Last edited by Jaykishan Shantilal Baldania 1 year ago

ISEE 2020 | Team Technum Opus | Basic Prototype

Hola, Welcome to our team's blog.

We are team Technum Opus and we introduced our team in our last blog post.

The basic prototype of **Money Lisa** is out!

Let us walk through the steps involved in reaching there:

- Requirements
- System Design
- Working Prototype APP

The details to each section are below. So Los Geht's...

Requirements

In our first meeting with the Customer, during our discussion, we got various requirements. We are going to include only the ***Essential Requirements*** in this phase. The user should be able to...

- ... add daily expenses
- ... view and filter those expenses
- ... choose category when adding expense
- ... add his/her own choice of category when adding expense
- ... add date when adding expense
- ... add details/notes when adding expense
- ... set the repetitive daily/weekly/monthly/yearly expense
- ... add mode of payment when adding expense
- ... choose currency when adding expense

User Stories:

As a team, we analysed each and every requirement of the customer, by going through it thoroughly and analyzing it by keeping the users into the picture. If we had any difficulty in understanding the requirements, we asked client to correct us if we misinterpreted it.

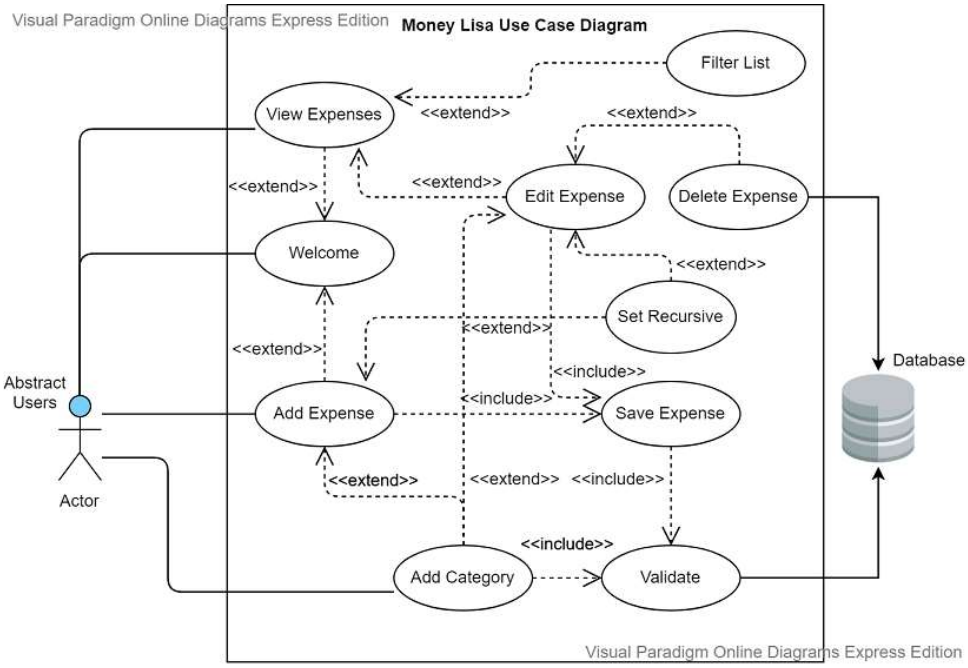
We made a Use Case Diagram as per requirements list and again verified the requirements with the client in next meeting. This is how we validated our assumptions of requirements.

In our last post, you can find our user stories as well. To give you a gist, I will tell you about them here too 😊

- As a user, I want to add my daily monetary transactions and save them to analyze and optimize my expenses.
- As a user, I want to assign categories (shopping, education, etc.) to transactions and also be able to edit those categories as per my requirements so that I can analyze my expenses in different areas.
- As a user, I want to assign date and time stamp to each transaction so that I can track the trends of my expenses with time.
- As a user, I would like to be able to filter the transactions list as per date, categories, range of amount or payment options so that I can focus and analyze on some narrowed list.
- As a user, I want to add notes to each transaction so that I can analyze the subjective information of the transactions in future.
- As a user, I should be able to add recursive transactions so as to save my time.
- As a business person, I want to assign mode of payment to each transaction so that I can analyze based on sources of debit.
- As a user, I prefer to see icons assigned to every category to save time while quickly scrolling through the entire list.
- As a hurried user, I want to delete or update the transactions so that I can correct some entries or eliminate them if I no longer need them.
- As a student, I should be able to set budget and would like to be notified in case the budget crosses the threshold so that I can plan further spending accordingly.
- As a charts lover, I want to visualize the transactions in graphs or other kinds of charts to save my analysis time using charts instead of lists.
- As a user, I want the help page to quickly understand how the application works and know its full capability.
- As a techie, I want to export or import data from/to CSV or Excel format to avoid wasting time entering all transactions or visualizing/analyzing transactions in other tools.
- As a student, I should be able to get information about my average expenses on per week or month basis so that I can quickly know that my spending is on track.
- As a working professional, I want to get information of my savings by having a functionality to enter income so that I can save my time calculating myself.
- As a frequent traveler, I should be able to assign currency to transactions and also want to convert transaction from one currency to another so that I can save my time of manually doing currency calculations.
- As a Product owner, I would like to have this application to be compatible with at least Android version Marshmellow, so that the application can be used by wide range of customers.

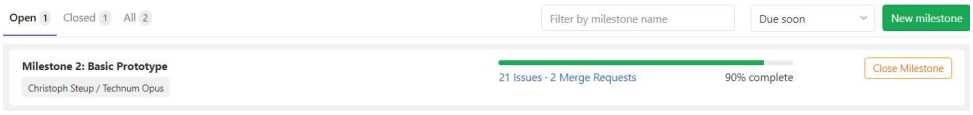
Use Case Diagram

Here, the interaction of the user with the system and their relationships are illustrated.



Timeline

Milestone 2 in GitLab status (while we are writing this)



Way ahead

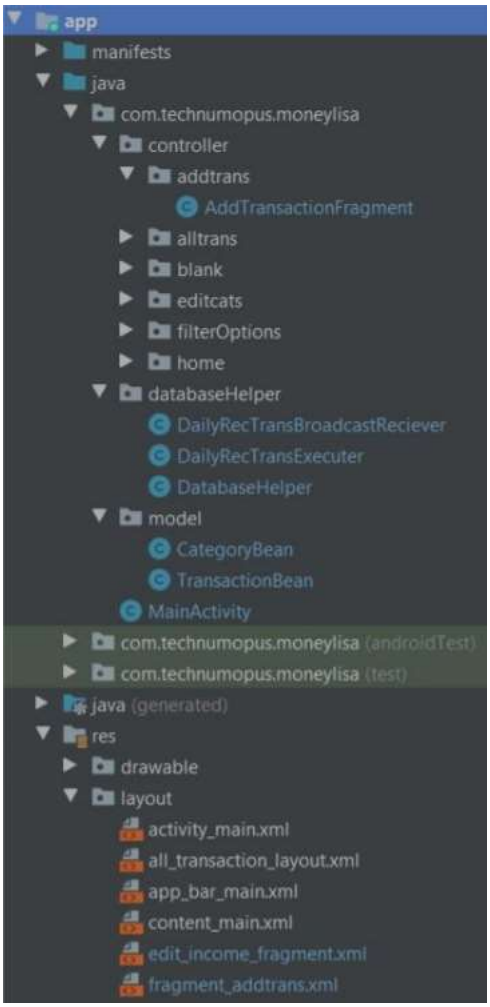
Time Line Overview

Milestones	Duration	Begin	End	21. Apr	28. Apr	05. Mai	12. Mai	19. Mai	26. Mai	02. Jun	09. Jun	16. Jun	23. Jun	30. Jun	07. Jul	14. Jul	21. Jul
Team Presentation	5	21. Apr	28. Apr														
Basic Prototype	28	28. Apr	26. Mai														
Basic Prototype Leverage	5	28. Mai	02. Jun														
Advanced Prototype	21	26. Mai	16. Jun														
Advanced Prototype Leverage	5	18. Jun	23. Jun														
Beta Prototype	21	09. Jun	30. Jun														
Beta Prototype Leverage	5	02. Jul	07. Jul														
Final Report	14	30. Jun	14. Jul														

System Design

System Architecture

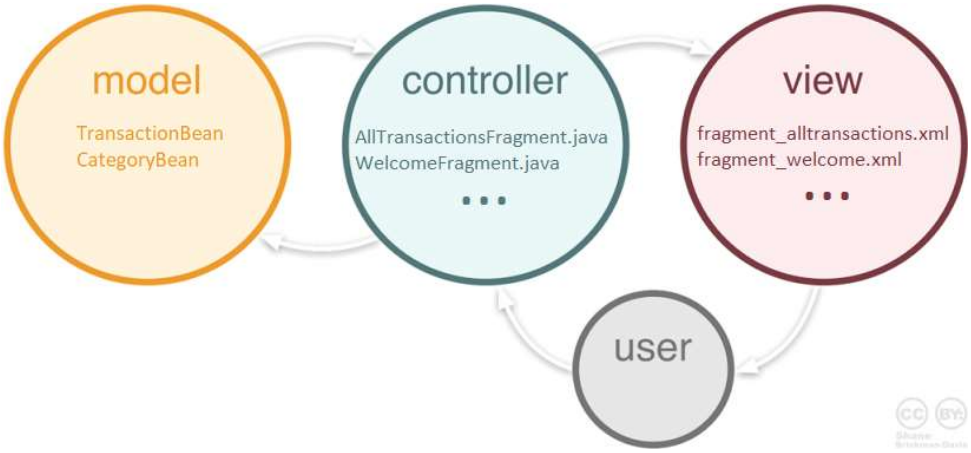
We have followed **Model View Controller** Architecture.



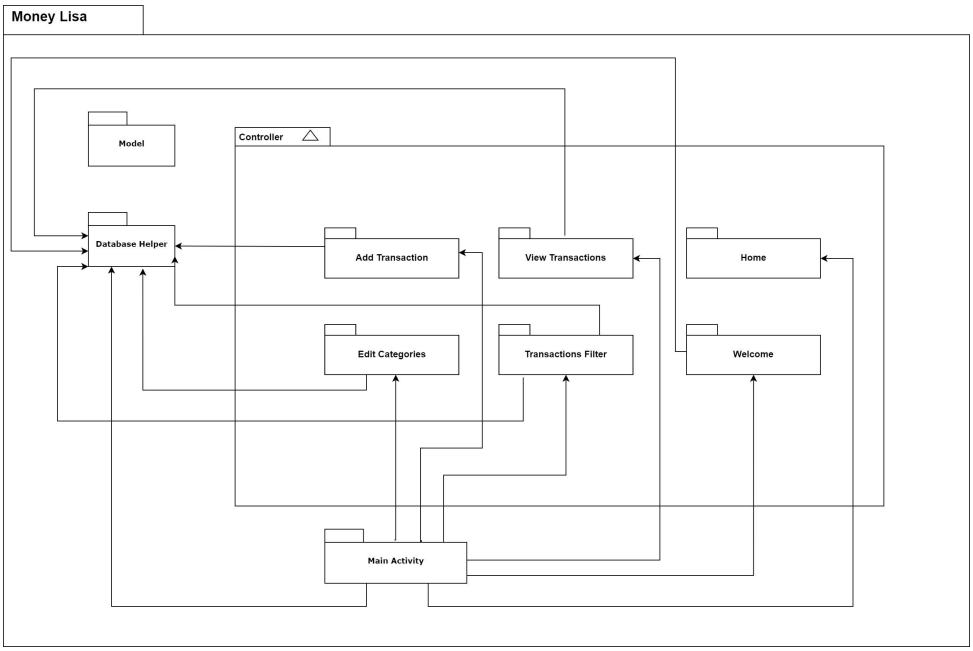
The Model-View-Controller (MVC) is an architectural pattern. It segregates the application into three components. Model, View and Controller.

- Model corresponds to the data related logic, that is being transferred between View and Controller. In our case, TransactionBean and CategoryBean are the Model classes which holds states of expense and category respectively and have corresponding getter and setter methods.
- View corresponds to the UI components of the application. In our cases xml files that corresponds to the fragments are the View component.
- Controller acts as an interface between View and Model components. It also connects to Data Access Object. All the business logic and data manipulations are done in this component. In our case, controller is Fragment classes.

- In MoneyLisa we are using MVC with Data Access Object. DAO is used to connect with database.



Package Diagram



Classes Overview

MainActivity

The starting point of the application. All the fragments are configured in the MainActivity. onCreate() method is executed when the activity is loaded. Database is initialized in this class. Every button in the app drawer loads from this class.

TransactionBean

It is the POJO class which holds the data related to each entry. It has getter and setter methods.

CategoryBean

It is the POJO class which holds the data related to each category the user inserts. It has getter and setter methods.

DataBaseHelper

It is the Data Access Object(DAO) often called as DAO. It is responsible for SQLite database interactions.

DailyRecursiveTransactionsExecuter

When the App starts for the first time in onCreate() method of MainActivity class, the DailyRecursiveTransactionsExecuter object will be instantiated. Then the methods of DailyRecursiveTransactionsExecuter such as setAlarm and setSchedule will schedule a task of updating recursive transactions automatically in database.

DailyRecursiveTransactionsBroadcastReciever

This is an Android Broadcast Receiver class which receives the trigger from object of DailyRecursiveTransactionsExecuter class. The actual task which needs to be performed will be implemented in onReceive method. This task involves adding entries to database every midnight for recursive transactions.

AddTransactionFragment

It is responsible to add new expenses in database. To perform add expense activity, it calls the method of DataBaseHelper class with an object of TransactionBean as a parameter.

AllTransactionsFragment

It is responsible to view expenses. It initializes AllTransactionsAdapter class which in turn fetches data by calling DataBaseHelper class.

AllTransactionsAdapter

It provides a binding from list of expense data set to views that are displayed within a RecyclerView in UI. It inherits RecyclerView.Adapter class. It has an inner class 'ViewHolder' which inherits from RecyclerView.ViewHolder class. It sets data into every list of Recycler View.

WelcomeFragment

It has the Welcome screen of the app. It has a continue button, which redirects to Home fragment of the app.

HomeFragment

Home fragment class has listeners for Add Expense and View Expense buttons. It calls fetchCurrMonthTxn() method of DataBaseHelper class, which returns a list of sum of amount of current month on the basis of different currencies. It instantiates CumulativeAmounttAdapter class and passes list of all cumulative transaction bean.

CumulativeAmountAdapter

It is an adapter class which provides binding from the list passed from HomeFragment to the RecyclerView of HomeFragment view.

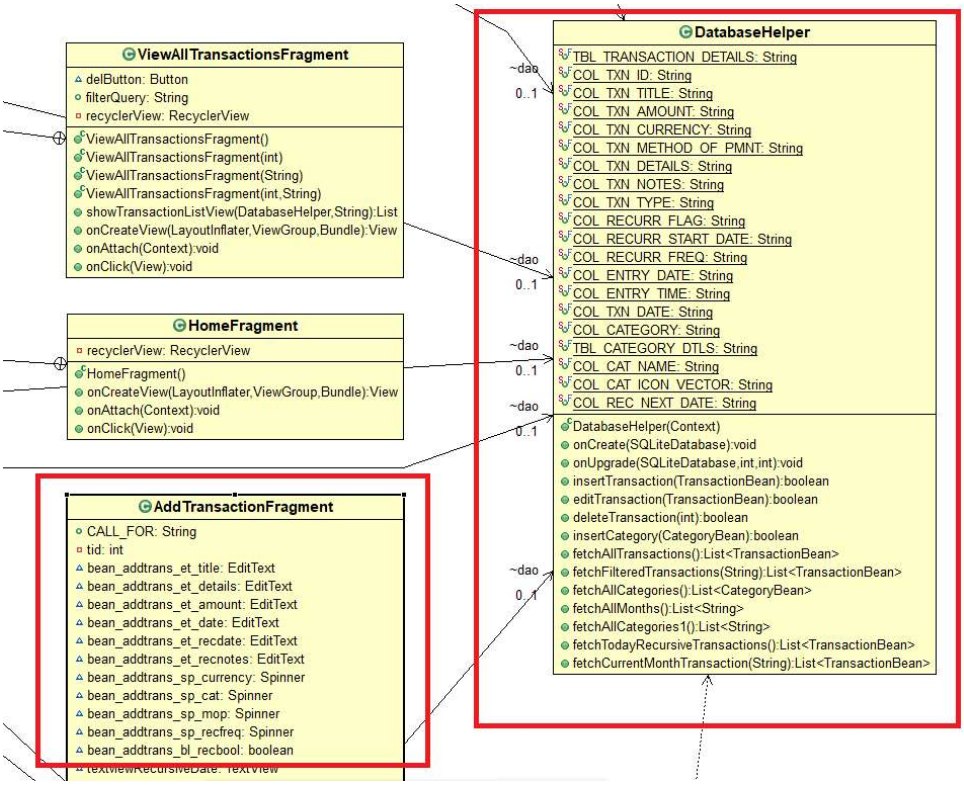
Class Diagram

Our complete Class Diagram is here: https://code.ovgu.de/steup/technum-opus/-/blob/patch-1/Diagrams/BasicPrototype/Class_Diagram_Phase1.pdf

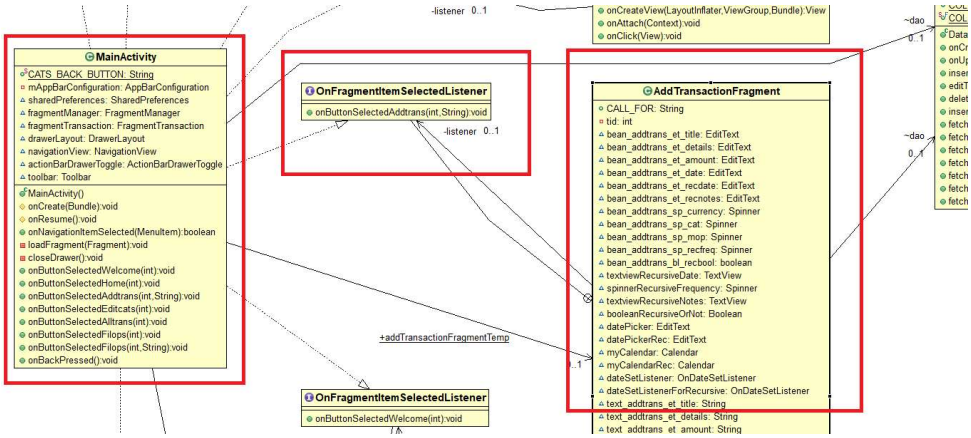
Some associations in Add Expense scenario:

Classes involved: MainActivity, OnFragmentItemSelectListener, AddTransactionFragment, DataBaseHelper

- AddTransactionFragment and DBHealper are associated with each other. AddTransactionFragment uses methods in this class to insert data which was provided by user in UI.



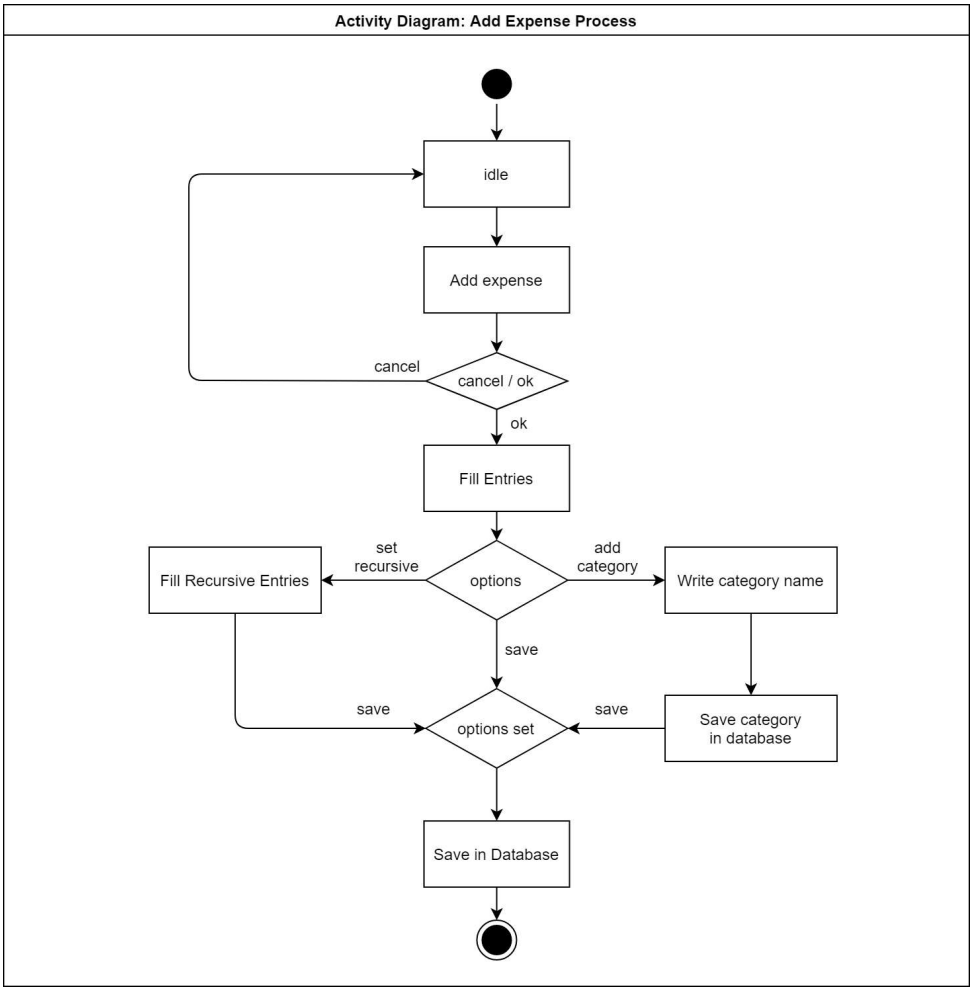
- OnFragmentItemSelectListener and AddTransactionFragment shows composition relationship. OnFragmentItemSelectListener is inner interface of AddTransactionFragment.
- OnFragmentItemSelectListener and AddTransactionFragment are also associated to each other.
- MainActivity realizes (provides implementation) the interface OnFragmentItemSelectListener.
- One to many association relationship between MainActivity and AddTransactionFragment class. It is 1 to many because the same fragment is instantiated for Add and Edit expenses.



Activity Diagram:

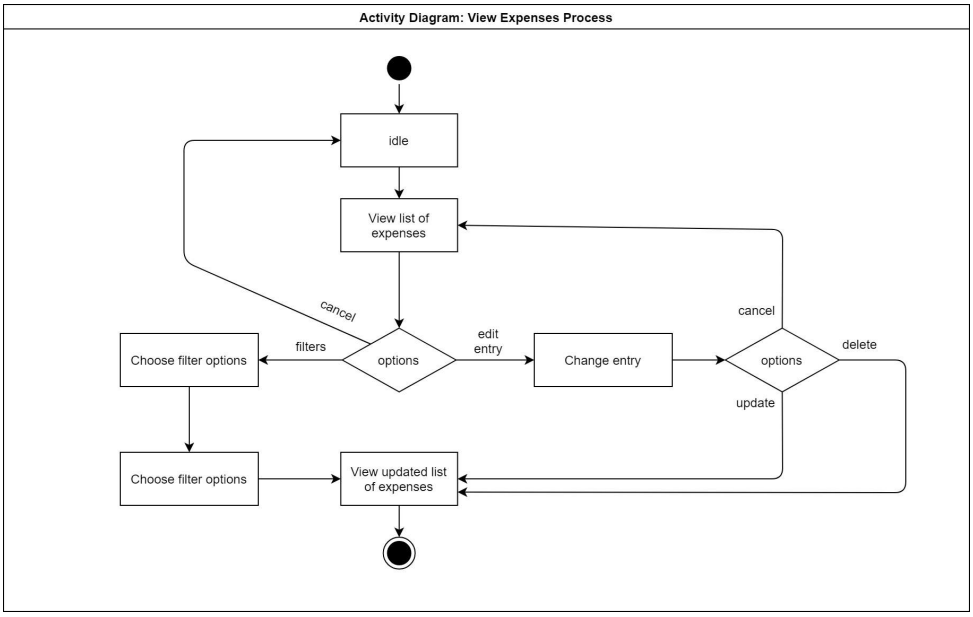
Adding Expense Process

When the user wants to add an expense, he/she can press ok and move to fill the entries, at this stage if a user cancels, they go back to the idle state. While filling the entries, the user has an option to set the expense as recursive or not. Here the user can also enter the category of his choice if he doesn't want to choose from the predefined categories. The updated category is saved in the database. After pressing save, the expense is also saved in the database and with this the activity stops.



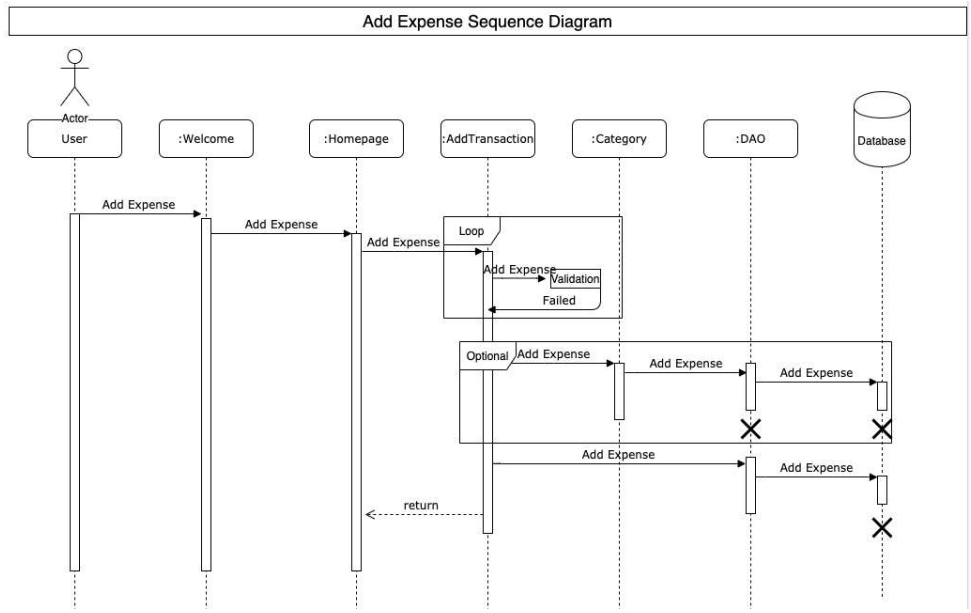
View Expenses Process

In the second case, after the idle state, the user enters into the view expense state. The user then can apply filters to view the expenses. The user can also edit/delete the saved expense, and view the updated expense list. Here he can also cancel and go back to the view expense state. This then ends the activity.

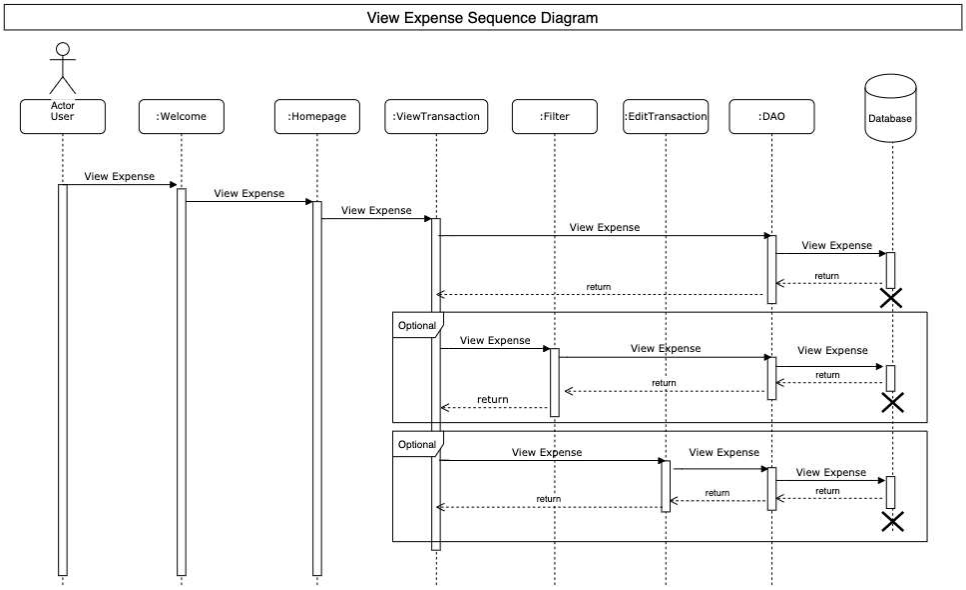


Sequence Diagram

Use Case - Add Expense



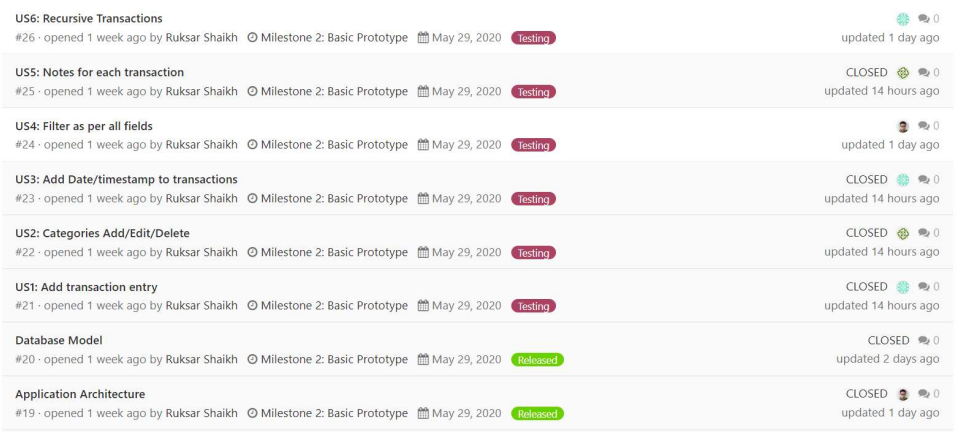
Use Case - View Expenses



Development Strategy

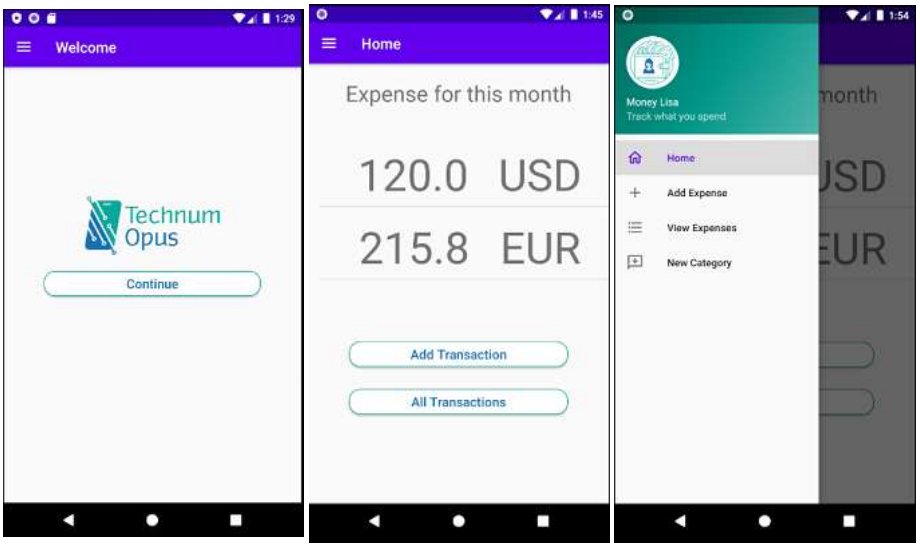
- Listing User Stories based on requirements
- Initializing the tasks related to App development and User Stories in GitLab
- Initializing the Milestones in GitLab
- Deciding the Labels for each task (for example, "Released", "Developing", "Testing", "Backlog")
- Planning each task in GitLab
- Changing status of tasks as per every day work
- Hosting a Source Code in GitLab
- Using GitLab VCS in Android Studio to push, pull and commit changes
- Working on different packages to avoid VCS conflicts

GitLab has really helped us to work synchronously and lets us know which all tasks are assigned to whom and what is pending. Most importantly, it helps us to identify our timelines as well. We love working as a team and we have created a enjoyable work atmosphere within the team.

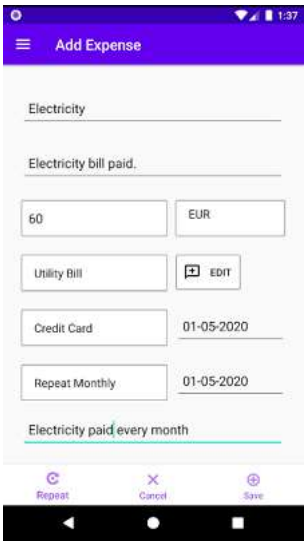


Working Prototype APP

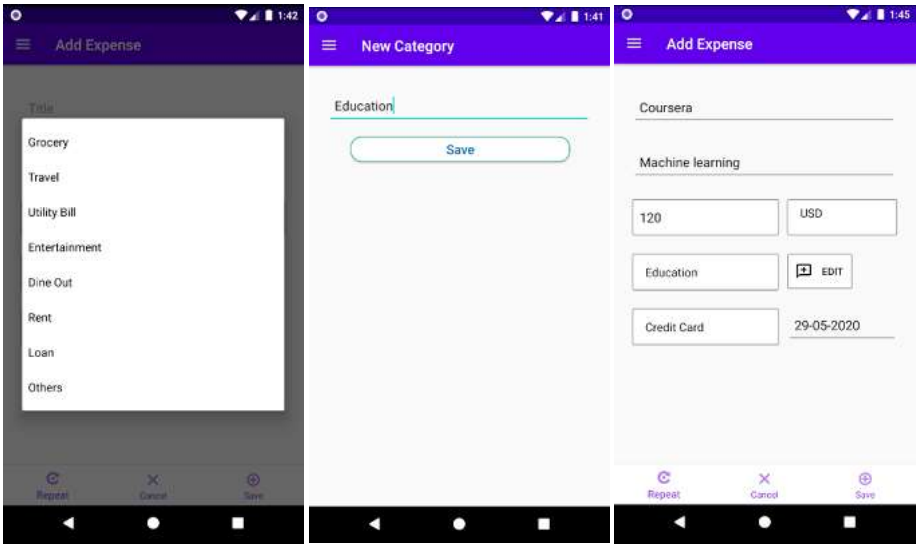
Welcome - Home - Navigation Drawer



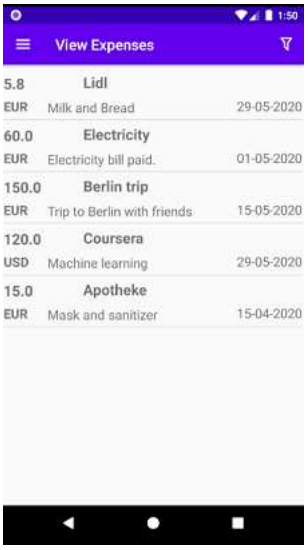
Add Expense



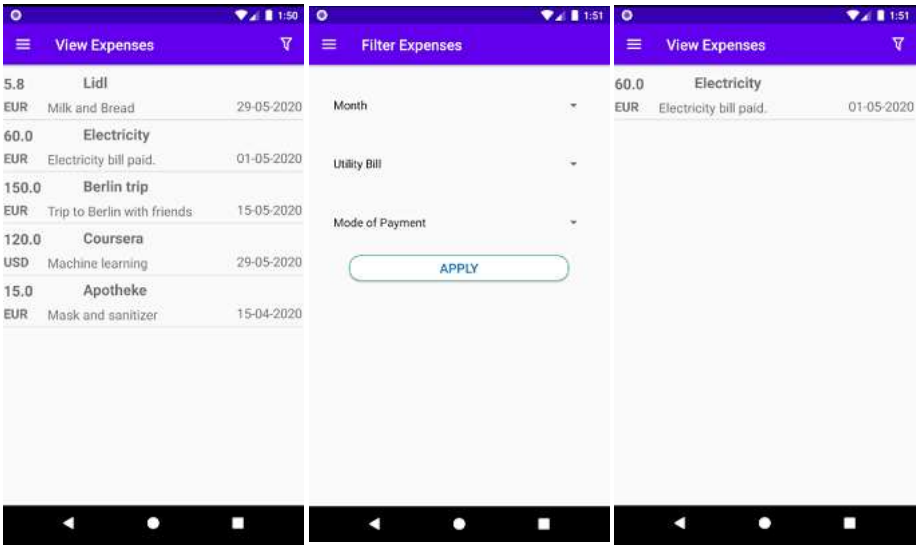
Add Category



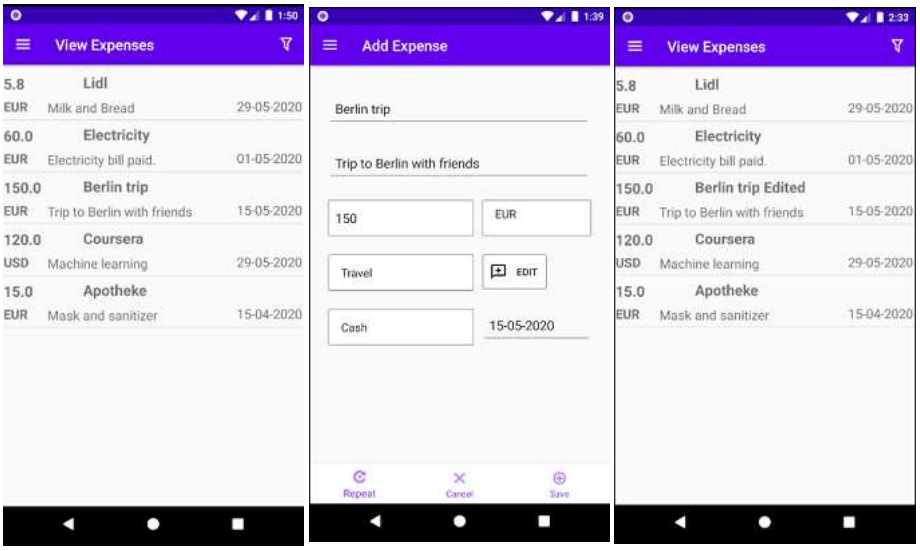
View Expenses



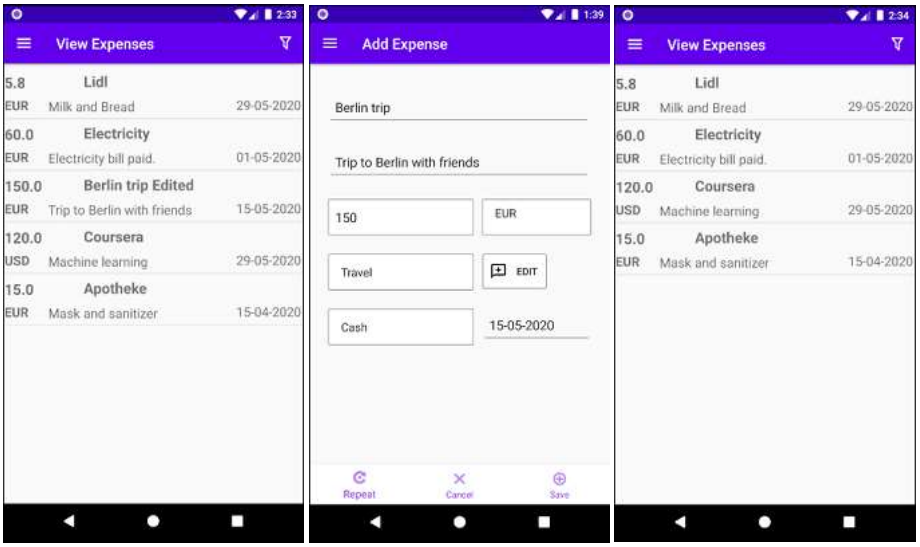
View Expenses - Filter



View Expenses - Edit



View Expenses - Delete


2:33
View Expenses
5.8 Lidl
EUR Milk and Bread 29-05-2020
60.0 Electricity
EUR Electricity bill paid. 01-05-2020
150.0 Berlin trip Edited
EUR Trip to Berlin with friends 15-05-2020
120.0 Coursera
USD Machine learning 29-05-2020
15.0 Apotheke
EUR Mask and sanitizer 15-04-2020
1:39
Add Expense
Berlin trip
Trip to Berlin with friends
150
EUR
Travel
EDIT
Cash
15-05-2020
Repeat
Cancel
Save
2:34
View Expenses
5.8 Lidl
EUR Milk and Bread 29-05-2020
60.0 Electricity
EUR Electricity bill paid. 01-05-2020
120.0 Coursera
USD Machine learning 29-05-2020
15.0 Apotheke
EUR Mask and sanitizer 15-04-2020