COMP-1424 Mobile Application Development

Coursework: M-Expenses Application

Amir Farkhadov
ID No: 001124832

Contents

Section 1

Features implemented in this project:

A	Fully implemented
B	Fully implemented
C	Fully implemented
D	Fully implemented
E	Fully implemented
F	Not implemented
G	Not implemented
H	"Edit" and "Delete" buttons for each expense ActionBar titles for each page Fragment layouts for future tablet version

Section 2

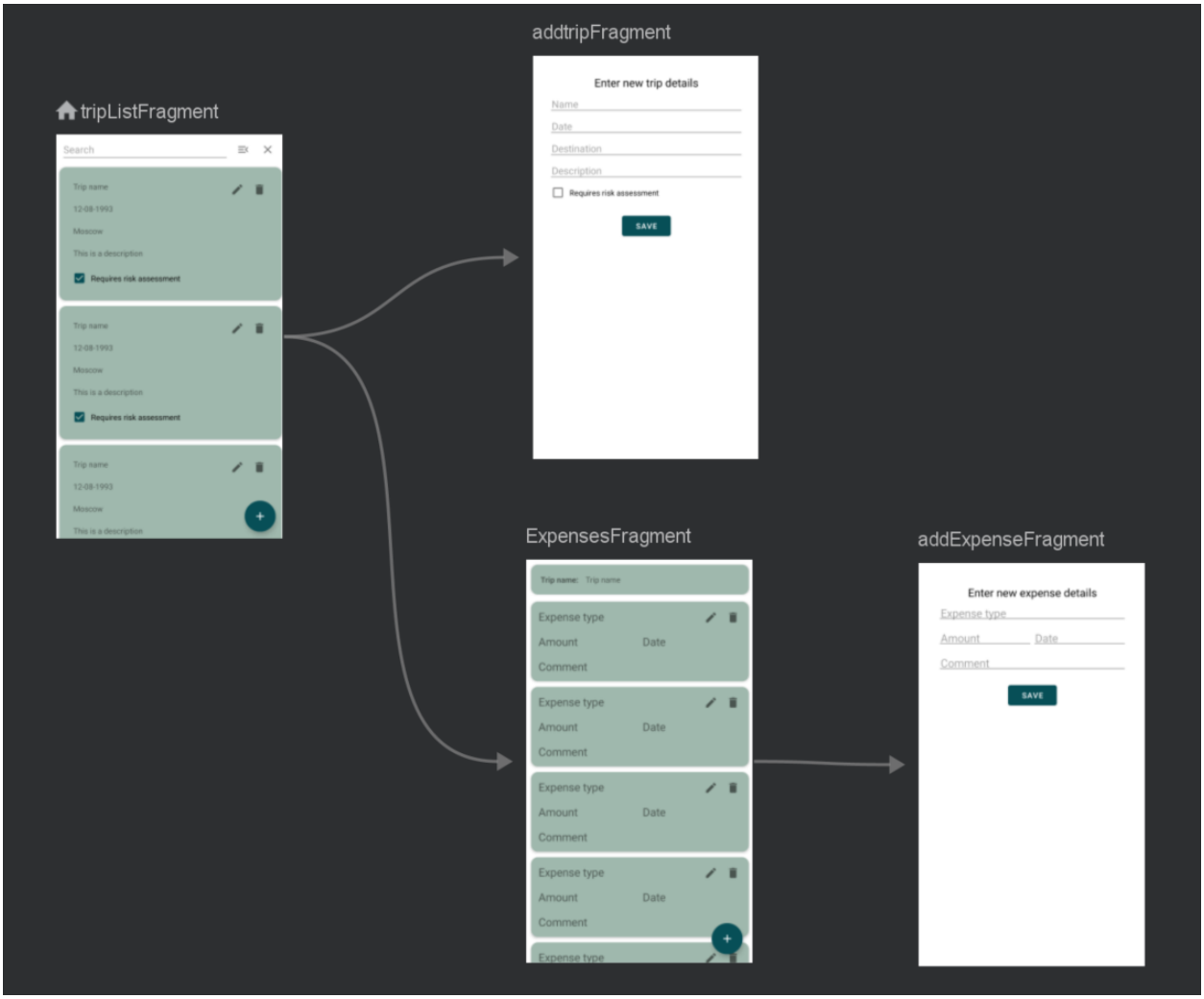
This coursework was prepared individually by Amir Farkhadov. My groupmates were not engaged in the process till the beginning of December when most of the functions were already implemented by me. Thus, I have decided to submit my work individually, as I consider it is difficult to bring valuable impact to this project in this short period of time. If all members of the group were engaged, this coursework could have been implemented fully and in less time.

Section 3

Application Overview

This application consist of two list fragments and two fragments for the addition of trips and expenses:

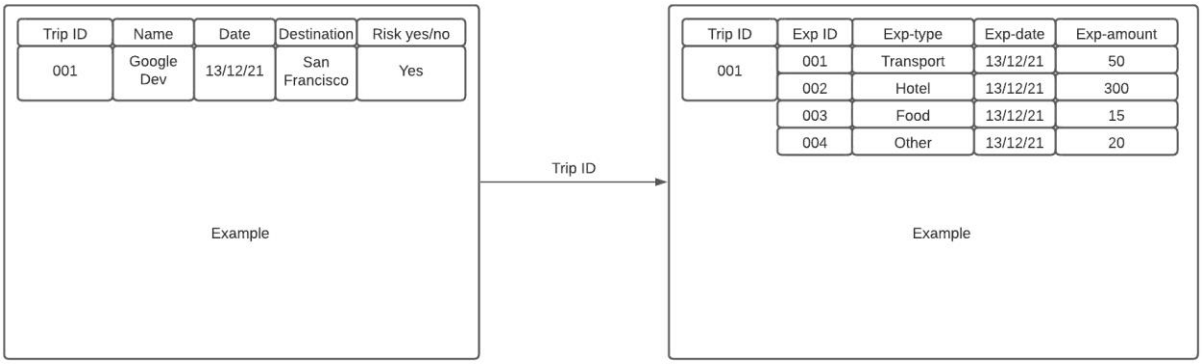
- Trips List Fragment – where a user can see available trips, edit details of this trip or delete all or particular trips.
- Add Trip Fragment – where a user can create a new trip and enter details, like Name, Destination, Date (current date as default), Risk Assessment, and Description.
- Expenses List Fragment – where a user can see available expenses to the particular trip along with the details of that expense (default date is the date of trip).
- Add Expense Fragment – where a user can create a new expense and enter details of it.



As it was not specified, this project could have been implemented by using activities. However, the practicality of this solution is questionable, because fragments can have the same functionality and simplify the adaptation process of the app for tablet (and foldable phones), and give more flexibility.

Each of these fragments has its RecyclerView for Trips and Expenses. It creates list elements dynamically and recycles unused views instead of destroying them, thus improving the performance and reducing power consumption of the app.

The SQLite database was fully implemented. As this app should be able to store a lot of trips, where each trip may have a number of expenses, it was decided to have two tables, one for Trips and one for Expenses, which will share Trip ID.



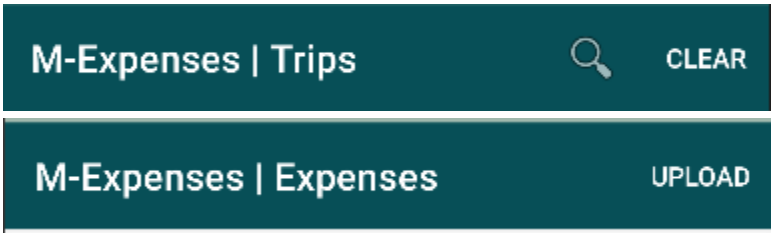
Searching and Advanced Searching were fully implemented. The simple search searches by trip name. For advanced searching, the dialog window was added, where the user can enter the name of the trip, destination, and date, and the program provides the most suitable result.

As required, the user can upload expenses stored on the device to the URL provided using the Retrofit library. In case of successful upload or failure, the user will get a corresponding notification.

Human computer interaction

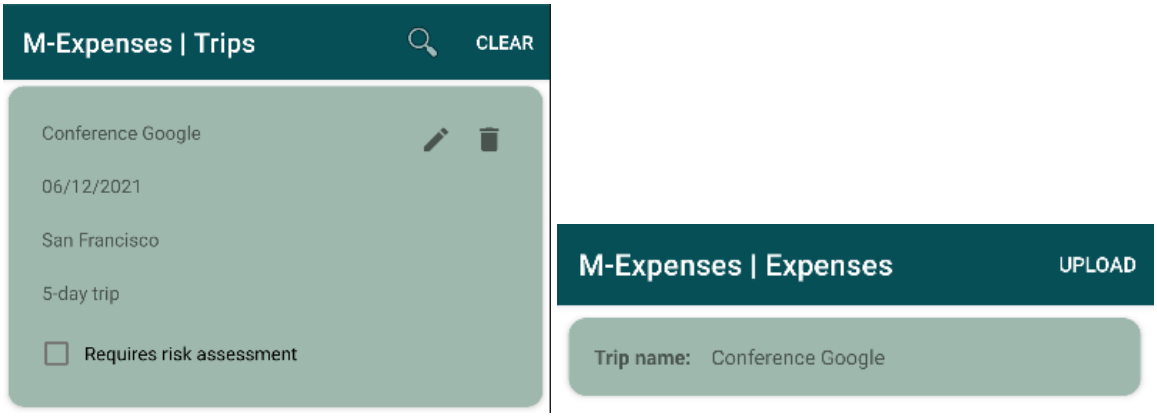
To achieve the best user experience, several UI elements were added:

The action bar - has a convenient title on every page, so the user will always be aware of what page he/she is on and what data he/she is about to enter.



The Floating Action Button – is an intuitive UI element that was introduced by Google many years ago and is still widely used in many different apps. Thus, the user always knows what that button means.

Trip name for expenses page – when a user creates a new trip and wants to add new expenses, the name of the trip will be placed on the top of the expenses page. Thus, a user sees and understands to which trip expenses are added.



Security

In this current stage, the app does not have any protection. The app can be more secure if the following protection mechanisms will be applied:

- User authentication – User provides some information to the system for validation, like username and password.
- Fingerprint/Face authentication or other biometric authentication.
- Session Handling – the session should not continue working when a user has switched from the app
- Data encryption – data will be only available to read for key owners.
- Source Code Encryption – modify the code, thus increasing the complexity for people to read.

Ability to run app on a range of screens

The user interface of the M-Expense application was built on Fragments. Fragments are independent components that encapsulate functionality. They operate in the context of the activity but have their lifecycle and UI and can be widely reused in different layouts. Fragments have a Fragment manager that is responsible for adding, deleting, replacing, and adding to the back stack.

Fragments were initially introduced to simplify adapting the layout for tablet screens. For mobile applications, there might be one fragment on the screen. Whereas on a tablet, there might be two fragments displayed at once. Thus, the only thing required to make this app adjustable for different screens size is to create a new layout file for wide screens and insert needed fragments.

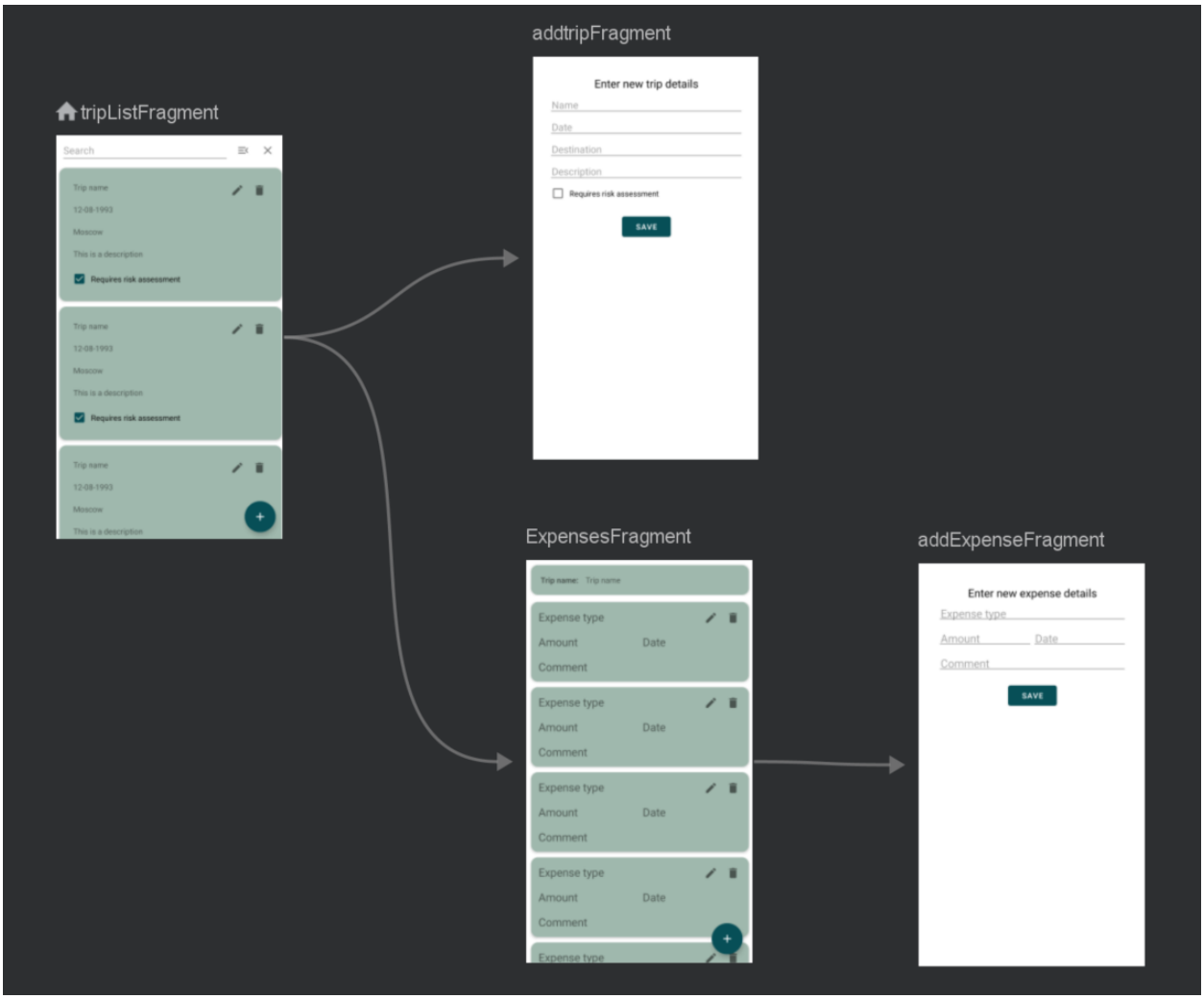
Change required for real-live deployment

If some company decide to use this application there are several functions that should be implemented:

- User authentication – so the company knows which employee adds the data
- Cross-platform – Application should be available on IOS, Android, and ideally as a Web Application, so every employee will have access to the app.
- Encryption – to increase security
- More appealing UI/UX
- Optimization and testing for bugs

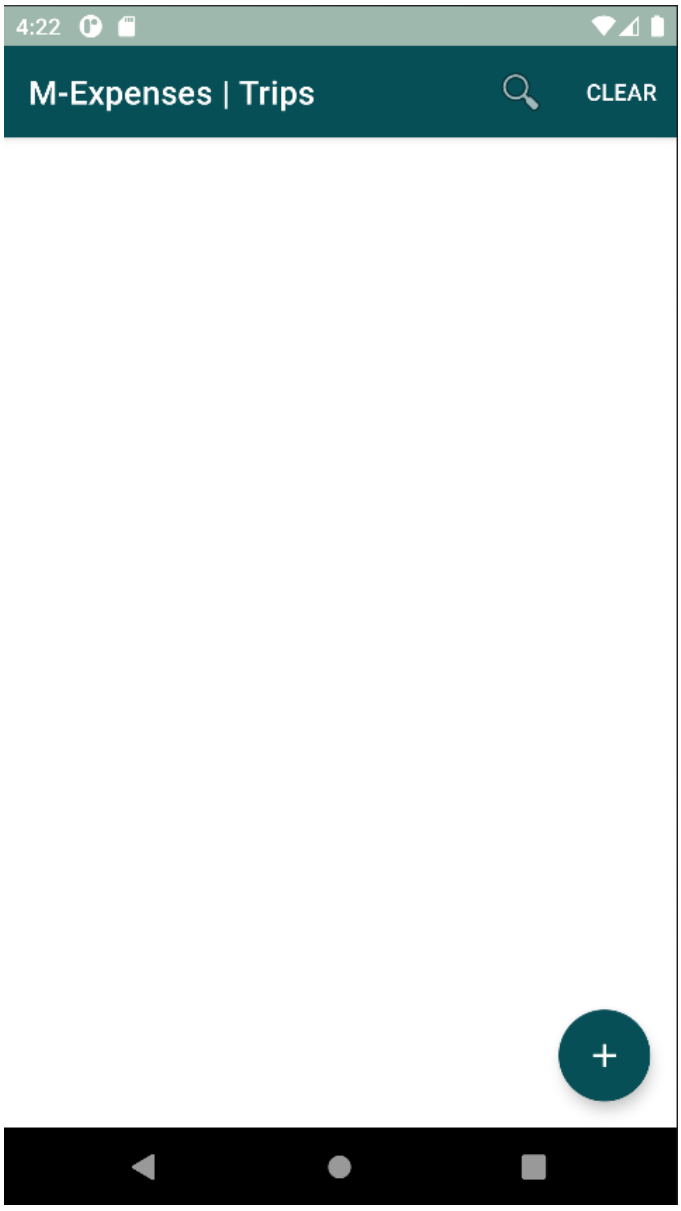
Section 4

Navigation Graph

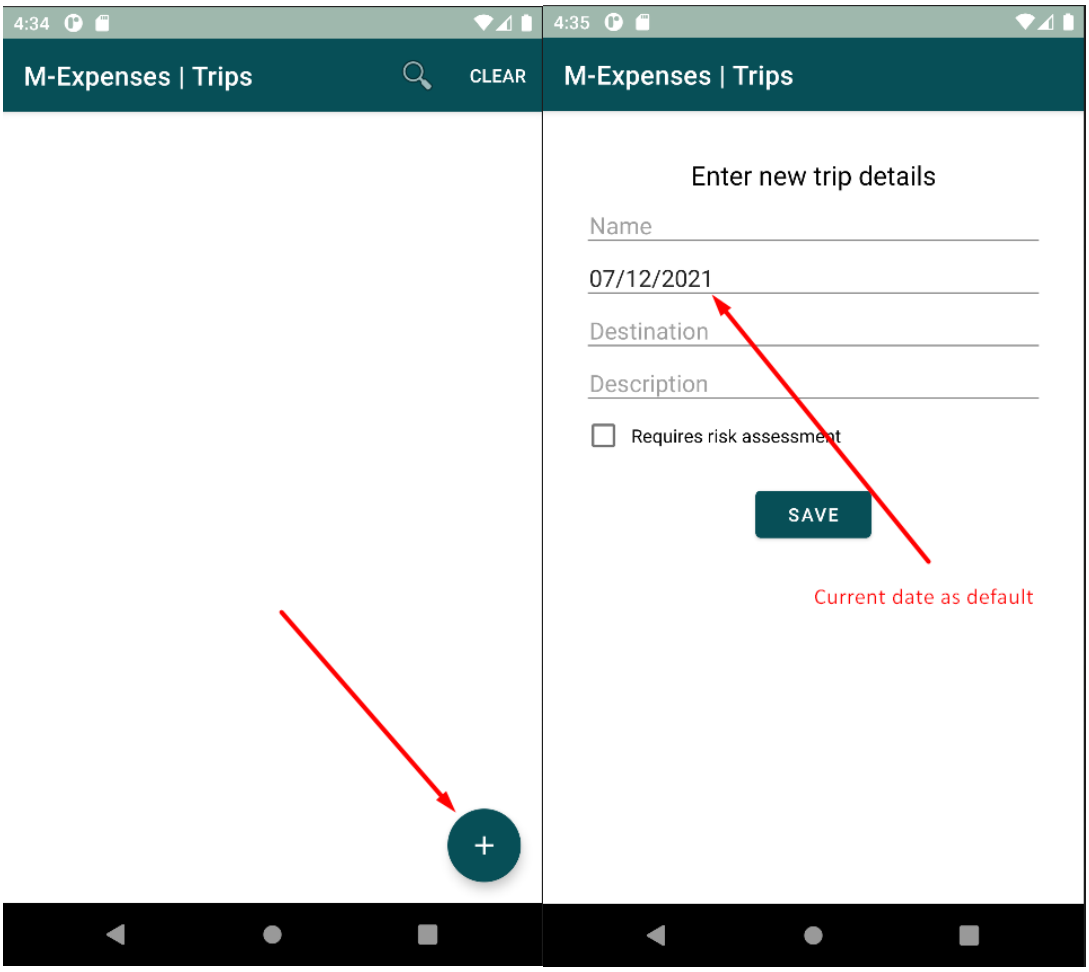


Step one: User opens the app.

Main screen displays “TripsListFragment” with all available trips.

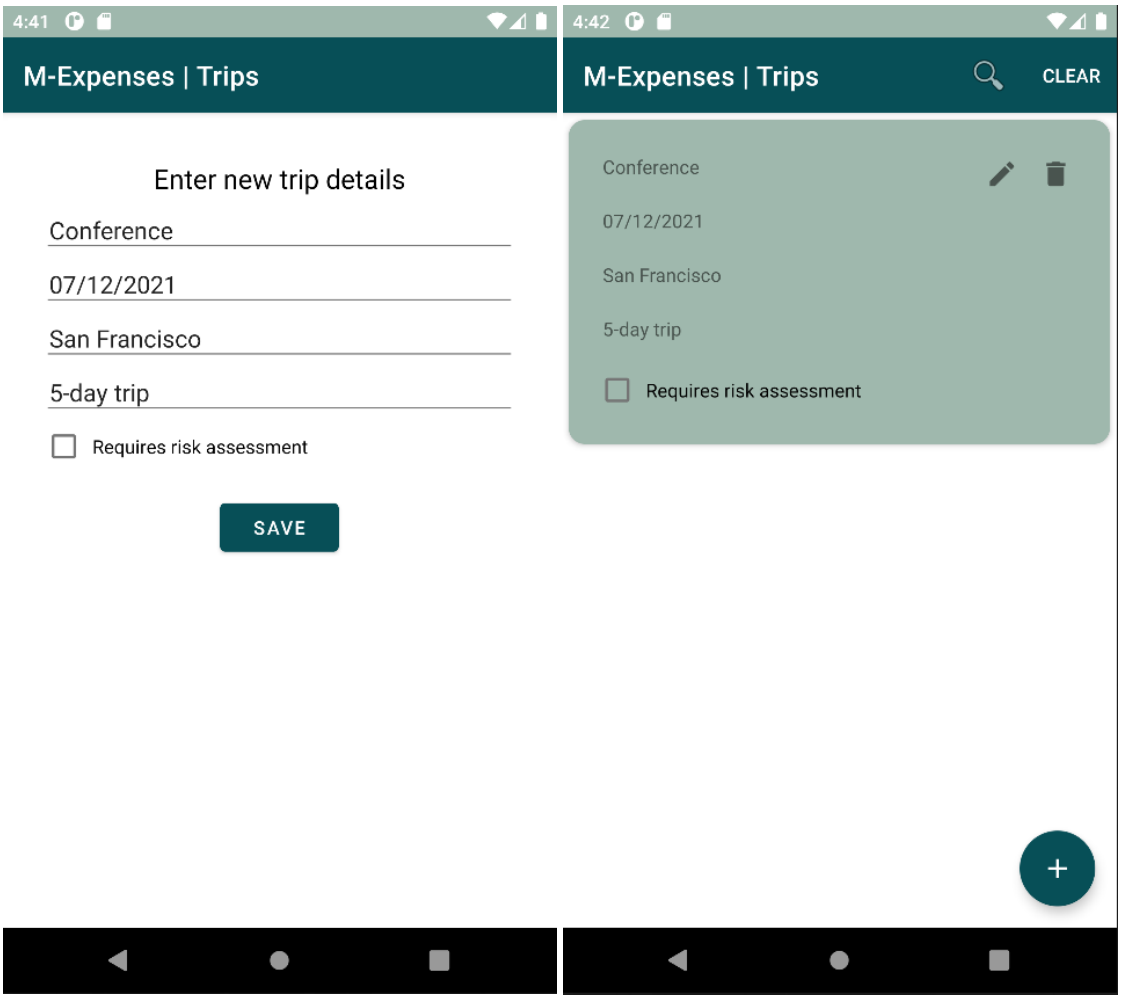


Step Two: User presses on “Add” button to create new trip.

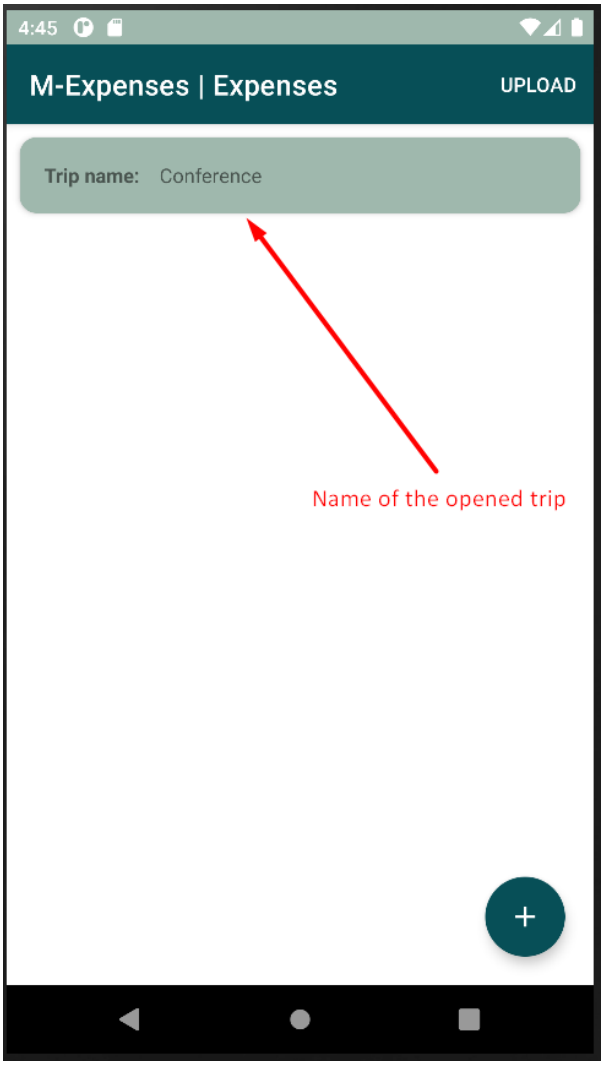


Step Three: User enters trip details and saves

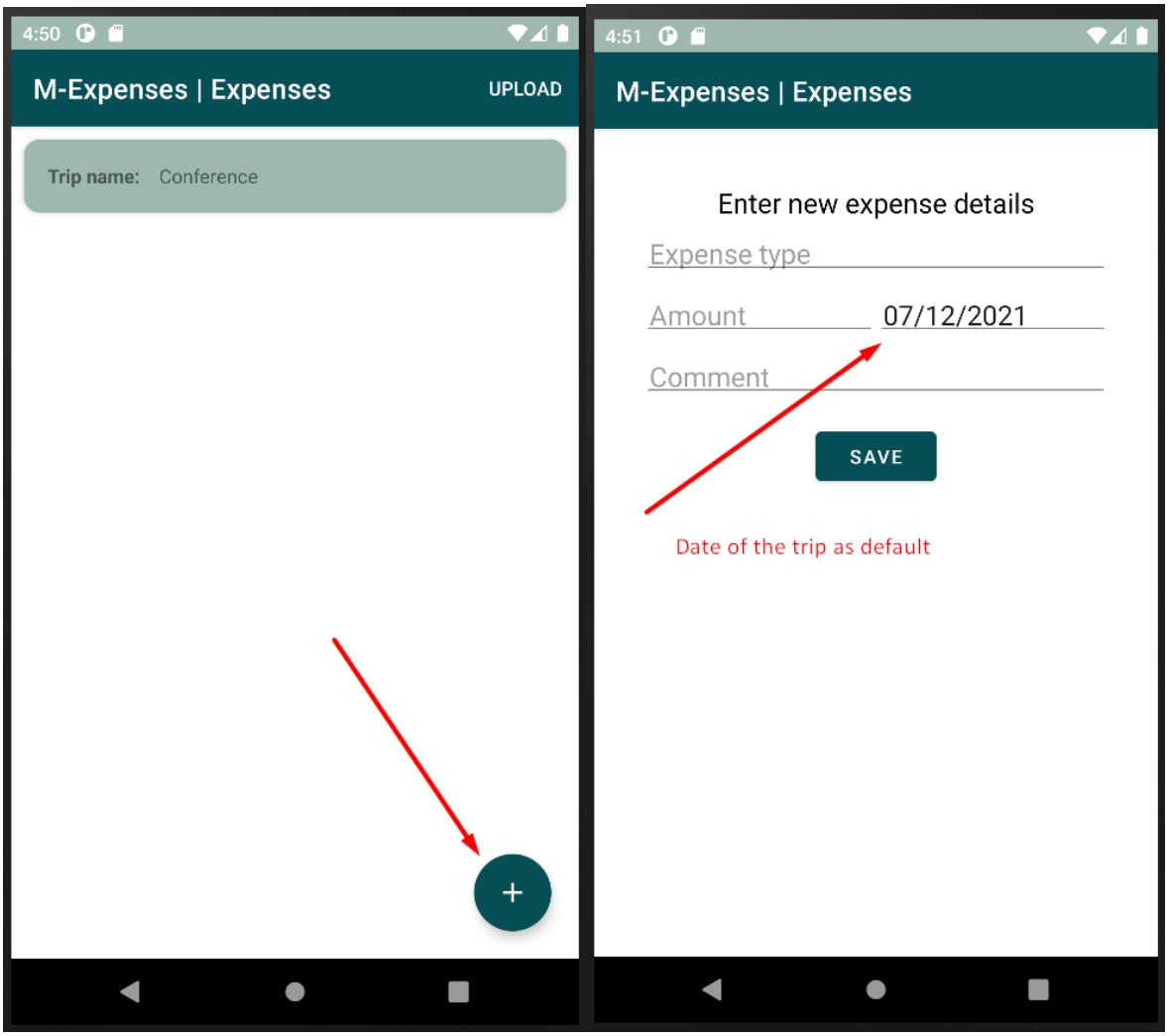
Created trip is displayed on the Main Screen (TripsListFragment).



Step Four: User clicks on trip and opens Expenses List.

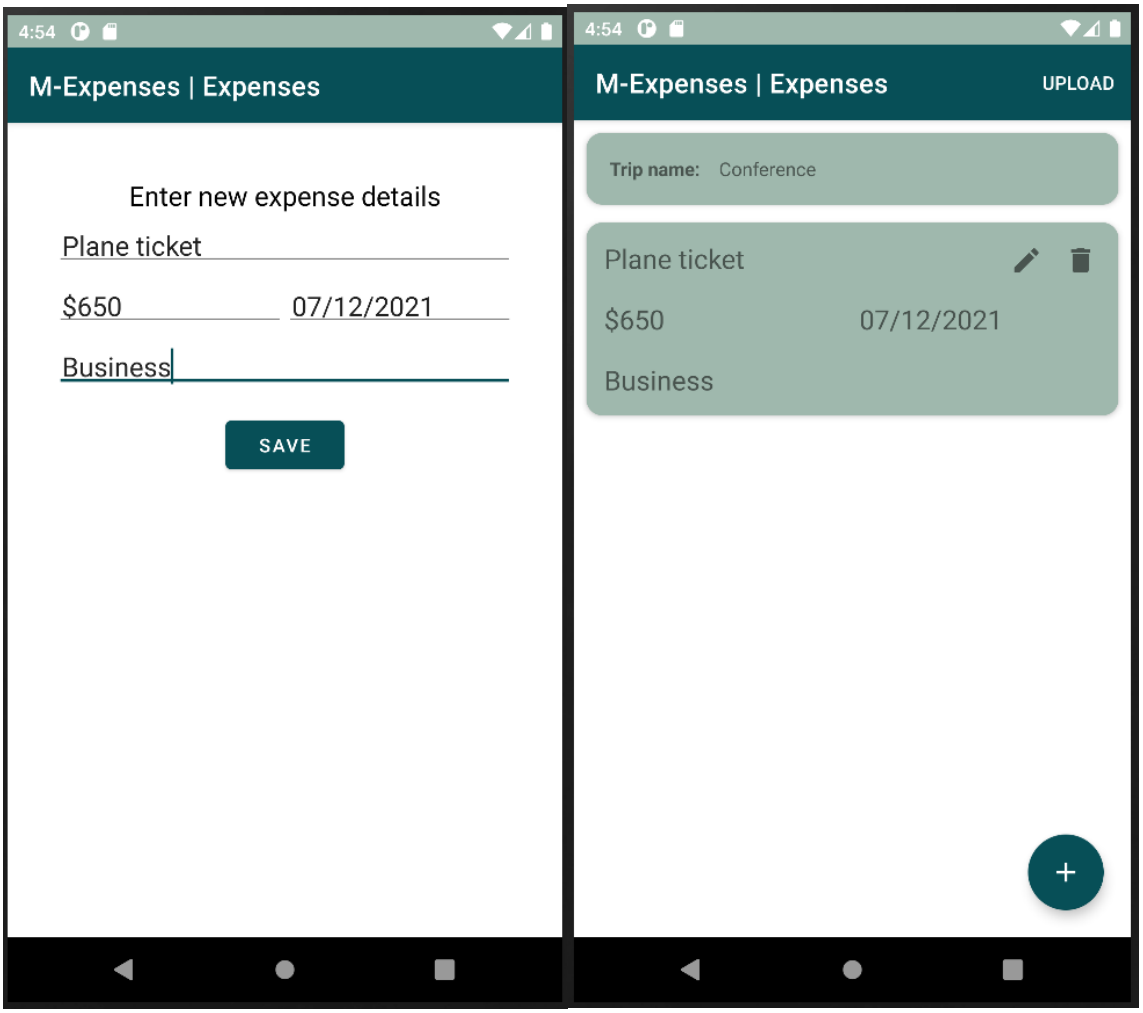


Step Five: User presses “Add” button to add new expense.



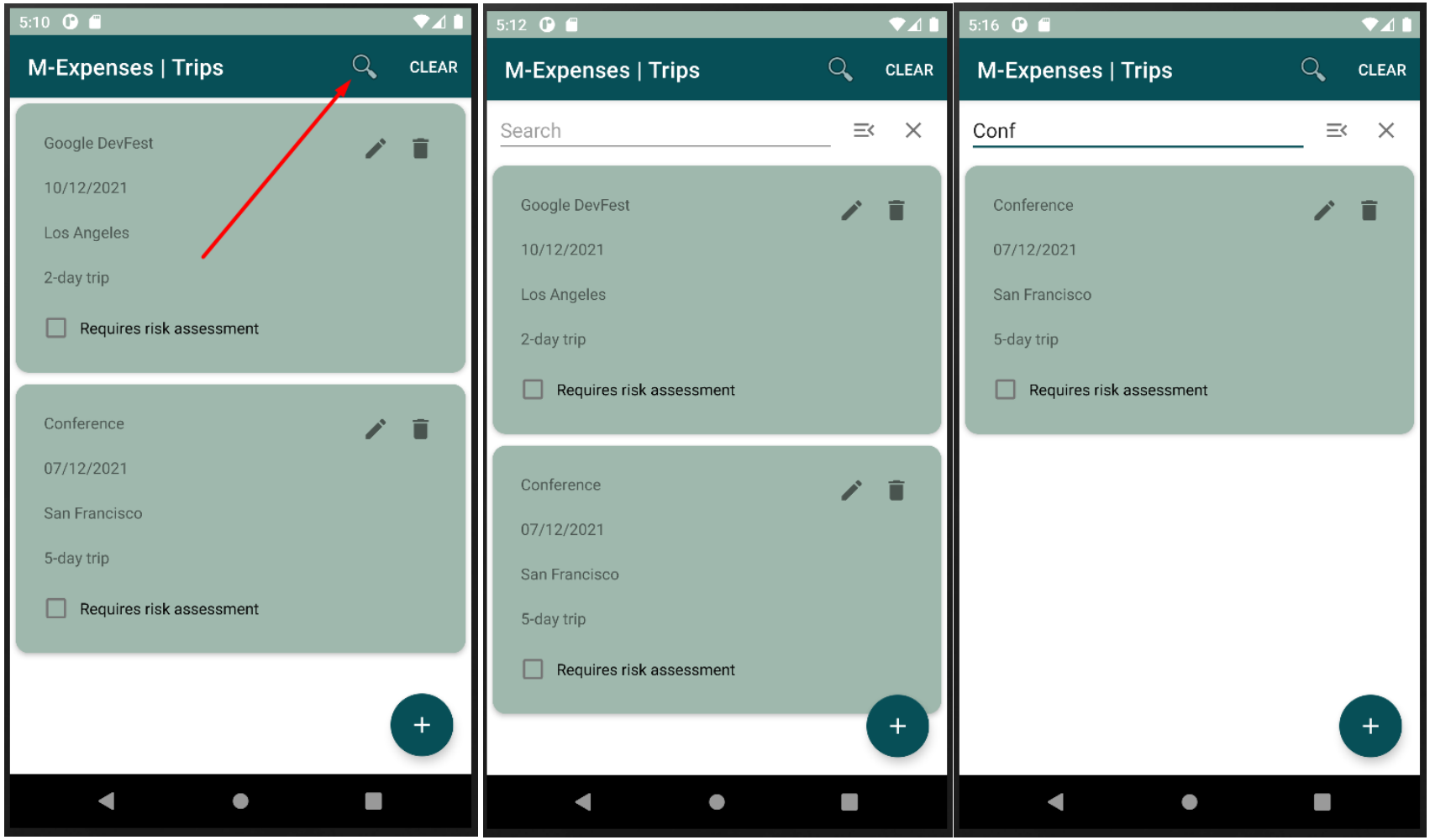
Step Six: User add expense details and saves

The added expense is displayed on Expenses Screen (Expenses Fragment).



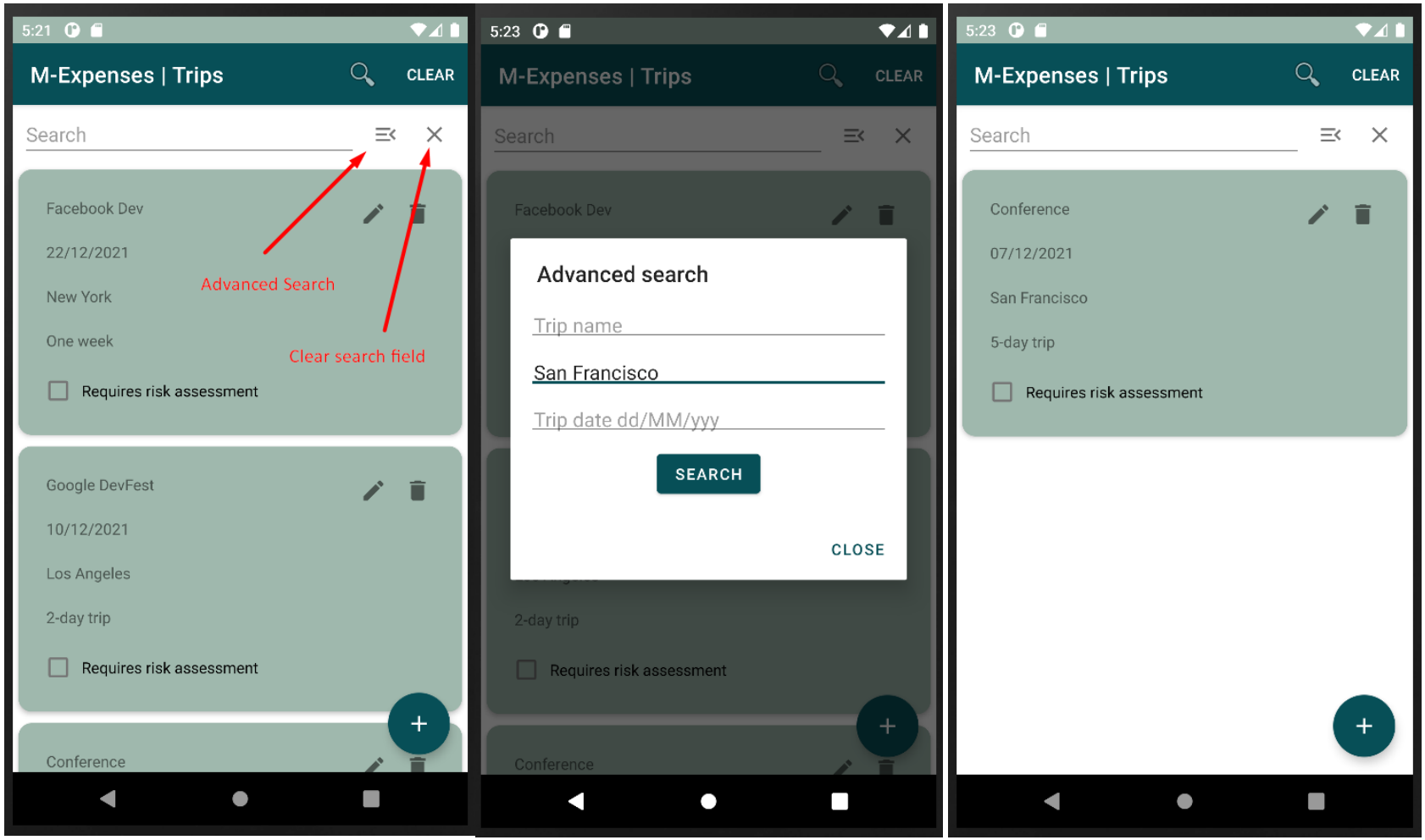
Features One: Search

The user clicks on the “Search” icon and the search line pops out. Search is done by Trip Name.

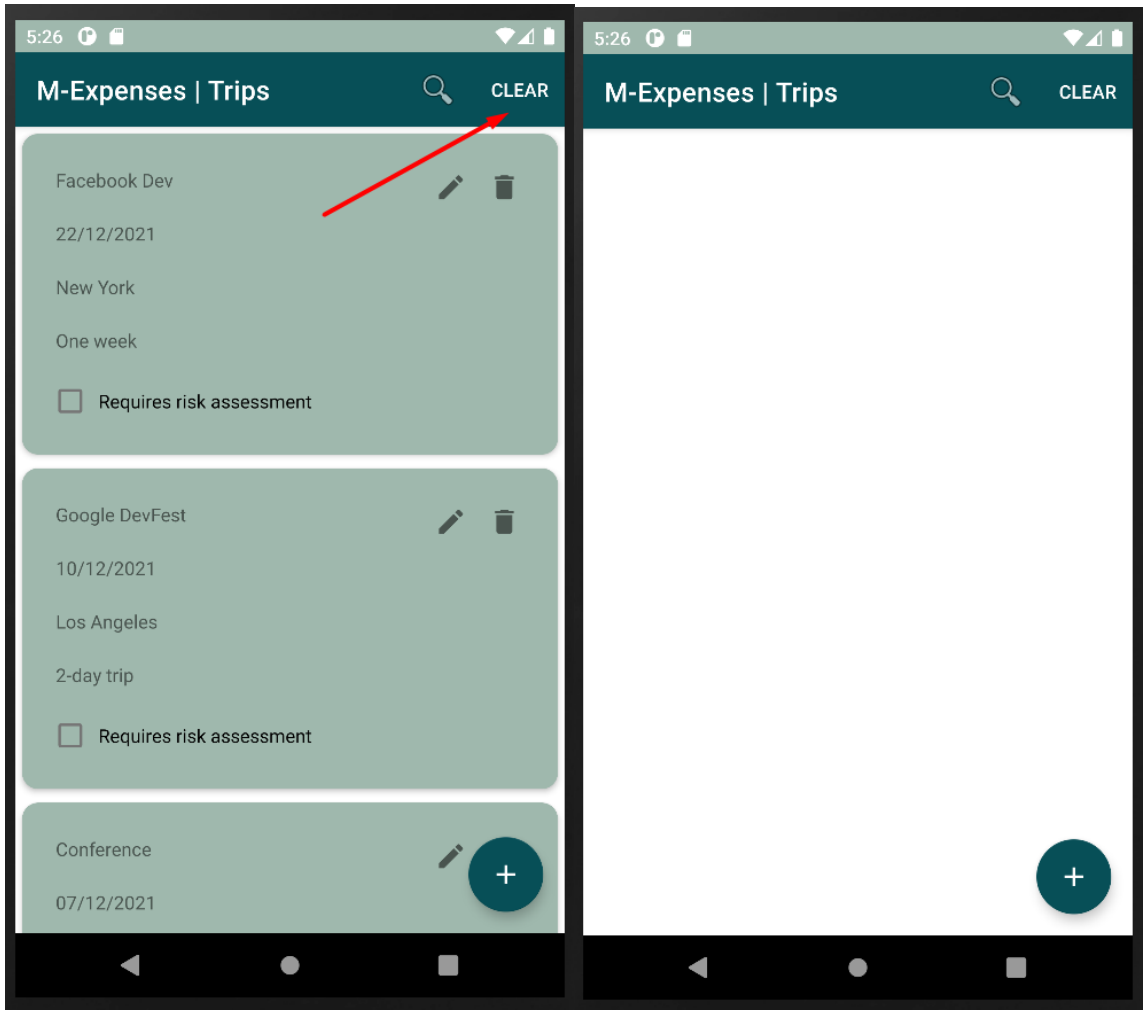


Feature Two: Advanced Searching

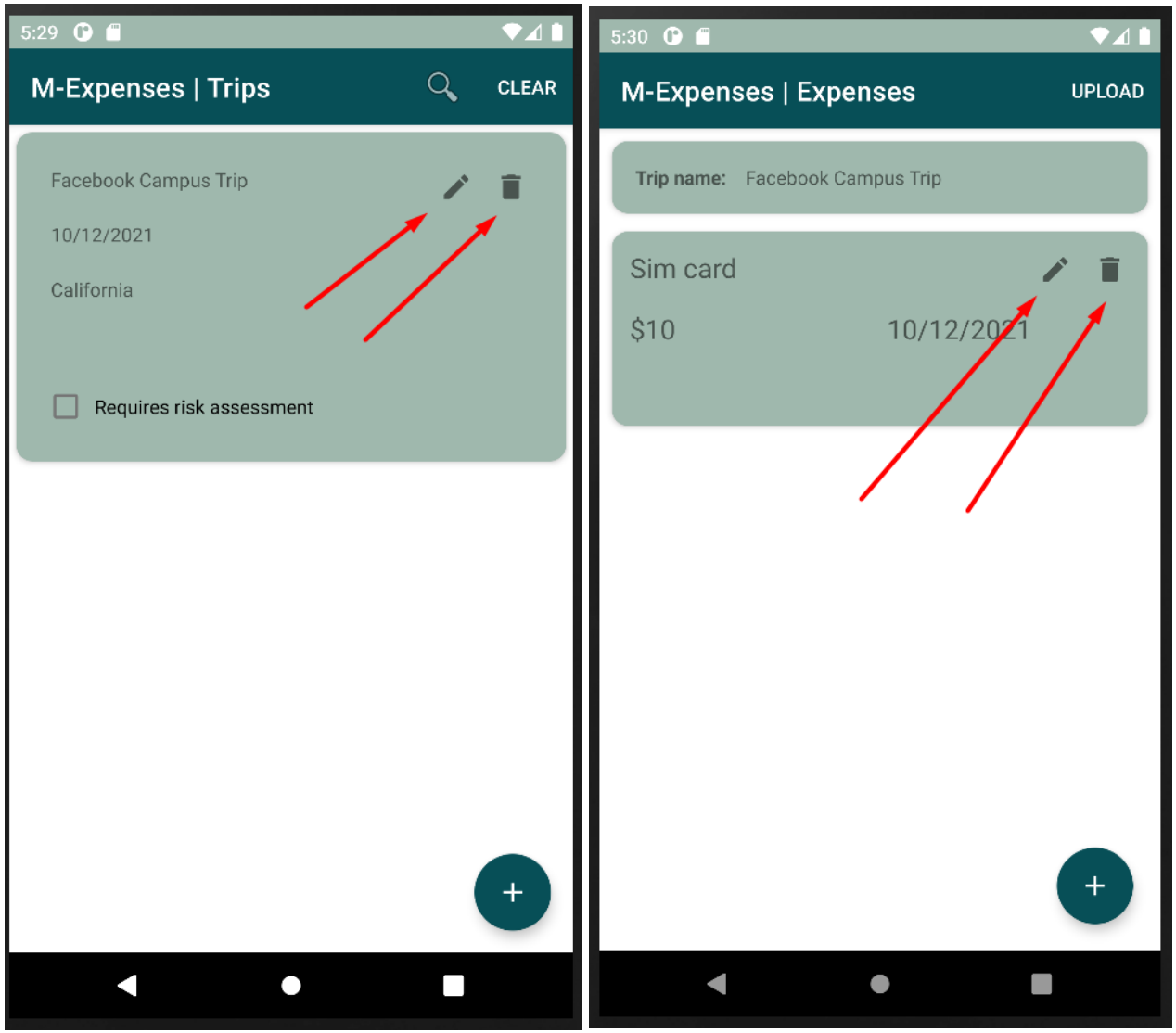
Searching done by Trip name, Destination and Date.



Feature Three: Clear – to delete all trips

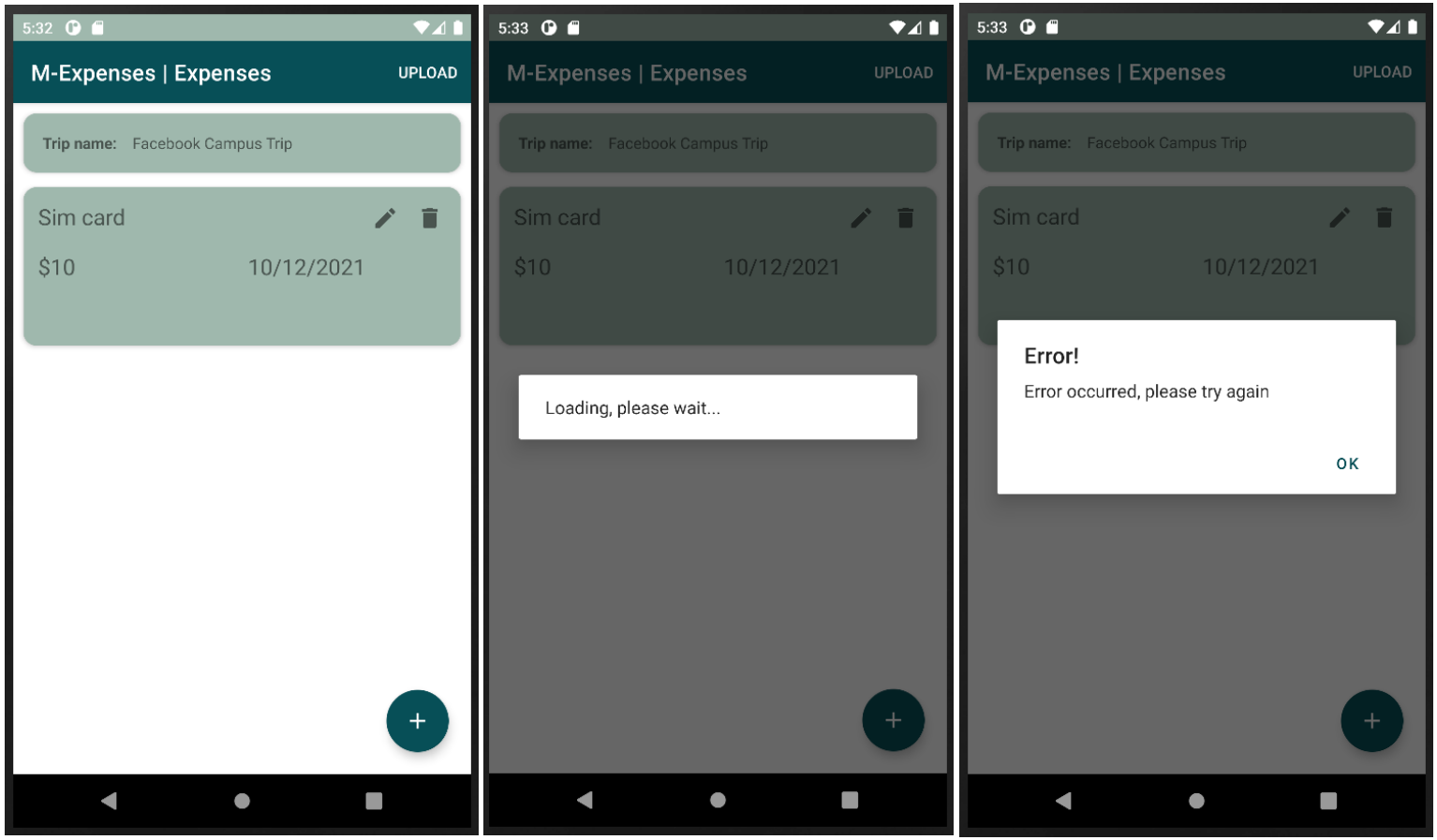


Feature Four: “Edit” and “Delete” buttons for Trips and Expenses

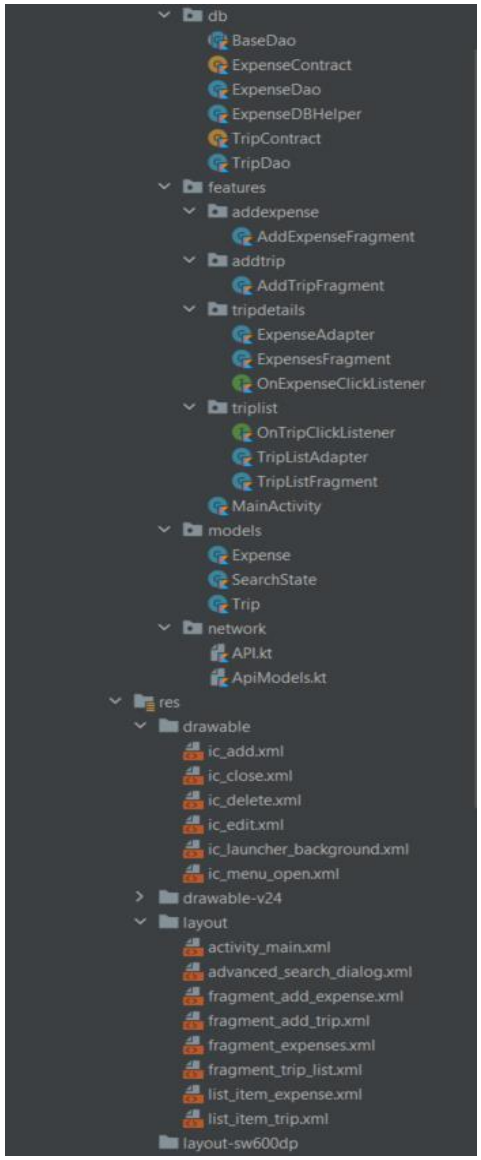


Feature Five: Upload Expenses Data

The user can upload entered expenses. If the user doesn't have an internet connection, he/she will get an error.



Section 5



Package name	**File Name**
db	BaseDao.kt
db	ExpenseContract.kt
db	ExpenseDao.kt
db	ExpenseDBHelper.kt
db	TripContract.kt
db	TripDao.kt
features	
features /addexpenses	AddExpensesFragment.kt
features /addtrip	AddTripFragment.kt
features /tripdetails	ExpenseAdapter.kt
features / tripdetails	ExpensesFragment.kt
features / tripdetails	OnExpenseClickListener.kt
features/ triplist	OnTripClickListener.kt
features/ triplist	TripListAdapter.kt
features/ triplist	TripListFragment.kt
features	MainActivity.kt
models	Expense.kt
models	SearchState.kt
models	Trip.kt
network	API.kt
Network	ApiModel.kt
drawable	ic_add.xml
drawable	ic_close.xml
drawable	ic_delete.xml
drawable	ic_edit.xml
drawable	ic_menu_open.xml
layout	activity_main.xml
layout	advanced_search_dialog.xml
layout	fragment_add_expense.xml

layout	fragment_add_trip.xml
layout	fragment_expense.xml
layout	fragment_trip_list.xml
layout	list_item_expense.xml
layout	list_item_trip.xml
navigation	nav_graph.xml