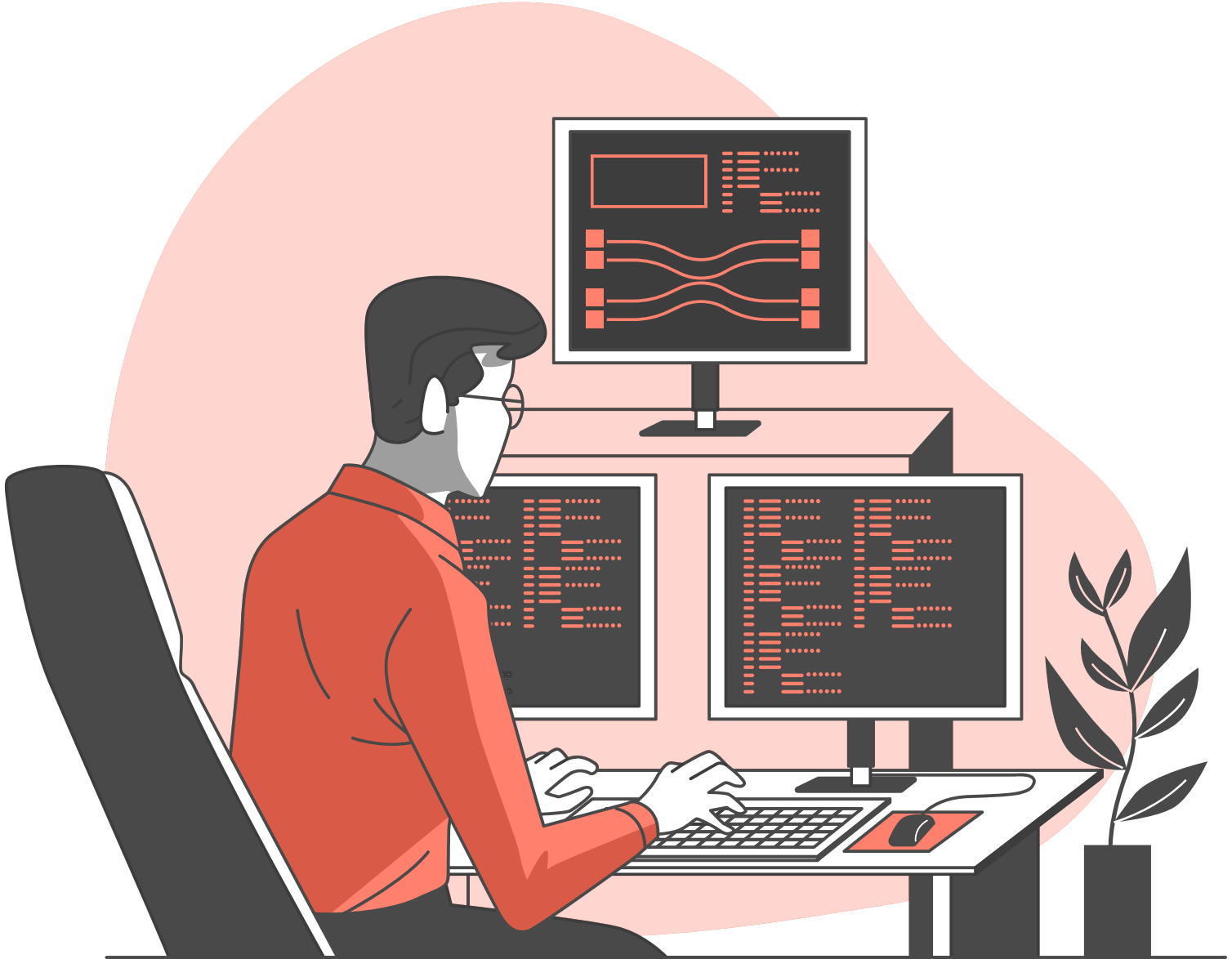
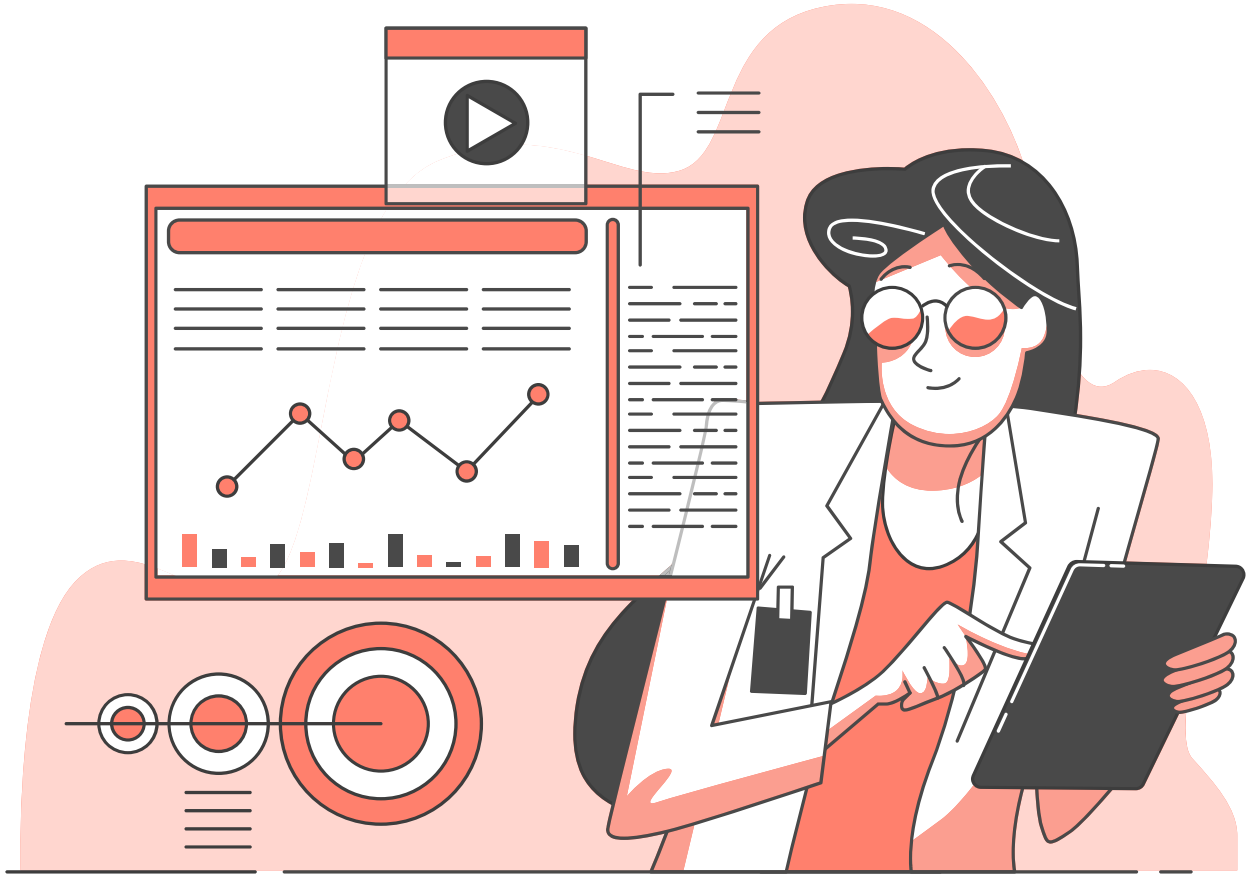


Personal Financing

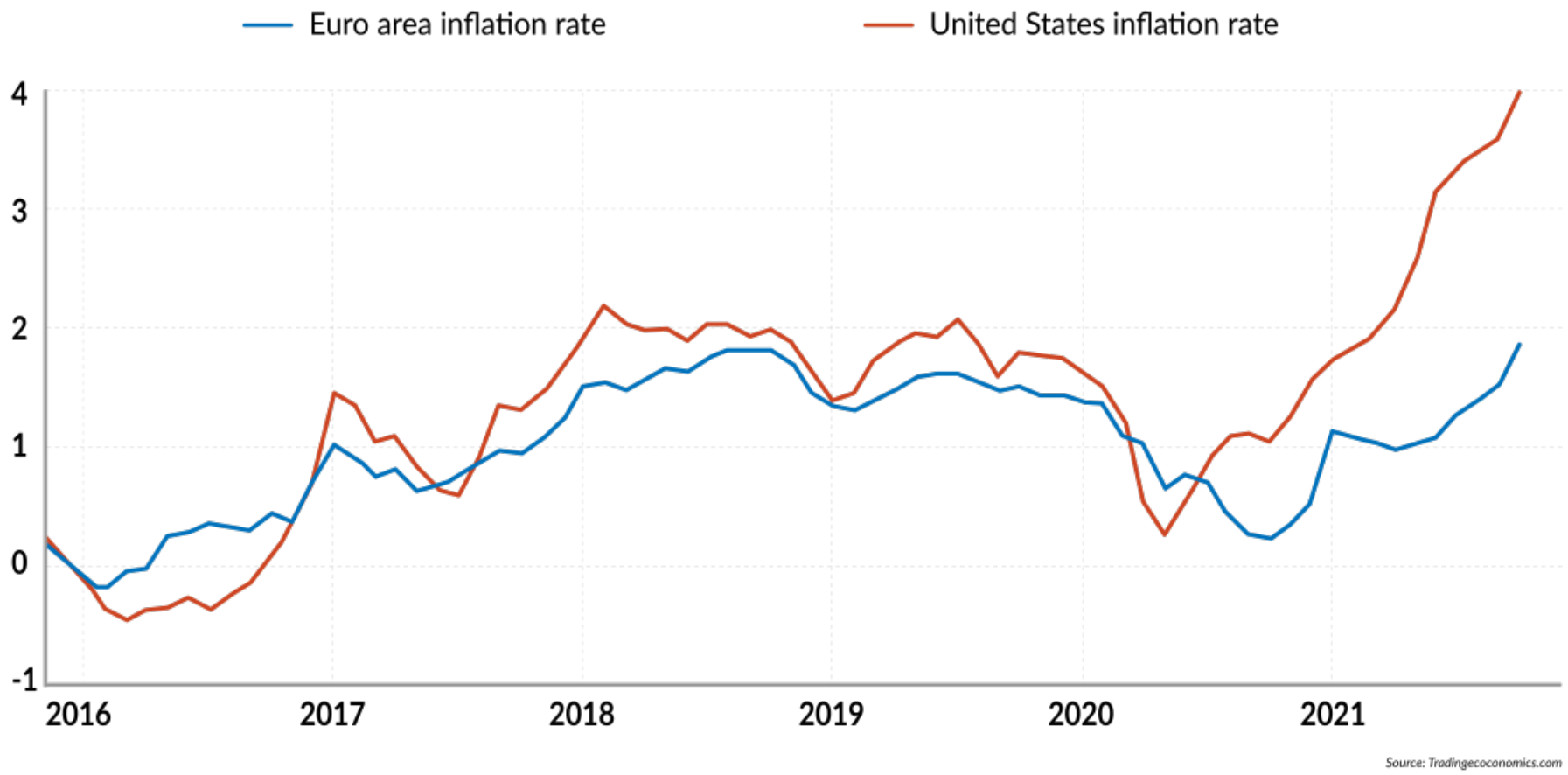
Gautam, Animesh
Ivlev, Andrei
Lee, Tsaichi
Pillay, Aru
Yang, Jiayi

Table of Contents

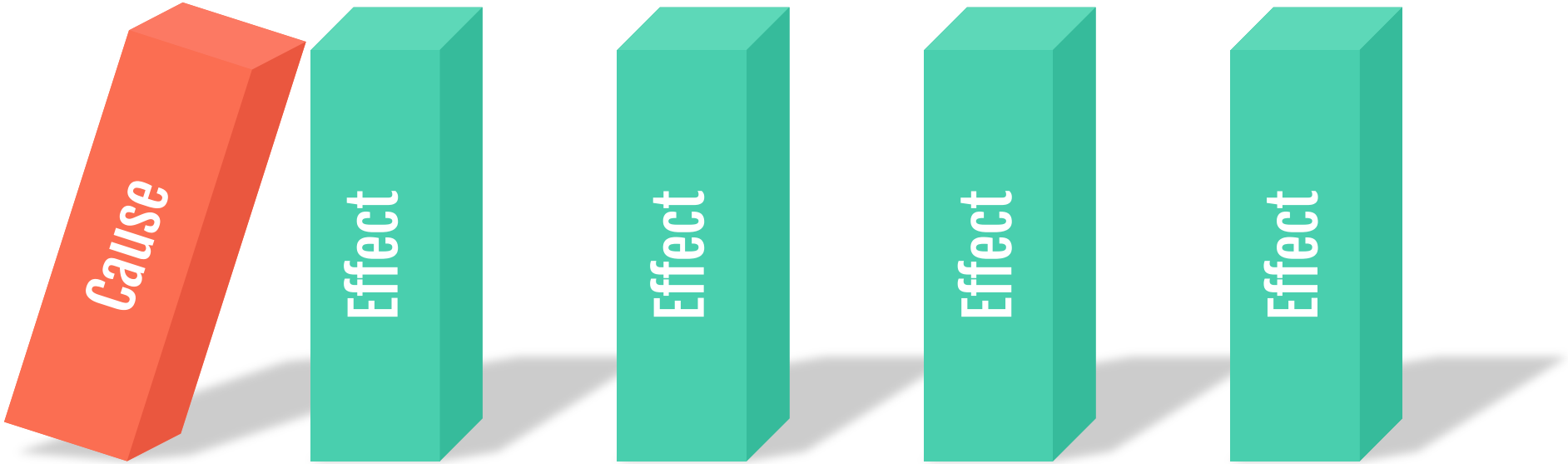


01
The Problem

In Jan 2022, Inflation in the US reached 7% and in the EU 5.1%



Consequences of not Budgeting



Overspending

Lack of Savings

Less Financial Control

More Debt

Stress

02

Our Solution

A personal financing and budgeting application which is secure and easy to use

03

Our Process

An N26 Bank Statement

Ferdinand Cafe Mastercard • Bars & Restaurants Original amount 10.90 CHF | Exchange rate 0.97248 Value Date 06.06.2022	06.06.2022	-10,60€
SUPERMER. EXTREMO ORIE Mastercard • Groceries Value Date 08.06.2022	08.06.2022	-14,25€
ESADE SANT CUGAT ARAMA Mastercard • Bars & Restaurants Value Date 08.06.2022	08.06.2022	-1,99€
ALLIANCE VENDING Mastercard • Groceries Value Date 09.06.2022	09.06.2022	-1,45€
FARMACIA FRANQUESA SOL Mastercard • Healthcare & Drug Stores Value Date 10.06.2022	10.06.2022	-2,99€

Preprocessing

```python
#create data processing function to process original balance dataset downloaded from ING
class Process:
    def __init__(self):
        df = pd.read_csv('statement-2022-05.csv')
        df.columns = ['Description', 'Date', 'Amount']

        #select the rows that contain the previous balance information
        previous_balance = df.loc[df['Description']=='Previous balance']['Date']

        #select the rows that describe the outgoing transaction
        outgoing_transactions = df.loc[df['Description']=='Outgoing transactions']['Date']

        #select the rows that describe the incoming transaction
        incoming_transactions = df.loc[df['Description']=='Incoming transactions']['Date']

        #select the rows that show the current balance
        new_balance = df.loc[df['Description']=='Your new balance']['Date']

        #get the row index of previous balance
        row = df[df['Description'] == 'Previous balance'].index.tolist()[0]

        #filter out the rows contain useless information
        df = df.iloc[:row]

        #drop out none values
        df = df[df['Description'].notna()]
        df = df.reset_index(drop=True)
        df = df.replace('Booking Date', np.nan)

        #get the indexes of the rows that contain useful transaction information
        first_list = df.index[df['Date'].notna() == True].tolist()
        second_list = [x+1 for x in first_list]
        index_list = first_list + second_list
        index_list = sorted(index_list)
        df = df.iloc[index_list]
        df = df.reset_index(drop=True)

        #change the form to get the number itself
        df['Description'] = np.where((df['Date'].isna() == True) & (df['Description'].str.contains(" • ", case=False) == Fa
                                     np.nan,
                                     df['Description'])
        df['Description'] = np.where(df['Description'].str.contains(" • ", case=False) == True,
                                     df['Description'].str.split(' • ').str[1],
                                     df['Description'])
        df['Description'] = df['Description'].fillna('Other')

        #select the rows that contain category information
        df['Category'] = np.where(df['Date'].isna() == False,
                                  df['Description'].shift(-1),
                                  np.nan)
        df = df[df['Category'].notna()]
        df = df.reset_index(drop=True)

        #get the number of spending in the numeric form instead of string
        df['Amount'] = df['Amount'].str.replace(',', '.')
        df['Amount'] = df['Amount'].str.replace('€', '')
        df['Amount'] = pd.to_numeric(df['Amount'],errors = 'coerce')
        df['Category'] = np.where(df['Amount'] > 0,
                                  'Income',
                                  df['Category'])

        #get the income seperately
        self.l_income = list(df[df['Category'] == 'Income']['Amount'])

        #select only spendings but not income
        df_cat = df[['Category', 'Amount']]
        df_cat = df_cat[df_cat['Category'] != 'Income']
        df_cat['Amount'] = abs(df_cat['Amount'])
        self.expense = df_cat.groupby('Category')['Amount'].apply(list).to_dict()
```

The Application

```python
class Budget(Process):
    def __init__(self):
        #self.expense = { 'Shopping' : [10,20], 'Other' : [30,40], 'Entertainment' : [50,60]}
        #self.l_income = [100,200,300]
        super().__init__()
        #get the total income and expense by suming they up and round them to 2 digits
        self.total_income = sum(self.l_income)
        self.total_income = round(self.total_income,2)
        self.total_expense = 0
        for x in self.expense.values():
            for y in x:
                self.total_expense += y
        self.total_expense = round(self.total_expense,2)

        #calculate the balance
        self.balance = self.total_income - self.total_expense
        self.balance = round(self.balance,2)

    #create display function to show the result
    def display_income(self):
        print("\nTotal income: {}".format(self.total_income))

    def display_expense(self):
        print("\nTotal expenses: {}".format(self.total_expense))

    def user_balance(self):
        print("\nUser balance: {}".format(self.balance))

    #get a report of the total expense of different categories
    def expense_distribution(self):
        for category,l in self.expense.items():
            cat_sum = 0
            for i in l:
                cat_sum += i
            print("\n{} : {} %".format(category, round((cat_sum / self.total_expense)*100,2)))

    #create a function to query the expense of a certain category
    def cat_expense(self):
        self.cat = {}
        for category,l in self.expense.items():
            cat_sum = 0
            for i in l:
                cat_sum += i
            self.cat[category] = cat_sum
        cat_required = input('Please enter the category expense you want to query ')
        if cat_required in self.cat.keys():
            print(round(self.cat[cat_required],2))
        else:
            print('Sorry, there is no such category')

    #create the fucntion for users' saving plan in amount of money
    def savings_basic(self):
        #ask about the saving goal
        self.basic_goal = int(input('\nEnter amount you want to save over the next year: '))

        #calculate the saving goal per month
        self.basic_monthly_amount = self.basic_goal / 12

        #get the percentage that have to be saved from the income
        self.basic_percent = (self.basic_monthly_amount * 100) / self.total_income
        print("You must save {} {} of your income per month".format(round(self.basic_percent,2),'%'))
        print("i.e. {} / month".format(round(self.basic_monthly_amount,2)))

        #compare to see if the balance can meet the saving goal per month
        if(self.basic_monthly_amount > self.balance):
            print("Based on your balance, you need to reduce expenses or increase income to reach your savings goals")
        else:
            print("Based on your balance, you are on track to reach your savings goals")

    #create a function to make a saving plan based in percentage of income
    def savings_advanced(self):
        self.adv_percent = int(input('\nEnter % ' + 'of income you want to save: '))
        self.adv_monthly_amount = (self.adv_percent * self.total_income) / 100
        print("You must save {} / month".format(self.adv_monthly_amount))
        #compare to see if the balance can meet the saving goal per month
        if(self.adv_monthly_amount > self.balance):
            print("Based on your balance, you need to reduce expenses or increase income to reach your savings goals")
        else:
            print("Based on your balance, you are on track to reach your savings goals")
```

The Main (UI)

```python
#the gate to call the function
if __name__ == '__main__':

    b = Budget()
    while True:
        #offers choices to get corresponding function
        print("1.  Display total income")
        print("2.  Display total expenses")
        print("3.  Display user balance")
        print("4.  Category wise distribution of expenses")
        print("5.  Query the expense of a certain category")
        print("6.  Savings suggestion")
        print("7.  Monthly savings plan")

        choice = int(input("\nSelect option: "))
        #call the corresponding fucntion according to the request
        if choice == 1:
            b.display_income()
        elif choice == 2:
            b.display_expense()
        elif choice == 3:
            b.user_balance()
        elif choice == 4:
            b.expense_distribution()
        elif choice == 5:
            b.cat_expense()
        elif choice == 6:
            b.savings_basic()
        elif choice == 7:
            b.savings_advanced()

        print()
        x = input("\nDo you want to continue? (y/n)")
        if x == 'N' or x == 'n':
            break
```

04

Functionalities

The UI

```
1.  Display total income
2.  Display total expenses
3.  Display user balance
4.  Category wise distribution of expenses
5.  Query the expense of a certain category
6.  Savings suggestion
7.  Monthly savings plan


Select option:
```

Option 1

```
1.  Display total income
2.  Display total expenses
3.  Display user balance
4.  Category wise distribution of expenses
5.  Query the expense of a certain category
6.  Savings suggestion
7.  Monthly savings plan

Select option: 1

Total income: 2640.25
```

Option 2

```
Do you want to continue? (y/n)y
1.  Display total income
2.  Display total expenses
3.  Display user balance
4.  Category wise distribution of expenses
5.  Query the expense of a certain category
6.  Savings suggestion
7.  Monthly savings plan

Select option: 2

Total expenses: 2286.7
```

Option 3

```
Do you want to continue? (y/n)y
1.  Display total income
2.  Display total expenses
3.  Display user balance
4.  Category wise distribution of expenses
5.  Query the expense of a certain category
6.  Savings suggestion
7.  Monthly savings plan

Select option: 3

User balance: 353.55
```

Option 4

```
Select option: 4

Bars & Restaurants : 15.51 %

Finance : 3.07 %

Groceries : 11.32 %

Leisure : 1.53 %

Multimedia & Telecom : 0.48 %

Other : 42.13 %

Shopping : 4.6 %

Transport : 21.35 %
```

Option 5

```
Do you want to continue? (y/n)y
1.  Display total income
2.  Display total expenses
3.  Display user balance
4.  Category wise distribution of expenses
5.  Query the expense of a certain category
6.  Savings suggestion
7.  Monthly savings plan

Select option: 5
Please enter the category expense you want to query Shopping
105.25
```

Option 6

```
Do you want to continue? (y/n)y
1.  Display total income
2.  Display total expenses
3.  Display user balance
4.  Category wise distribution of expenses
5.  Query the expense of a certain category
6.  Savings suggestion
7.  Monthly savings plan

Select option: 6

Enter amount you want to save over the next year: 10000
You must save 31.56 % of your income per month
i.e. 833.33 / month
Based on your balance, you need to reduce expenses or increase income to reach your savings goals
```

Option 7 (a)

```
Do you want to continue? (y/n)y
1.  Display total income
2.  Display total expenses
3.  Display user balance
4.  Category wise distribution of expenses
5.  Query the expense of a certain category
6.  Savings suggestion
7.  Monthly savings plan

Select option: 7

Enter % of income you want to save: 10
You must save 264.025 / month
Based on your balance, you are on track to reach your savings goals
```

Option 7 (b)

```
Do you want to continue? (y/n)y
1.  Display total income
2.  Display total expenses
3.  Display user balance
4.  Category wise distribution of expenses
5.  Query the expense of a certain category
6.  Savings suggestion
7.  Monthly savings plan

Select option: 7

Enter % of income you want to save: 34
You must save 897.685 / month
Based on your balance, you need to reduce expenses or increase income to reach your savings goals
```

Thank You!

Appendix 1

	A	B	C	D	E	F
1	Bank Statement Nr. 06/2022,,					
2	01.06.2022 until 30.06.2022,,					
3	Description,Booking Date,Amount					
4	ALLIANCE VENDING,01.06.2022,"-1,20â,¬"					
5	Mastercard â€¢ Bars & Restaurants,,					
6	Value Date 01.06.2022,,					
7	EL CORTE INGLES PLAZA,01.06.2022,"-7,54â,¬"					
8	Mastercard â€¢ Shopping,,					
9	Value Date 01.06.2022,,					
10	Andrei Ivlev,01.06.2022,"-300,00â,¬"					
11	Outgoing Transfers,,					
12	IBAN: LT793250046411044865 â€¢ BIC: REVOLT21XXX,,					
13	Sent from N26,,					
14	Value Date 01.06.2022,,					
15	N26,01.06.2022,"-1,99â,¬"					
16	Instant bank transfer fee,,					
17	Value Date 01.06.2022,,					
18	BON PREU, S.A.U.,01.06.2022,"-2,59â,¬"					
19	Mastercard â€¢ Groceries,,					
20	Value Date 01.06.2022,,					
21	MARKET VIA LAIETANA,01.06.2022,"-3,72â,¬"					
22	Mastercard â€¢ Groceries,,					
23	Value Date 01.06.2022,,					
24	ESADE CREAPOLIS ARAMAR,02.06.2022,"-7,35â,¬"					
25	Mastercard â€¢ Bars & Restaurants,,					
26	Value Date 02.06.2022,,					
27	LA RUSTICA,02.06.2022,"-2,89â,¬"					
28	Mastercard â€¢ Bars & Restaurants,,					
29	Value Date 02.06.2022					

Why Pre-processing was necessary