subtractly.

An Android Subscription Tracker

Presented:
Anirudh J B - 1BM17IS009
Irene Komal P - 1BM17IS033

THE APPLICATION

Subscriptions are fast growing business models for online service platforms where users avail services by paying on a periodic basis.

Subscription models come in many variations based on how many people can split it, the payment cycles and different access levels to content.

Today, most people split subscriptions with friends and family and share the costs across multiple online service platforms.

There is a need to track these transactions and simplify them.

FUNCTIONAL REQUIREMENTS (1)

* The user should be able to enter new subscriptions and their relevant details including subscription type, start date, billing cycle and amount to be paid.
* The user should be able to delete subscriptions that they no longer subscribe to.
* The user should be able to add members to a given subscription.
* The user should be able to delete subscriptions and obtain a reconfigured share-split with the remaining members.

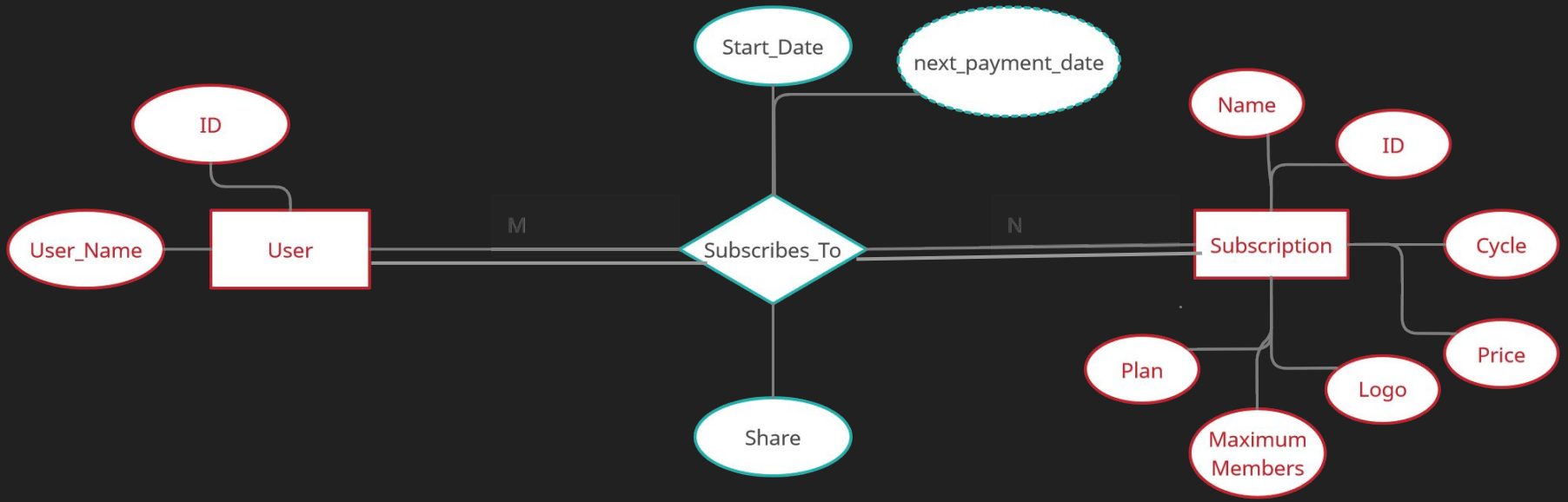
FUNCTIONAL REQUIREMENTS (2)

* The user should be able to view all their subscriptions and further details on any subscription they choose.
* The user should be able to have easy access to the deadlines both on an individual subscription level as well as in terms of all their collective subscriptions.
* The user should be able to view latest news and trends related to their subscription platform.
* The user should be able to view personalized insights about their spending and content consumption.

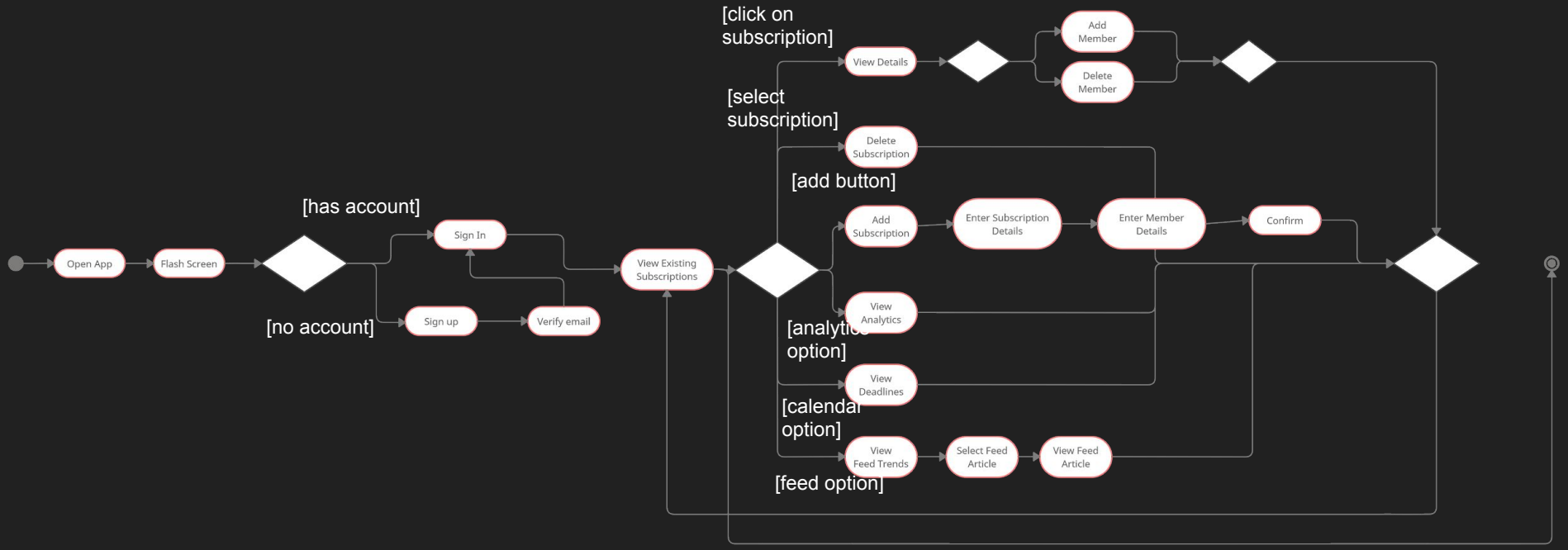
NON FUNCTIONAL REQUIREMENTS

* The application must provide ease-of-use by allowing users to manage subscriptions with intuitive user interfaces and without arduous setup and registration processes.
* The application must provide secure storage and processing of data by ensuring minimal information outsourced to network-reliant services.
* Majority of the application functionality must be available irrespective of network conditions.
* The application must function across all android devices with SDK version greater than 24.
* The application should work in both light and dark modes to improve user accessibility.

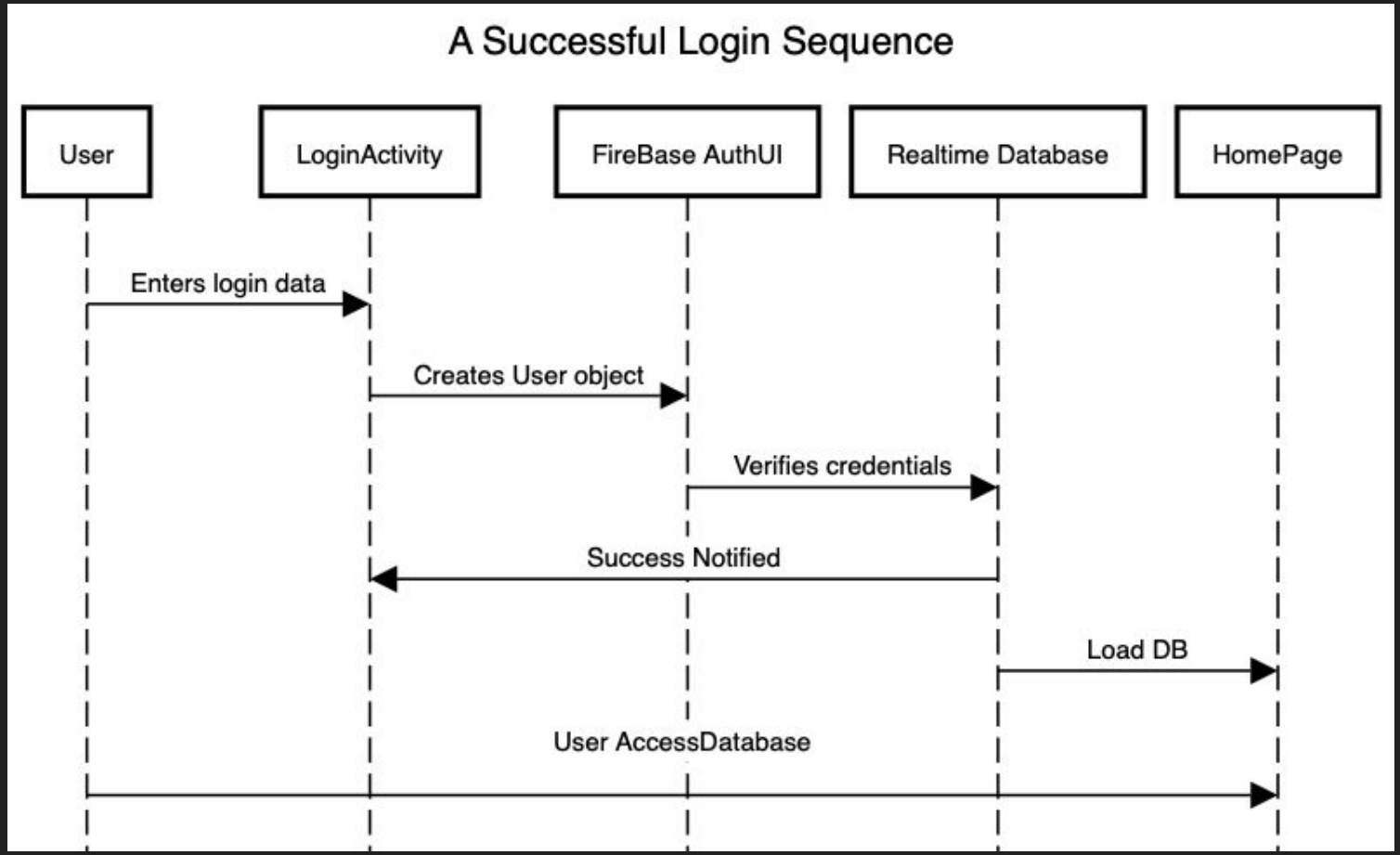
ER DIAGRAM


Start_Date
next_payment_date
ID
Name
ID
User_Name
User
M
Subscribes_To
N
Subscription
Cycle
Price
Plan
Logo
Maximum Members
Share

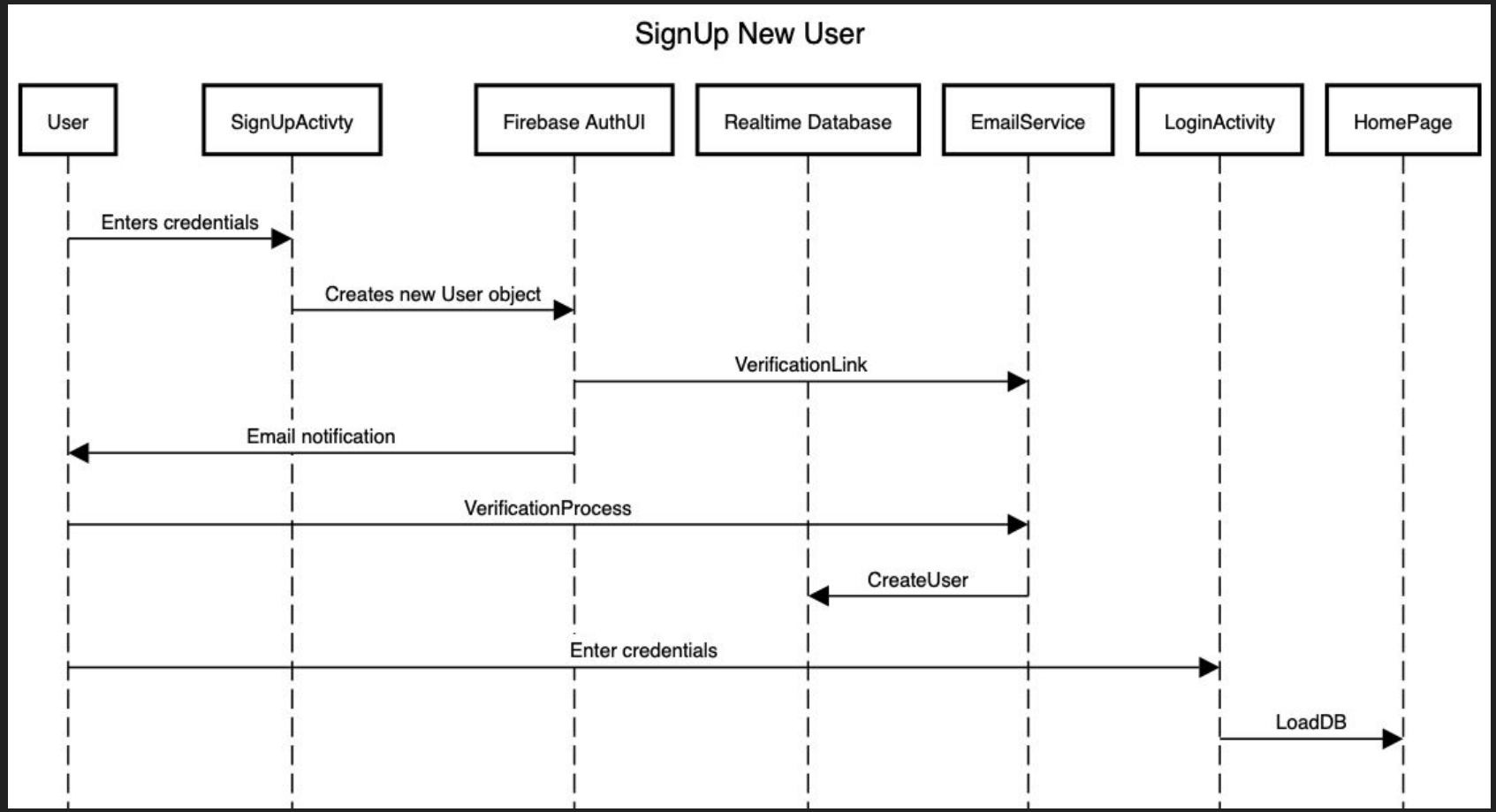
ACTIVITY DIAGRAM



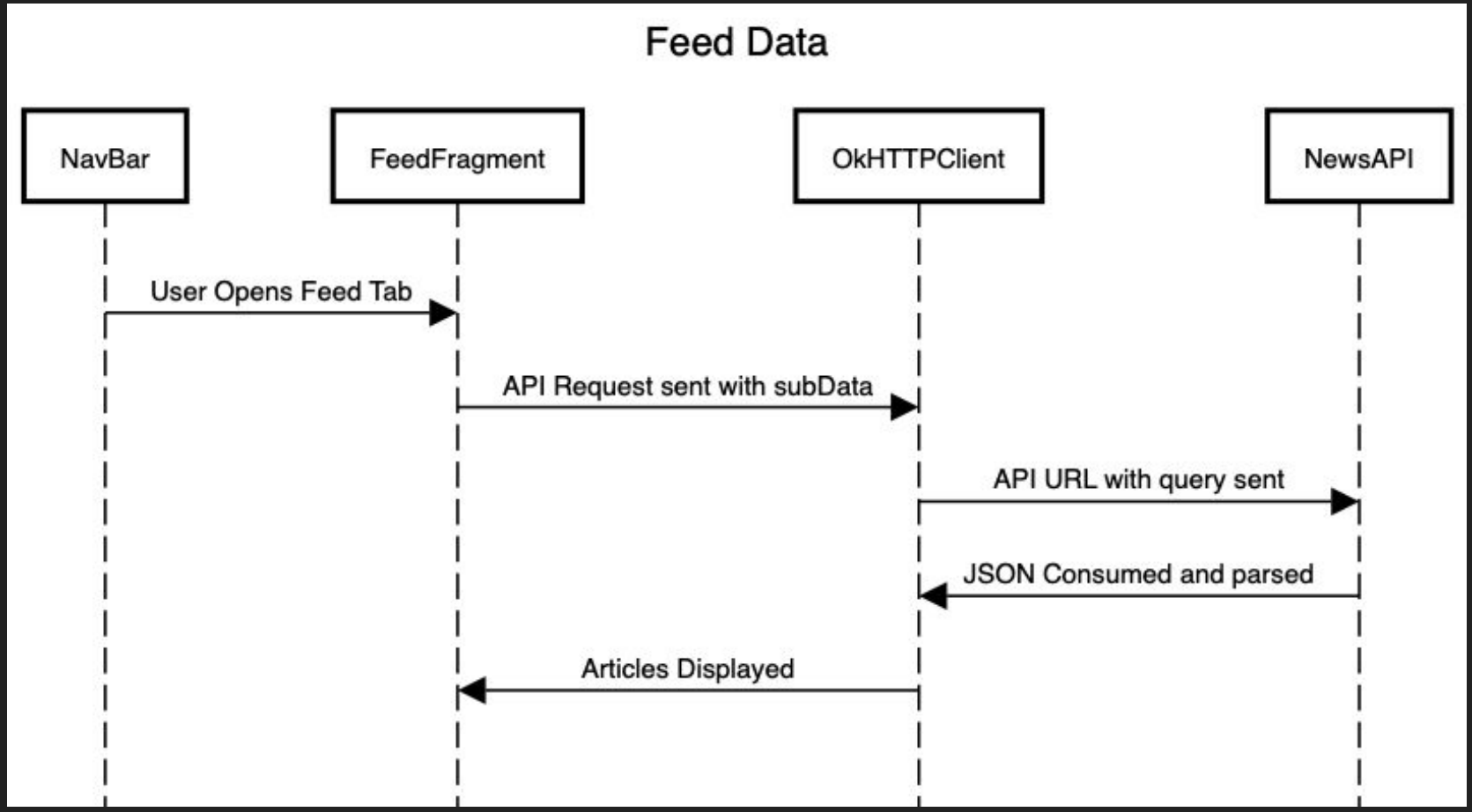
SEQUENCE DIAGRAMS



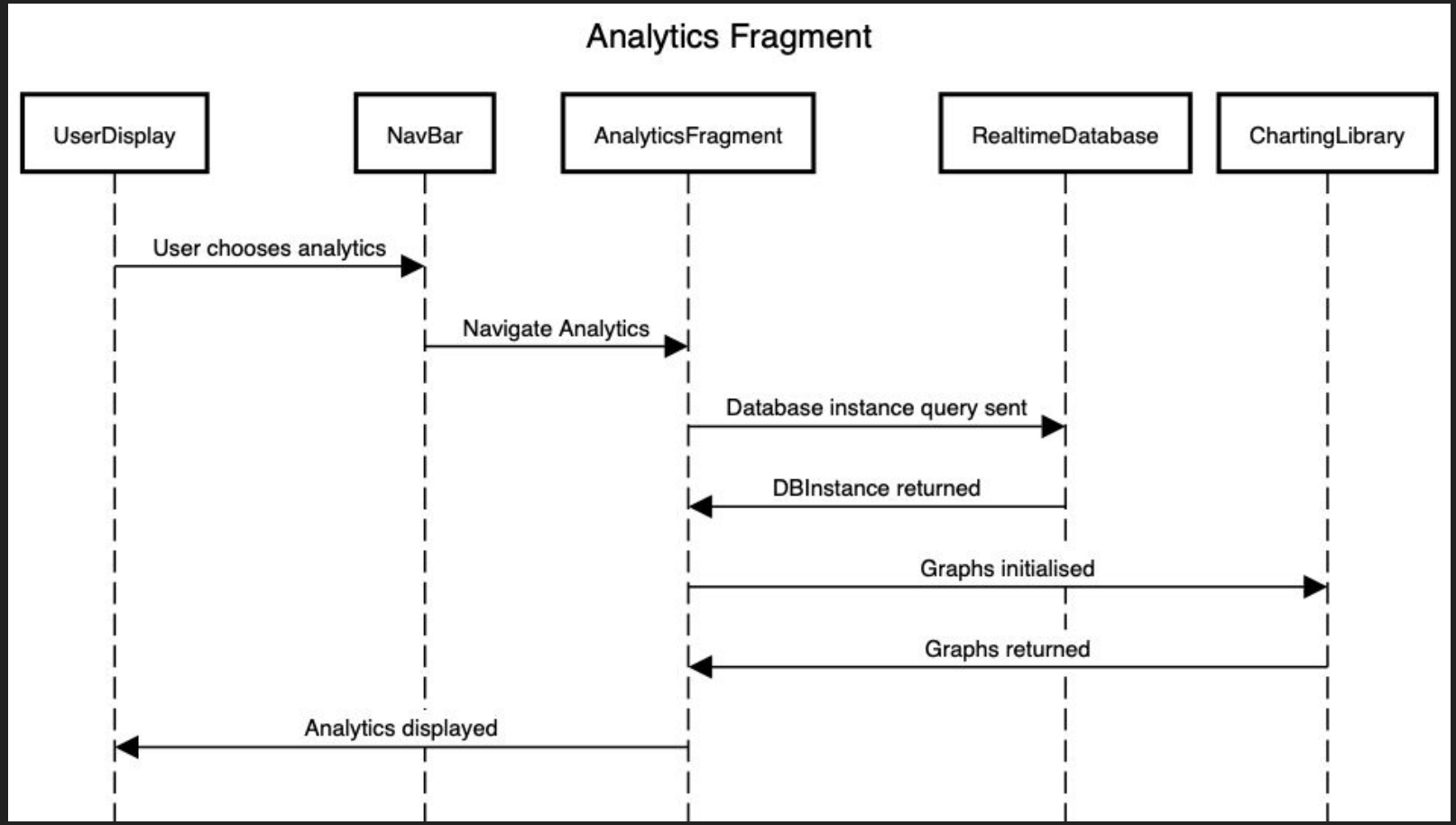
SEQUENCE DIAGRAMS



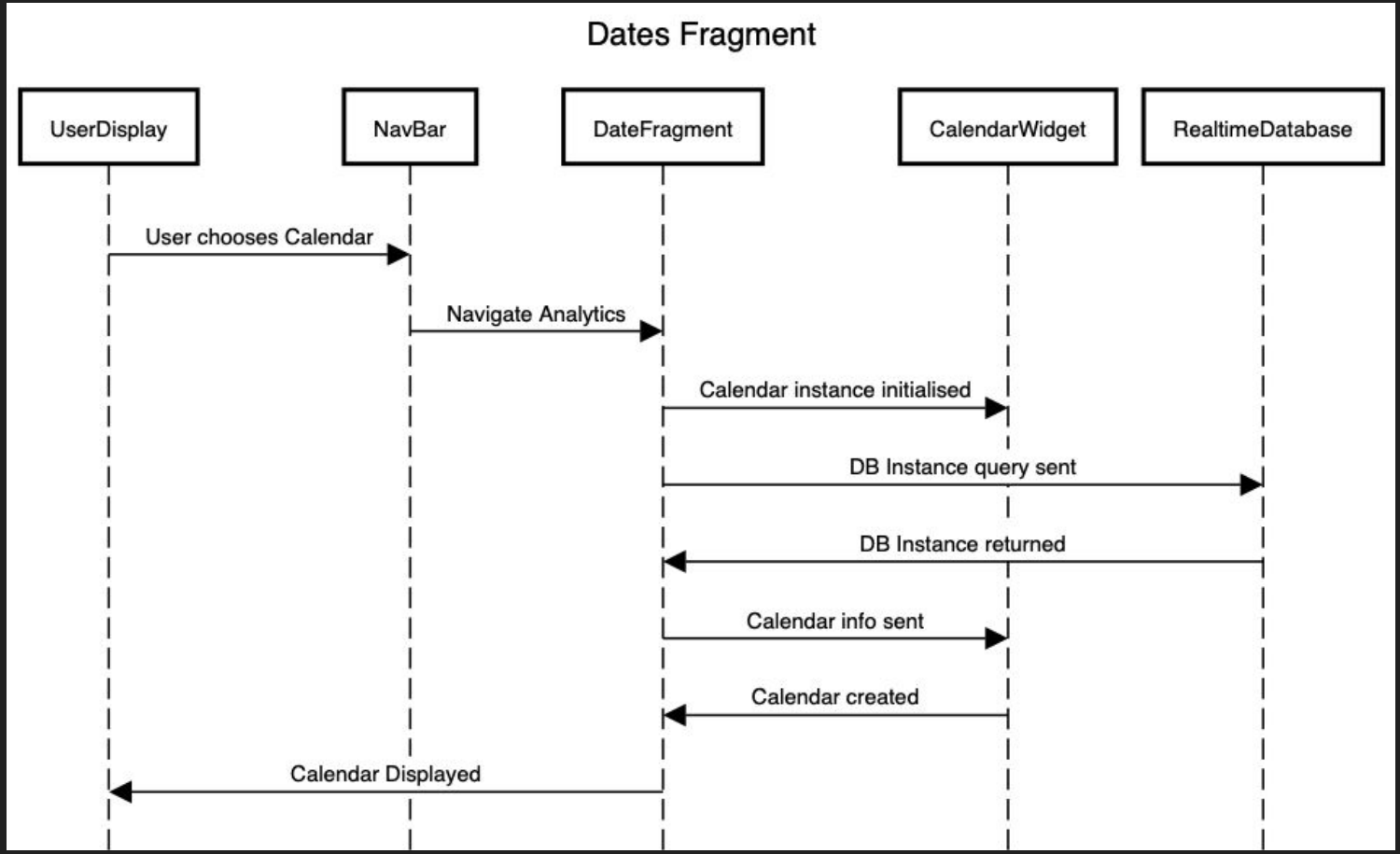
SEQUENCE DIAGRAMS



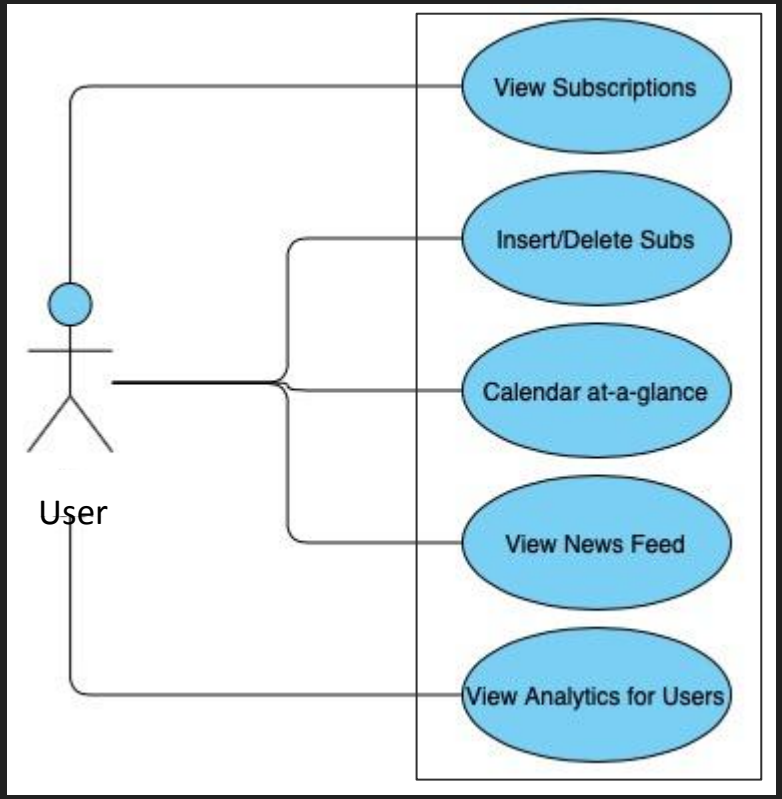
SEQUENCE DIAGRAMS



SEQUENCE DIAGRAMS



USE-CASE DIAGRAM



CODE SNIPPETS (1)

Login Authentication

```
mAuth = FirebaseAuth.getInstance();
mUser = FirebaseAuth.getInstance().getCurrentUser();
mAuthListner = new FirebaseAuth.AuthStateListener() {
    @Override
    public void onAuthStateChanged(@NonNull FirebaseAuth firebaseAuth) {
        if (mUser != null) {
            Intent intent = new Intent( packageContext: LoginActivity.this, HomeFragment.class);
            intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
            startActivity(intent);
        } else {
            Log.d(TAG,  msg: "AuthStateChanged:Logout");
        }

    }
};
```

Async Task Outline for Db query & RecyclerView Populate

```
public class AsyncGetSubs extends AsyncTask<Void, View, Void> {
    @Override
    protected Void doInBackground(Void... voids) {
            FirebaseDatabase database = FirebaseDatabase.getInstance();
            DatabaseReference myRef = database.getReference( path: "subscriptions");
            myRef.addValueEventListener(new ValueEventListener() {
                @Override
                public void onDataChange(DataSnapshot dataSnapshot) {

                    for(DataSnapshot ds : dataSnapshot.getChildren()) {...}

                }
                @Override
                public void onCancelled(DatabaseError error) {...}
            });
        return null;
    }
    @Override
    protected void onProgressUpdate(View... v) { homeView.addView(v[0]); }
}
```

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

        LinearLayout scr = findViewById(R.id.memberlist);
        LayoutInflater inflater = LayoutInflater.from(addSubActivity.this);
        EditText memname = (EditText) findViewById(R.id.memberName);
        EditText memprice = (EditText) findViewById(R.id.subPriceText);
        String price = memprice.getText().toString();
        String name = memname.getText().toString();
        membersToAdd.add(new MembersFB(name));
        View inflatedLayout1 = inflater.inflate(R.layout.member_entry,  root: null,  attachToRoot: false);
        TextView setname = inflatedLayout1.findViewById(R.id.memberThingy);
        TextView setprice = inflatedLayout1.findViewById(R.id.membershare);
        setprice.setText("");
        setname.setText(name);
        scr.addView(inflatedLayout1);
        memname.setText("");
    }
});
```

Adding New Members

CODE SNIPPETS (2)

```
List<Entry> entries = new ArrayList<~>();
for (int i=0;i<12;i++) {
    entries.add(new Entry(i,money.get(i)));
}
LineDataSet dataSet = new LineDataSet(entries,  label: "Costs"); // add entries to dataset
dataSet.setLineWidth(2f);
dataSet.setColor(R.color.colorPrimary);
dataSet.setMode(LineDataSet.Mode.CUBIC_BEZIER);
dataSet.setCubicIntensity(0.2f);
dataSet.setDrawFilled(true);
dataSet.setFillColor(R.color.colorPrimary);
dataSet.setFillAlpha(80);
Legend legend = chart.getLegend();
legend.setEnabled(false);
LineData lineData = new LineData(dataSet);
XAxis xAxis = chart.getXAxis();
String [] months={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sept","Oct","Nov","Dec"};
xAxis.setValueFormatter(new IndexAxisValueFormatter(months));
xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
chart.getDescription().setEnabled(false);
chart.setData(lineData);
chart.invalidate();
```

Configuring LineGraph

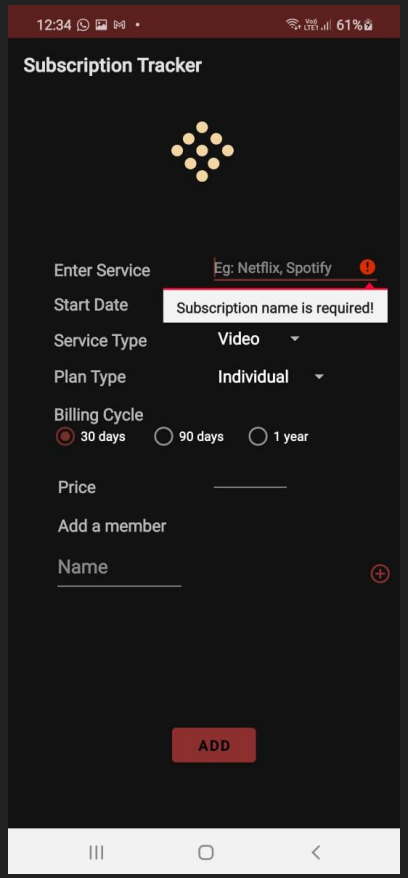
Adding Calendar Events

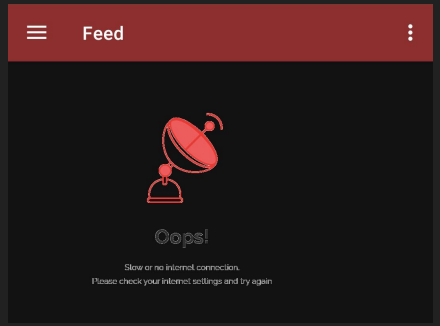
Fetching News Articles

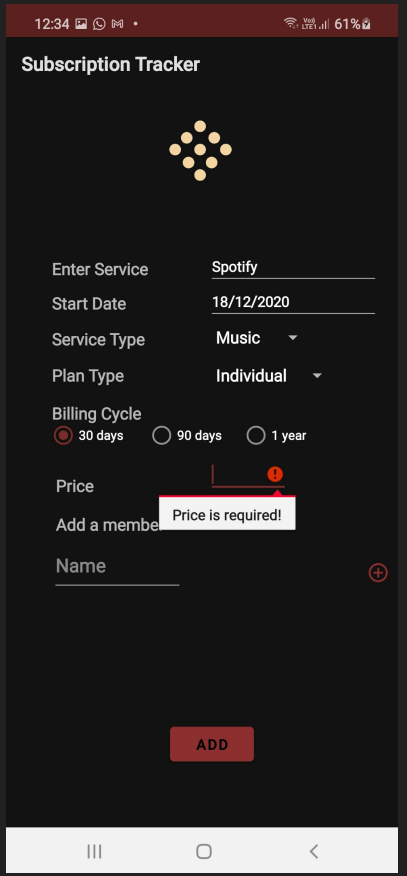
```
int day, month, year;
day = c.get(Calendar.DATE);
month = c.get(Calendar.MONTH);
year = c.get(Calendar.YEAR);
System.out.println(subscription.name + ", " + day + ", " + month + ", " + year);
calendar.set(year, month, day);
events.add(new EventDay(calendar, getResources().getDrawable(subscription.icon)));
calendars.add(calendar);
//+next 5 pay dates
for(int i=1;i<6;i++){
    Calendar x=Calendar.getInstance();
    x.set(year,month,day);
    x.add(Calendar.DAY_OF_MONTH, amount: i*subscription.billing_cycle);
    events.add(new EventDay(x, getResources().getDrawable(subscription.icon)));
    calendars.add(x);
}
////////
CalendarView calendarView = getActivity().findViewById(R.id.calendarView);
calendars.add(calendar);
calendarView.setHighlightedDays(calendars);
calendarView.setEvents(events);
```

```
for(int i=0;i<topics.length-1;i++)
{
    System.out.println("for "+topics[i]+" get articles");
    String url="https://newsapi.org/v2/everything?qInTitle="+topics[i]+"&from="+today+"&to="+past+"&lang
    try {
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder()
                .url(url)
                .build();
        Response responses = null;
        try {
            responses = client.newCall(request).execute();
        } catch (IOException e) {
            e.printStackTrace();
        }
        jsonData = responses.body().string();
        Jobject = new JSONObject(jsonData);
        Jarray = Jobject.getJSONArray( name: "articles");
        //get individual articles data
        for (int j = 0; j < Jarray.length(); j++) {...}
    }catch(Exception e) {
        e.printStackTrace();
    }
```

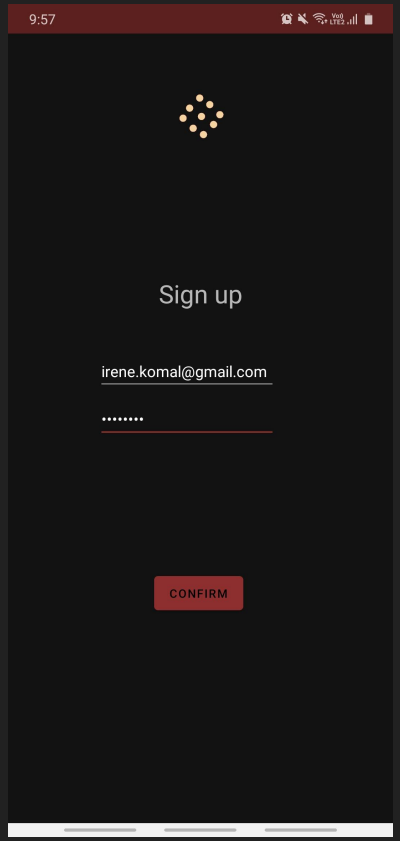
TESTING

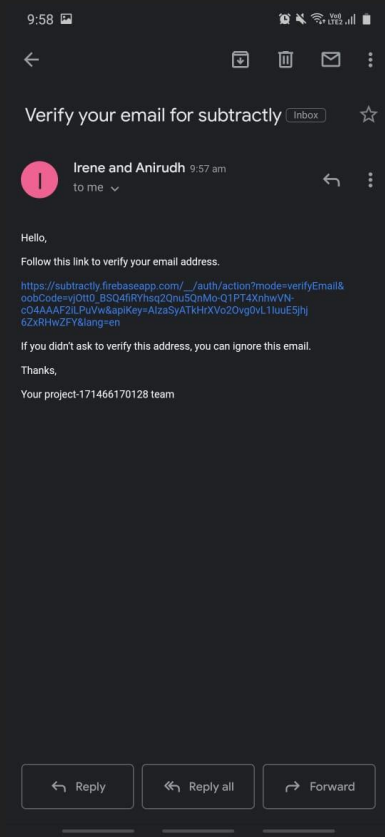

12:34
61%
Subscription Tracker
Enter Service
Eg: Netflix, Spotify
Start Date
Subscription name is required!
Service Type
Video
Plan Type
Individual
Billing Cycle
30 days
90 days
1 year
Price
Add a member
Name
ADD

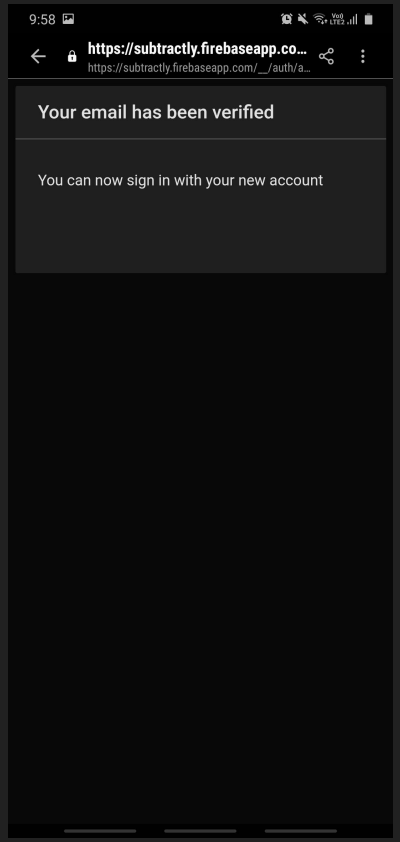

Feed
Oops!
Slow or no internet connection.
Please check your internet settings and try again


12:34
61%
Subscription Tracker
Enter Service
Spotify
Start Date
18/12/2020
Service Type
Music
Plan Type
Individual
Billing Cycle
30 days
90 days
1 year
Price
Price is required!
Add a member
Name
ADD

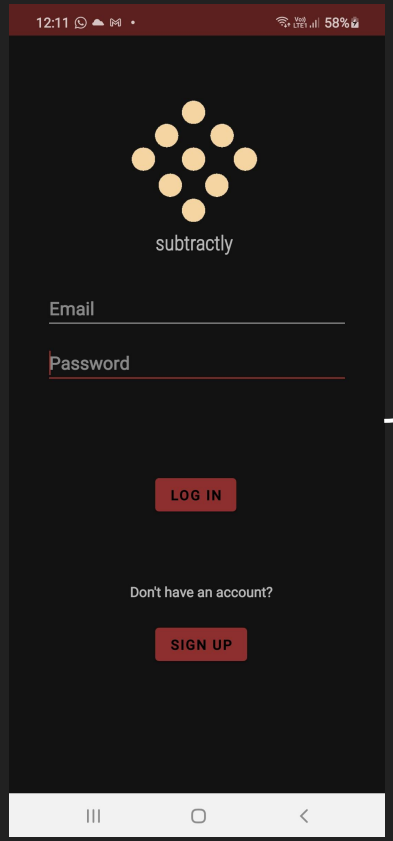
SIGN UP


9:57
Sign up
irene.komal@gmail.com
CONFIRM


9:58
Verify your email for subtractly
Inbox
Irene and Anirudh 9:57 am
to me
Hello,
Follow this link to verify your email address.
https://subtractly.firebaseapp.com/__/auth/action?mode=verifyEmail&oobCode=vjOtt0_BSQ4fiRYhsq2Qnu5QnMo-Q1PT4XnhwVN-c04AAAF2iLPuVw&apiKey=AIzaSyATkHrXVo2Ovg0vL1IuuE5jhj6ZxRHwZFY&lang=en
If you didn't ask to verify this address, you can ignore this email.
Thanks,
Your project-171466170128 team
Reply
Reply all
Forward

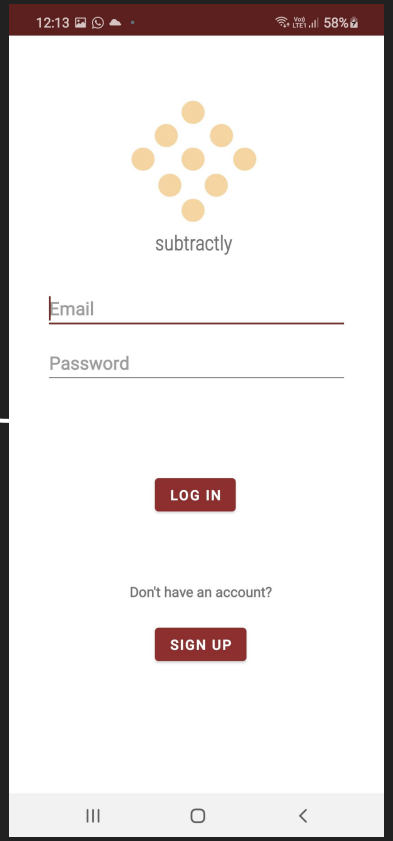

9:58
https://subtractly.firebaseapp.co...
https://subtractly.firebaseapp.com/__/auth/a...
Your email has been verified
You can now sign in with your new account

SCREENSHOTS

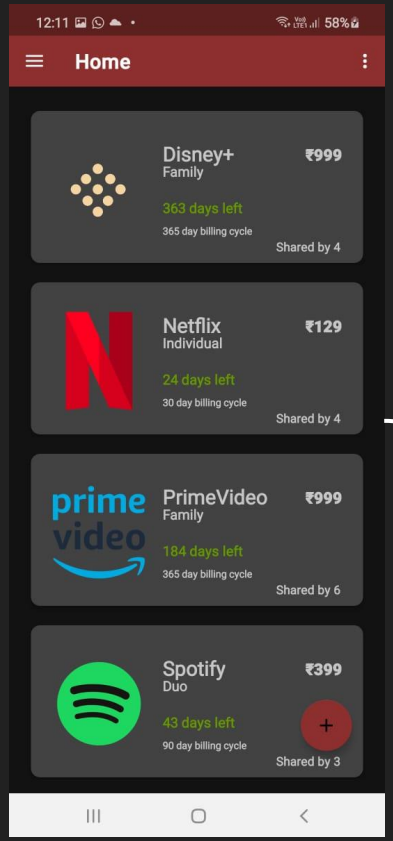


FireBase AuthUI

Fragments



SCREENSHOTS



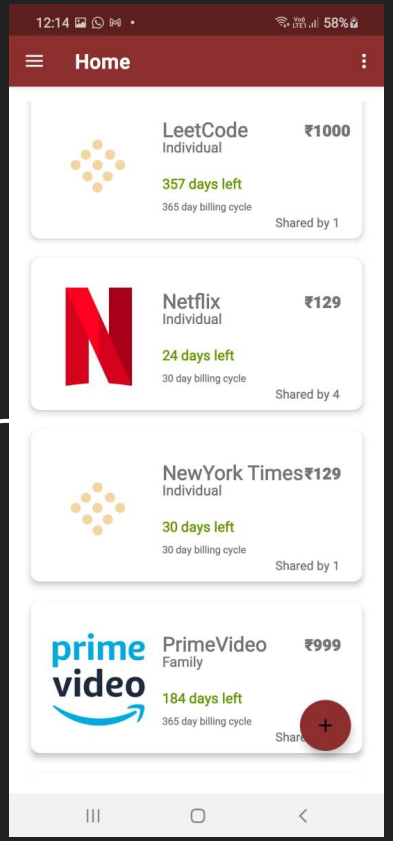
RealtimeDatabase

Fragments

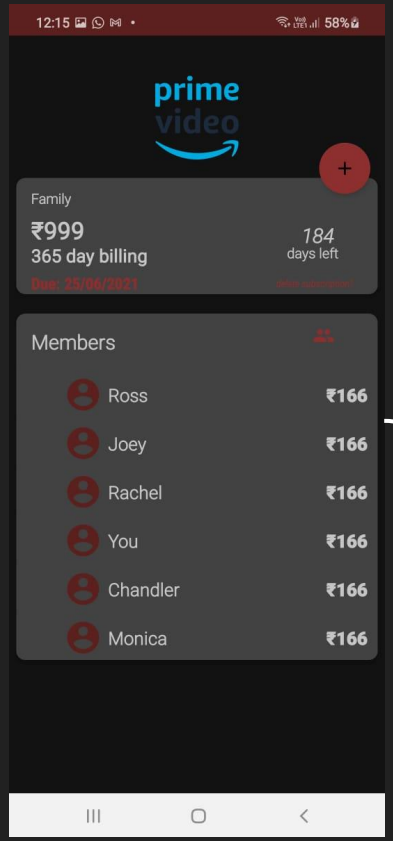
CardView

RecyclerView

Adapters



SCREENSHOTS



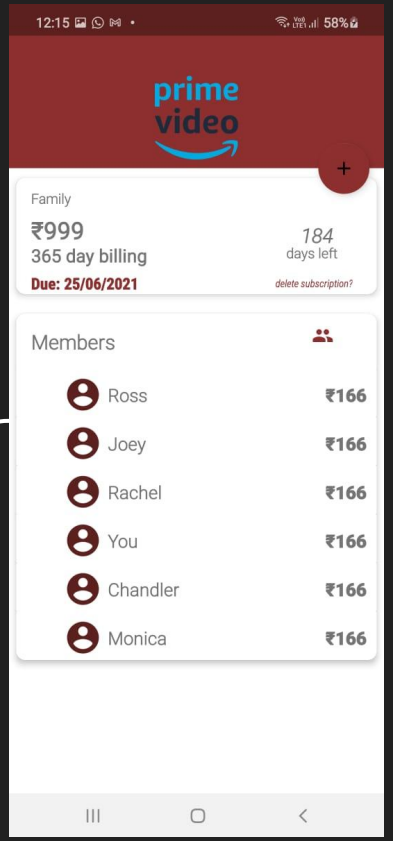
RealtimeDatabase

Fragments

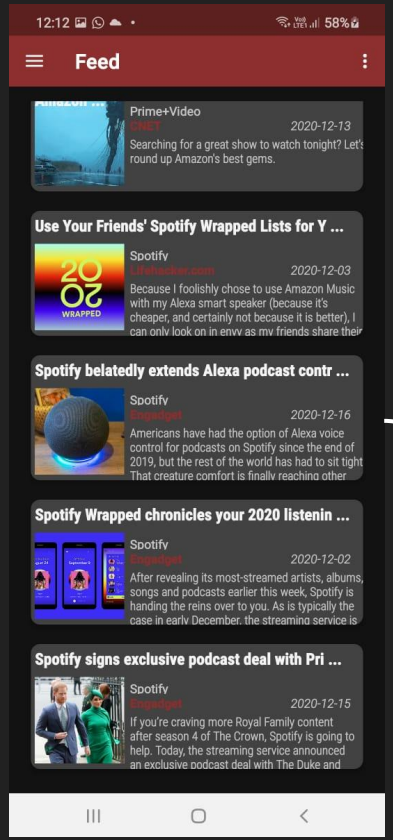
CardView

ScrollView

Dialogs



SCREENSHOTS



OkHTTP Client

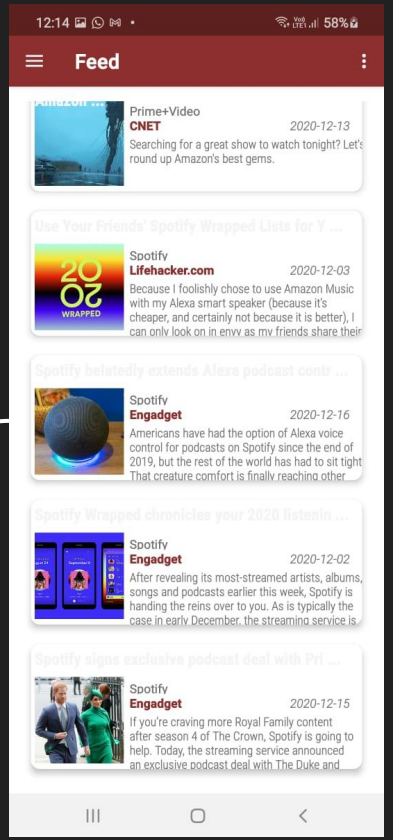
NewsAPI

Fragments

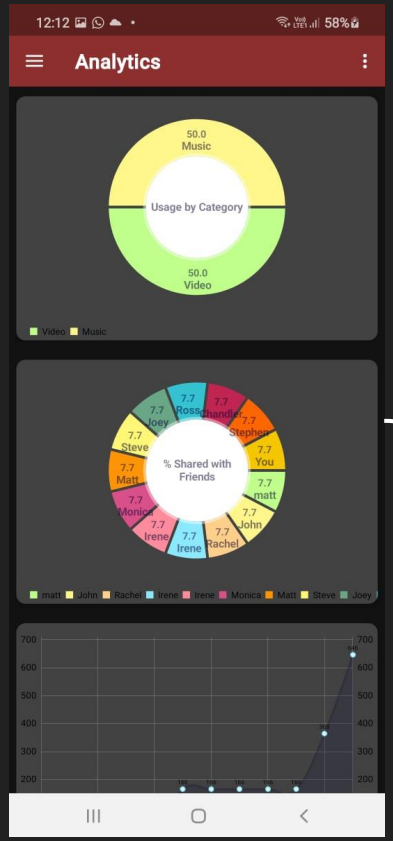
CardView

NavBar

WebBrowser



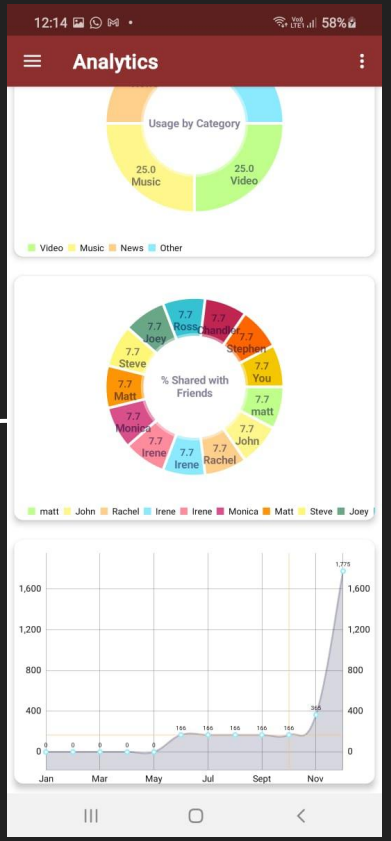
SCREENSHOTS



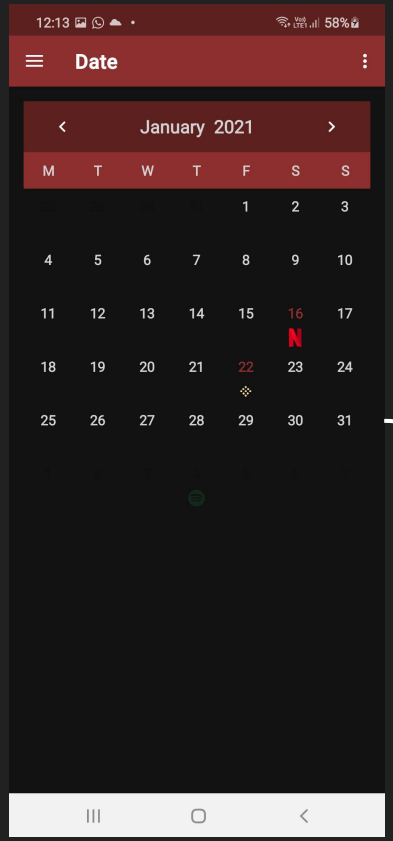
Realtime Database

Fragments

AndroidChart



SCREENSHOTS

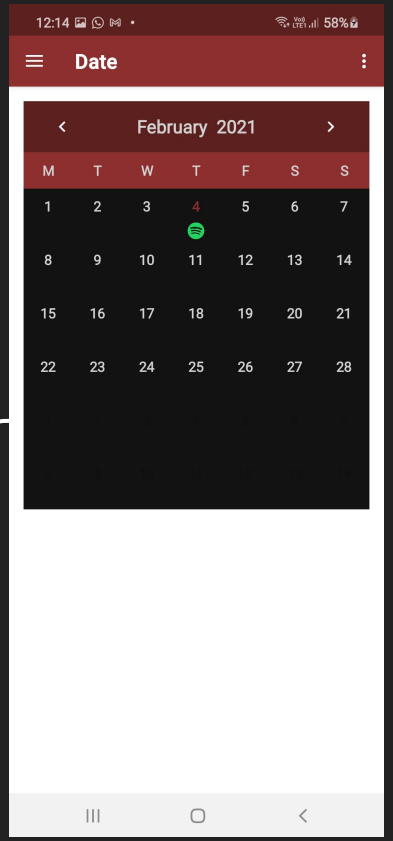


Realtime Database

Fragments

MaterialCalendarView

NavBar




DEMO