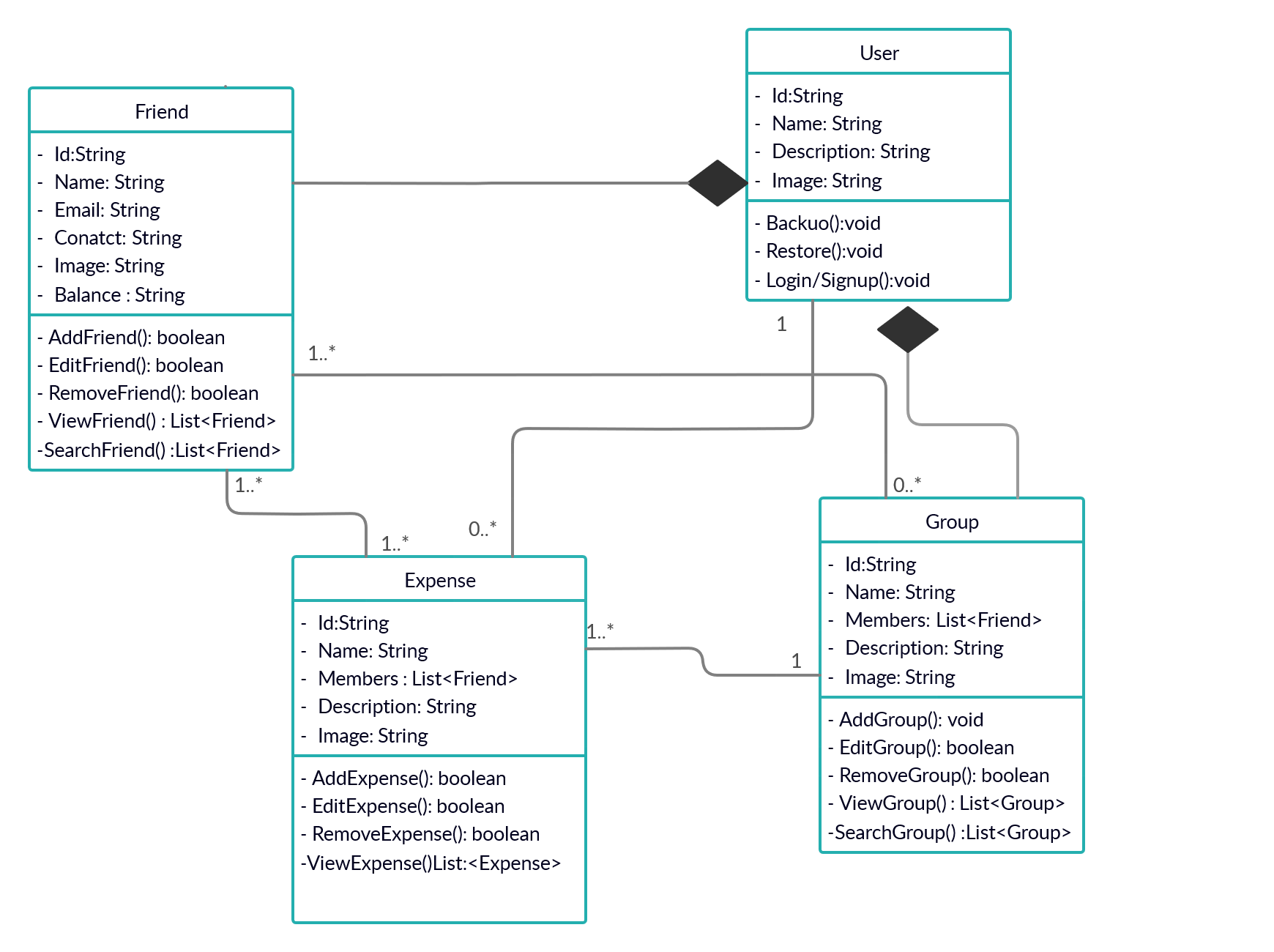
Output:view particular record

**Design**

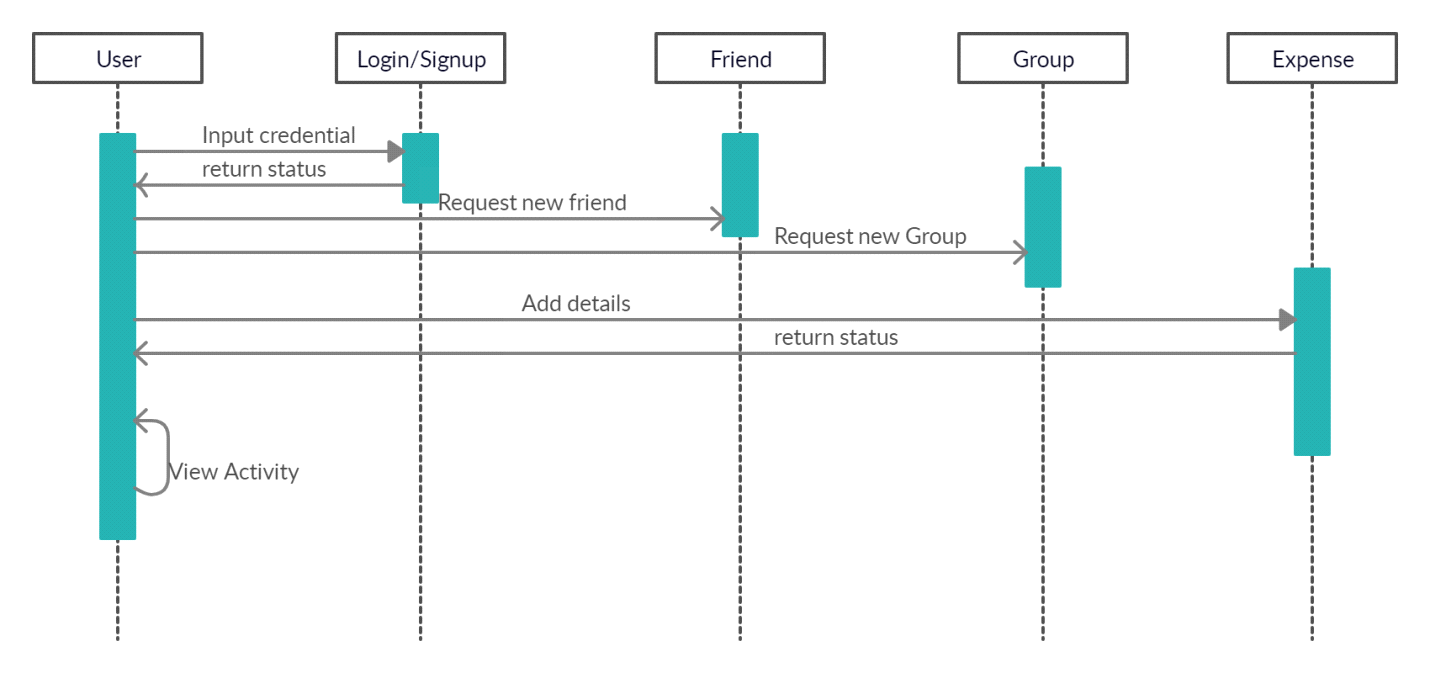
**Usecase Diagram :**



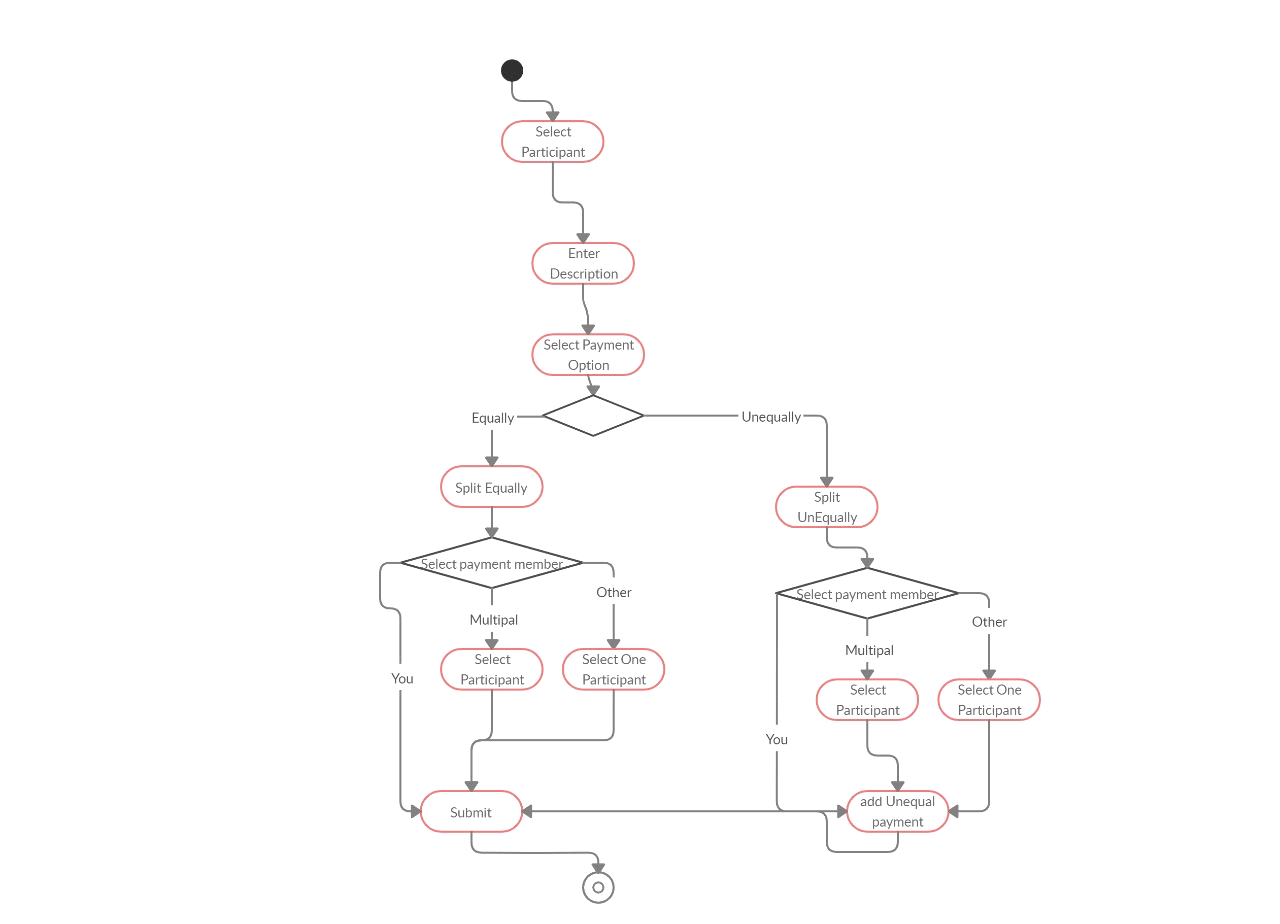
**Class Diagram :**



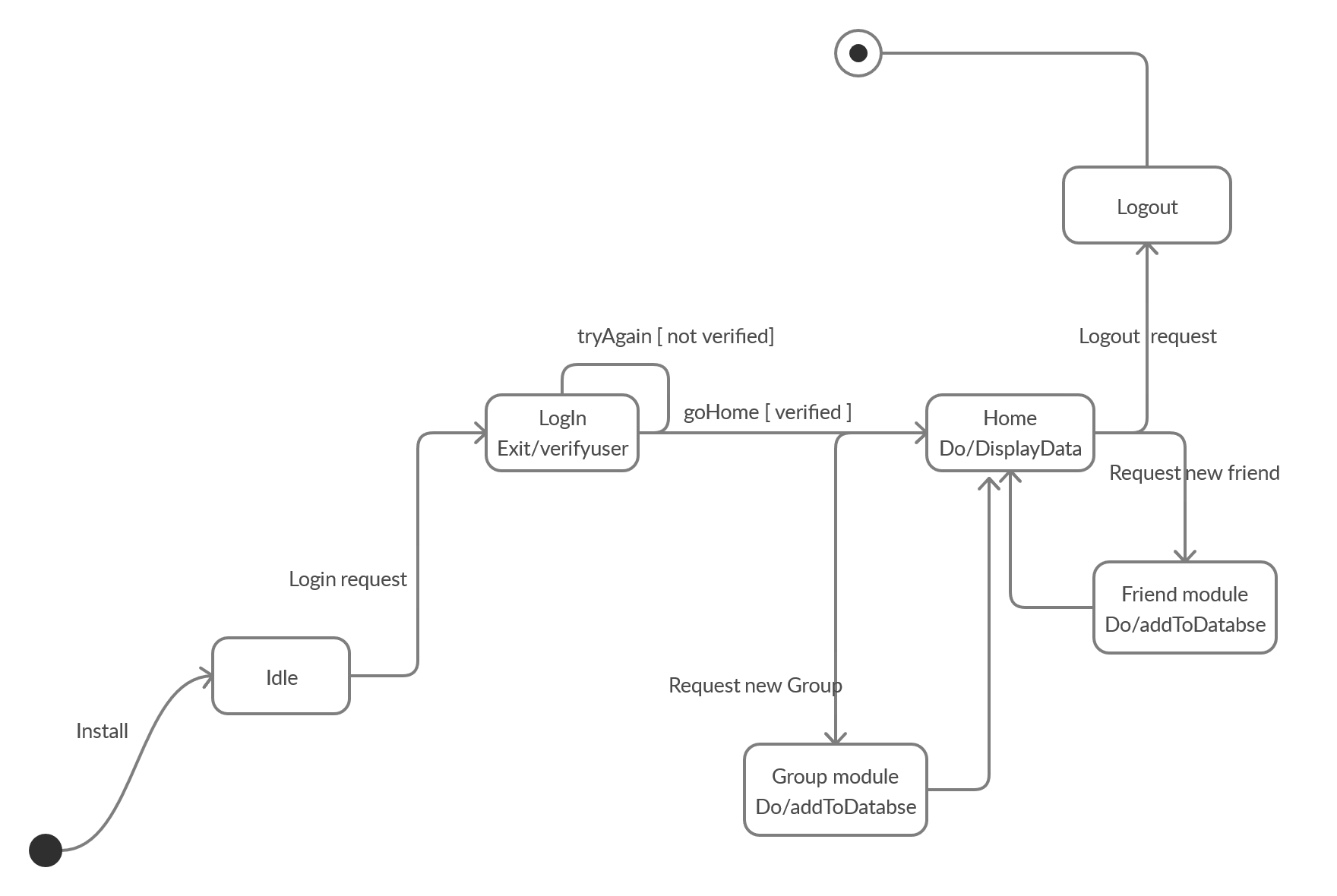
**Sequence Diagram :**



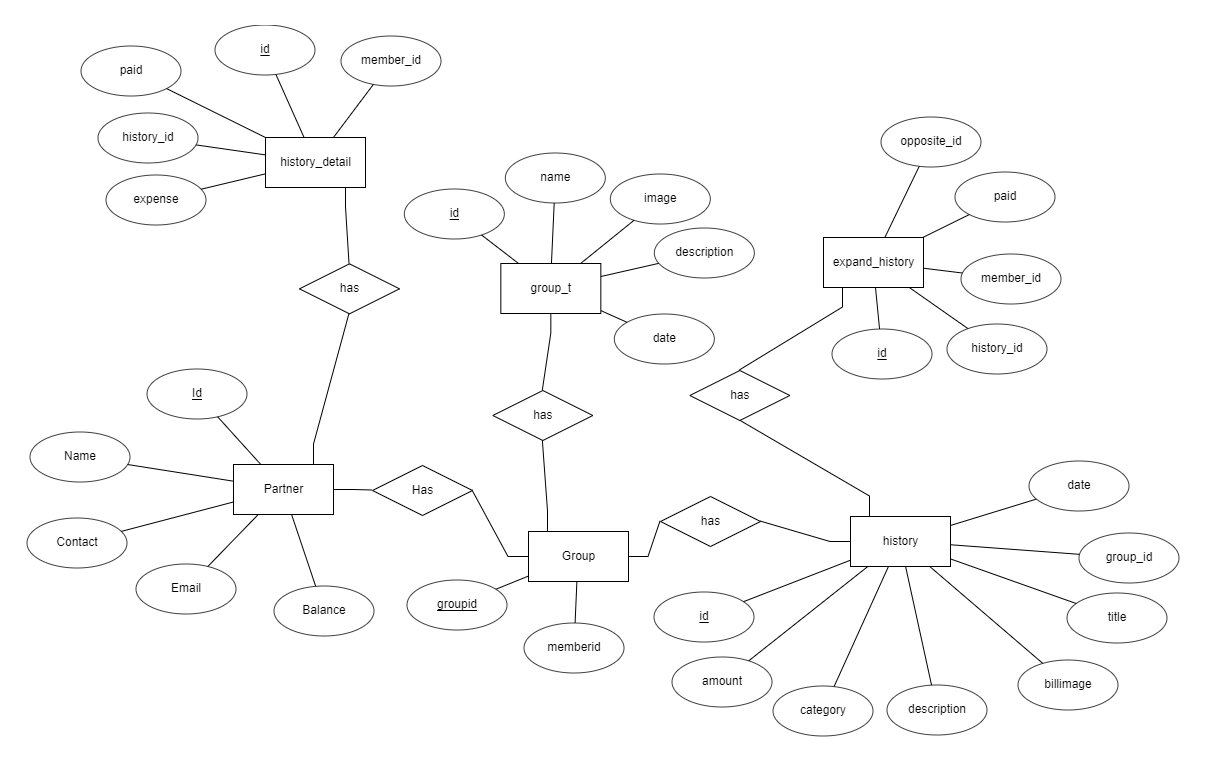
**Activity Diagram :**



**State Diagram :**



**E-R Diagram :**



**Data dictionary :**

**Partner**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr\_no.** | **Name** | **Data Type** | **Width** | **Required** | **Unique** | **PK/FK** | **Referenced Table** |
| **1** | **Id** | **Text** | **20** | **YES** | **YES** | **PK** | **-** |
| **2** | **Name** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **3** | **Profile** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **4** | **Contact** | **Integer** | **-** | **YES** | **-** | **-** | **-** |
| **5** | **Email** | **Text** | **50** | **YES** | **YES** | **-** | **-** |
| **6** | **Balance** | **NUMBER** | **10** | **YES** | **-** | **-** | **-** |

**History\_detail**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr\_no.** | **Name** | **Data Type** | **Width** | **Required** | **Unique** | **PK/FK** | **Referenced Table** |
| **1** | **Id** | **Text** | **20** | **YES** | **YES** | **PK** | **-** |
| **2** | **History\_id** | **Text** | **50** | **YES** | **-** | **FK** | **History** |
| **3** | **Member\_id** | **Text** | **50** | **YES** | **-** | **FK** | **Partner** |
| **3** | **Paid** | **Number** | **10** | **YES** | **-** | **-** | **-** |
| **4** | **Expence** | **Number** | **50** | **YES** | **-** | **-** | **-** |

**Group\_member**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr\_no.** | **Name** | **Data Type** | **Width** | **Required** | **Unique** | **PK/FK** | **Referenced Table** |
| **1** | **Group\_id** | **Text** | **20** | **YES** | **YES** | **PK** | **-** |
| **2** | **Member\_id** | **Text** | **50** | **YES** | **-** | **FK** | **Partner** |

**Expand\_history**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr\_no.** | **Name** | **Data Type** | **Width** | **Required** | **Unique** | **PK/FK** | **Referenced Table** |
| **1** | **Id** | **Text** | **20** | **YES** | **YES** | **PK** | **-** |
| **2** | **History\_id** | **Text** | **50** | **YES** | **-** | **FK** | **History** |
| **3** | **Member\_id** | **Text** | **50** | **YES** | **-** | **FK** | **Partner** |
| **4** | **Paid** | **Number** | **50** | **YES** | **-** | **-** | **-** |
| **5** | **Opposite\_id** | **Text** | **10** | **YES** | **-** | **-** | **-** |

**Group\_t**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr\_no.** | **Name** | **Data Type** | **Width** | **Required** | **Unique** | **PK/FK** | **Referenced Table** |
| **1** | **Id** | **Text** | **20** | **YES** | **YES** | **PK** | **-** |
| **2** | **Name** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **3** | **Image** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **4** | **Description** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **5** | **Date** | **Text** | **20** | **YES** | **-** | **-** | **-** |

**History**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Sr\_no.** | **Name** | **Data Type** | **Width** | **Required** | **Unique** | **PK/FK** | **Referenced Table** |
| **1** | **Id** | **Text** | **20** | **YES** | **YES** | **PK** | **-** |
| **2** | **Amount** | **Number** | **50** | **YES** | **-** | **-** | **-** |
| **3** | **Date** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **4** | **Billimage** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **5** | **Group\_id** | **Text** | **50** | **YES** | **-** | **FK** | **Group** |
| **6** | **Description** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **7** | **Title** | **Text** | **50** | **YES** | **-** | **-** | **-** |
| **8** | **Category** | **Text** | **50** | **YES** | **-** | **-** | **-** |

**Implementation Detail**

**Authentication Module:**

**Sign Up Activity**

This activity is used to store user’s data to the database and enables the user to login to system. All the fields in this module contains validations.

Input : User’s Information

Output: User Registration and redirect to login page

**Sign In Activity**

This activity is takes users credentials and then verifies it with registered users, If user is not registered invalid credential is shown and if user is registered then it will match the database and user will login. All the fields in this module contains validations.

Input : User’s Credentials

Output: Logging user

**Forget Password Activity**

In this activity user gives its email id and if the email id is valid and match the database. Then it send the link to the that registered email.

Input : User’s email

Output: Check your mailbox

**Manage Friends Module:**

**All Friends Activity**

In this activity all the friends will be shown with their expenses.

Input : All Friends

Output: Show the List

**Add Friend Activity**

In this user can add new friends. User can add his friend by his contact or email id.

If that contact or email will be valid and that will be in database then it will send the request to that friend.

Input : User’s email or contact

Output: Friend is added

**Manage Group Module:**

**All Groups Activity**

In this activity all the group will be shown with their expenses.

Input : All group

Output: Show the List

**Add Group Activity**

In this activity user can create new group. For the user have add their friends name and name of the group.

Input : Add Friends

Output: Group is created

**Expense Module:**

**Manage Expense Activity**

Using this activity user can add new expenses,can edit and delete also . For that user have add all the valid information and name of friends. In this there is option for how to split. By that if user wants to add different amount in all the friends then it can choose split unequal. There is also option for who paid, using that user can choose multiple pairs.

Input : Add amount and friends

Output: Expense Added

**Backup Module:**

**Create Backup Activity**

If the user wants to take the backup of all expenses. Then using create backup user can create the backup.

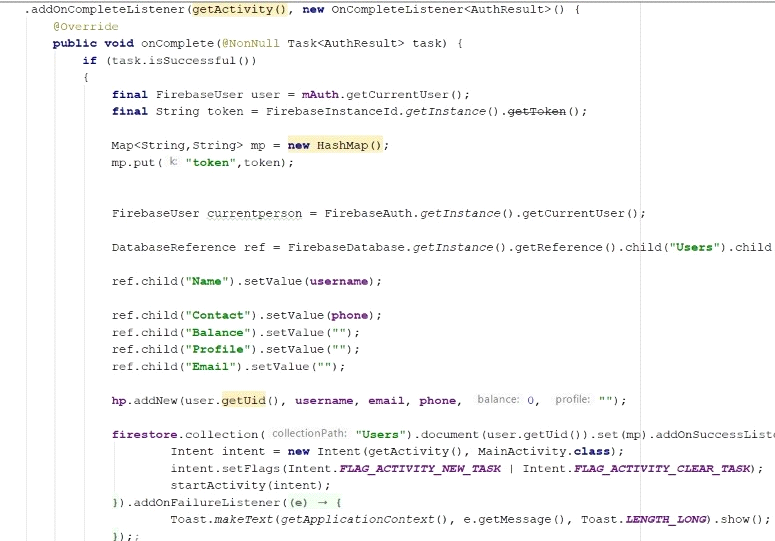
Input : Click get Backup

Output: Created succefully

**Function Prototype:**

* Loginfragment()
* signupfragmet()
* MainActivity()
* AddExpenseActivity()
* AddInGroup()
* display\_History()
* display\_groups()
* delete\_expense()
* display\_friends()

**1) SignIn(Through contact no.)**

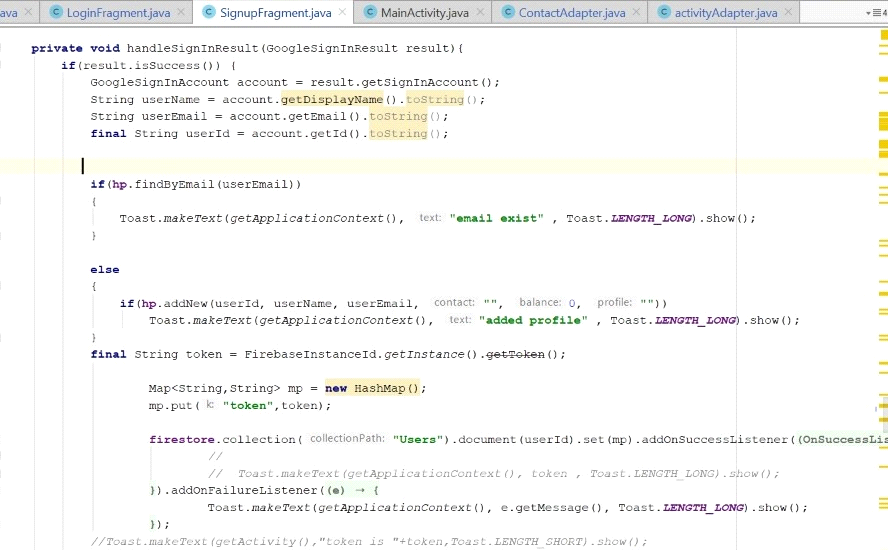


**Input:-** task

**Output:-** Success or failure Message will be Toast to user display.

**Description:-** Basically here input is task. task is object of one process of authentication.when use try to signIn using ContectNo. then he has to pass through OTP verification process.one OTP is send to use and when he enter correct OTP then Authentication is done by Firebase server.When that process is completed then one task object is passed to "*onComplete*" method. This method get User refrence from that task Object.and bind all information of user to "ref" object that you can see in snap.then according to userid those information is stored to Firebase server and new user is successfully registered.

**2) SignIn (through Google)**

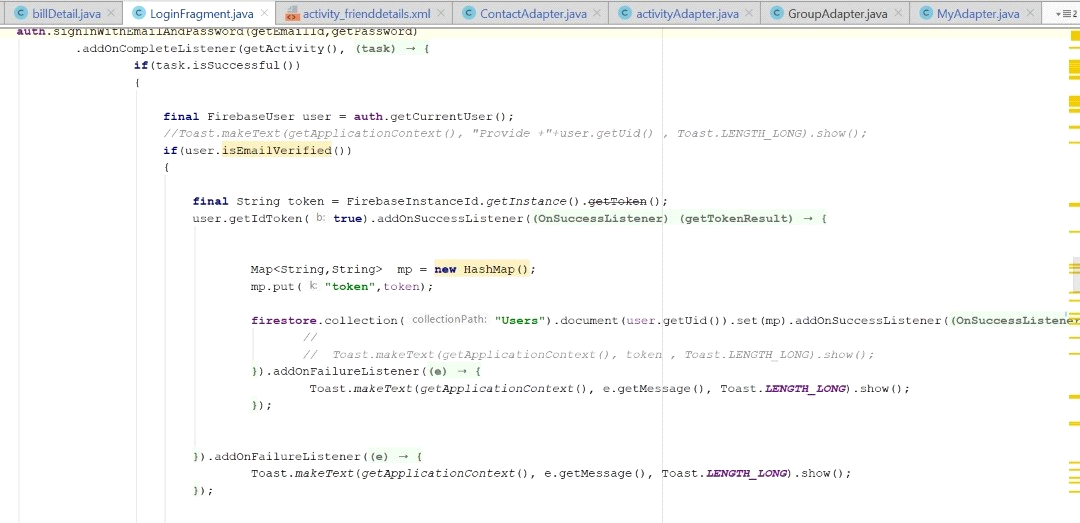


**Input:-** GoogleSignInResult result

**output:-** Success or failure Message will be Toast to user display.

**Description:-** Here when user try to Signin using GoogleId then google server will authenticate email id and create one result object.That object is passed to "*handleSigninResult"* function.this function verify that email is already exists on the server or not.According to that one Firebase token is genrated and user will be registered.

**3) LogIn**

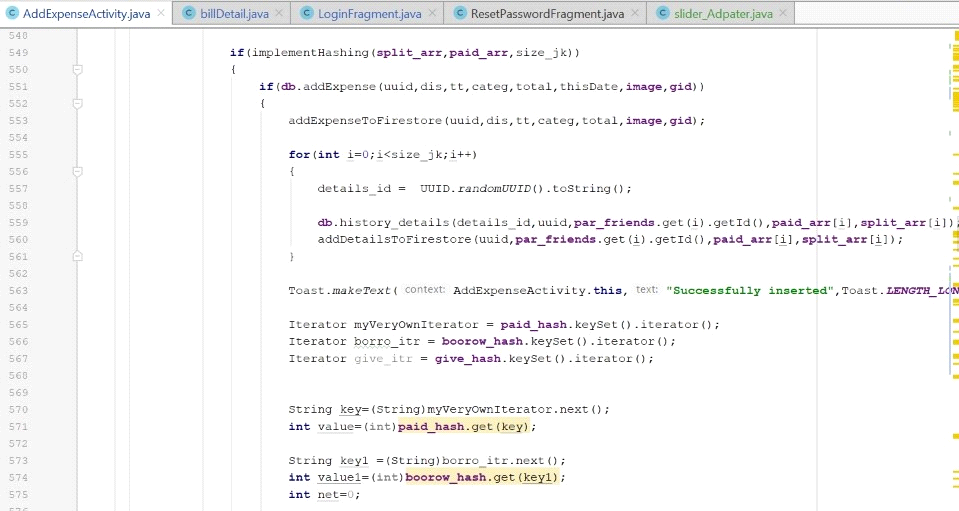


**Input:-** EmailId and Password

**Output:-** LogIn Responce

**Description:-** Firebase server will create one user object by authentication.and if "*isEmailIsVerified*" fuction is true for that user object then firebase server create login Token.and according to that token user will get loggedin.

**5) Add Expence**

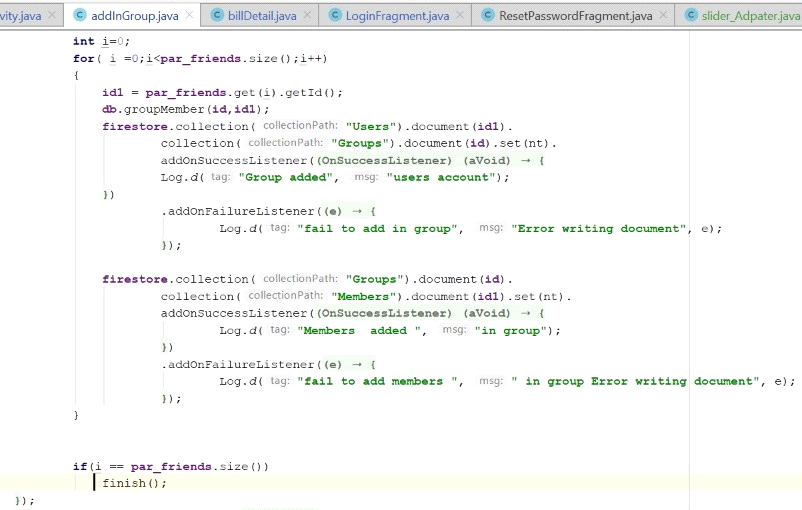


**Input:-** uuid,dis,tt,categ,thisDate,total,image,gid

**output:-** one successful message will be toast and one History of this expance will be added.

**Description:-** Here when all input of this function is valid then according to expance type Owes money is distributed in participants Tab.and then one history is genrated of this Expence and Added is History Tab.and "successfully inserted" message will be toast at the bottom of screen.

6) Add Group

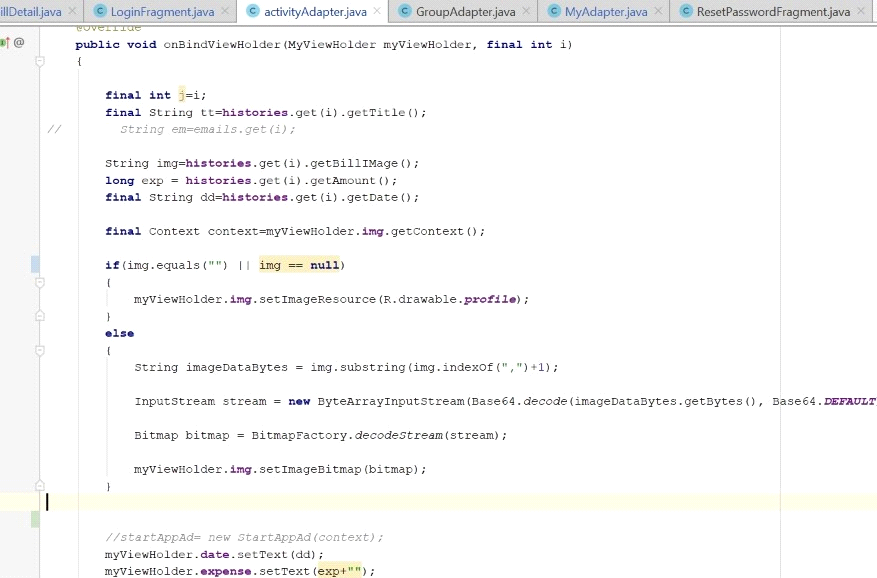


**Input:-** Title,Description,participants,Image

**Output:-** Toast message

**Description:-** here two id is created one is "*id*" and second is "*id1*".one is for group and second is for participants.first group is added to Database and then participants will be hased to that group.

**7) Activity Adpter**



**Input:-** MyViweHolder,histories

**Output:-** This adpter is use to display history,activitys

**Description:-** from teo object that is passed to this fuction is used to extract information such as image,amount,name.. etc. and this function will bind those information with Adpter information.

**8) Display Activity**



**Input:-** Adpter

**Output:-** Activity will be appear in RecycleView

**Description:-** For display the activity into recycler view will have to bind information to adpter and we have to pass that adpter to recycler view.so information binding to adpter we have seen in privious snapshot.

**Testing**

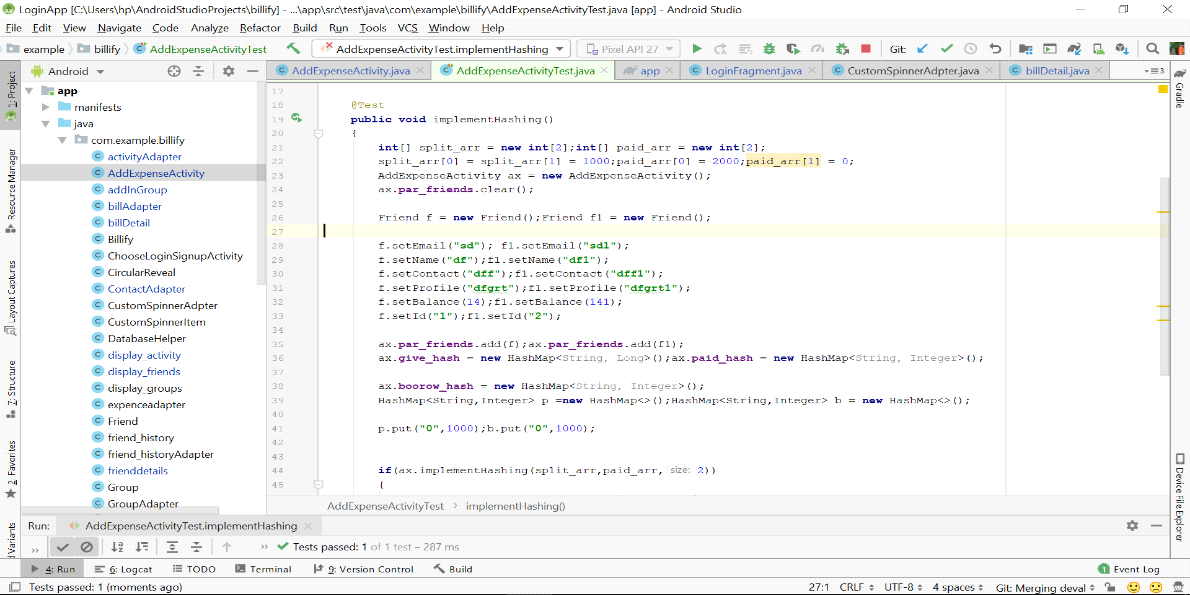
**Unit Testing(Junit framework)**

* Junit framework is inbuilt framework available in java for unit testing.

**Demo - Split expense**



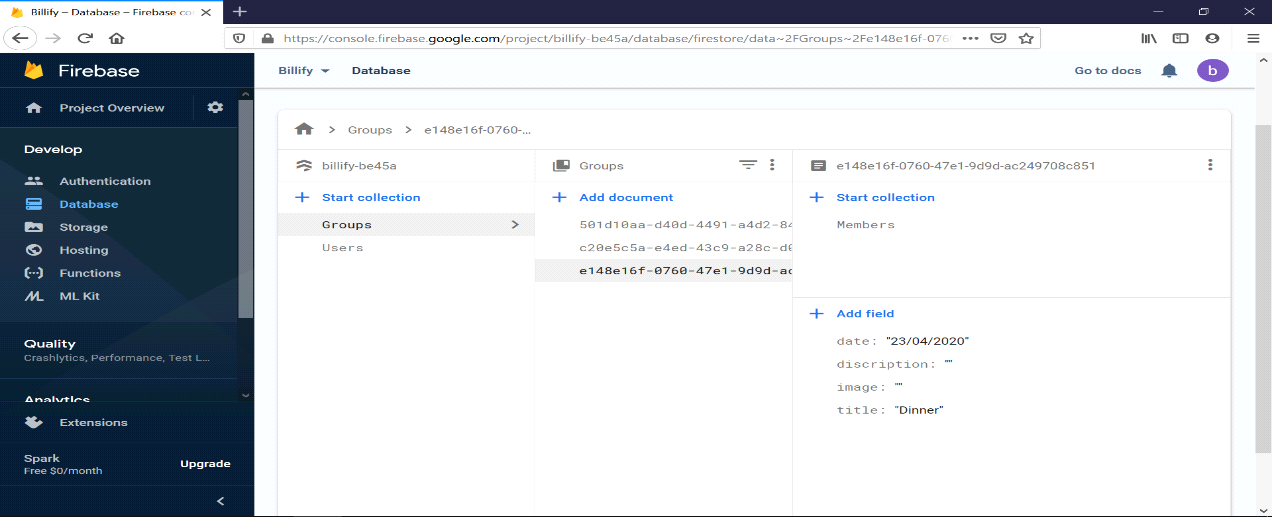
Test case failed



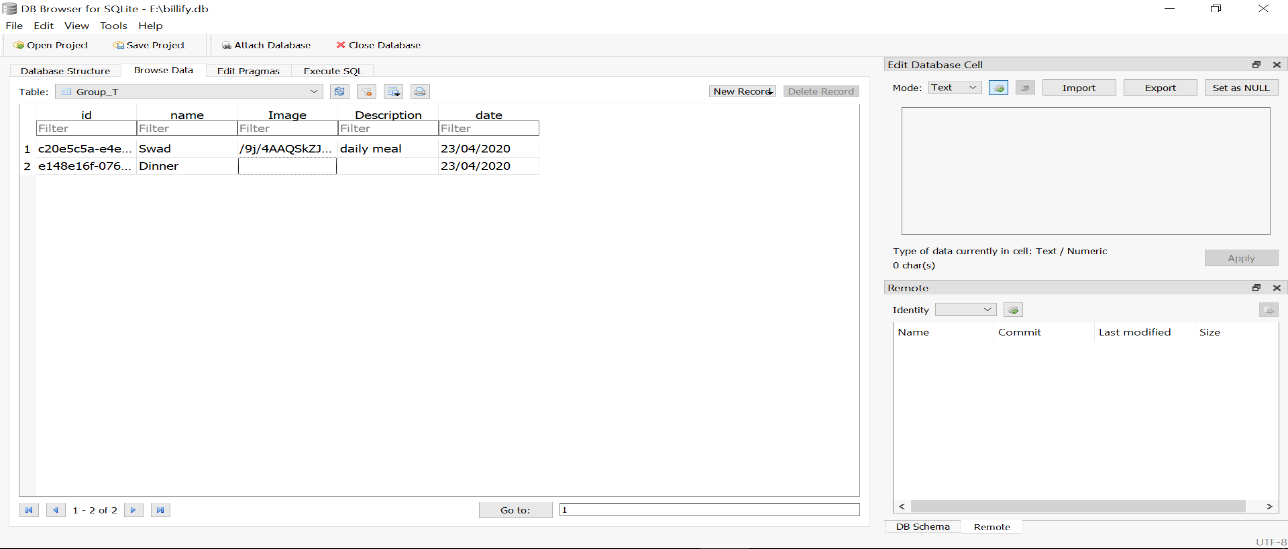
Test case Passed

**Integration Testing – Testing with database**

Demo – View group



Firestore database( Online Database)

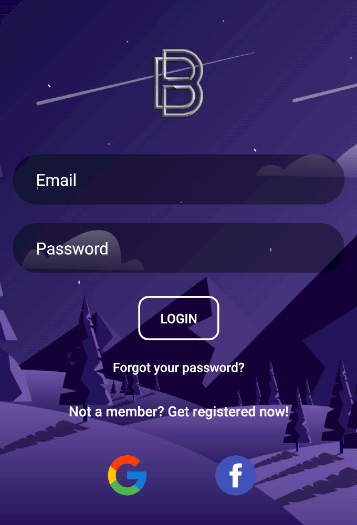


Sqlite database( Offline Database)

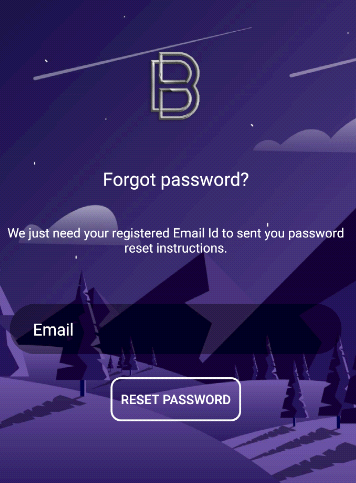
**Screen-Shots**

**Login Module**

SigninSignup

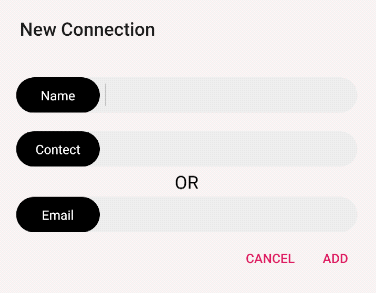


Forgot Password

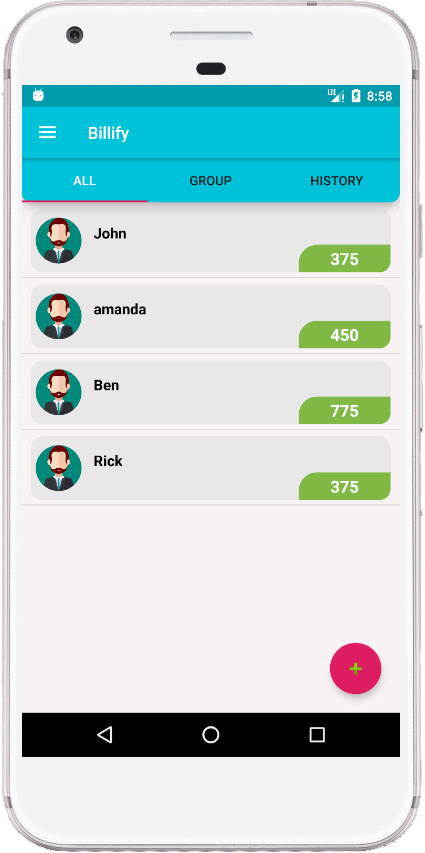


**Manage Friend Module**

Add Friend

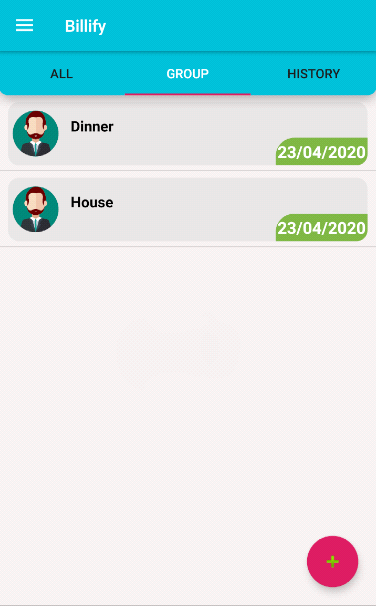
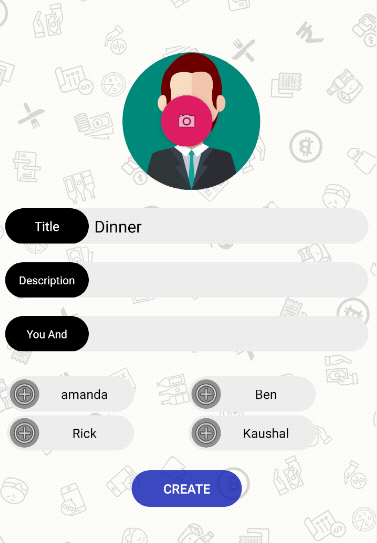


Show All Friends



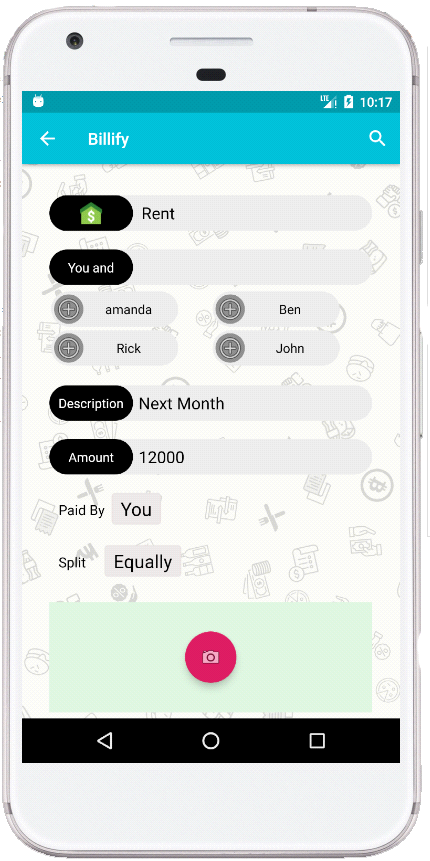
**Manage Group Module**

Add Group Show Groups

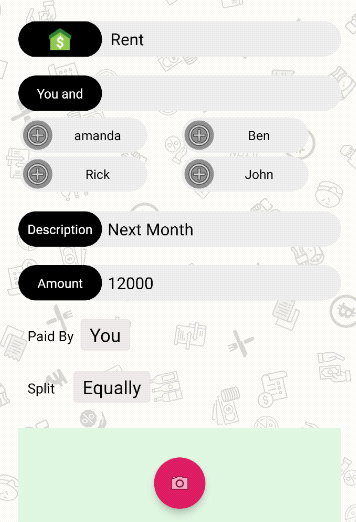


**Expenses Module**

Add Expense

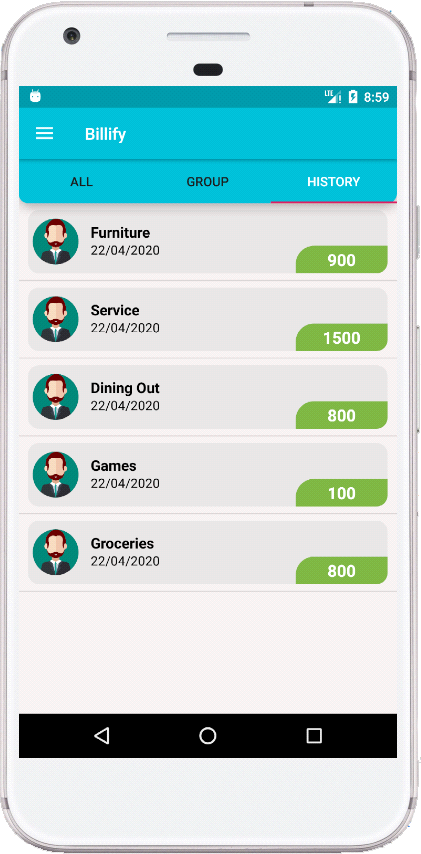
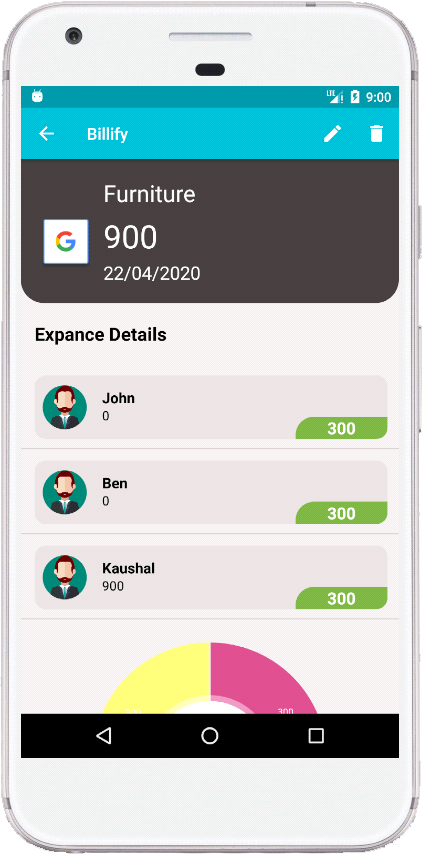


Add Expense Help

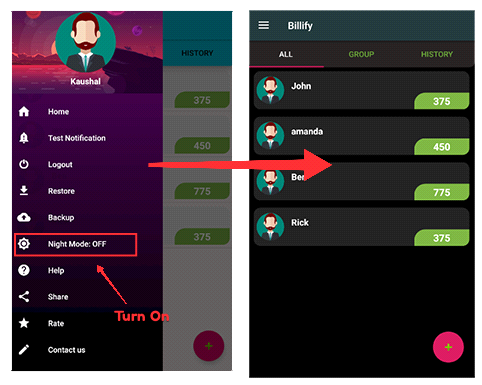


**History**

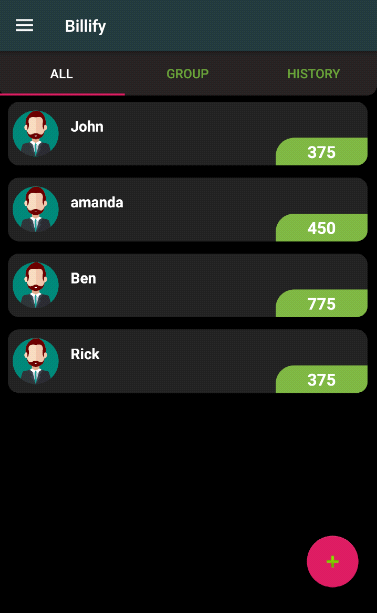
Show History History Detail

**Dark Mode**



After Dark Mode Applied



**Miscellaneous**

**Deployment Details :**

There are step by step given below that you have to follow to pulish or deploy your android app to play store,

1. Sign up for a Google Play Developer account.

* Pay registration fee.
* Complete your account details.

<https://play.google.com/apps/publish>

2. Then folllow given below link to set up account,

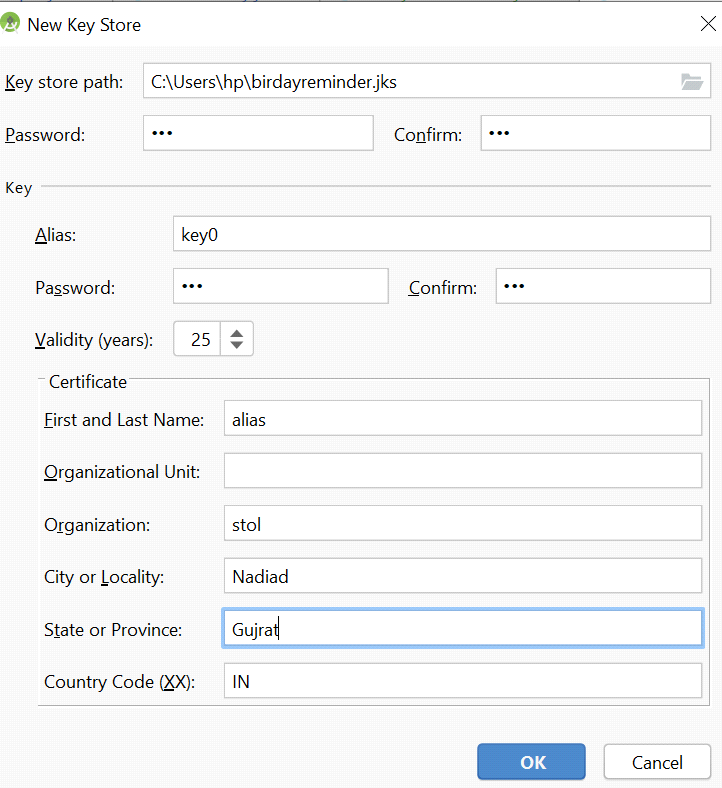
<https://support.google.com/googleplay/android-developer/answer/113469?hl=en>

3. In android studio Generate signed bundle,

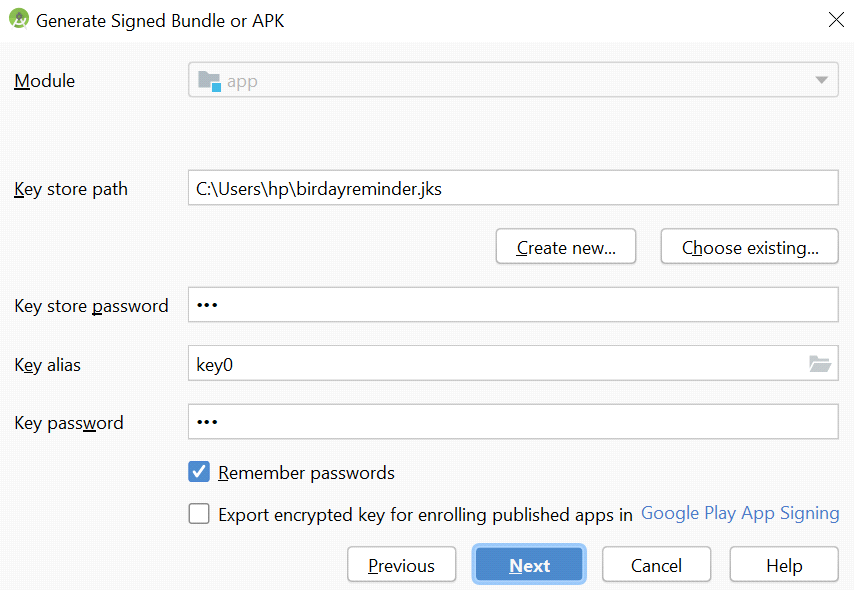
* Go to Build -> Generate Signed Bundle/APK... in Android Studio.
* Select Android App Bundle -> Next



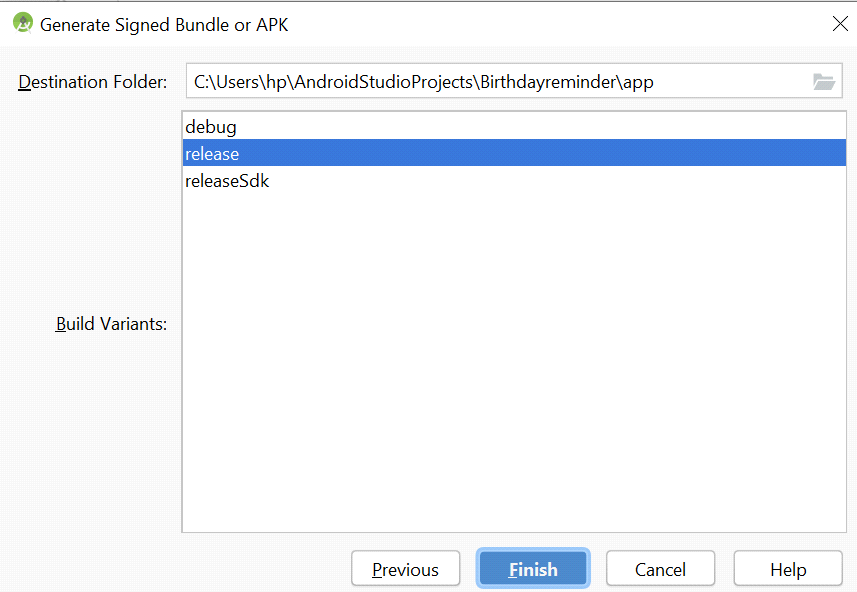
* Next enter details as like shown below and click OK -> Next.



* Enter details as previous and click Nextbutton.



* Select release and click Finish



* Now, Signed apk(bundle) is ready.

4. Now go to your google play developer console

* Click 'Create Application'
* Select a default language and add a title for your app. Type the name of your app as you want it to appear on Google Play.
* Create your app's store listing,

<https://support.google.com/googleplay/android-developer/answer/188189>

* Take the content rating questionnaire,

<https://support.google.com/googleplay/android-developer/answer/188189> and set up

* Set up pricing and distribution.

<https://support.google.com/googleplay/android-developer/answer/6334373>

* Now, Create a release

<https://support.google.com/googleplay/android-developer/answer/7159011?hl=en>

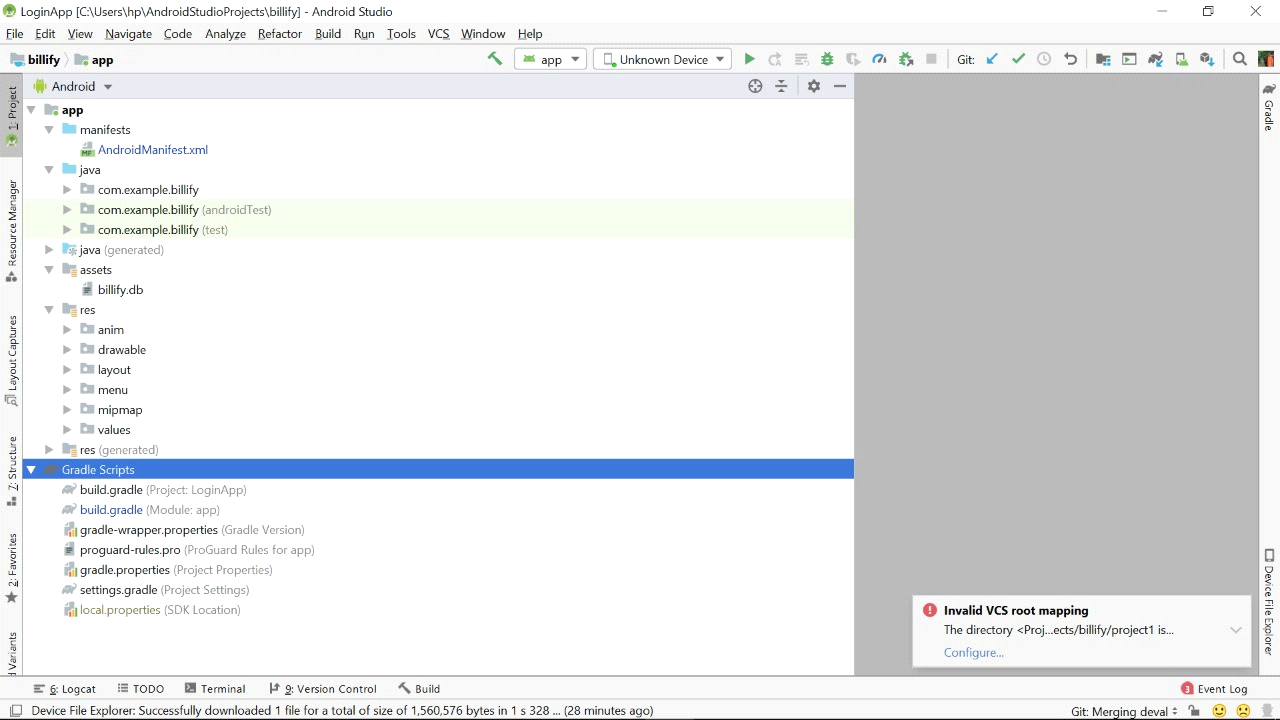
* Prepare your app's release
* Review and start roll out your release

**# With in maximum 7 days application will be live**

**Folder structure of the project :**

Android Studio is the official IDE (Integrated Development Environment) developed by JetBrains community which is freely provided by Google for android app development.

After completing the setup of Android Architecture we can create android application in the studio. We need to create new project for each sample application and we should understand about the folder structure. It look like this:



**Detailed explanation of every file and folder :**

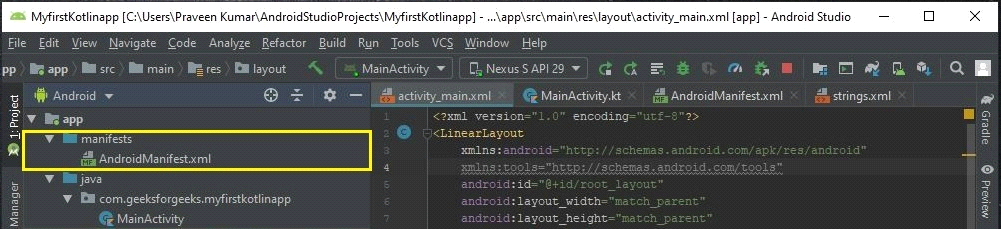
The android project contains different type of app modules, source code files and resource files. We will explore all the folders and files in android app.

* Manifests Folder
* Java Folder
* res (Resources) Folder
* Drawable Folder
* Layout Folder
* Mipmap Folder
* Values Folder
* Gradle Scripts

**Manifests Folder**

Manifests folder contains **AndroidManifest.xml** for our creating the android application. This file contains information about our application such as android version, metadata, states package for Kotlin file and other application components. It acts as an intermediator between android OS and our application.

Following is the mainfests folder structure in android application.



**AndroidManifest.xml**

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

<manifest xmlns:android="http:// schemas.android.com/apk/res/android"

    package="com.geeksforgeeks.myapplication">

    <application

        android:allowBackup="true"

        android:icon="@mipmap/ic\_launcher"

        android:label="@string/app\_name"

        android:roundIcon="@mipmap/ic\_launcher\_round"

        android:supportsRtl="true"

        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity">

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />

            </intent-filter>

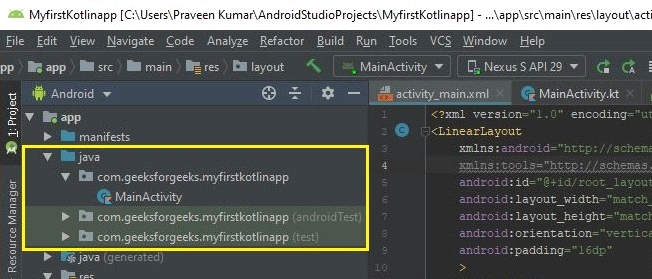
        </activity>

    </application>

</manifest>

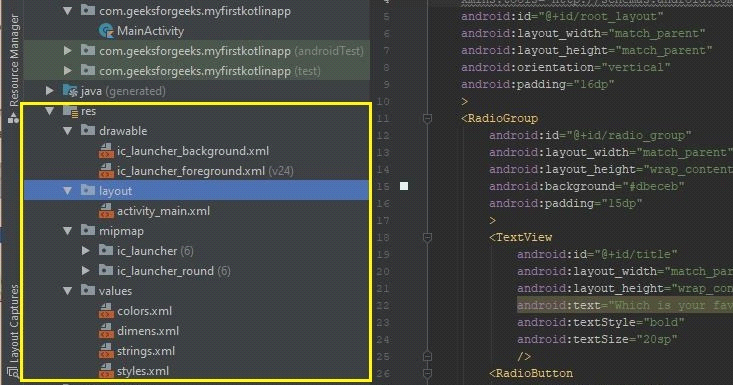
**Java Folder**

Java folder contains all the java and Kotlin source code (.java) files which we create during the app development, including other Test files. If we create any new project using Kotlin, by default the class file MainActivity.kt file will create automatically under the package name “com.geeksforgeeks.myfirstkotlinapp” like as shown below.



**Resource(res) folder**

Resource folder is the most important folder because it contains all the non-code sources like images, XML layouts, UI strings for our android application.



**Res/drawable folder**

It contains the different type of images used for the development of the application. We need to add all the images in drawable folder for the application development.

**Res/layout folder**

Layout folder contains all XML layout files which we used to define the user Interface of our application. It contains the **activity\_main.xml** file.

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

<androidx.constraintlayout.widget.ConstraintLayout

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

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

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

    android:layout\_width="match\_parent"

    android:layout\_height="match\_parent"

    tools:context=".MainActivity">

    <TextView

        android:layout\_width="wrap\_content"

        android:layout\_height="wrap\_content"

        android:text="Hello World!"

        app:layout\_constraintBottom\_toBottomOf="parent"

        app:layout\_constraintLeft\_toLeftOf="parent"

        app:layout\_constraintRight\_toRightOf="parent"

        app:layout\_constraintTop\_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>

**Res/midmap folder**

This folder contains launcher.xml files to define icons which are used to show on the home screen. It contains different density type of icons depends upon the size of the device such as hdpi, mdpi, xhdpi.

**Res/values folder**

Values folder contains a number of XML files like strings, dimens, colors and styles definitions. One of the most important file is **strings.xml** file which contains the resources

<resources>

    <string name="app\_name">NameOfTheApplication</string>

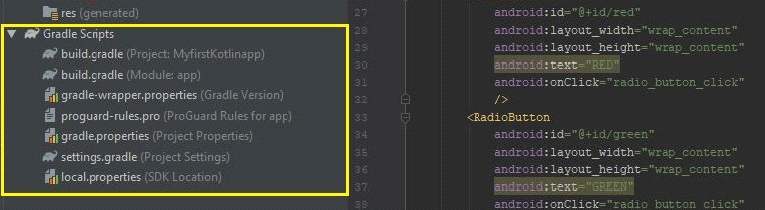
    <string name="checked">Checked</string>

    <string name="unchecked">Unchecked</string>

</resources>

**Gradle Scripts folder**

Gradle means automated build system and it contains number of files which are used to define a build configuration which can be apply to all modules in our application. In build.gradle (Project) there are buildscripts and in build.gradle (Module) plugins and implementations are used to build configurations that can be applied to all our application modules.



**Owner Ship of Module:**

Here all the activities and detailed work distribution is given.

* Deval Italiya: -
* Authentication Module and Security
* Firebase base login using mail or contact
* Firebase logout
* Forget password
* Login and logout using google
* Login and logout using facebook
* Security
* Prijen Khokhani: -
* Group and Expense Module
* Add friend
* Add, update and delete expense
* Backup management
* Create and update group
* History Management
* Kaushal Italiya: -
* Designing
* Dark theme mode
* Light theme mode
* Handling Orientation Chenges

**Conclusion**

The functionality implemented in the system was done after understanding all the system modules according to the requirements.

Functionalities that are successfully implemented in the system are:

**Main Features**

**-**Split expenses

**-**Group expense

**-**History

-Backup/Restore

-Notification

**Advantages**

**-**Easy expense management

**-**Analysis of expenses

**-**Split expense on one tap

-Group for daily expense

After the implementation and coding of system, comprehensive testing was performed on the system to determine the loopholes and possible flaws in the system.

**Limitation and Future extension**

**Limitations**

-Anyone can update or delete payment without any other participants permission.

-Any group member can add new members.

-Payment directly through this app

-This app is available for only android users.

**Future extension**

-Payment mode through this will be implemented

-For ios platform and web also availble in nearer future.

-Performance of this app will be optimised in future.

**Bibliography**

**Reference**

<https://developer.android.com/guide>

<https://developers.facebook.com/docs>

<https://console.developers.google.com/cloud-resource-manager>

<https://firebase.google.com/>