Expense Tracker

Increment 4

Team Members

1) Apuroop Reddy
2) Blake Simpson
3) Vang Lia

1) Story of the Project

Main story of the project is to help individuals keep track of their expenses and bills.

Certain individuals such as tenants, store owners or any person who spends more amount of money or receive.

Most of them find it difficult to manage or track their expenses when they need them the most to review it accordingly.

All of them face this when they want to calculate or track their expenses week or month or year wise.

In all the areas where several stores, individuals are present without a proper application to check their income or spending's on items.

By tracking the expenses people get to know where they are spending their money more and can appropriately adjust the spending on more necessary items.

I see lot of people, including ourselves difficult to keep track of our spending's on items weekly or monthly and cannot compare those expenses manually when we need to see the difference on spending's as time goes on.

2) **Data for the Project**

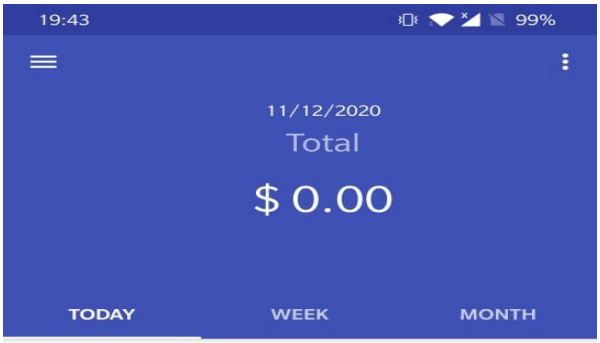
Data set is about people who want to track their expenditures and income. Sampled data set is self-curated and some of the reference is taken from city wise daily,weekly,monthly expenditure of numerous families. Nobody will be oversampled or under sampled. There is no identifiable information that is at risk by disclosing above dataset information as it is self-curated.

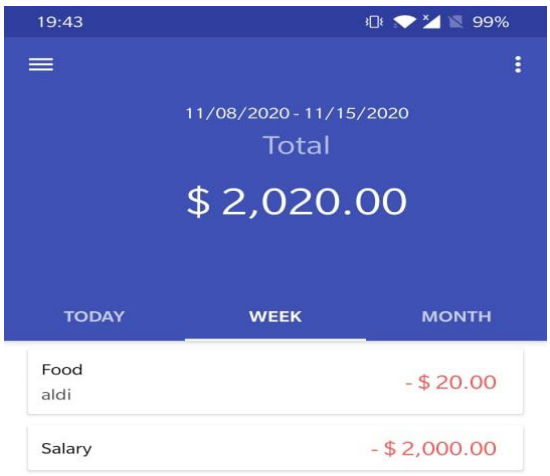
Mainly recorded events are expenditure and income, this data set contains targeted events ,activities , behaviors like type of items(General,clothes,entertainment,fuel,sports,travel, food) on which money is spent on a particular day and is recorded.

These activities take place depending on the need of a certain person. Data collected is real time depending on the person and all the expenses and income are tracked. As data entered and stored is real time it becomes old as time goes on and new data is added. Generalization can be done on a daily, weekly, monthly basis.

As it is self-curated data, we can assume that activity took place with one of the people working on this project. If possible, the whole data can be compared or analyzed with other data sets of the individuals that are nearer to him/her. It can be used as a local or regional or national data set taking some of the dataset variables into account.
As written data on paper seemed inappropriate and hard to track as we are talking about long term tracking.

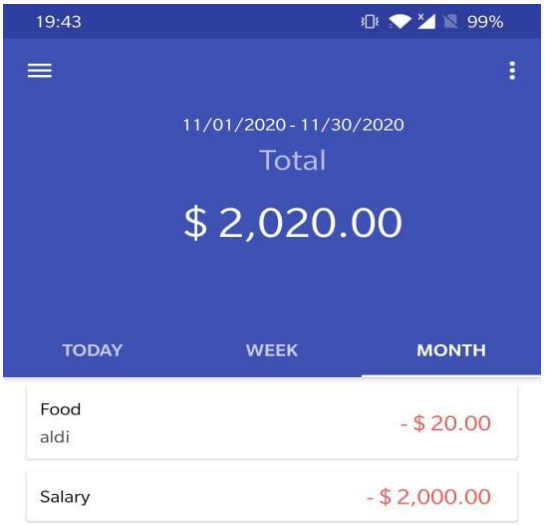
3) **Working Screens**

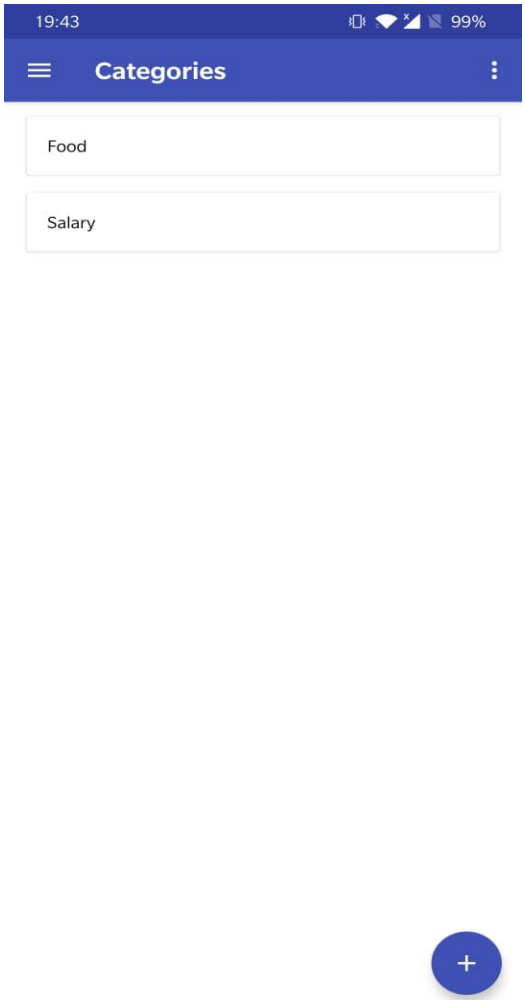




<u>3.1</u> Daily expenses page

<u>3.2</u> Weekly expenses list page

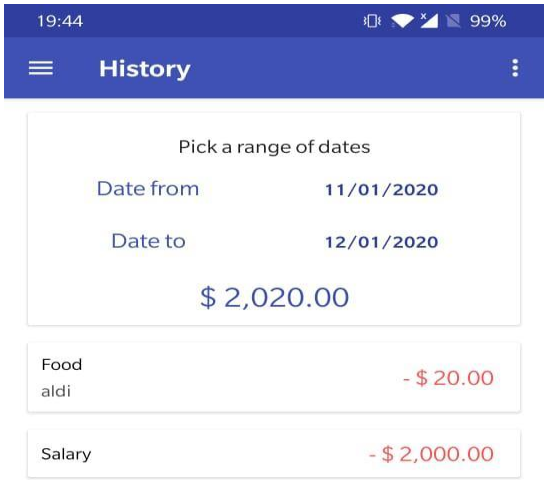




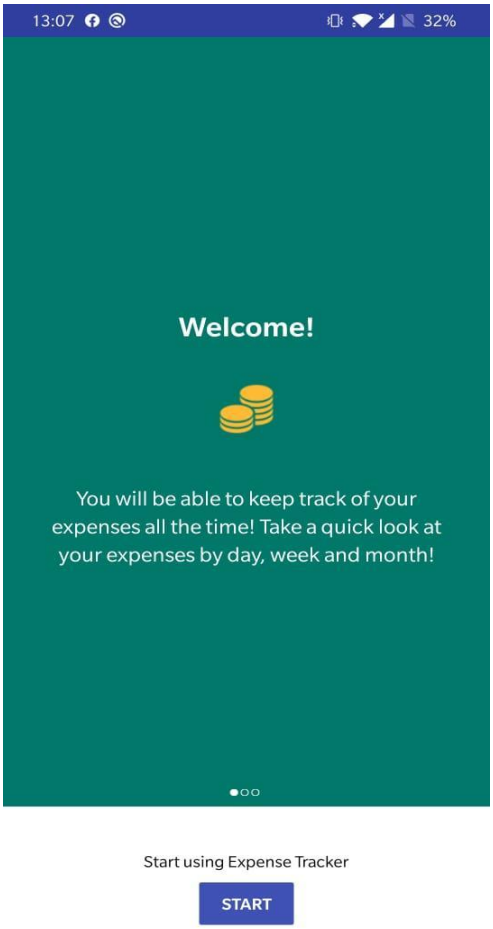
3.3 Monthly expenses page

3.4 Categories page

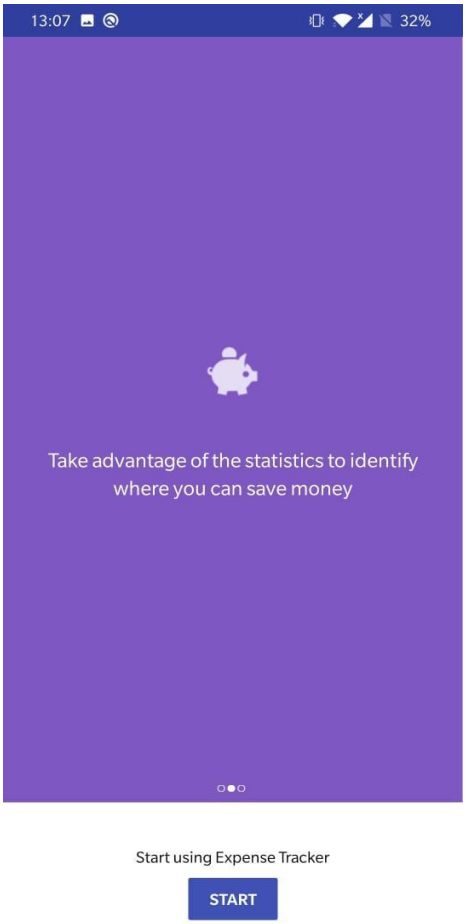
3.5 Statistics page(graph view)



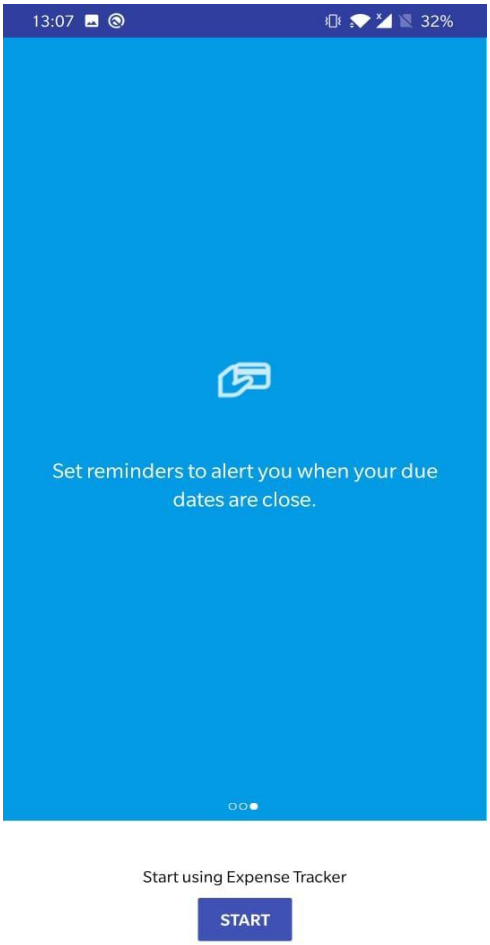
3.6 History Page



3.7.a Welcome Page

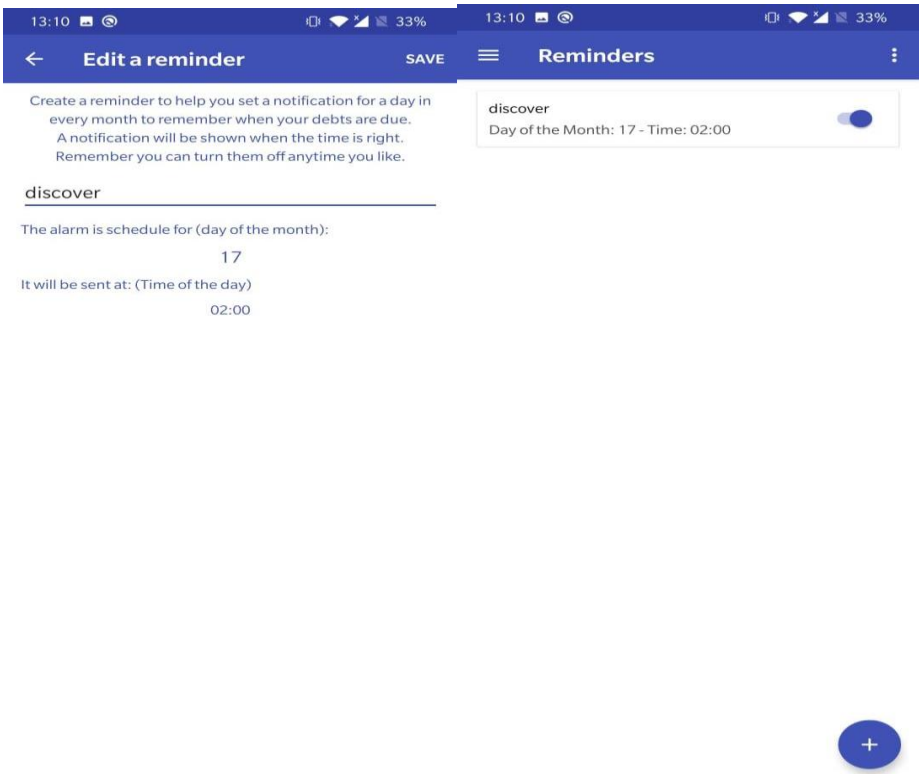


3.7.b Welcome page

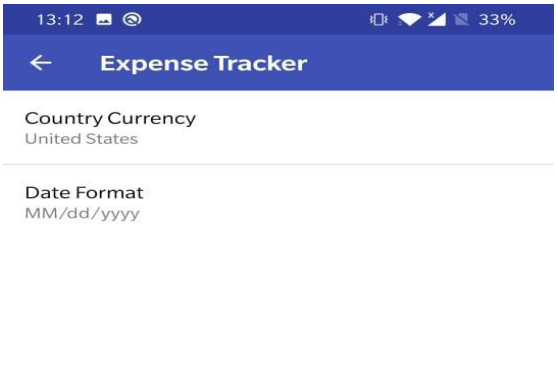


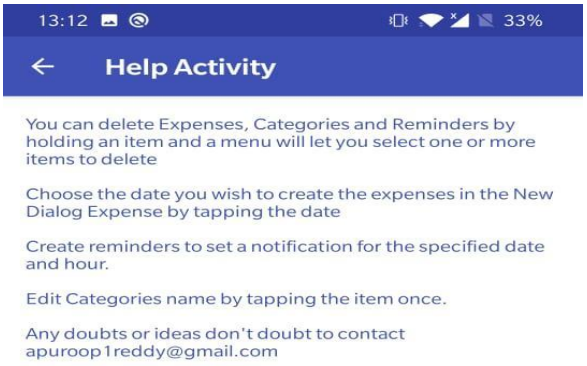
3.7.c Welcome Page

3.8 Reminders Activity

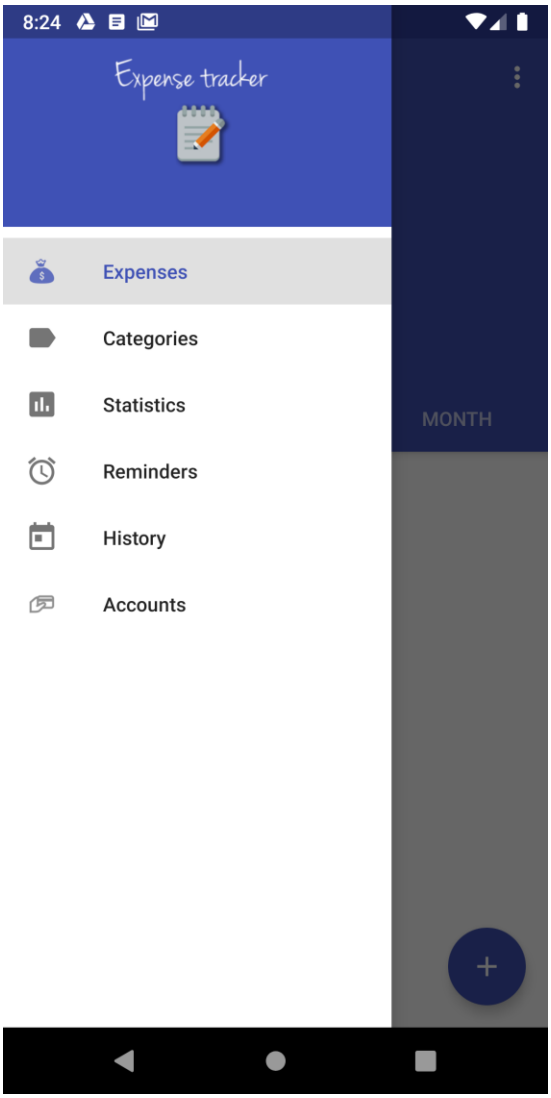


3.8.a Creating Reminders

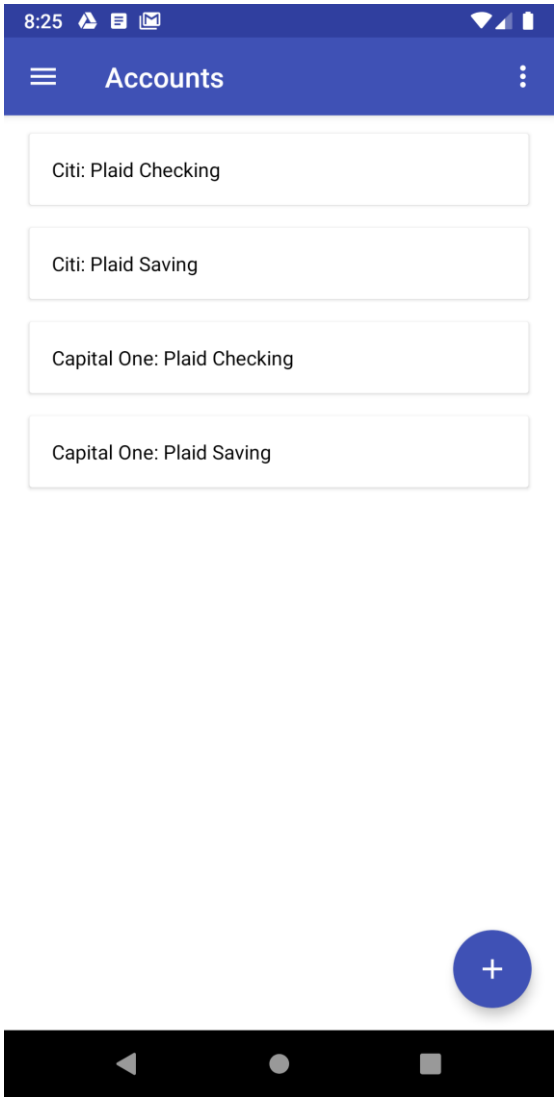




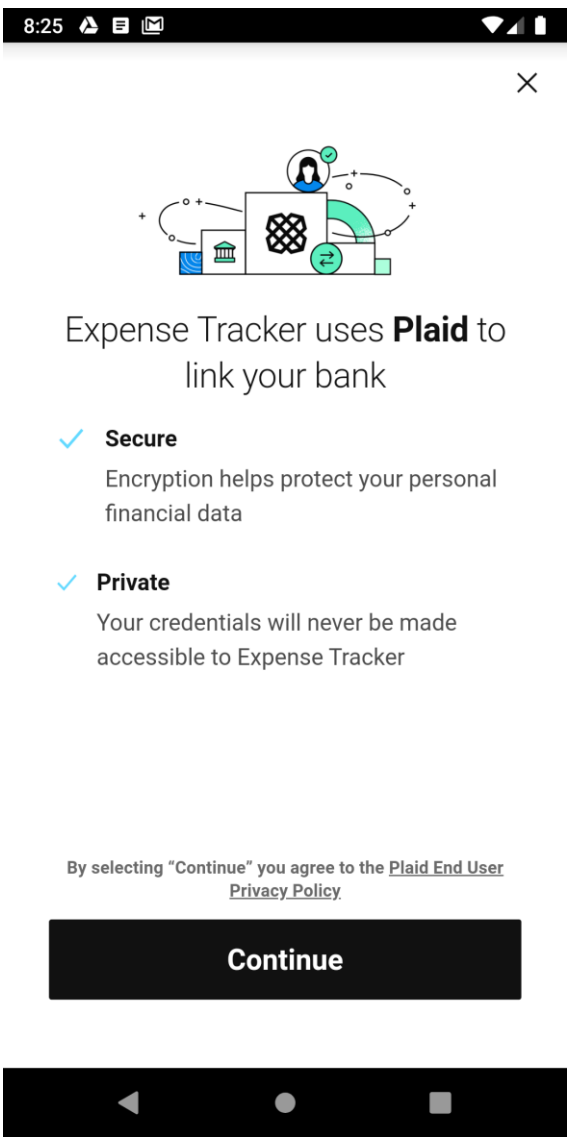
3.9 Settings and Help Screen

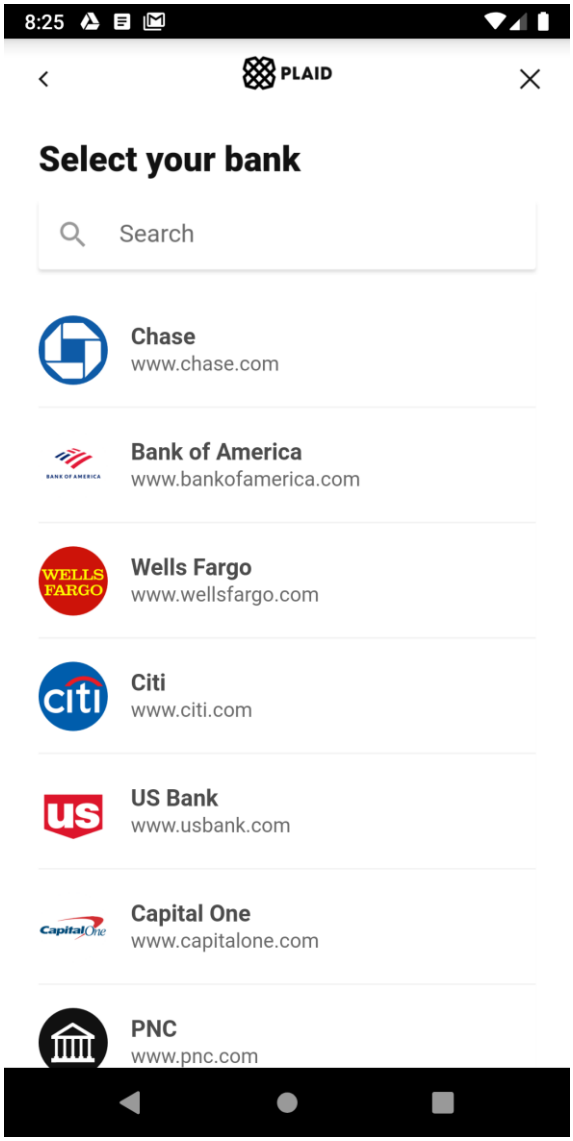


6. a Account page view

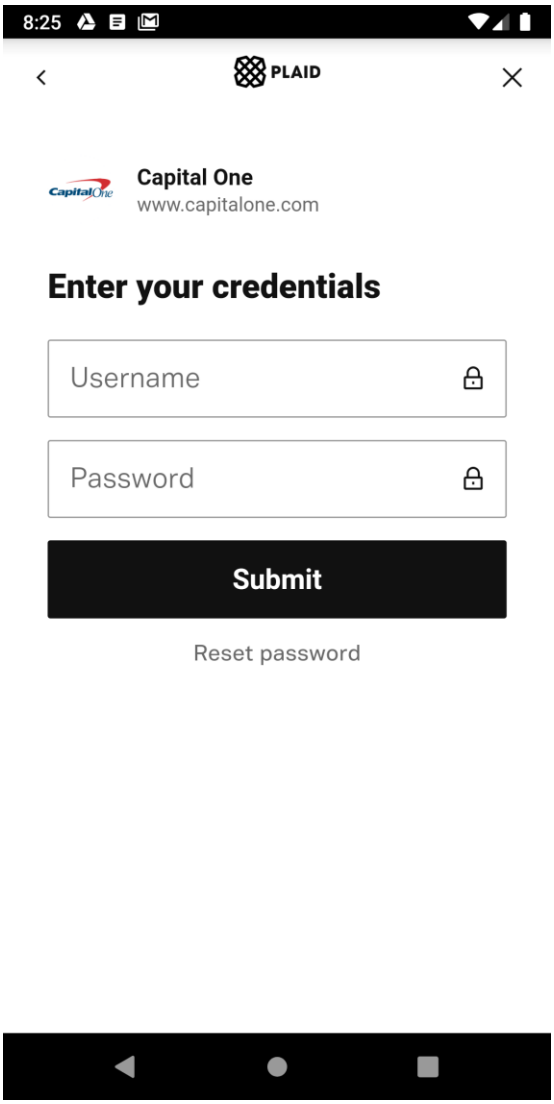


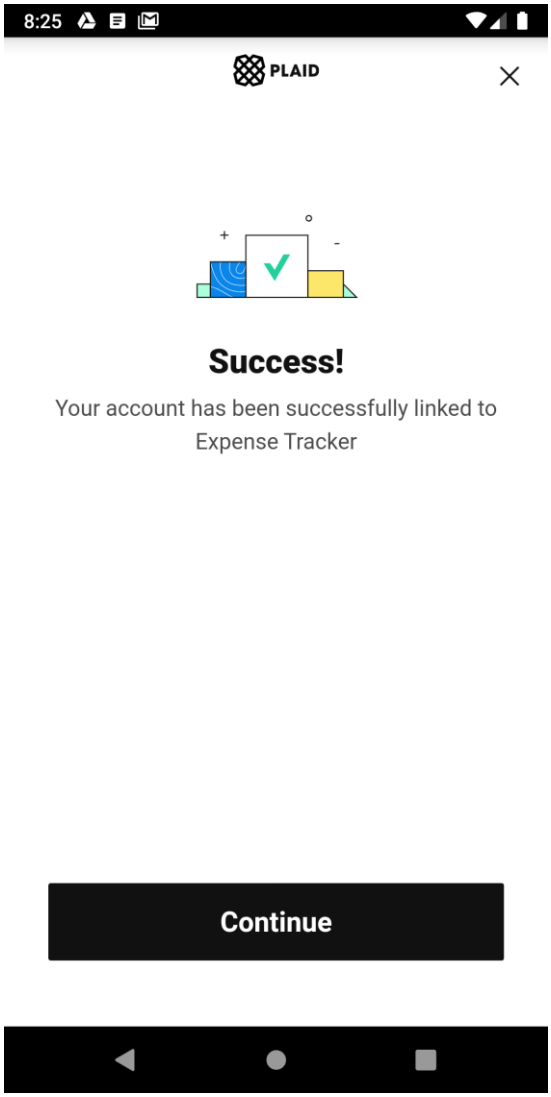
6. b Add account





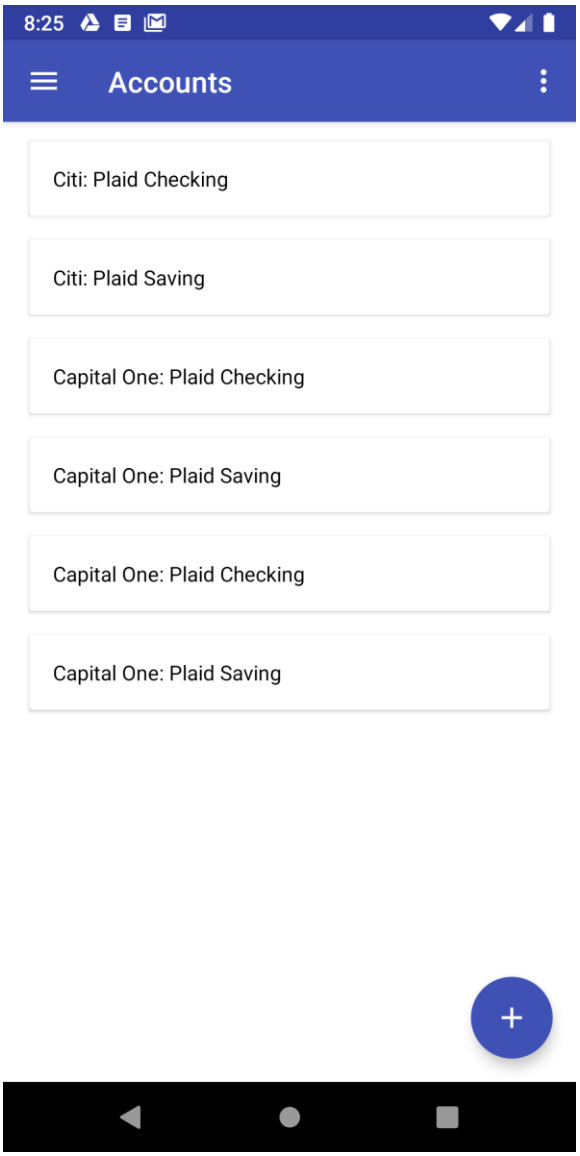
6. c Link any bank account



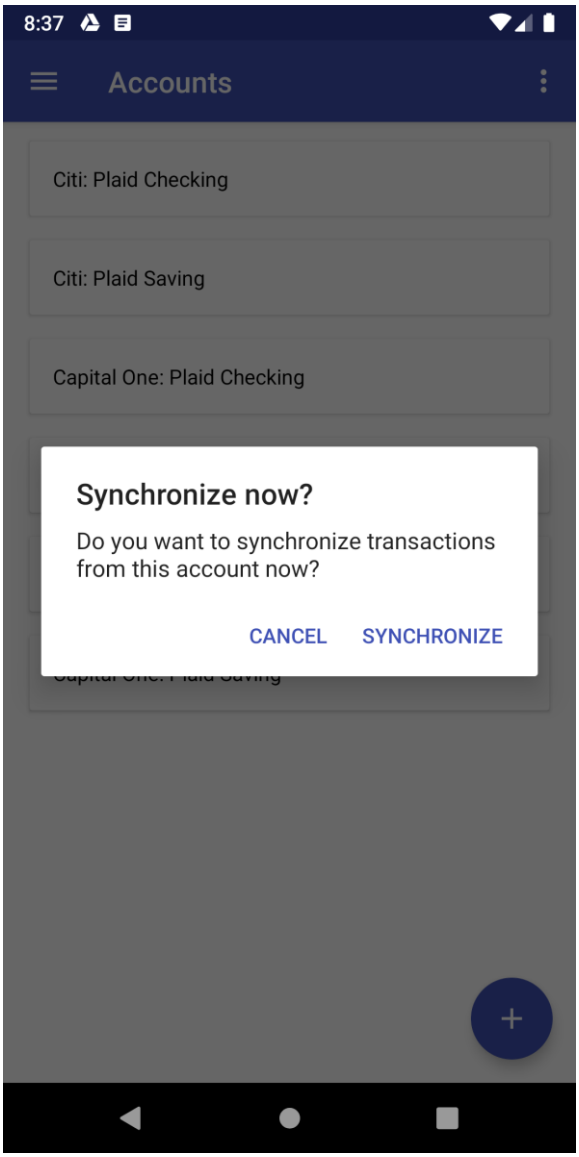


6. d Enter user's bank information

6. e Account as been linked successfully



6. f New bank account added to screen



6. g Sync transaction to detail page

4) Improvements

a) Interactive welcome pages have been added for users who login into the app for the first time.
b) Reminders functionality has been included to set alarms for scheduled payments.
c) Settings option contains functionality to change currency and date format.
d) Help page provides information on what can be done using this application.
e) integrated plaid api to connect to bank accounts and synchronise data from multiple bank accounts.

5) Important Code Snippets

a) Using MpPieChartLibrary api's (Rf 11.e) to view statistics or graphs

```
private BarChart bcCategories;

private List<Category> mCategoryList;
private SelectDateFragment selectDateFragment;

public static StatisticsFragment newInstance() { return new StatisticsFragment(); }

public StatisticsFragment() {
}

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


@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View rootView = inflater.inflate(R.layout.fragment_statistics, container,  attachToRoot: false);
    pcCategories = (PieChart) rootView.findViewById(R.id.pc_categories);
    bcCategories = (BarChart) rootView.findViewById(R.id.bc_categories);
    tvPcCategoriesEmpty = (TextView)rootView.findViewById(R.id.tv_bar_chart_category_empty);
    tvBcCategoriesEmpty = (TextView)rootView.findViewById(R.id.tv_pie_categories_chart_empty);
    selectDateFragment = (SelectDateFragment)getChildFragmentManager().findFragmentById(R.id.select_date_fragment);
    selectDateFragment.setSelectDateFragment(this);
    return rootView;
}

@Override
public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    mMainActivityListener.setMode(MainActivity.NAVIGATION_MODE_STANDARD);
    mMainActivityListener.setTitle("Statistics");
```

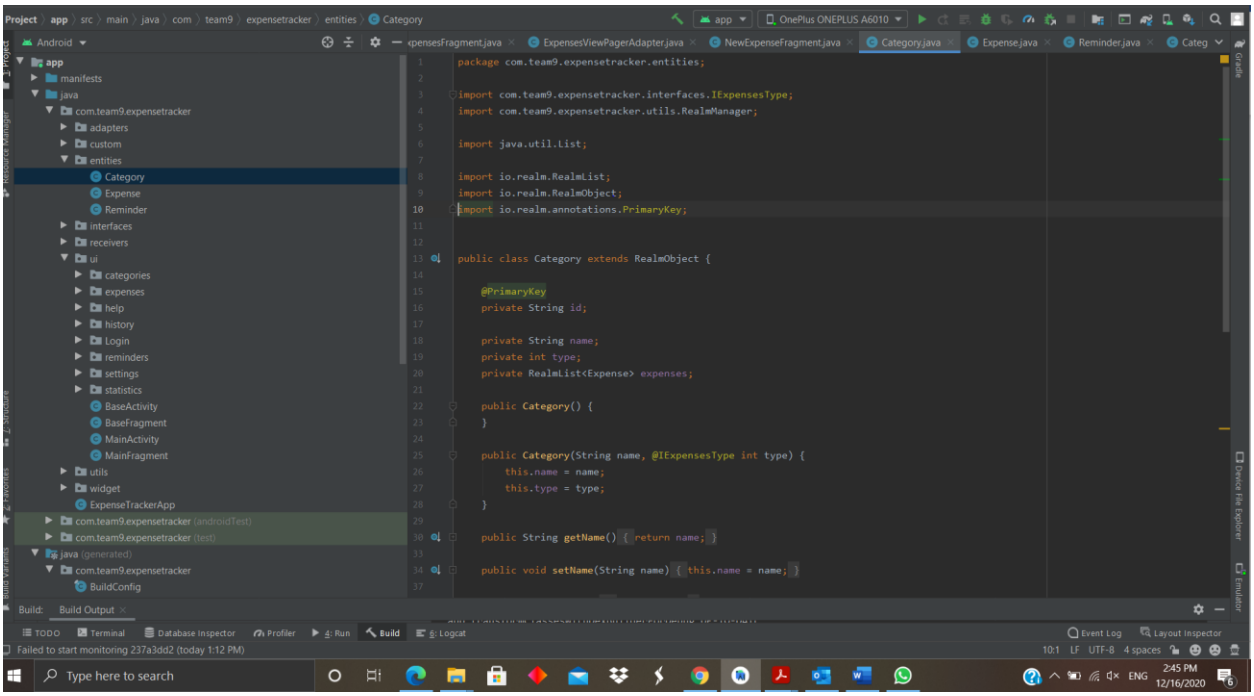
b) Welcome page Layouts will be called from below snippet.

```
private static final int NUM_PAGES = 3;

public WelcomePagerAdapter(FragmentManager fm) { super(fm); }

@Override
public Fragment getItem(int position) {
    WelcomePage tp = null;
    switch (position) {
        case 0:
            tp = WelcomePage.newInstance(R.layout.layout_welcome_first);
            break;
        case 1:
            tp = WelcomePage.newInstance(R.layout.layout_welcome_second);
            break;
        case 2:
            tp = WelcomePage.newInstance(R.layout.layout_welcome_third);
            break;
    }
    return tp;
}

@Override
public int getCount() { return NUM_PAGES; }
```

c) For saving the inserted expense.

```
private void onSaveExpense() {
    if (mCategoriesSpinnerAdapter.getCount() > 0 ) {
        if (!Util.isEmptyField(etTotal)) {
            Category currentCategory = (Category) spCategory.getSelectedItem();
            String total = etTotal.getText().toString();
            String description = etDescription.getText().toString();
            if (mUserActionMode == IUserActionsMode.MODE_CREATE) {
                RealmManager.getInstance().save(new Expense(description, selectedDate, mExpenseType, currentCategory, F
            } else {
                Expense editExpense = new Expense();
                editExpense.setId(mExpense.getId());
                editExpense.setTotal(Float.parseFloat(total));
                editExpense.setDescription(description);
                editExpense.setCategory(currentCategory);
                editExpense.setDate(selectedDate);
                RealmManager.getInstance().update(editExpense);
            }
            // update widget if the expense is created today
            if (DateUtils.isToday(selectedDate)) {
                Intent i = new Intent(getActivity(), ExpensesWidgetProvider.class);
                i.setAction(ExpensesWidgetService.UPDATE_WIDGET);
                getActivity().sendBroadcast(i);
            }
            getTargetFragment().onActivityResult(getTargetRequestCode(), Activity.RESULT_OK,  data: null);
            dismiss();
        } else {
            DialogManager.getInstance().showShortToast("Please enter a total");
        }
    } else {
        DialogManager.getInstance().showShortToast("Please create categories to start saving expenses");
    }
}
```

d) Entities folder contain all database related files.



e) Entities folder contain all accounts.

f) Plaid API

```
class PlaidHelper {
   private static final String CLIENT_ID = "";
   private static final String PUBLIC_KEY = "";
   private final PlaidClient client;

   PlaidHelper() {
       client = PlaidClient.newBuilder().clientId(CLIENT_ID)
               .sandboxBaseUrl().build();
   }

   private String getUserId() {
       Context context = ExpenseTrackerApp.getContext();
       SharedPreferences preferences = context

.getSharedPreferences(context.getString(R.string.plaid_file_key),
Context.MODE_PRIVATE);
       String userIdKey = context.getString(R.string.plaid_user_id_key);
       String userId = preferences.getString(userIdKey, null);
       if (userId != null) {
           return userId;
       }

       userId = UUID.randomUUID().toString();
       SharedPreferences.Editor editor = preferences.edit();
       editor.putString(userIdKey, userId);
       editor.apply();
       return userId;
   }
```

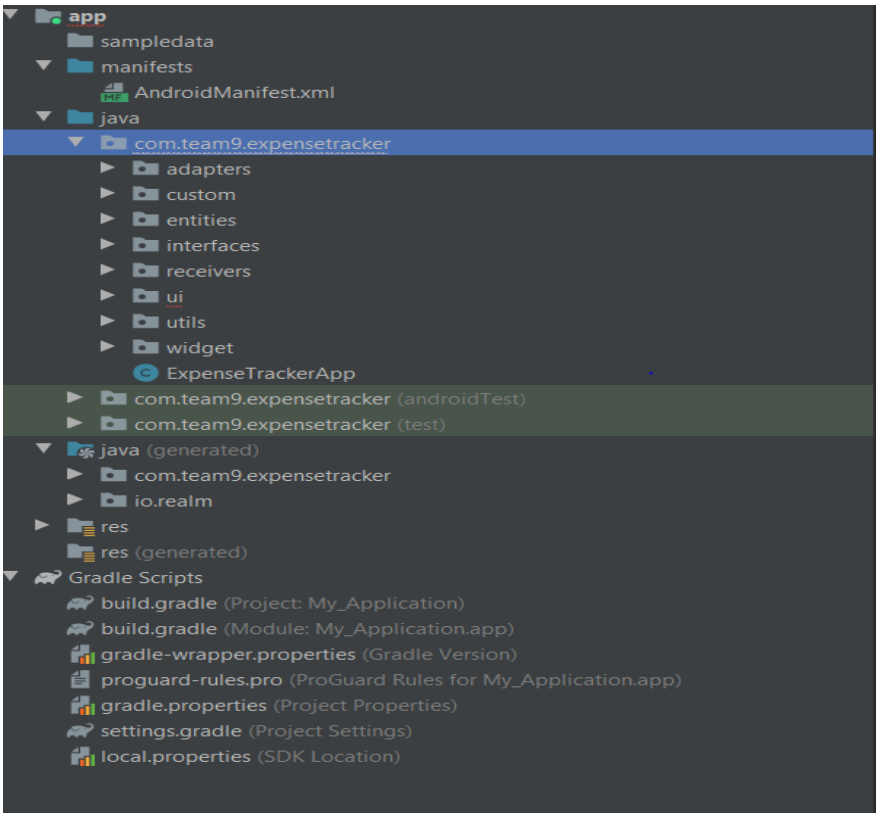
```
   CompletableFuture<LinkTokenConfiguration> getLinkToken() {
       return CompletableFuture.supplyAsync(() -> {
           try {
               Map<String, LinkTokenCreateRequest.SubtypeFilters>
accountFilters = new HashMap<>();
               accountFilters.put("depository", new
LinkTokenCreateRequest.SubtypeFilters(singletonList("all")));
               accountFilters.put("credit", new
LinkTokenCreateRequest.SubtypeFilters(singletonList("all")));

               LinkTokenCreateRequest request = new LinkTokenCreateRequest(
                       new LinkTokenCreateRequest.User(getUserId()),

ExpenseTrackerApp.getContext().getString(R.string.app_name),
                       Arrays.asList("auth", "transactions"),
singletonList("US"), "en")

.withAndroidPackageName(ExpenseTrackerApp.getContext().getPackageName())
                       .withAccountFilters(accountFilters);

               return new LinkTokenConfiguration.Builder()

.token(client.service().linkTokenCreate(request).execute().body().getLinkTok
en())
                       .build();
           } catch (IOException e) {
               e.printStackTrace();
               throw new UncheckedIOException(e);
           }
       });
   }

   CompletableFuture<String> getAccessToken(String publicToken) {
       return CompletableFuture.supplyAsync(() -> {
           try {
               return client.service().itemPublicTokenExchange(new
ItemPublicTokenExchangeRequest(publicToken))
                       .execute().body().getAccessToken();
           } catch (IOException e) {
               e.printStackTrace();
               throw new UncheckedIOException(e);
           }
       });
   }

   CompletableFuture<List<TransactionsGetResponse.Transaction>>
getTransactions(Account account) {
       String accessToken = account.getAccessToken();
       String plaidId = account.getPlaidId();
       return CompletableFuture.supplyAsync(() -> {
           try {
               Map<String, LinkTokenCreateRequest.SubtypeFilters>
accountFilters = new HashMap<>();
               accountFilters.put("depository", new
LinkTokenCreateRequest.SubtypeFilters(singletonList("all")));
```

```
                accountFilters.put("credit", new
LinkTokenCreateRequest.SubtypeFilters(singletonList("all")));

                List<TransactionsGetResponse.Transaction> transactions = new
ArrayList<>();

                TransactionsGetResponse body;
                do {
                    TransactionsGetRequest request = new
TransactionsGetRequest(accessToken,
                            Date.from(Instant.now().minus(60,
ChronoUnit.DAYS)),
                            Date.from(Instant.now()))
                            .withAccountIds(singletonList(plaidId))
                            .withCount(100)
                            .withOffset(transactions.size());

                    body =
client.service().transactionsGet(request).execute().body();

                    if (body == null) {
                        // error occurred in plaid
                        break;
                    }

                    transactions.addAll(body.getTransactions());
                } while (body.getTotalTransactions() < transactions.size());

                return transactions;
            } catch (IOException e) {
                e.printStackTrace();
                throw new UncheckedIOException(e);
            } catch (Exception e) {
                e.printStackTrace();
                throw new RuntimeException(e);
            }
        });
    }
}
```

6) Working Modules

Task 1: setting up the project

a) Create an android project with simple activity and name it.
b) Adding all the required library dependencies to build.gradle file as the project progresses.
c) Create a suitable project structure.(Rf 11.c)
d) Realm database is used for handling data persistence.



Project Structure

Task 2: Implementing UI for Each Activity

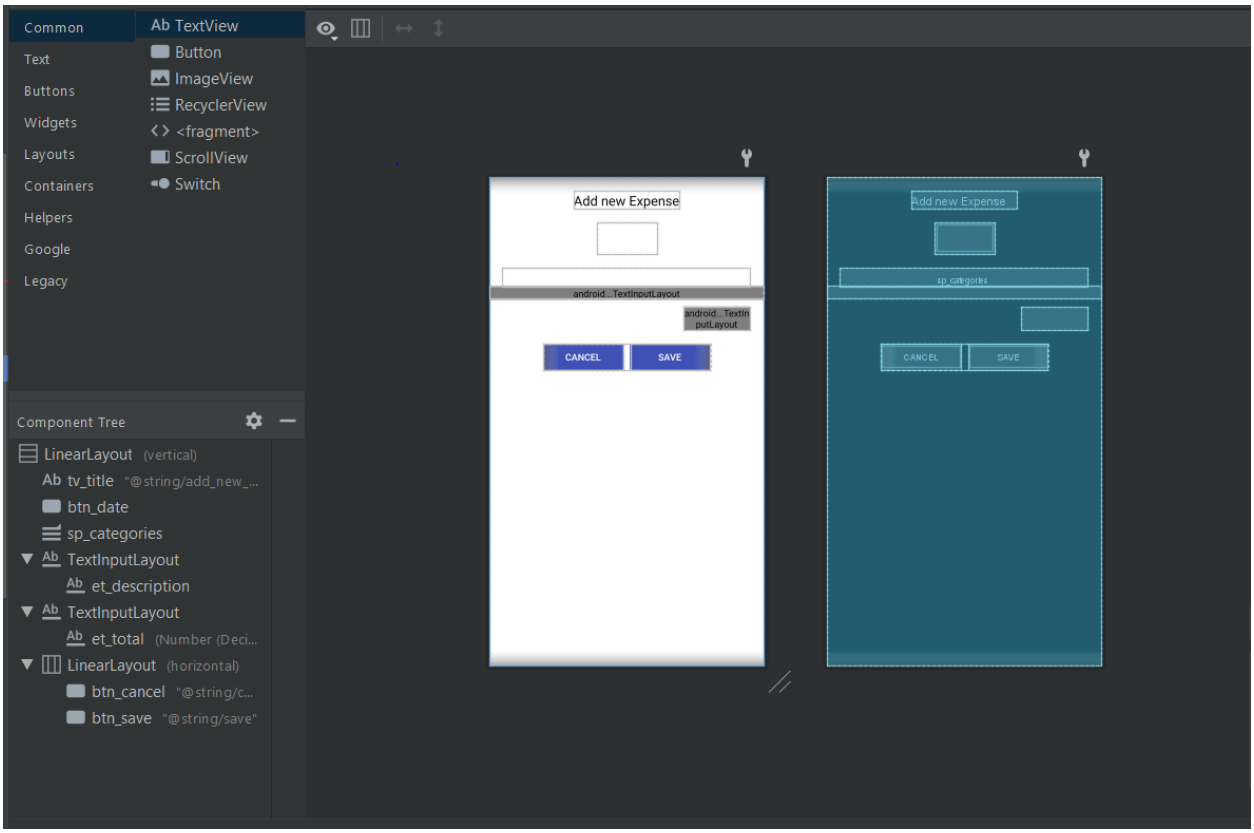
a) Design UI for Main Activity

b) Design UI for Expenses Activity

c) Design UI for Expense Detail that will show last known expenses in the same category as a summary.

d) Design UI for Categories Fragment

e) Design UI for Statistics fragment



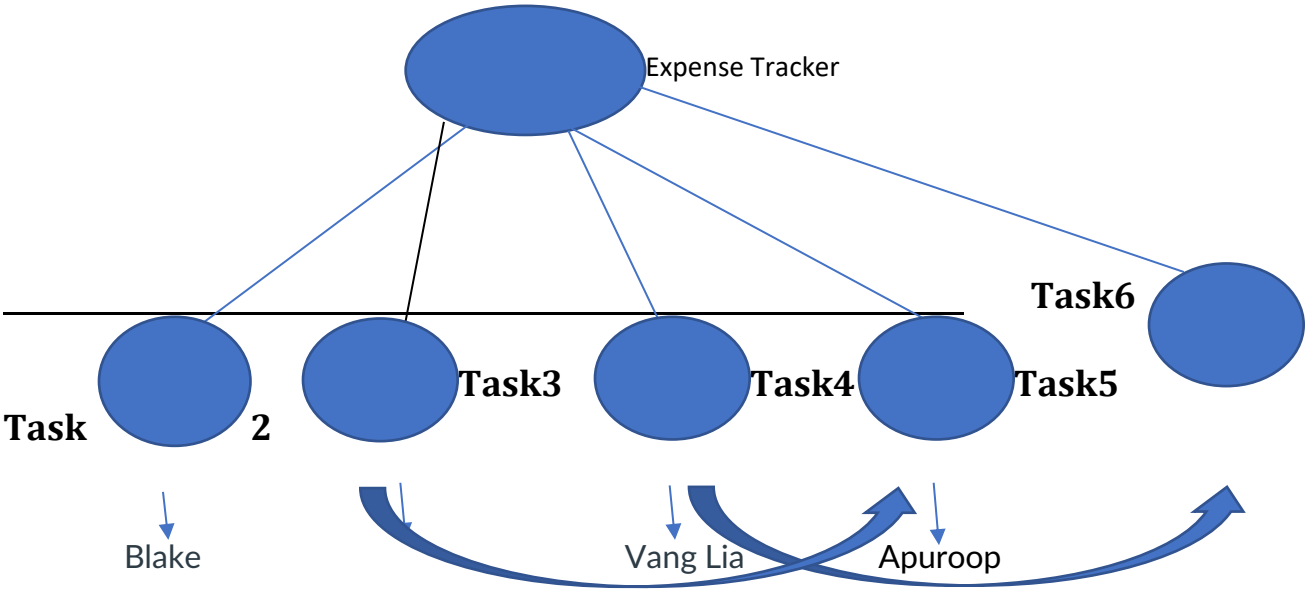
UI LAYOUTS

Task 3: Expenses View
Implementing logics for Adding and removing Expenses.
a) Add only expenses from screen
b) Remove expense from labeled view after going to the detail page.

Task 4: Categories View
a) Categories add and remove from label view.
b) Detail of category with graph showing last used in the present week.

Task 5: Statistics View
a) Select the dates to make the query based on expenses.
b) Displaying of Graphs for expenses made that month.
c) Displaying of Graphs for expenses made by category.

Task 6: Accounts View
a) Add bank checking/saving accounts and remove accounts
b) Sync and add transactions to page



7) Any issues, blockages with the project

A. Plaid API has a learning curve with Realm database.
B. some issues with displaying statistics after plaid api data is integrated.

8) GitHub link for your project

https://github.com/Apuroop1/Mobile-Fall-2020/tree/main/Project

https://github.com/lvp4b/Mobile-Fall-2020?organization=lvp4b&organization=lvp4b

9) Video Link for project

https://www.youtube.com/watch?v=AjEPrREzjWY&feature=youtu.be

https://drive.google.com/file/d/1Gae-lwNKMDXpe9FoGbY43L4D_IgdeGYt/view?usp=sharing

10) Presentation link for project

https://drive.google.com/file/d/16SHcrkuhE1savmM4PiYMxUdcaZk8rDbu/view?usp=sharing

11) References

a) https://www.kaggle.com/saurav9786/incomeexpenditure-dataset
b) https://flaviocopes.com/sample-app-ideas/
c) https://guides.codepath.com/android/Organizing-your-Source-Files
d) https://realm.io/docs/java/0.72.0/
e) https://github.com/PhilJay/MPAndroidChart
f) https://plaid.com/docs/