



**PERSONAL EXPENSE TRACKER APPLICATION**  
**NALAIYA THIRAN PROJECT BASED LEARNING**

**ON**

**PROFESSIONAL READINESS FOR INNOVATION,  
EMPLOYABILITY AND ENTREPRENEURSHIP**

**PROJECT REPORT**

***Submitted by***

K.MARUTHI (820619104027)

R.MOHAMED FAYAZ (820619104029)

S.MOHAMED RIYAS (820619104031)

K.PUNITHAN (820619104044)

**COMPUTER SCIENCE AND ENGINEERING**  
**ARASU ENGINEERING COLLEGE, KUMBAKONAM**

**ANNA UNIVERSITY: CHENNAI 600 02**

**NOVEMBER-2022**

# **CONTENTS**

## **1. INTRODUCTION**

**1.1 Project Overview 1.2 Purpose**

## **2. LITERATURE SURVEY**

**2.1 Existing problem**

**2.2 References**

**2.3 Problem Statement Definition**

## **3. IDEATION & PROPOSED SOLUTION**

**3.1 Empathy Map Canvas**

**3.2 Ideation & Brainstorming**

**3.3 Proposed Solution**

**3.4 Problem Solution fit**

## **4. REQUIREMENT ANALYSIS**

**4.1 Functional requirement**

**4.2 Non-Functional requirements**

## **5. PROJECT DESIGN**

**5.1 Data Flow Diagrams**

## **5.2 Solution & Technical Architecture**

## **5.3 User Stories**

# **6. PROJECT PLANNING & SCHEDULING**

## **6.1 Sprint Planning & Estimation**

## **6.2 Sprint Delivery Schedule**

## **6.3 Reports from JIRA**

# **7. CODING & SOLUTIONING (Explain the features added in the project along with code)**

## **7.1 Feature 1**

## **7.2 Feature 2**

## **7.3 Database Schema (if Applicable)**

# **8. TESTING**

## **8.1 Test Cases**

## **8.2 User Acceptance Testing**

# **9. RESULTS**

## **9.1 Performance Metrics**

# **10. ADVANTAGES & DISADVANTAGES**

**11. CONCLUSION**

**12. FUTURE SCOPE**

**13. APPENDIX Source Code**

**GitHub & Project Demo Link**

# 1. INTRODUCTION

## 1.1. Project Overview

The most demanded skills for data scientists Python, R, SQL, and the list goes on and on. There are many surveys and reports that show some good statistics on popular data skills. In this post, I am going to gather first-hand information by scraping data science jobs from indeed.ca, analyze top skills required by employers, and make job recommendations by matching skills from resume to posted jobs. It will be fun! Python code can be found on my [GitHub](#). This post is part of a series of people analytics experiments I am putting together: [Job skill match](#) (Recruitment ) [Employee attrition prediction](#) (Employee Management) Pay gap by gender, ethnicity, profession (Employee Compensation) FUTURE WORK Organizational network analysis (ONA) FUTURE WORK Web Scraping I like scraping data from Internet because it is free! But you should not abuse it just because it is free. There are basic rules to follow so we do not get our cyber friends and reject ourselves from visiting their websites. All can be summed in two words: BE NICE. Read their terms and conditions on the website first, and space out your requests so that their website does not get hit too hard. Here is a very good [web scraping 101](#). I scraped data science jobs from indeed.ca, the largest global recruiting website. I gathered data scientist/engineer/analyst jobs posted in the last 30 days (09/19/2018 – 10/19/2018), in 6 major Canadian cities, i.e. Toronto, Montreal, Vancouver, Ottawa, Calgary, and Edmonton. I used Selenium Webdriver to automate web scraping and saved results in a local JSON file. In total 367 job postings were retrieved, and it took about 20 minutes. Job Description Keyword Extraction For each job, I tokenized its job description, cleaned up the list by removing words defined in the NLTK list of stopwords, and finally filtered on a list of popular data science related skill words. Bar chart below shows top 30 data skills required by most employers . More than 60% jobs requires SQL, 48% requires Python, and 32% requires R. No surprise. It is worth noting that Excel is actually quite a common requirement, which I would not normally considered a serious data skill. Agile is also among the top required skills. AWS knowledge is more demanded than Azure or GCP.

## 1.2. Purpose

An expense tracker is a desktop application that keeps track of all your expenses and stores all the information regarding them, including the person to whom you have paid the money (also called payee) and the reason why you paid the money. The objective of this project is to create a GUI based Expense Tracker. To build this, you will need an intermediate understanding of the Tkinter library, SQL language and its commands, and basic understanding of the message box module, ttk.Treeview widget and tkcalendar library.

## 2. LITERATURE SURVEY

### 2.1. Existing problem

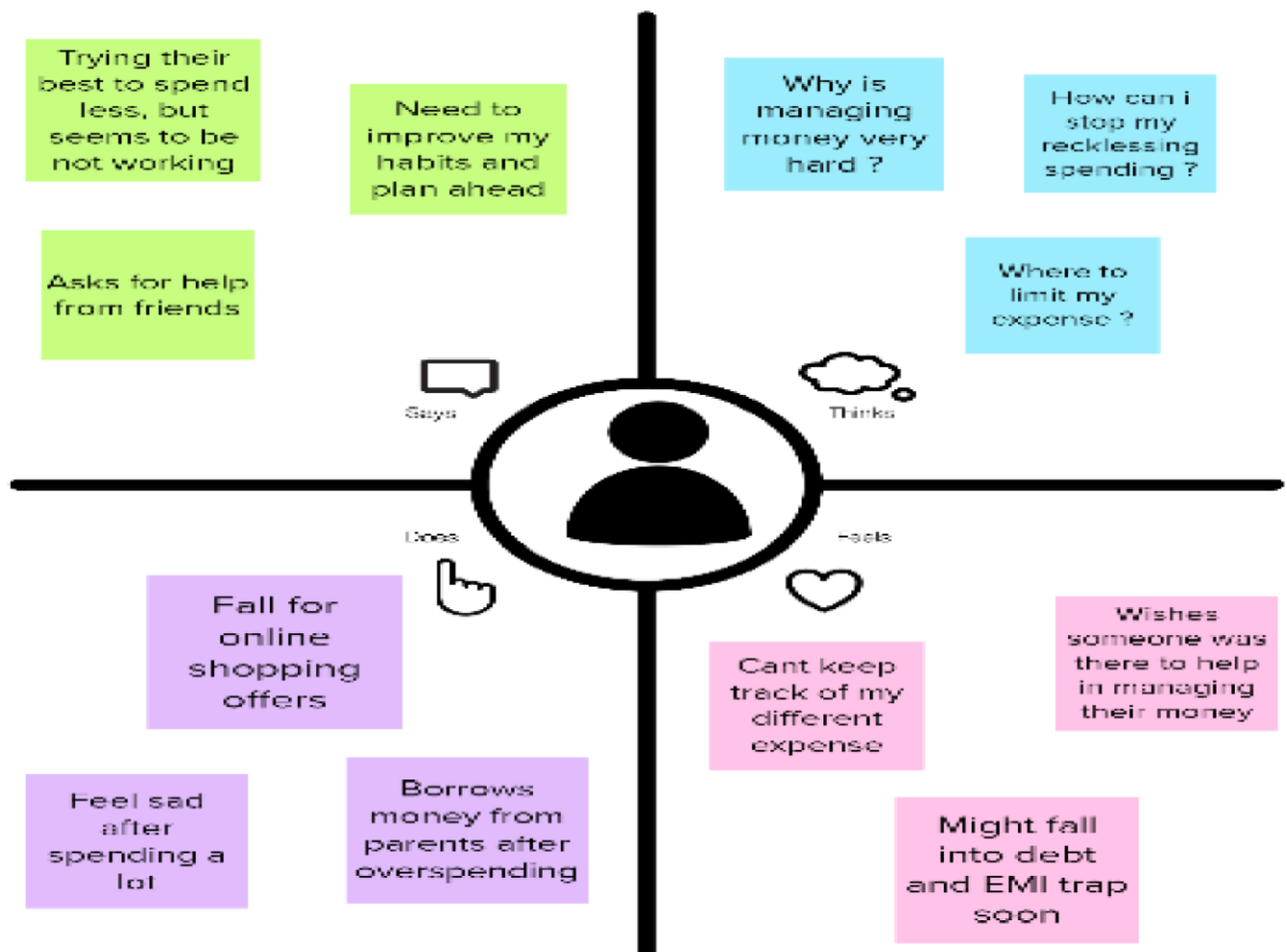
Money View App reads all of the transactional SMS messages and provides you with real-time visibility into your finances. This app unearths the hidden financial data that sits idly in SMS logs and makes excellent use of it. This personal finance manager app acts as a proactive budget planner, assisting you in staying on top of your budget, bills, and finances. The personal finance app was designed for simple, real-time budget and financial tracking, making it one of the best expense tracker apps in India. You can use the budget planner and spending tracker to keep track of your personal and business financial transactions, review financial data on a daily/weekly/monthly basis, and manage your assets. Monefy tracks the user's expenses and compares them to the monthly income and the budget planner. Monefy's money manager app keeps your monthly budget in top shape. As a result, it could also serve as the best expense tracker app. Wallet can automatically track your daily expenses by syncing your bank account, view weekly expense reports, plan your shopping expenses, and share specific features with your loved ones. You can manage your money with a wallet from anywhere and at any time. Walnut automates and secures the tracking of your monthly expenses. You can stay within your budget, pay your bills on time, and save more money each month by using the Walnut app. They also provide personal loans.

### 2.2. References

1. <https://moneyview.in/insights/best-personal-finance-management-apps-in-india>
2. <https://www.factmr.com/report/personal-finance-mobile-app-market>
3. <https://www.moneytap.com/blog/best-money-management-apps/>
4. <https://relevant.software/blog/personal-finance-app-like-mint/>
5. <https://www.onmanorama.com/lifestyle/news/2022/01/11/financial-literacy-trend-among-todays-youth-investment.html>
6. <https://www.livemint.com/money/personal-finance/96-indian-parents-feel-their-children-lack-financial-know-how-survey-11661336110855.html>
7. <https://economictimes.indiatimes.com/smallbiz/money/importance>

## Personal Expense Tracker Application

- Analysing the mindset of a typical target audience of this solution



8. [https://www.news18.com/news/education-career/only-27-adults-16-7-of-in](https://www.news18.com/news/education-career/only-27-adults-16-7-of-indian-teenagers-financially-literate-4644893.html)

[dian-teenagers-financially-literate-4644893.html](https://www.news18.com/news/education-career/only-27-adults-16-7-of-indian-teenagers-financially-literate-4644893.html)

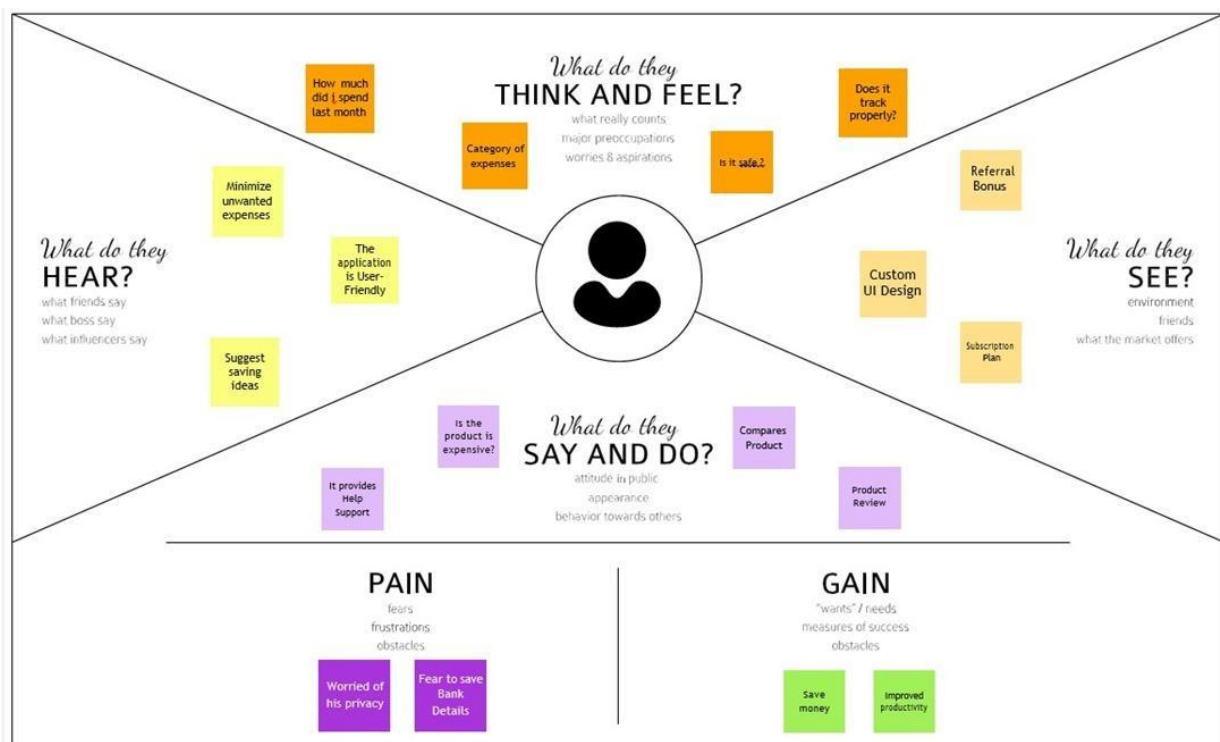
### 2.3. Problem Statement Definition

Modern education does not focus on finance management. This is primarily due to lack of resources and the Indian value system on giving money to children. Failing to teach this valuable knowledge had left many Indians to recklessly spend their income and fall into vicious cycles of EMI and debt. Many of them are just a month's salary away from bankruptcy. This issue is tackled by providing a web application for where people can plan their monthly expenses into categories, set alerts and get visual insights from their spending patterns. Who does the problem affect Young adults and earning

middle class citizens? What is the issue? Lack of financial literacy among people When does the issue occur Primarily when the person moves from college to job and starts earning their own money. Where is the issue occurring Especially among young engineers who are newly exposed to consumer centric market and services.

## 1. IDEATION & PROPOSED SOLUTION

### 3.1. Empathy Map Canvas



### 3.2. Ideation & Brainstorming





## Brainstorm & ideaprioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

- 10 minutes to prepare
- 1 hour to collaborate
- 2-8 people recommended



### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

10 minutes



#### Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.



#### Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.



#### Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

Open article →



### Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

5 minutes



#### Key rules of brainstorming

To run an smooth and productive session



### Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

Kavinaya N

Day to Day reminder about account balance via mail

Afra Thahseen J

If the user spent high, then send mail

Abdul Waseem Nihazi KW

If they logged in to the web app, we can send some alert

Jayant PS

Sending mail when exceeds the limit



### Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. In the last 10 minutes, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

20 minutes





### 3.3. Proposed Solution

**Problem Statement** Building a personal finance tracking application that will imbibe good spending habits into students. (Problem to be solved)

**Idea / Solution description** To build a web application that is deployed in IBM cloud and leverage mailing service like sendgrid to implement the same

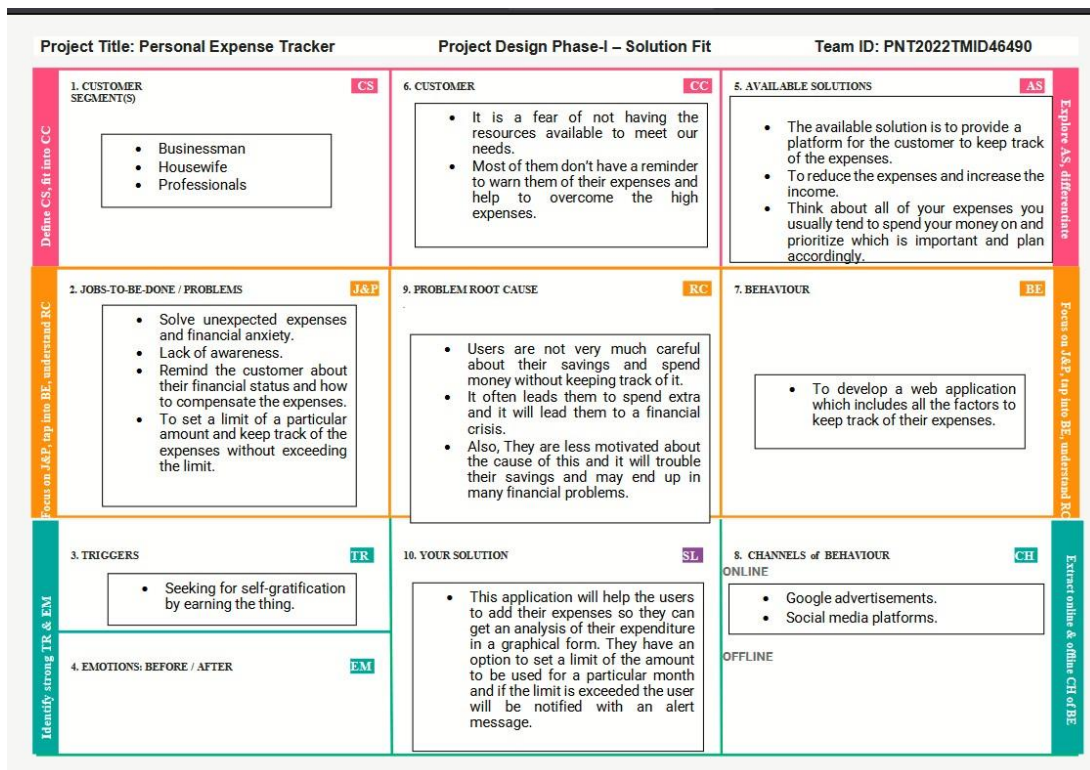
**Novelty / Uniqueness** The stats generated with visual graphs are more effective than log books. It also helps in using technology to gain better insights from Social Impact / Customer Satisfaction

**Better financial knowledge is gained.** Gamified approach can be used to give self satisfaction. Reduced chances of bad debt in future

**Business Model (Revenue Model)** Subscription can be incorporated to access premium tools within the app.

**Scalability of the Solution as deployment.** As the application is containerized for It can be easily scaled in a cloud service provider like IBM.

### 3.4. Problem Solution



## 4. REQUIREMENT ANALYSIS

### 4.1. Functional requirement

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIn
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	Forgot Password	Resetting the password by sending an OTP to user's mail
FR-4	User Login	Login through Login form

FR - 5	Dashboard	User can add the expense and can evaluate them using the provided options
FR - 6	Result Page	Shows the user result

### Non-functional Requirements:

?? A non-functional requirement defines the quality attribute of a software system

?? It places constraint on "How should the software system fulfil the functional requirements?" ?? It is not mandatory

?? Applied to system as a whole

?? Usually more difficult to define

?? Helps you verify the performance of the software

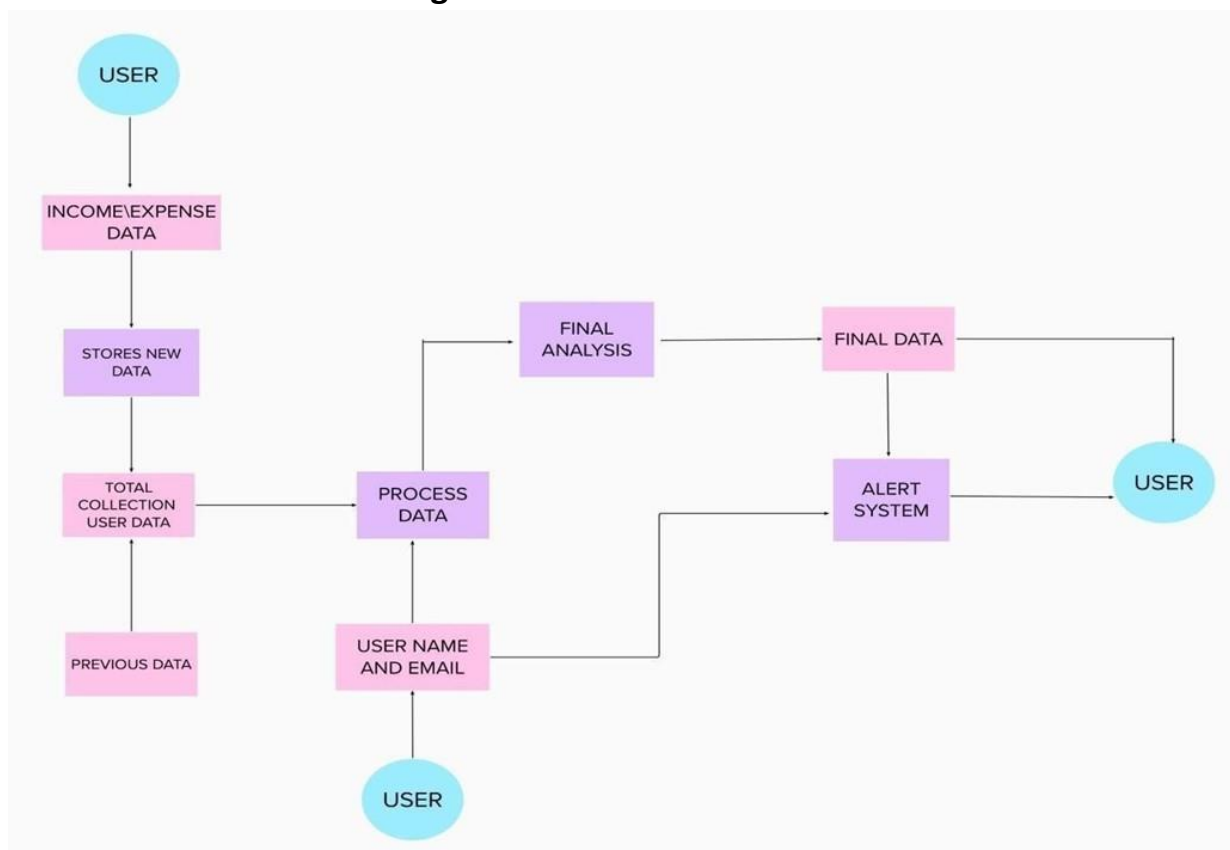
Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	<b>Usability</b>	Customers can use the application in almost all the web browsers. Application is with good looking and detailed UI, which makes it more friendly to use.
NFR-2	<b>Security</b>	Customers are asked to create an account for themselves using their email which is protected with an 8 character-long password, making it more secure.

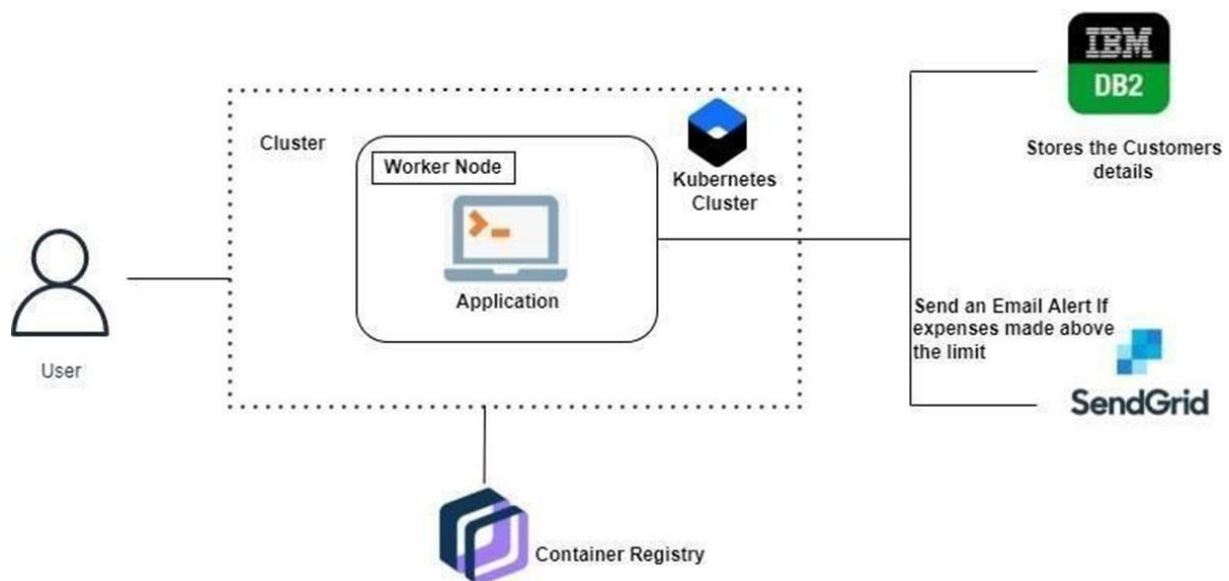
NFR-3	<b>Reliability</b>	If customer face any problem , then they can sent queries messages
NFR-4	<b>Performance</b>	Customers will have a smooth experience while using the application, as it is simple and is well optimised.
NFR-5	<b>Availability</b>	Application is available 24/7 as it is hosted on IBM Cloud
NFR-6	<b>Scalability</b>	In future, may be cross-platform mobile applications can be developed as the user base grows.

## 5. PROJECT DESIGN

## 5.1. Data Flow Diagrams



## 5.2. Solution & Technical Architecture



S.No.	Component	Description	Technology
1.	User Interface	The user can Interact with the application with use of Chatbot	HTML, CSS, JavaScript / Angular Js / React Js etc.

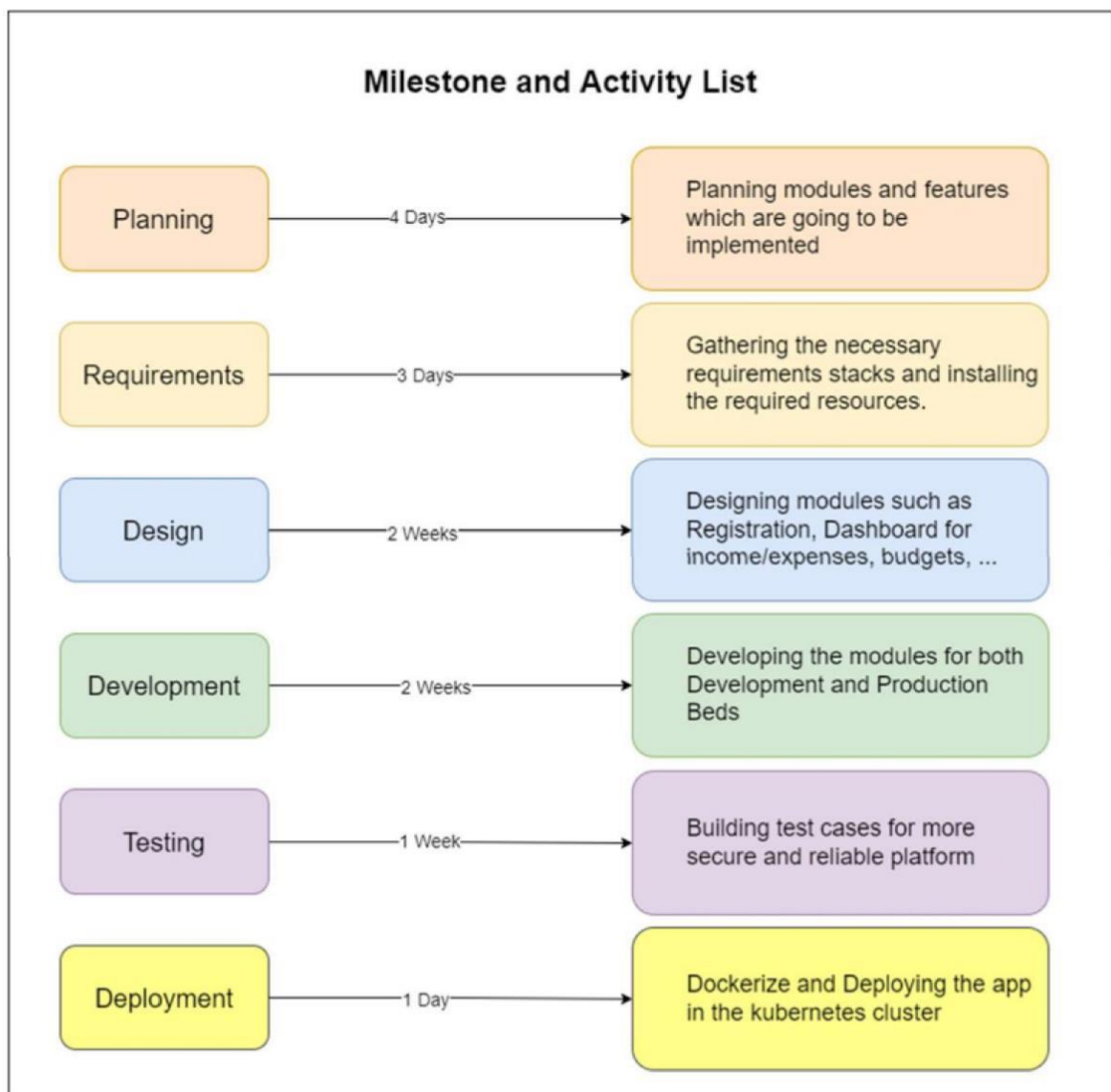
2.	Application Logic-1	The application contains the sign in/sign up where the user will login into the main dashboard	Java / Python
3.	Application Logic-2	Dashboard contains the fields like Add income, Add Expenses, Save Money	IBM Watson STT service
4.	Application Logic-3	The user will get the expense report in the graph form and also get alerts if the expense limit exceeds	IBM Watson Assistant, SendGrid
5.	Database	The Income and Expense data are stored in the MySQL database	MySQL, NoSQL, etc.
6.	Cloud Database	With use of Database Service on Cloud, the User data are stored in a well secured Manner	IBM DB2, IBM Cloudant etc.

S.No.	Characteristics	Description	Technology
1.	Open-Source Frameworks	Flask Framework in Python is used to implement this Application	Python-Flask
2.	Security Implementations	This Application Provides high security to the user Financial data. It can be done by using the Container Registry in IBM cloud	Container Registry, Kubernetes Cluster
3.	Scalable Architecture	Expense Tracker is a life time access supplication. It's	Container Registry, Kubernetes Cluster
		demand will increase when the user's income are high	

4.	Availability	This application will be available to the user at any part of time	Container Registry, Kubernetes Cluster
----	--------------	--	--

## 6. PROJECT PLANNING & SCHEDULING

### 6.1. Sprint Planning & Estimation



### 6.2. Sprint Delivery Schedule



Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint 1	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming my password.	2	High	Mohamed Fayaz
		USN-2	As a user, I will receive confirmation email once I have registered for the application	1	High	Maruthi
	Login	USN-3	As a user, I can log into the application by entering email & password	1	High	Mohamed Riyas
	Dashboard	USN-4	Logging in takes to the dashboard for the logged user.	2	High	Punithan
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						
Sprint 2	Workspace	USN-1	Workspace for personal expense tracking	2	High	Punithan
	Charts	USN-2	Creating various graphs and statistics of customer's data	1	Medium	Mohamed Riyas
	Connecting to IBM DB2	USN-3	Linking database with dashboard	2	High	Mohamed Fayaz
		USN-4	Making dashboard interactive with JS	2	High	Mohamed Riyas
Sprint-3		USN-1	Wrapping up the server side works of frontend	1	Medium	Punithan
	Watson Assistant	USN-2	Creating Chatbot for expense tracking and for clarifying user's query	1	Medium	Mohamed Fayaz
	SendGrid	USN-3	Using SendGrid to send mail to the user about their expenses	1	Low	Mohamed Riyas
		USN-4	Integrating both frontend and backend	2		Mohamed Riyas
Bug fixes, routine checks and improvisation by everyone in the team *Intended bugs only						

<b>Sprint-4</b>	Docker	USN-1	Creating image of website using docker/	2	High	Mohamed Riyas
	Cloud Registry	USN-2	Uploading docker image to IBM Cloud registry	2	High	Mohamed Fayaz
	Kubernetes	USN-3	Create container using the docker image and hosting the site	2	High	Punithan
	Exposing	USN-4	Exposing IP/Ports for the site	2	High	Mohamed Riyas

### Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

### Velocity

We have a 6-day sprint duration, and the velocity of the team is 20 (points per sprint).  
Calculating the team's average velocity (AV).

$$AV = \frac{\text{Total Story Points}}{\text{Sprint duration}} = \frac{20}{6}$$

3.33 velocity

## 1. CODING & SOLUTIONING

### 7.1. Feature 1

**Handle Documents** It's time to stop using paper and excel spreadsheets for keeping records of your cash payments and online transactions! Papers are tough to handle and more dangerous for the environment. On the other hand, excel sheets may offer an online solution but don't do much help in money handling. So, it's better to develop money management software that collects insights from the data and helps make business decisions.

**Tracks Receipts** You always can't find the cash and digital payments made by you and this is a big issue with tracking expenses. So, if you want to keep a track of your monetary investments, you should go using a business expense tracker app. This helps store all receipts by only clicking their images in your expenses handling app.

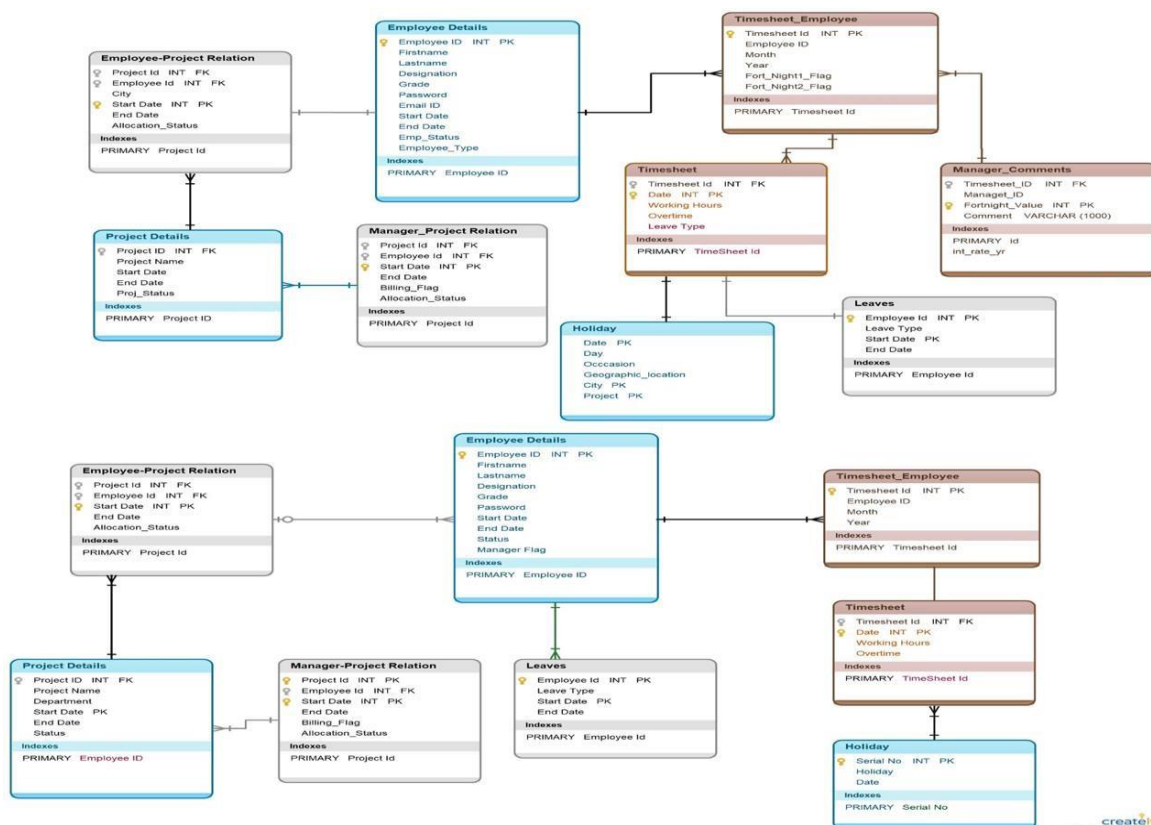
**Prevents Data Losses and Frauds** Manual handling of personal expenses and finances can't check every transaction detail

accurately. For this reason, fraud cases happen many times. Using an expense tracking and budgeting app, the workflow of money and finance handling becomes automated. This not just prevents fraudulence but also makes the procedure more accurate and transparent.

## 7.2. Feature 2

Mitigates Human Errors We cannot afford mistakes when it comes to handling budgets and finances. However, humans may make some errors because of misunderstanding, carelessness, or negligence. With the help of an expense handling app, you can lower as well as prevent every mistake caused because of carelessness. Offers Precise Analytics Excel spreadsheets might track and store data and create helpful charts or graphs from it but still have no advanced functionality. On the other hand, human engagement brings the possibility of mistakes. So, it's best to build a business expense tracker app that carries out prediction analysis and helps you make efficient business decisions.

## 7.3. Database Schema



## 8. TESTING

### 8.1. Test Cases

With a concerted effort, I conducted research on general well-being to have a rudimentary grasp on health management, as well as the existing job/skill recommender apps in order to get an understanding of what is already existing in the market, the characteristics, specialties, and usability. There are a considerable

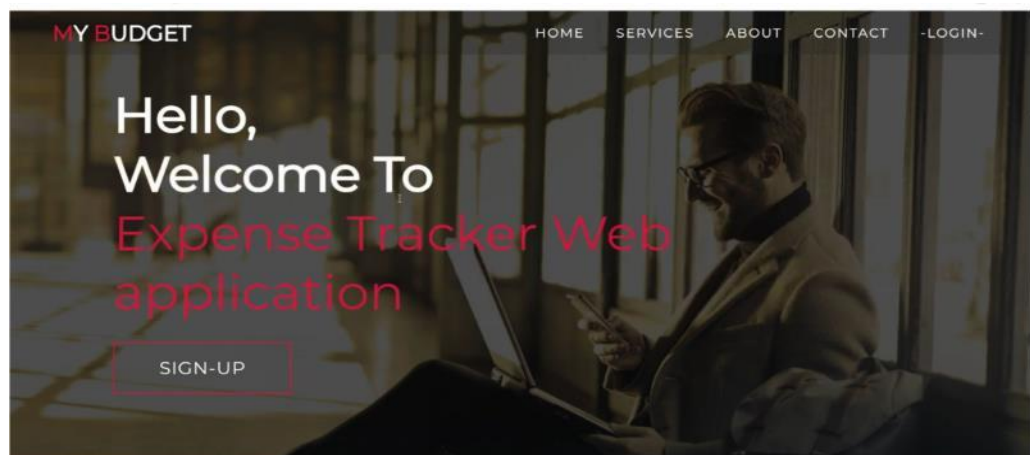
number of job/skill recommender apps tracking apps existing in the market. They aim to track daily spends intake by logging info given to achieve users' preset goals. To log spend, users can input the expense in the app, or scan the barcode of a package. Most apps allow users to connect with associated activities apps to track spend progress. With a premium upgrade, users can get access to tailor-made saving according to health goals or specified way to spend. In order to build a realistic initial target group, I wanted to conduct some usability tests with 5 users that regularly engage in buying activity and spending tracking, including both first-time and regular users of money planning. I asked these individuals to perform tasks related to general usage


## 8.2. User Acceptance Testing

Must-have features of a Job/Skill recommender app I wanted to address the user pain points by including (and improving) the core features of the application. Personal profiles After downloading the app, a user needs to register and create an account. At this stage, users should fill in personal information like name, gender, age, height, weight, spend preferences, spend logging and dashboard. Allowing users to analyze their spending habits. They should be able to log expenses and money intake and see their progress on a dashboard that can track overall spends. Push notifications Push notifications are an effective tool for increasing user engagement and retention. To motivate users to keep moving toward their goals, it's pertinent to deliver information on their progress toward the current goal and remind them to log what they spend on. money counter Enabling the application to calculate spent amount of users have gone and done based on the data they've logged. Barcode scanner Let users count money and see accurate spend information via a built-in barcode scanner.


## 9. RESULTS


### 9.1. Performance Metrics






# Hello, Friend












OR

Sign-up with








☐ I read and agree to [Terms & Conditions](#)

**CREATE ACCOUNT**

[Already have an account? Sign in](#)

## Glad to see you



**Welcome.** Please Fill in the blanks for sign up

MyBudget
Home
Add
History
LIMIT
Report
User

### Add Expense

Date


Expense name

Expense Amount

Pay-Mode

Category

Add



MyBudget
Home
Add
History
LIMIT
Report
User

### EXPENSES

2021-05-25 22:14:00	biryani	₹ 500	payment	food	Edit	Delete
2021-05-25 17:50:00	eeeeee	₹ 52825433	cash	entertainment	Edit	Delete
2021-05-25 14:50:00	qdfwegg	₹ 25786558	debitcard	food	Edit	Delete
2021-05-25 11:49:00	ffffff	₹ 25574242	onlinebanking	business	Edit	Delete
2021-05-23 18:49:00	v-game	₹ 60000	onlinebanking	EMI	Edit	Delete

MyBudget

HomeAddHistoryLIMITReport

User

Edit Expense

Date

25-05-2021 12:20

Expense name

skodacar

Expense Amount

52825433

debitcard

entertainment

food  
Entertainment  
Business  
Rent  
EMI  
other

MyBudget

HomeAddHistoryLIMITReport

User

Currently your MONTHLY limit is ₹ 900000

ENTER the MONTHLY LIMIT to avoid over EXPENSES

ENTER

MyBudget

HomeAddHistoryLIMITReport

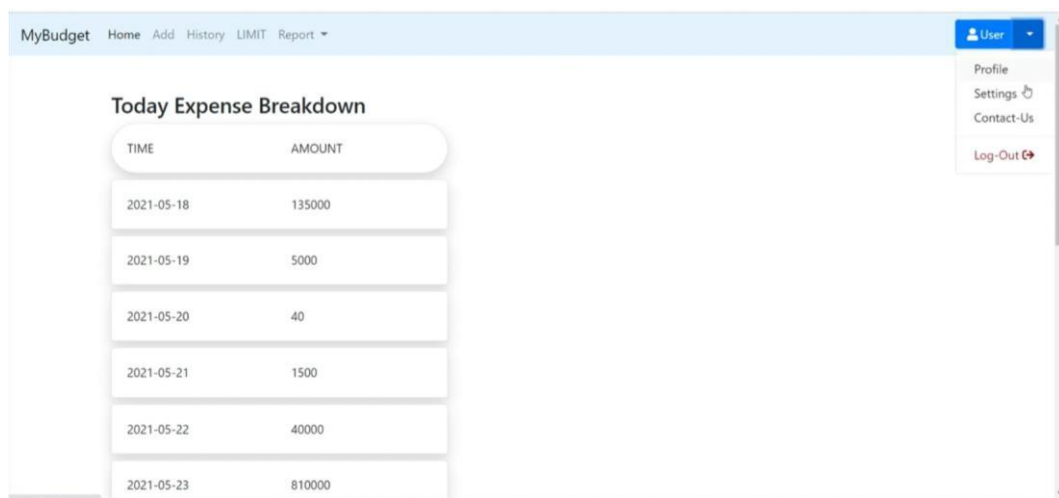
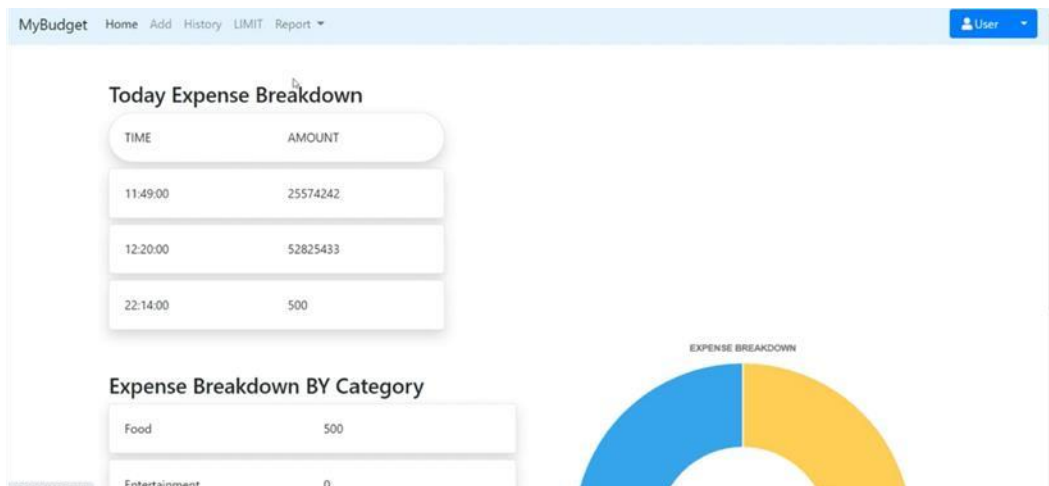
User

Currently your MONTHLY limit is ₹ 900000

ENTER the MONTHLY LIMIT to avoid over EXPENSES

1000000

ENTER



## 1. ADVANTAGES & DISADVANTAGES

### 10.1 Advantages:-

I have a cheaper cell phone plan, using a smaller provider than the big monopoly providers. I don't watch tv, so no cable, though my husband has a Netflix account and I'll watch comedy specials or scifi series on occasion. I don't subscribe to any music streaming or video games. Come to think of it, I've never been that much into entertainment – I don't buy tickets to concerts, sports games, or movies (I am in the minority for sure – a lot of people around me love watching movies but I forget movies so fast so I hardly go to the theater). What do I do for fun? I take walks with my family, work in the garden, read, surf online, nap, and I have a lot of occupations to keep me busy. I don't buy books, I borrow them at the library. I always have a lot of books on hold or on renewal for my family. I listen to a ton of audiobooks when I drive to and from work everyday, they are such a source of delight for me. I don't have healthcare premiums – I live in Canada. I don't eat out too much – we do batch cooking, I always bring my own lunch, snacks, and coffee in a thermos to work. I am adamant about perfectly planned meals that incorporate fiber, protein, fats, carbs, and high water content fruit. Plus I am very picky about what I buy at the grocery store. We do not eat processed junk food or soda, I mostly buy high-quality meat, fruit, and dairy. Rice is very economical. I also grind my own coffee beans and bring my coffee to work, I NEVER buy Starbucks and I never eat at the cafeteria. Why should I pay more for inferior food and drink prepared by people who don't have health as a priority? I have a SodaStream, which is one of life's joys. I love fizzy sparkling water, and now I never have to buy club soda again. I can use tap water and my SodaStream carbonates it for me! I even

drink more water now because of it, and I bring it in a water bottle when I'm out and about. I don't shop for clothes – at age 41, I have enough clothes already and still fit and wear the same size clothes as when I was a teenager. Since I always use a drying rack instead of the dryer, my clothes never wear out either. Over the years I've donated the ones that I don't wear, and kept the ones that I do. Plus, the hospital provides sterile scrubs for work which is free! I am happy with my wardrobe and usually wear cheap Uniqlo leggings with a dress that is 20 years old.

#### 10.2 Disadvantages:-

Your information is less secure, and probably being used and sold. If the service is free, then the product is you. Mint.com, like other financial apps, is a free service. They have to pay their bills somehow, so regardless of what their privacy policy may or may not say, just assume that your spending history and trends are going to be recorded and analyzed, by someone, somewhere. Now, you shouldn't have to worry about credit card fraud or [identity theft](#), these companies are large enough and secure enough that you'll never have to worry about something like that. Just recognize that your information, most likely anonymous, will be used and potentially even sold. Personally, I have no problem with that, but if you do, then make sure you avoid these types of services.

Automating everything to do with your finances can make you financially lazy. If your bills are paid automatically and your finances are track automatically, then what is there left for you to do? Not a lot, to be honest. So you might stop [caring about what you're spending](#) and where your money is going. Eventually you may look at your Mint data and realize that you've blown your budget over the last two months, but by then it is too late. So if you do choose to use this program, ensure that you are also being diligent in checking in on your finances. Set up a weekly or biweekly check for yourself to go through your finances and hit on all the important points.

## 2. CONCLUSION

In this paper, we proposed a framework for job recommendation task. This framework facilitates the understanding of job recommendation process as well as it allows the use of a variety of text processing and recommendation methods according to the preferences of the job recommender system designer. Moreover, we also contribute making publicly available a new dataset containing job seekers profiles and job vacancies. Future directions of our work will focus on performing a more exhaustive evaluation considering a greater amount of methods and data as well as a comprehensive evaluation of the impact of each professional skill of a job seeker on the received job recommendation.

## 3. FUTURE SCOPE

Changing face of expense management software Year-on-year, modern expense management software underwent a continuous evolution from traditional back-office function to strategic internal set of processes. But would it be sufficient to meet the needs of next-gen companies? Have you ever thought about what the next-generation software should look like? As the requirements of companies evolve continuously, the software should undergo a series of changes to meet the growing needs of next generation companies. The next-generation travel and expense (T & E) management apps should not only just accelerate the expense management process but also should come with mobile and cloud integration capabilities that add tremendous value to the business bottom line. Future T & E management software should be able to provide greater visibility into spending and standardize critical procedures. Next-gen expense management A recent T & E study unveiled that visibility and intelligence are the two key aspects that companies look forward to understanding spending associated with business travel. Analytics is also on the priority list of the



best-in-class organizations. The motto is not just to enhance the existing process but also to leverage analytical capabilities and visibility that can help companies drive efficiency, forecast and plan better for corporate finances. Apparently, as per research, integration, analytics and mobile apps are the three key factors that can help companies succeed at a faster pace. When incorporated, these factors add edge and value to the businesses. Integration between corporate cards and expense management software increases transparency and makes the process effortless throughout the expense report cycle. Analytics Increased intelligence provides you with an unheralded level of visibility into travel spending and enhances overall T & E intelligence. Companies can measure the true performance of any business trip by evaluating the ROI. Efficiency complimented by intelligence proves to be a great way to take the business to new heights. Need for mobile applications Mobile apps provide employees with the opportunity to manage expenses on the go. It gives both the companies and employees the flexibility that they need in managing the expense related activities. In fact, it increases accuracy as the power of technology is put into the hands of both employees and employers. On a final note, the next generation software should be something that gives users an unprecedented experience in expense management and allows organizations to emphasize on what's really important to make their businesses better.

## 13. APPENDIX

### 13.1. Source Code

```
package com.github.ematiyuk.expensetracer.activities;
```

```
import android.os.Bundle; import android.support.annotation.LayoutRes; import
android.support.annotation.Nullable; import android.support.v4.app.Fragment; import
android.support.v4.app.FragmentManager; import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar; import com.github.ematiyuk.expensetracer.R;
public abstract class BaseFragmentActivity extends AppCompatActivity { protected Toolbar
mToolbar;
```

```
@LayoutRes
```

```
protected int getLayoutResId() { return
```

```
R.layout.activity_base;
```

```
}
```

```
@Override protected void onCreate(@Nullable Bundle savedInstanceState) {
```

```
super.onCreate(savedInstanceState); setContentView(getLayoutResId());
```

```
// Set a Toolbar to replace the ActionBar mToolbar = (Toolbar) findViewById(R.id.toolbar);
setSupportActionBar(mToolbar);
```

```
}
```

```
protected void insertFragment(Fragment fragment) {
```

```
// Insert the fragment by replacing any existing fragment
```

```
FragmentManager fragmentManager = getSupportFragmentManager(); fragmentManager.beginTransaction()
```

```
.replace(R.id.content_frame, fragment)
```

```
.commit();
```

```

} } package com.github.ematiyuk.expensetracer.activities; import android.os.Bundle;

import android.support.annotation.Nullable; import android.support.v7.app.ActionBar; import
com.github.ematiyuk.expensetracer.fragments.CategoryEditFragment; public class CategoryEditActivity
extends BaseFragmentActivity {

/* Important: use onCreate(Bundle savedInstanceState)

* instead of onCreate(Bundle savedInstanceState, PersistableBundle persistentState) */ @Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
super.onCreate(savedInstanceState); insertFragment(new CategoryEditFragment());
setupActionBar();
}

private void setupActionBar() {
ActionBar actionBar = getSupportActionBar(); if (actionBar != null) {

// Show the Up button in the action bar (toolbar). actionBar.setDisplayHomeAsUpEnabled(true);
}

} } package com.github.ematiyuk.expensetracer.activities;

import android.os.Bundle; import android.support.annotation.Nullable; import
android.support.v7.app.ActionBar; import
com.github.ematiyuk.expensetracer.fragments.ExpenseEditFragment; public class ExpenseEditActivity
extends BaseFragmentActivity {

/* Important: use onCreate(Bundle savedInstanceState)

* instead of onCreate(Bundle savedInstanceState, PersistableBundle persistentState) */ @Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
super.onCreate(savedInstanceState); insertFragment(new ExpenseEditFragment());
setupActionBar();
}

private void setupActionBar() {
ActionBar actionBar = getSupportActionBar(); if (actionBar != null) {

// Show the Up button in the action bar (toolbar). actionBar.setDisplayHomeAsUpEnabled(true);
}

} } package com.github.ematiyuk.expensetracer.activities;

import android.content.Intent; import android.content.res.Configuration; import android.os.Bundle; import
android.support.annotation.IdRes; import android.support.annotation.LayoutRes; import
android.support.annotation.Nullable; import android.support.design.widget.NavigationView; import
android.support.v4.app.Fragment; import android.support.v4.view.GravityCompat; import
android.support.v4.widget.DrawerLayout; import android.support.v7.app.ActionBarDrawerToggle; import
android.view.MenuItem;

```

```

import com.github.ematiyuk.expensetracer.fragments.CategoryFragment; import
com.github.ematiyuk.expensetracer.R; import
com.github.ematiyuk.expensetracer.fragments.ReportFragment; import
com.github.ematiyuk.expensetracer.fragments.TodayFragment; public
class MainActivity extends BaseFragmentActivity {

private DrawerLayout mDrawerLayout; private NavigationView mNavDrawer; private ActionBarDrawerToggle
mDrawerToggle;

@Override

@LayoutRes protected int getLayoutResId() { return
R.layout.activity_main;
}

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

mDrawerLayout = (DrawerLayout) findViewById(R.id.drawer_layout); mNavDrawer =
(NavigationView) findViewById(R.id.nav_drawer); mDrawerToggle = setupDrawerToggle();

// Tie DrawerLayout events to the ActionBarDrawerToggle mDrawerLayout.addDrawerListener(mDrawerToggle);
// Setup drawer view setupDrawerContent(mNavDrawer);
// Select TodayFragment on app start by default loadTodayFragment();
}

@Override protected void onPause() { super.onPause(); closeNavigationDrawer();
}

@Override public boolean onOptionsItemSelected(MenuItem item) {

// Pass the event to ActionBarDrawerToggle, if it returns // true, then it has handled the app icon
touch event if (mDrawerToggle.onOptionsItemSelected(item)) { return true; }

return super.onOptionsItemSelected(item);
}

@Override protected void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

// Sync the toggle state after onRestoreInstanceState has occurred. mDrawerToggle.syncState();
}

@Override public void onConfigurationChanged(Configuration newConfig) {

super.onConfigurationChanged(newConfig); // Pass any configuration change to the drawer toggle
mDrawerToggle.onConfigurationChanged(newConfig);
}
}

```

```

@Override public void onBackPressed() { if
(!closeNavigationDrawer()) {
Fragment currentFragment = getSupportFragmentManager()
.findFragmentById(R.id.content_frame); if (!(currentFragment instanceof TodayFragment)) {
loadTodayFragment();
} else {
// If current fragment is TodayFragment then exit super.onBackPressed();
}
}}

private ActionBarDrawerToggle setupDrawerToggle() { return new ActionBarDrawerToggle(this,
mDrawerLayout, mToolbar,

R.string.drawer_open, R.string.drawer_close); }

private void setupDrawerContent(NavigationView navigationView) {
navigationView.setNavigationItemSelectedListener( new NavigationView.OnNavigationItemSelectedListener()
{
@Override public boolean onNavigationItemSelectedListener(Menuitem menuitem) { selectDrawerItem(menuitem);
return true;
}
});

} private void selectDrawerItem(Menuitem menuitem) { closeNavigationDrawer();
switch(menuitem.getItemId()) { case R.id.nav_today:
loadFragment(TodayFragment.class, menuitem.getItemId(),
menuitem.getTitle()); break; case R.id.nav_report:
loadFragment(ReportFragment.class, menuitem.getItemId(),
menuitem.getTitle()); break; case R.id.nav_categories:
loadFragment(CategoryFragment.class, menuitem.getItemId(),
menuitem.getTitle()); break; case R.id.nav_settings:
startActivity(new Intent(MainActivity.this, SettingsActivity.class)); break; default:
loadFragment(TodayFragment.class, menuitem.getItemId(), menuitem.getTitle());
}}

private boolean closeNavigationDrawer() { boolean drawerIsOpen =
mDrawerLayout.isDrawerOpen(GravityCompat.START); if (drawerIsOpen) {
mDrawerLayout.closeDrawer(GravityCompat.START);

```

```

    } return drawerIsOpen;
}

public void hideNavigationBar() { closeNavigationDrawer();
}

private void loadFragment(Class fragmentClass, @IdRes int navDrawerCheckedItemId,
CharSequence toolbarTitle) { Fragment fragment =
null; try { fragment = (Fragment)
fragmentClass.newInstance();
} catch (Exception e) {
e.printStackTrace(); } insertFragment(fragment);
// Highlight the selected item mNavDrawer.setCheckedItem(navDrawerCheckedItemId); //
Set action bar title setTitle(toolbarTitle);
}

private void loadTodayFragment() { loadFragment(TodayFragment.class, R.id.nav_today,
getResources().getString(R.string.nav_today)); } } package
com.github.ematiyuk.expensetracer.activities;

import android.os.Bundle; import android.support.annotation.Nullable; import
android.support.v7.app.ActionBar;

import com.github.ematiyuk.expensetracer.R; import
com.github.ematiyuk.expensetracer.fragments.SettingsFragment; public class SettingsActivity extends
BaseFragmentActivity {

@Override protected void onCreate(@Nullable Bundle savedInstanceState) {
super.onCreate(savedInstanceState); insertFragment(new SettingsFragment());
setTitle(R.string.nav_settings); setupActionBar();
}

private void setupActionBar() {
ActionBar actionBar = getSupportActionBar(); if
(actionBar != null) {
// Show the Up button in the action bar (toolbar). actionBar.setDisplayHomeAsUpEnabled(true);
}
} } package com.github.ematiyuk.expensetracer.adapters;

import android.content.Context; import android.database.Cursor; import android.view.View; import
android.view.ViewGroup; import android.widget.TextView;

```

```

import com.github.ematiyuk.expensetracer.R; import
com.github.ematiyuk.expensetracer.providers.ExpensesContract; import
com.github.ematiyuk.expensetracer.utils.Utils; import
com.twotoasters.sectioncursoradapter.SectionCursorAdapter; public
class SectionExpenseAdapter extends SectionCursorAdapter { private
String mCurrency; public SectionExpenseAdapter(Context context) {
super(context, null, 0);
}

public void setCurrency(String currency) { mCurrency
= currency; notifyDataSetChanged();
}

@Override protected Object
getSectionFromCursor(Cursor cursor) {

String dateStr = cursor.getString(cursor.getColumnIndexOrThrow(ExpensesContract.Expenses.DATE)); return
Utils.getSystemFormatDateString(mContext, dateStr);
}

@Override protected View newSectionView(Context context, Object item, ViewGroup parent) {
return getLayoutInflater().inflate(R.layout.expense_report_section_header, parent, false); }

@Override protected void bindSectionView(View convertView, Context context, int position,
Object item) {

((TextView) convertView).setText((String) item);
}

@Override protected View newItemView(Context context, Cursor cursor, ViewGroup
parent) { return getLayoutInflater().inflate(R.layout.expense_list_item, parent, false);
}

@Override protected void bindItemView(View convertView, Context context, Cursor
cursor) {

// Find fields to populate in inflated template

TextView tvExpenseValue = (TextView) convertView.findViewById(R.id.expense_value_text_view);

TextView tvExpenseCurrency = (TextView)
convertView.findViewById(R.id.expense_currency_text_view);

TextView tvExpenseCatName = (TextView)
convertView.findViewById(R.id.expense_category_name_text_view);

```

```

// Extract values from cursor float expValue =
cursor.getFloat(cursor.getColumnIndexOrThrow(ExpensesContract.Expenses.VALUE));

String categoryName =
cursor.getString(cursor.getColumnIndexOrThrow(ExpensesContract.Categories.NAME));

// Populate views with extracted values tvExpenseValue.setText(Utils.formatToCurrency(expValue));
tvExpenseCatName.setText(categoryName); tvExpenseCurrency.setText(mCurrency);

} } package com.github.ematiyuk.expensetracer.adapters; import android.content.Context; import
android.database.Cursor; import android.support.v4.widget.CursorAdapter; import
android.view.LayoutInflater; import android.view.View; import android.view.ViewGroup; import
android.widget.TextView;

import com.github.ematiyuk.expensetracer.providers.ExpensesContract; import
com.github.ematiyuk.expensetracer.R; import com.github.ematiyuk.expensetracer.utils.Utils; public class
SimpleExpenseAdapter extends CursorAdapter { private String mCurrency; public
SimpleExpenseAdapter(Context context) {
super(context, null, 0);
}

public void setCurrency(String currency) { mCurrency
= currency; notifyDataSetChanged();
}

// The newView method is used to inflate a new view and return it
@Override public View newView(Context context, Cursor cursor, ViewGroup parent) {
return LayoutInflater.from(context).inflate(R.layout.expense_list_item, parent, false);
}

// The bindView method is used to bind all data to a given view
@Override public void bindView(View view, Context context, Cursor cursor) {
// Find fields to populate in inflated template
TextView tvExpenseValue = (TextView) view.findViewById(R.id.expense_value_text_view);
TextView tvExpenseCurrency = (TextView) view.findViewById(R.id.expense_currency_text_view);
TextView tvExpenseCatName = (TextView)
view.findViewById(R.id.expense_category_name_text_view); // Extract values from cursor float
expValue = cursor.getFloat(cursor.getColumnIndexOrThrow(ExpensesContract.Expenses.VALUE));

String categoryName =
cursor.getString(cursor.getColumnIndexOrThrow(ExpensesContract.Categories.NAME));

```

```

// Populate views with extracted values tvExpenseValue.setText(Utils.formatToCurrency(expValue));
tvExpenseCatName.setText(categoryName); tvExpenseCurrency.setText(mCurrency);

} } package com.github.ematiyuk.expensetracer.db;

import android.content.ContentValues; import android.content.Context; import
android.database.sqlite.SQLiteDatabase; import android.database.sqlite.SQLiteOpenHelper; import
com.github.ematiyuk.expensetracer.R; import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.Categories; import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.Expenses; public class ExpenseDbHelper
extends SQLiteOpenHelper {

private static final int DATABASE_VERSION = 1; private static final String DATABASE_NAME =
"expense_tracer.db";

public static final String CATEGORIES_TABLE_NAME = "categories"; public static final String
EXPENSES_TABLE_NAME = "expenses"; private Context mContext; public
ExpenseDbHelper(Context ctx) { super(ctx, DATABASE_NAME, null, DATABASE_VERSION);
mContext = ctx;
}

@Override

public void onCreate(SQLiteDatabase db) { db.execSQL(CategoriesTable.CREATE_TABLE_QUERY);

// Fill the table with predefined values
CategoriesTable.fillTable(db, mContext);

db.execSQL(ExpensesTable.CREATE_TABLE_QUERY);
}

@Override public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {

/* Temporary (dummy) upgrade policy */ db.execSQL(ExpensesTable.DELETE_TABLE_QUERY);
db.execSQL(CategoriesTable.DELETE_TABLE_QUERY); onCreate(db);
}

private static final class CategoriesTable { public
static final String CREATE_TABLE_QUERY =
"CREATE TABLE " + CATEGORIES_TABLE_NAME + " (" +
Categories._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
+
Categories.NAME + " TEXT NOT NULL);"; public static final
String DELETE_TABLE_QUERY = "DROP TABLE IF EXISTS " +
CATEGORIES_TABLE_NAME + ";";

public static void fillTable(SQLiteDatabase db, Context ctx) {

```



```

String[] predefinedNames = ctx.getResources().getStringArray(R.array.predefined_categories);
ContentValues values = new ContentValues(); for (String name : predefinedNames) {
    values.put(Categories.NAME, name); db.insert(CATEGORIES_TABLE_NAME, null, values); }
}
}

private static final class ExpensesTable { public
    static final String CREATE_TABLE_QUERY =
        "CREATE TABLE " + EXPENSES_TABLE_NAME + " (" +
        Expenses._ID + " INTEGER PRIMARY KEY AUTOINCREMENT, "
        +
        Expenses.VALUE + " FLOAT NOT NULL, " +
        Expenses.DATE + " DATE NOT NULL, " +
        Expenses.CATEGORY_ID + " INTEGER NOT NULL);"; public
    static final String DELETE_TABLE_QUERY =
        "DROP TABLE IF EXISTS " + EXPENSES_TABLE_NAME + ";"; } } package
com.github.ematiyuk.expensetracer.fragments;

import android.content.ContentUri; import android.content.ContentValues; import
android.database.Cursor; import android.net.Uri; import android.os.Bundle; import
android.support.annotation.Nullable; import android.support.v4.app.Fragment; import
android.support.v4.app.LoaderManager; import android.support.v4.content.CursorLoader; import
android.support.v4.content.Loader; import android.view.KeyEvent; import android.view.LayoutInflater;
import android.view.Menu; import android.view.MenuInflater; import android.view.MenuItem; import
android.view.View; import android.view.ViewGroup; import android.widget.EditText; import
android.widget.Toast;

import com.github.ematiyuk.expensetracer.providers.ExpensesContract.Categories; import
com.github.ematiyuk.expensetracer.R; public class CategoryEditFragment extends
    Fragment implements
        LoaderManager.LoaderCallbacks<Cursor> { public static final String EXTRA_EDIT_CATEGORY =
            "com.github.ematiyuk.expensetracer.edit_category";

    private EditText mCatNameEditText; private long mExtraValue; @Override public void
        onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState); setHasOptionsMenu(true);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
    {

```

```

// Inflate layout for this fragment
View rootView = inflater.inflate(R.layout.fragment_category_edit, container, false);
mCatNameEditText = (EditText) rootView.findViewById(R.id.category_name_edit_text);

// Set listener on Done (submit) button on keyboard clicked
mCatNameEditText.setOnKeyListener(new View.OnKeyListener() { @Override
public boolean onKey(View view, int keyCode, KeyEvent event) { if
((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode
== KeyEvent.KEYCODE_ENTER)) { checkEditTextForEmptyField(mCatNameEditText);
return true;
}
return false;
}
});
return rootView;
}

@Override public void onActivityCreated(@Nullable Bundle savedInstanceState) {
super.onActivityCreated(savedInstanceState);          mExtraValue          =
getActivity().getIntent().getLongExtra(EXTRA_EDIT_CATEGORY, -1);
// Create a new category if (mExtraValue < 1) { getActivity().setTitle(R.string.add_category);
// Edit existing category
} else { getActivity().setTitle(R.string.edit_category); setCategoryData();
}
}

@Override public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
super.onCreateOptionsMenu(menu, inflater); inflater.inflate(R.menu.fragment_category_edit, menu);
}

@Override public boolean onOptionsItemSelected(MenuItem item) { switch (item.getItemId()) { case
R.id.done_category_edit_menu_item:

```

```

if (checkEditTextForEmptyField(mCatNameEditText)) {
    // Create a new category if (mExtraValue < 1) { insertNewCategory();
    // Edit existing category
} else { updateCategory(mExtraValue);
}

getActivity().finish();
}

return true; default:

return super.onOptionsItemSelected(item);
}}

private boolean checkEditTextForEmptyField(EditText editText) { String inputText =
editText.getText().toString().trim(); if (inputText.length() == 0) {

editText.setError(getResources().getString(R.string.error_empty_field));
mCatNameEditText.selectAll(); return false;

} else {

return true;

}}

private void setCategoryData() { getLoaderManager().initLoader(0,
null, this);

}

@Override

public CursorLoader onCreateLoader(int id, Bundle args) {

String[] projectionFields = new String[] {

Categories._ID,

Categories.NAME

};

Uri singleCategoryUri =

ContentUri.withAppendedId(Categories.CONTENT_URI, mExtraValue);

return new CursorLoader(getActivity(), singleCategoryUri, projectionFields, null, null, null

);

```

```

@Override public void onLoadFinished(Loader<Cursor> loader, Cursor data) { int
categoryNameIndex = data.getColumnIndex(Categories.NAME); data.moveToFirst();

String categoryName = data.getString(categoryNameIndex);
mCatNameEditText.setText(categoryName);
}

@Override public void onLoaderReset(Loader loader) { mCatNameEditText.setText("");
}

private void insertNewCategory() {
ContentValues insertValues = new ContentValues(); insertValues.put(Categories.NAME,
mCatNameEditText.getText().toString()); getActivity().getContentResolver().insert(
Categories.CONTENT_URI, insertValues
);

Toast.makeText(getActivity(), getResources().getString(R.string.category_added),
Toast.LENGTH_SHORT).show();
}

private void updateCategory(long id) {
ContentValues updateValues = new ContentValues(); updateValues.put(Categories.NAME,
mCatNameEditText.getText().toString());

Uri categoryUri =
ContentUris.withAppendedId(Categories.CONTENT_URI, id);

getActivity().getContentResolver().update( categoryUri, updateValues, null, null
);

Toast.makeText(getActivity(), getResources().getString(R.string.category_updated),
Toast.LENGTH_SHORT).show();
} } package com.github.ematiyuk.expensetracer.fragments;

import android.content.ContentUris; import android.content.DialogInterface; import
android.content.Intent; import android.database.Cursor; import android.net.Uri; import
android.os.Bundle; import android.support.annotation.Nullable; import
android.support.v4.app.Fragment; import android.support.v4.app.LoaderManager; import
android.support.v4.content.CursorLoader; import android.support.v4.content.Loader; import
android.support.v4.widget.SimpleCursorAdapter; import android.support.v7.app.AlertDialog;
import android.view.ContextMenu; import android.view.LayoutInflater; import android.view.Menu;
import android.view.MenuInflater; import android.view.MenuItem; import android.view.View;
import android.view.ViewGroup; import android.widget.AdapterView; import
}

```

```

android.widget.AdapterView.AdapterContextMenuInfo; import android.widget.ListView; import
android.widget.Toast;

import com.github.ematiyuk.expensetracer.providers.ExpensesContract.Categories; import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.Expenses; import
com.github.ematiyuk.expensetracer.R; import
com.github.ematiyuk.expensetracer.activities.CategoryEditActivity; public

class CategoryFragment extends Fragment implements

LoaderManager.LoaderCallbacks<Cursor> { private ListView mCategoriesView; private
SimpleCursorAdapter mAdapter; private View mProgressBar;

@Override public void onCreate(@Nullable Bundle savedInstanceState) {

super.onCreate(savedInstanceState); setHasOptionsMenu(true);

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
{

// Inflate layout for this fragment

View rootView = inflater.inflate(R.layout.fragment_category, container, false); mCategoriesView
= (ListView)

rootView.findViewById(R.id.categories_list_view); mProgressBar =
rootView.findViewById(R.id.categories_progress_bar);
mCategoriesView.setEmptyView(rootView.findViewById(R.id.categories_empty
_list_view)); mCategoriesView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override public void onItemClick(AdapterView<?> parent, View view, int position, long
id) { prepareCategoryToEdit(id);

}

});

rootView.findViewById(R.id.add_category_button_if_empty_list).setOnClickListener(new
View.OnClickListener() {

@Override public void onClick(View view) { prepareCategoryToCreate();

}

}); registerContextMenu(mCategoriesView); return

rootView;

```

```

@Override public void onActivityCreated(@Nullable Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);

    mAdapter = new SimpleCursorAdapter(getActivity(), R.layout.category_list_item, null, new String[] {
        Categories.NAME }, new int[] { R.id.category_name_list_item}, 0);
    mCategoriesView.setAdapter(mAdapter);

    // Initialize the CursorLoader getLoaderManager().initLoader(0,
    null, this);
}

@Override public void onResume() { super.onResume(); reloadCategoryList();
}

@Override public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    super.onCreateOptionsMenu(menu, inflater); inflater.inflate(R.menu.fragment_category, menu);
}

@Override public boolean onOptionsItemSelected(MenuItem item) { switch (item.getItemId()) {
    case R.id.new_category_menu_item: prepareCategoryToCreate(); return true; default:
return super.onOptionsItemSelected(item);
}
}

@Override public void onCreateContextMenu(ContextMenu
menu, View v,
ContextMenu.ContextMenuInfo menuInfo) { super.onCreateContextMenu(menu, v, menuInfo);
getActivity().getMenuInflater().inflate(R.menu.category_list_item_context, menu);
}

@Override public boolean
onContextItemSelected(MenuItem item) {
    AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo(); switch
    (item.getItemId()) { case R.id.delete_category_menu_item:
deleteCategory(info.id); return true; default:
return super.onContextItemSelected(item);
}
}
}
}

```

```

@Override public Loader<Cursor> onCreateLoader(int id,
Bundle args) {
String[] projectionFields = new String[] {
Categories._ID,
Categories.NAME
};
return new CursorLoader(getActivity(), Categories.CONTENT_URI,
projectionFields, null, null, null
);
}

@Override public void onLoadFinished(Loader<Cursor> loader, Cursor data) {
// Hide the progress bar mProgressBar.setVisibility(View.GONE); mAdapter.swapCursor(data);
}

@Override public void onLoaderReset(Loader<Cursor> loader) { mAdapter.swapCursor(null);
}

private void reloadCategoryList() { // Show the progress bar
mProgressBar.setVisibility(View.VISIBLE); // Reload data by restarting the cursor loader
getLoaderManager().restartLoader(0, null, this);
} private int deleteSingleCategory(long categoryId) { Uri uri =
ContentUris.withAppendedId(Categories.CONTENT_URI, categoryId);

// Defines a variable to contain the number of rows deleted int rowsDeleted;

// Deletes the category that matches the selection criteria rowsDeleted =
getActivity().getContentResolver().delete( uri, // the URI of the row to delete null, // where
clause null // where args
);

reloadCategoryList(); showMessage(getResources().getString(R.string.category_deleted));

return rowsDeleted;

private int deleteAssociatedExpenses(long categoryId) {
String selection = Expenses.CATEGORY_ID + " = ?"; String[] selectionArgs = {
String.valueOf(categoryId) }; return
getActivity().getContentResolver().delete( Expenses.CONTENT_URI,
selection, selectionArgs

```

```

);
}

private void deleteCategory(final long categoryId) { new
AlertDialog.Builder(getActivity())
.setTitle(R.string.delete_category)
.setMessage(R.string.delete_cat_dialog_msg)
.setNeutralButton(android.R.string.cancel, null)
.setPositiveButton(R.string.delete_string, new DialogInterface.OnClickListener() { @Override
public void onClick(DialogInterface dialogInterface, int i) { int expenseRowsDeleted =
deleteAssociatedExpenses(categoryId);
String statusMsg = getResources().getQuantityString(
R.plurals.expenses_deleted_plurals_msg, expenseRowsDeleted,
expenseRowsDeleted); showStatusMessage(statusMsg);
deleteSingleCategory(categoryId);
}
})
.setIcon(R.drawable.ic_dialog_alert)
.show(); }

private void showStatusMessage(CharSequence text) {
Toast.makeText(getActivity(), text, Toast.LENGTH_SHORT).show();
}

private void prepareCategoryToCreate() { startActivity(new
Intent(getActivity(), CategoryEditActivity.class));
}

private void prepareCategoryToEdit(long id) {
}
}

```



```

Intent intent = new Intent(getActivity(), CategoryEditActivity.class);
intent.putExtra(CategoryEditFragment.EXTRA_EDIT_CATEGORY, id); startActivity(intent);
}} package com.github.ematiyuk.expensetracer.fragments;

import android.app.DatePickerDialog; import android.app.Dialog; import android.os.Bundle; import
android.support.v4.app.AlertDialog; import java.util.Calendar; public class
DatePickerFragment extends DialogFragment { private static DatePickerDialog.OnDateSetListener
mListener;

@Override public Dialog onCreateDialog(Bundle savedInstanceState) { // Use the current date as the
default date in the picker final Calendar c = Calendar.getInstance(); int year = c.get(Calendar.YEAR);
int month = c.get(Calendar.MONTH); int day = c.get(Calendar.DAY_OF_MONTH);

// Create a new instance of DatePickerDialog and return it return new
DatePickerDialog(getActivity(), mListener, year, month, day);
}

public static DatePickerFragment

newInstance(DatePickerDialog.OnDateSetListener listener) { mListener = listener; return new
DatePickerFragment();
}} package com.github.ematiyuk.expensetracer.fragments;

import android.content.ContentUris; import android.content.ContentValues; import
android.database.Cursor; import android.net.Uri; import android.os.Bundle; import
android.support.annotation.Nullable; import android.support.v4.app.Fragment; import
android.support.v4.app.LoaderManager; import android.support.v4.content.CursorLoader; import
android.support.v4.content.Loader; import android.support.v4.widget.SimpleCursorAdapter; import
android.support.v7.widget.AppCompatSpinner; import android.view.KeyEvent; import
android.view.LayoutInflater; import android.view.Menu; import android.view.MenuInflater; import
android.view.MenuItem; import android.view.View; import android.view.ViewGroup; import
android.widget.AdapterView; import android.widget.ArrayAdapter; import android.widget.EditText;
import android.widget.Toast;

import com.github.ematiyuk.expensetracer.providers.ExpensesContract.Categories; import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.Expenses; import
com.github.ematiyuk.expensetracer.R; import com.github.ematiyuk.expensetracer.utils.Utils;
import java.util.ArrayList; import java.util.Date;

public class ExpenseEditFragment extends Fragment implements
LoaderManager.LoaderCallbacks<Cursor> { public static final String EXTRA_EDIT_EXPENSE =
"com.github.ematiyuk.expensetracer.edit_expense"; private static final int EXPENSE_LOADER_ID =
1; private static final int CATEGORIES_LOADER_ID = 0; private EditText mExpValueEditText; private
AppCompatSpinner mCategorySpinner; private SimpleCursorAdapter mAdapter; private View
mCatProgressBar; private long mExtraValue; private long mExpenseCategoryId = -1;

```

```

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

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState)
{
    // Inflate layout for this fragment

    View rootView = inflater.inflate(R.layout.fragment_expense_edit, container, false);

    mExpValueEditText = (EditText)

    rootView.findViewById(R.id.expense_value_edit_text); mCatProgressBar =
    rootView.findViewById(R.id.cat_select_progress_bar); mCategorySpinner = (AppCompatSpinner)

    rootView.findViewById(R.id.category_choose_spinner); setEditTextDefaultValue();

    // Set listener on Done (submit) button on keyboard clicked
    mExpValueEditText.setOnKeyListener(new View.OnKeyListener() {
        @Override public boolean onKey(View view, int keyCode, KeyEvent event) {
            if ((event.getAction() == KeyEvent.ACTION_DOWN) && (keyCode
            == KeyEvent.KEYCODE_ENTER)) { checkValueFieldForIncorrectInput(); return true;
        }
        return false;
    }

    });

    mCategorySpinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override public void onItemSelected(AdapterView<?> parent, View view, int pos, long id)
        { mExpenseCategoryId = id;
        }

        @Override public void onNothingSelected(AdapterView<?> parent) { }

    });

    return rootView;
}

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

```

```

mAdapter = new SimpleCursorAdapter(getActivity(), android.R.layout.simple_spinner_item, null,
new String[] { Categories.NAME }, new int[] { android.R.id.text1 }, 0);

// Specify the layout to use when the list of choices appears
mAdapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

// Apply the adapter to the spinner mCategorySpinner.setAdapter(mAdapter);

mExtraValue = getActivity().getIntent().getLongExtra(EXTRA_EDIT_EXPENSE, -
1);

// Create a new expense if (mExtraValue < 1) { getActivity().setTitle(R.string.add_expense);
loadCategories();

// Edit existing expense
} else { getActivity().setTitle(R.string.edit_expense); loadExpenseData();
}
}

@Override public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
super.onCreateOptionsMenu(menu, inflater); inflater.inflate(R.menu.fragment_expense_edit,
menu);
}

@Override public boolean onOptionsItemSelected(MenuItem item) { switch (item.getItemId()) { case
R.id.done_expense_edit_menu_item:
if (checkForIncorrectInput()) { // Create a new expense if (mExtraValue < 1) { insertNewExpense(); //
Edit existing expense
} else { updateExpense(mExtraValue);
}
getActivity().finish();
}
return true; default:
return super.onOptionsItemSelected(item);
}}

private boolean checkForIncorrectInput() { if
(!checkValueFieldForIncorrectInput()) {
mExpValueEditText.selectAll(); return false;
}
}

```

```

// Future check of other fields return
true; }

private boolean checkValueFieldForIncorrectInput() { String etValue =
mExpValueEditText.getText().toString(); try { if (etValue.length() == 0) {
mExpValueEditText.setError(getResources().getString(R.string.error_empty_fiel
d )); return false;
} else if (Float.parseFloat(etValue) == 0.00f) {
mExpValueEditText.setError(getResources().getString(R.string.error_zero_value
));
return false;
}
} catch (Exception e) {
mExpValueEditText.setError(getResources().getString(R.string.error_incorrect_in put)); return false;
}
return true; } private void
loadCategories() {
// Show the progress bar next to category spinner mCatProgressBar.setVisibility(View.VISIBLE);
getLoaderManager().initLoader(CATEGORIES_LOADER_ID, null, this);
}

private void loadExpenseData() { getLoaderManager().initLoader(EXPENSE_LOADER_ID,
null, this); loadCategories();
}

private void setEditTextDefaultValue() { mExpValueEditText.setText(String.valueOf(0));
mExpValueEditText.selectAll();
}

@Override public CursorLoader onCreateLoader(int id,
Bundle args) { String[] projectionFields = null; Uri uri =
null; switch (id) { case EXPENSE_LOADER_ID:
projectionFields = new String[] {
Expenses._ID,
Expenses.VALUE,

```

```

Expenses.CATEGORY_ID
};

uri = ContentUris.withAppendedId(Expenses.CONTENT_URI, mExtraValue); break;

case CATEGORIES_LOADER_ID: projectionFields = new String[] {
Categories._ID,
Categories.NAME
};

uri = Categories.CONTENT_URI; break; } return new
CursorLoader(getActivity(), uri, projectionFields, null,
null, null
);
}

@Override public void onLoadFinished(Loader<Cursor> loader, Cursor data) { switch (loader.getId())
{ case EXPENSE_LOADER_ID: int expenseValueIndex = data.getColumnIndex(Expenses.VALUE); int
expenseCategoryIdIndex = data.getColumnIndex(Expenses.CATEGORY_ID);

data.moveToFirst(); mExpenseCategoryId = data.getLong(expenseCategoryIdIndex);
updateSpinnerSelection();

mExpValueEditText.setText(String.valueOf(data.getFloat(expenseValueIndex)));
mExpValueEditText.selectAll(); break; case CATEGORIES_LOADER_ID:

// Hide the progress bar next to category spinner mCatProgressBar.setVisibility(View.GONE); if
(null == data || data.getCount() < 1) { mExpenseCategoryId = -1;

// Fill the spinner with default values
ArrayList<String> defaultItems = new ArrayList<>();

defaultItems.add(getResources().getString(R.string.no_categories_string));

ArrayAdapter<String> tempAdapter = new
ArrayAdapter<String>(getActivity(), android.R.layout.simple_spinner_item, defaultItems);

mCategorySpinner.setAdapter(tempAdapter);

// Disable the spinner mCategorySpinner.setEnabled(false);

} else {

// Set the original adapter mCategorySpinner.setAdapter(mAdapter); // Update spinner data
mAdapter.swapCursor(data); // Enable the spinner mCategorySpinner.setEnabled(true);
updateSpinnerSelection();

```

```

    } break;

    }

    }

    @Override public void onLoaderReset(Loader<Cursor> loader) { switch (loader.getId()) { case
    EXPENSE_LOADER_ID: mExpenseCategoryId = -1; setEditTextDefaultValue(); break; case
    CATEGORIES_LOADER_ID: mAdapter.swapCursor(null); break;

    }}

    private void updateSpinnerSelection() { mCategorySpinner.setSelection(0); for (int pos
    = 0; pos < mAdapter.getCount(); ++pos) {

    if (mAdapter.getItemId(pos) == mExpenseCategoryId) { // Set spinner item selected according to the
    value from db mCategorySpinner.setSelection(pos); break;

    }

    }}

    private void insertNewExpense() {

    ContentValues insertValues = new ContentValues(); insertValues.put(Expenses.VALUE,
    Float.parseFloat(mExpValueEditText.getText().toString())); insertValues.put(Expenses.DATE,
    Utils.getDateString(new Date())); // Put
    current date (today) insertValues.put(Expenses.CATEGORY_ID, mExpenseCategoryId);
    getActivity().getContentResolver().insert( Expenses.CONTENT_URI, insertValues
    );

    Toast.makeText(getActivity(), getResources().getString(R.string.expense_added),
    Toast.LENGTH_SHORT).show();

    }

    private void updateExpense(long id) {

    ContentValues updateValues = new ContentValues(); updateValues.put(Expenses.VALUE,
    Float.parseFloat(mExpValueEditText.getText().toString()));
    updateValues.put(Expenses.CATEGORY_ID, mExpenseCategoryId); Uri expenseUri =
    ContentUris.withAppendedId(Expenses.CONTENT_URI,
    id);

    getActivity().getContentResolver().update( expenseUri, updateValues, null, null
    );

    Toast.makeText(getActivity(), getResources().getString(R.string.expense_updated),
    Toast.LENGTH_SHORT).show();

```

```

} } package com.github.ematiyuk.expensetracer.providers; import
android.net.Uri; import android.provider.BaseColumns; public
final class ExpensesContract {
/**
    * The authority for the expenses provider
*/ public static final String AUTHORITY =
"com.github.ematiyuk.expensetracer.provider";
/**
    * The content:// style URI for expenses provider
*/ public static final Uri AUTHORITY_URI = Uri.parse("content://" +
AUTHORITY);
/**
    * The contract class cannot be instantiated
*/ private ExpensesContract(){} public static class Categories implements
BaseColumns, CategoriesColumns {
/**
    * This utility class cannot be instantiated
*/ private Categories() {}
/**
    * The content:// style URI for this table
*/ public static final Uri CONTENT_URI =
Uri.withAppendedPath(AUTHORITY_URI, "categories");
/**
    * The MIME type of {@link #CONTENT_URI} providing a directory
ofcategories.
*/ public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.ematiyuk.expensetracer.provider.expense_category"; /**
    * The MIME type of a {@link #CONTENT_URI} sub-directory of a
singlecategory.
*/ public static final String CONTENT_ITEM_TYPE =

```

```

"vnd.android.cursor.item/vnd.ematiyuk.expensetracer.provider.expense_category"
;
/**
     * Sort by ascending order of _id column (the order as items were added). */
    public static final String DEFAULT_SORT_ORDER = _ID + " ASC";
}

public static class Expenses implements BaseColumns, ExpensesColumns {
/**
     * This utility class cannot be instantiated
    */ private Expenses() {}
/**
     * The content:// style URI for this table
    */ public static final Uri CONTENT_URI =
Uri.withAppendedPath(AUTHORITY_URI, "expenses");
/** * The MIME type of {@link #CONTENT_URI} providing a directory of expenses.
    */ public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.ematiyuk.expensetracer.provider.expense";
/**
     * The MIME type of a {@link #CONTENT_URI} sub-directory of a
    single expense.
    */ public static final String CONTENT_ITEM_TYPE =
"vnd.android.cursor.item/vnd.ematiyuk.expensetracer.provider.expense";
/**
     * Sort by descending order of date (the most recent items are at the end).
    */ public static final String DEFAULT_SORT_ORDER = DATE + " ASC";
/**
     * Expense sum value column name to return for joined tables */ public static
    final String VALUES_SUM = "values_sum";
}

public static class ExpensesWithCategories implements BaseColumns {

```



```

/**
     * This utility class cannot be instantiated.
*/ private ExpensesWithCategories() {}
/**
     * The content:// style URI for this table.
*/ public static final Uri CONTENT_URI =
Uri.withAppendedPath(AUTHORITY_URI, "expensesWithCategories");
/**
     * The MIME type of {@link #CONTENT_URI} providing a directory of expenses
with categories.
*/ public static final String CONTENT_TYPE =
"vnd.android.cursor.dir/vnd.ematiyuk.expensetracer.provider.expense_with_category"; /**
     * The MIME type of a {@link #CONTENT_URI} sub-directory of a single expense
with a category. */
// public static final String CONTENT_ITEM_TYPE =
//
"vnd.android.cursor.item/vnd.ematiyuk.expensetracer.provider.expense_with_category"; /**
     * The content:// style URI for this joined table to filter items by a specific date.
*/ public static final Uri DATE_CONTENT_URI = Uri.withAppendedPath(CONTENT_URI, "date"); /**
     * The content:// style URI for this joined table to filter items by a specific date
range.
*/
public static final Uri DATE_RANGE_CONTENT_URI = Uri.withAppendedPath(CONTENT_URI,
"dateRange");
/**
     * The content:// style URI for getting sum of expense values* for this joined
table by "date" filter.
*/ public static final Uri SUM_DATE_CONTENT_URI = Uri.withAppendedPath(DATE_CONTENT_URI,
"sum");
/**
     * The content:// style URI for getting sum of expense values* for this joined
table by "date range" filter.

```

```

*/

public static final Uri SUM_DATE_RANGE_CONTENT_URI =
Uri.withAppendedPath(DATE_RANGE_CONTENT_URI, "sum");
}

protected interface CategoriesColumns { String NAME = "name";
}

protected interface ExpensesColumns {
String VALUE = "value";
String DATE = "date";
String CATEGORY_ID = "category_id";
} } package com.github.ematiyuk.expensetracer.providers;

import android.content.ContentProvider; import android.content.ContentUris; import
android.content.ContentValues; import android.content.UriMatcher; import
android.database.Cursor; import android.database.sqlite.SQLiteDatabase; import
android.database.sqlite.SQLiteOpenHelper; import android.net.Uri;

import com.github.ematiyuk.expensetracer.db.ExpenseDbHelper; import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.Categories; import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.Expenses; import
com.github.ematiyuk.expensetracer.providers.ExpensesContract.ExpensesWithCategories;
import static com.github.ematiyuk.expensetracer.db.ExpenseDbHelper.CATEGORIES_TABLE
_NAME;

import static
com.github.ematiyuk.expensetracer.db.ExpenseDbHelper.EXPENSES_TABLE_
NAME;

public class ExpensesProvider extends ContentProvider { public static final
int EXPENSES = 10; public static final int EXPENSES_ID = 11; public static
final int CATEGORIES = 20; public static final int CATEGORIES_ID = 21;

public static final int EXPENSES_WITH_CATEGORIES = 30; public static final int
EXPENSES_WITH_CATEGORIES_DATE = 31; public static final int
EXPENSES_WITH_CATEGORIES_DATE_RANGE =
32; public static final int EXPENSES_WITH_CATEGORIES_SUM_DATE = 33; public static final int
EXPENSES_WITH_CATEGORIES_SUM_DATE_RANGE = 34;

private SQLiteOpenHelper mDbHelper; private SQLiteDatabase mDatabase; private
static final UriMatcher sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);

```

```

static {
    sUriMatcher.addURI(ExpensesContract.AUTHORITY, "expenses",
        EXPENSES); sUriMatcher.addURI(ExpensesContract.AUTHORITY, "expenses/#",
        EXPENSES_ID); sUriMatcher.addURI(ExpensesContract.AUTHORITY, "categories",
        CATEGORIES); sUriMatcher.addURI(ExpensesContract.AUTHORITY,
        "categories/#", CATEGORIES_ID);
    sUriMatcher.addURI(ExpensesContract.AUTHORITY,
        "expensesWithCategories", EXPENSES_WITH_CATEGORIES);
    sUriMatcher.addURI(ExpensesContract.AUTHORITY,
        "expensesWithCategories/date", EXPENSES_WITH_CATEGORIES_DATE);
    sUriMatcher.addURI(ExpensesContract.AUTHORITY,
        "expensesWithCategories/dateRange",
        EXPENSES_WITH_CATEGORIES_DATE_RANGE); sUriMatcher.addURI(ExpensesContract.AUTHORITY,
        "expensesWithCategories/date/sum",
        EXPENSES_WITH_CATEGORIES_SUM_DATE); sUriMatcher.addURI(ExpensesContract.AUTHORITY,
        "expensesWithCategories/dateRange/sum",
        EXPENSES_WITH_CATEGORIES_SUM_DATE_RANGE);
}

```

```

/*

```

```

        * SELECT expenses._id, expenses.value, categories.name, expenses.date

```

```

        * FROM expenses JOIN categories

```

```

        * ON expenses.category_id = categories._id

```

```

*/ private static final

```

```

String

```

```

BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY =

```

```

"SELECT " + EXPENSES_TABLE_NAME + "." + Expenses._ID + ", " +

```

```

EXPENSES_TABLE_NAME + "." + Expenses.VALUE + ", " +

```

```

CATEGORIES_TABLE_NAME + "." + Categories.NAME + ", " +

```

```

EXPENSES_TABLE_NAME + "." + Expenses.DATE + " FROM " +

```

```

EXPENSES_TABLE_NAME + " JOIN " +

```

```

CATEGORIES_TABLE_NAME + " ON " +

```

```

EXPENSES_TABLE_NAME + "." + Expenses.CATEGORY_ID + "
= " +
CATEGORIES_TABLE_NAME + "." + Categories._ID;
/**
     * <p>
     *   Initializes the provider.
     * </p>
 *
     * <i>Note</i>: provider is not created until a
     *   {@link android.content.ContentResolver ContentResolver} object tries
     *   to access it.
 *
     * @return <code>true</code> if the provider was successfully loaded,
     <code>false</code> otherwise
 */
@Override public boolean onCreate() { mDbHelper = new
ExpenseDbHelper(getContext()); return true;
}
@Override
public Cursor query(Uri uri, String[] projection, String selection, String[]
selectionArgs, String sortOrder) {
Cursor cursor;
String table;
String rawQuery; mDatabase = mDbHelper.getReadableDatabase(); switch (sUriMatcher.match(uri))
{ // The incoming URI is for all of categories case CATEGORIES:
table = CATEGORIES_TABLE_NAME; sortOrder =
(sortOrder == null || sortOrder.isEmpty())
? Categories.DEFAULT_SORT_ORDER
: sortOrder; break;
// The incoming URI is for a single row from categories case CATEGORIES_ID:
table = CATEGORIES_TABLE_NAME;

```

```

// Defines selection criteria for the row to query selection = Categories._ID + " = ?"; selectionArgs =
new String[]{ uri.getLastPathSegment() }; break; // The incoming URI is for all of expenses case
EXPENSES: table = EXPENSES_TABLE_NAME;

sortOrder = (sortOrder == null || sortOrder.isEmpty())
? Expenses.DEFAULT_SORT_ORDER
: sortOrder; break;

// The incoming URI is for a single row from expenses case EXPENSES_ID:
table = EXPENSES_TABLE_NAME;

// Defines selection criteria for the row to query selection = Expenses._ID + " = ?"; selectionArgs =
new String[]{ uri.getLastPathSegment() }; break;

// The incoming URI is for all expenses with categories case EXPENSES_WITH_CATEGORIES: /*
        * SELECT expenses._id, expenses.value,
        * categories.name,expenses.date
        * FROM expenses JOIN categories
        * ON expenses.category_id = categories._id
*/ return
mDatabase.rawQuery(BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY, null);

// The incoming URI is for the expenses with categories for a specific date case
EXPENSES_WITH_CATEGORIES_DATE:
/*
        * SELECT expenses._id, expenses.value,
        * categories.name,expenses.date
        * FROM expenses JOIN categories* ON expenses.category_id =
        * categories._id * WHERE expense.date = ?
*/ rawQuery =
BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY + "
WHERE " +
EXPENSES_TABLE_NAME + "." + Expenses.DATE + " = ?"; return mDatabase.rawQuery(rawQuery,
selectionArgs);

// The incoming URI is for the expense values sum for a specific date range case
EXPENSES_WITH_CATEGORIES_SUM_DATE:
/*

```

```

        * SELECT SUM(expenses.value) as values_sum* FROM expenses
        WHERE expenses.date = ?

*/ rawQuery =

"SELECT SUM(" + EXPENSES_TABLE_NAME + "." +
Expenses.VALUE + ") as " +
Expenses.VALUES_SUM + " FROM " + EXPENSES_TABLE_NAME +
" WHERE " + EXPENSES_TABLE_NAME + "." +
Expenses.DATE + " = ?"; return mDatabase.rawQuery(rawQuery, selectionArgs);

// The incoming URI is for the expenses with categories for a specific date range case
EXPENSES_WITH_CATEGORIES_DATE_RANGE:

/*

        * SELECT expenses._id, expenses.value,
        categories.name,expenses.date

        * FROM expenses JOIN categories *      ON expenses.category_id =
        categories._id *      WHERE expense.date BETWEEN ? AND ?

*/ rawQuery =

BASE_SELECT_JOIN_EXPENSES_CATEGORIES_QUERY + "
WHERE " +
EXPENSES_TABLE_NAME + "." + Expenses.DATE + "
BETWEEN ? AND ?"; return
mDatabase.rawQuery(rawQuery, selectionArgs);

// The incoming URI is for the expense values sum for a specific date range case
EXPENSES_WITH_CATEGORIES_SUM_DATE_RANGE:

/*

        * SELECT SUM(expenses.value) as values_sum

        * FROM expenses WHERE expense.date BETWEEN ? AND ?

*/ rawQuery =

"SELECT SUM(" + EXPENSES_TABLE_NAME + "." +
Expenses.VALUE + ") as " +
Expenses.VALUES_SUM + " FROM " +
EXPENSES_TABLE_NAME +

```

```

" WHERE " + EXPENSES_TABLE_NAME + "." +
Expenses.DATE + " BETWEEN ? AND ?"; return mDatabase.rawQuery(rawQuery, selectionArgs);
default:
throw new IllegalArgumentException("Unknown Uri provided.");
}

cursor = mDatabase.query( table, projection, selection, selectionArgs, null, null, sortOrder
);

return cursor;
}

@Override

public Uri insert(Uri uri, ContentValues values) {

String table; Uri contentUri; switch (sUriMatcher.match(uri)) { // The incoming URI is for all of
categories case CATEGORIES:

table = CATEGORIES_TABLE_NAME; contentUri = Categories.CONTENT_URI; break;

// The incoming URI is for all of expenses case EXPENSES:

table = EXPENSES_TABLE_NAME; contentUri = Expenses.CONTENT_URI; break;

// The incoming URI is for a single row from categories case CATEGORIES_ID:

// The incoming URI is for a single row from expenses case EXPENSES_ID:

throw new UnsupportedOperationException("Inserting rows with

specified IDs is forbidden."); case EXPENSES_WITH_CATEGORIES: case

EXPENSES_WITH_CATEGORIES_DATE:

case EXPENSES_WITH_CATEGORIES_DATE_RANGE: case EXPENSES_WITH_CATEGORIES_SUM_DATE:

case EXPENSES_WITH_CATEGORIES_SUM_DATE_RANGE:

throw new UnsupportedOperationException("Modifying joined results is forbidden."); default:

throw new IllegalArgumentException("Unknown Uri provided.");

} mDatabase = mDbHelper.getWritableDatabase(); long

newRowID = mDatabase.insert( table, null, values

);

Uri newItemUri = ContentUris.withAppendedId(contentUri, newRowID); return

(newRowID < 1) ? null : newItemUri;

}

@Override

```

```

public int delete(Uri uri, String selection, String[] selectionArgs) { String table; switch
(sUriMatcher.match(uri)) {

// The incoming URI is for a single row from categories case CATEGORIES_ID:

table = CATEGORIES_TABLE_NAME;

// Defines selection criteria for the row to delete selection = Categories._ID + " = ?"; selectionArgs =
new String[]{ uri.getLastPathSegment() }; break; // The incoming URI is for all of expenses case
EXPENSES:

table = EXPENSES_TABLE_NAME;

break;

// The incoming URI is for a single row from expenses case
EXPENSES_ID:

table = EXPENSES_TABLE_NAME;

// Defines selection criteria for the row to delete selection = Expenses._ID + " = ?"; selectionArgs =
new String[]{ uri.getLastPathSegment() }; break;

// The incoming URI is for all of categories case CATEGORIES:

throw new UnsupportedOperationException("Removing multiple rows
from the table is forbidden."); case EXPENSES_WITH_CATEGORIES: case
EXPENSES_WITH_CATEGORIES_DATE:

case EXPENSES_WITH_CATEGORIES_DATE_RANGE: case EXPENSES_WITH_CATEGORIES_SUM_DATE:
case EXPENSES_WITH_CATEGORIES_SUM_DATE_RANGE:

throw new UnsupportedOperationException("Modifying joined results is forbidden."); default:

throw new IllegalArgumentException("Unknown Uri provided.");

} mDatabase = mDbHelper.getWritableDatabase(); return

mDatabase.delete( table, selection, selectionArgs

);

}

@Override

public int update(Uri uri, ContentValues values, String selection, String[] selectionArgs)

{ String table; switch (sUriMatcher.match(uri)) {

// The incoming URI is for a single row from categories case CATEGORIES_ID:

table = CATEGORIES_TABLE_NAME;

// Defines selection criteria for the row to delete selection = Categories._ID + " = ?"; selectionArgs =
new String[]{ uri.getLastPathSegment() }; break;

```



```

// The incoming URI is for a single row from expenses case EXPENSES_ID:
table = EXPENSES_TABLE_NAME;

// Defines selection criteria for the row to delete selection = Expenses._ID + " = ?"; selectionArgs =
new String[]{ uri.getLastPathSegment() }; break;

// The incoming URI is for all of categories case CATEGORIES:

// The incoming URI is for all of expenses case EXPENSES:

throw new UnsupportedOperationException("Updating multiple table rows is forbidden."); case
EXPENSES_WITH_CATEGORIES: case EXPENSES_WITH_CATEGORIES_DATE:

case EXPENSES_WITH_CATEGORIES_DATE_RANGE: case EXPENSES_WITH_CATEGORIES_SUM_DATE:
case EXPENSES_WITH_CATEGORIES_SUM_DATE_RANGE:

throw new UnsupportedOperationException("Modifying joined results is forbidden."); default:

throw new IllegalArgumentException("Unknown Uri provided.");

} mDatabase = mDbHelper.getWritableDatabase(); return
mDatabase.update( table, values, selection, selectionArgs
);
}

@Override

public String getType(Uri uri) { final int match =
sUriMatcher.match(uri); switch (match) { case
CATEGORIES: return Categories.CONTENT_TYPE; case
CATEGORIES_ID:
return Categories.CONTENT_ITEM_TYPE; case
EXPENSES: return Expenses.CONTENT_TYPE;
case EXPENSES_ID:
return Expenses.CONTENT_ITEM_TYPE; case EXPENSES_WITH_CATEGORIES: case
EXPENSES_WITH_CATEGORIES_DATE: case EXPENSES_WITH_CATEGORIES_DATE_RANGE: case
EXPENSES_WITH_CATEGORIES_SUM_DATE:
case EXPENSES_WITH_CATEGORIES_SUM_DATE_RANGE: return
ExpensesWithCategories.CONTENT_TYPE; default:
return null;
}
} } package com.github.ematiyuk.expensetracer.utils; import android.content.Context;

```

```

import java.text.NumberFormat; import java.text.ParseException; import
java.text.SimpleDateFormat; import java.util.Date; import java.util.Locale;

public class Utils {

    public static String getSystemFormatDateString(Context context, Date date) {
        java.text.DateFormat dateFormat =
        android.text.format.DateFormat.getDateFormat(context); return dateFormat.format(date);
    }

    public static String getSystemFormatDateString(Context context, String dateString)
    { java.text.DateFormat dateFormat =
    android.text.format.DateFormat.getDateFormat(context); return
    dateFormat.format(stringToDate(dateString));
    }

    public static String getDateString(Date date) {
        SimpleDateFormat dateFormat = new SimpleDateFormat("MM/dd/yy", Locale.US);
        try {
            return dateFormat.format(date);
        } catch (Exception pe) { pe.printStackTrace(); return "no_date";
        }
    }

    private static Date stringToDate(String dateString) {
        SimpleDateFormat dateFormat = new SimpleDateFormat("MM/dd/yy",
        Locale.US); try { return
        dateFormat.parse(dateString);
        } catch (ParseException pe) { pe.printStackTrace(); return null;
        } }

    public static String formatToCurrency(float value) { final NumberFormat numberFormat =
    NumberFormat.getNumberInstance(); numberFormat.setMaximumFractionDigits(2);
    numberFormat.setMinimumFractionDigits(2); return numberFormat.format(value);
    } } package com.github.ematiyuk.expensetracer; import
    android.app.Application; import android.test.ApplicationTestCase;

    /**
    * <a href="http://d.android.com/tools/testing/testing_android.html">Testing

```

## Fundamentals</a>

```
*/ public class ApplicationTest extends ApplicationTestCase<Application> {  
  
public ApplicationTest() { super(Application.class);  
  
} } package com.github.ematiyuk.expensetracer; import org.junit.Test; import static  
org.junit.Assert.*;  
  
/**  
  
* To work on unit tests, switch the Test Artifact in the Build Variants view.  
  
*/ public class ExampleUnitTest {  
  
@Test  
public void addition_isCorrect() throws Exception { assertEquals(4, 2 + 2);  
  
}  
  
}
```

13.2.

GitHub & Project Demo Link

GITHUB: <https://github.com/IBM-EPBL/IBM-Project-28318-1660110565>