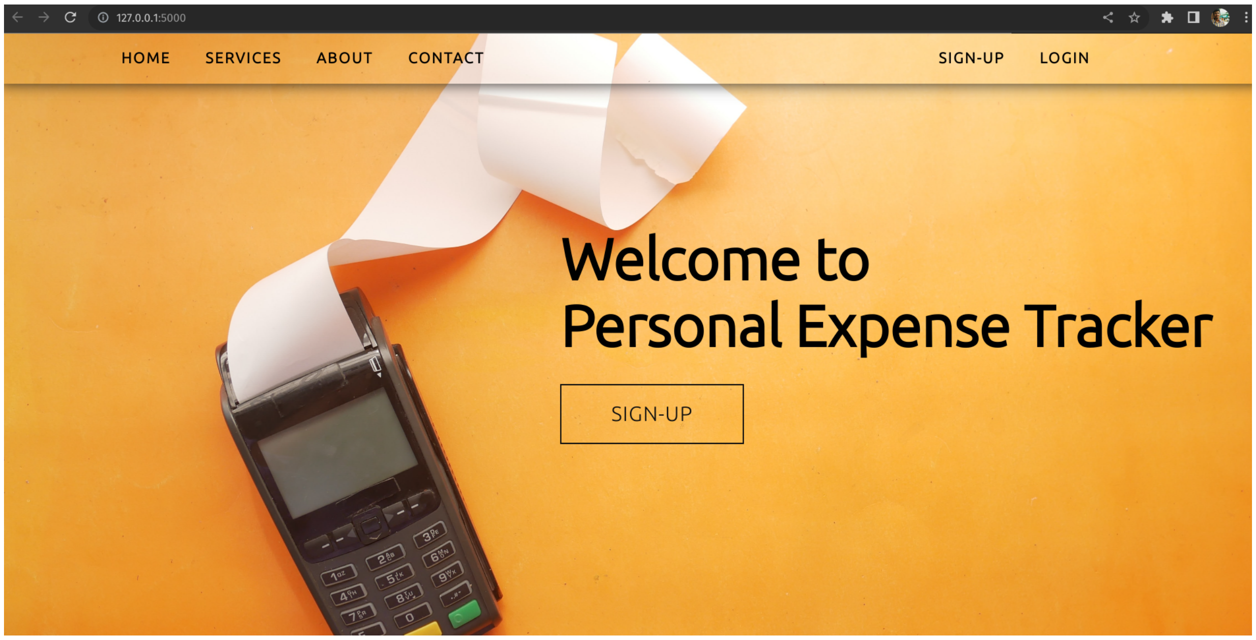
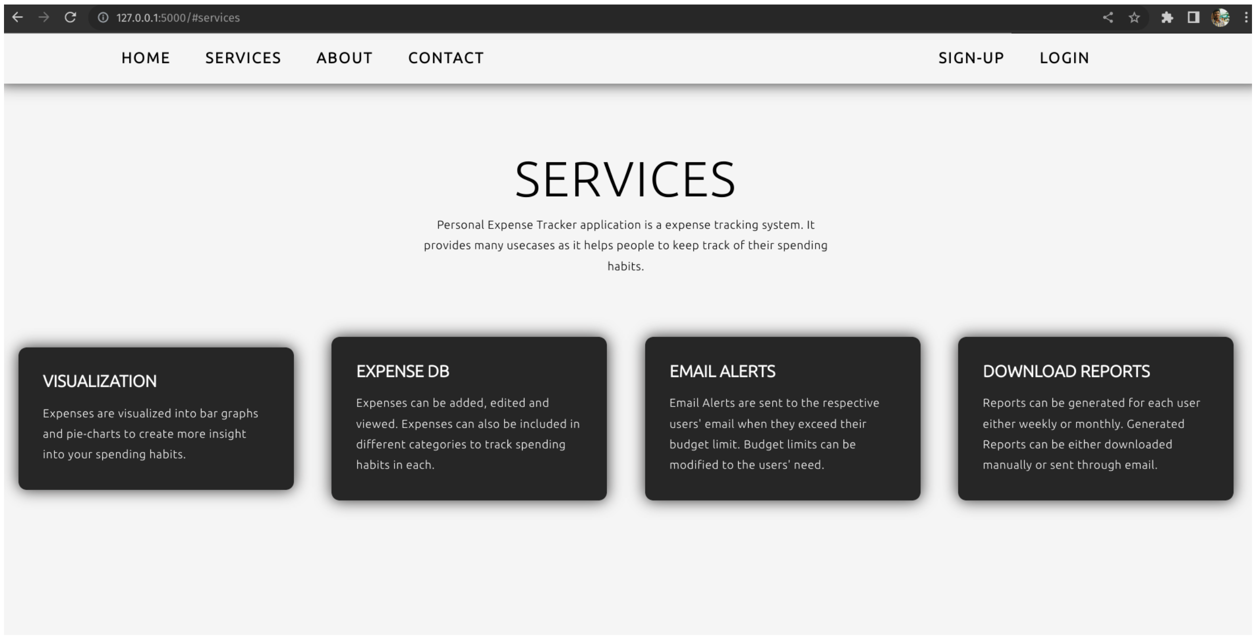
DEVELOPMENT PHASE
SPRINT - 1

Team ID	PNT2022TMID35639
Project Name	Personal Expense Tracker Application

SCREENSHOTS:





SERVICES

Personal Expense Tracker application is a expense tracking system. It provides many usecases as it helps people to keep track of their spending habits.

VISUALIZATION

Expenses are visualized into bar graphs and pie-charts to create more insight into your spending habits.

EXPENSE DB

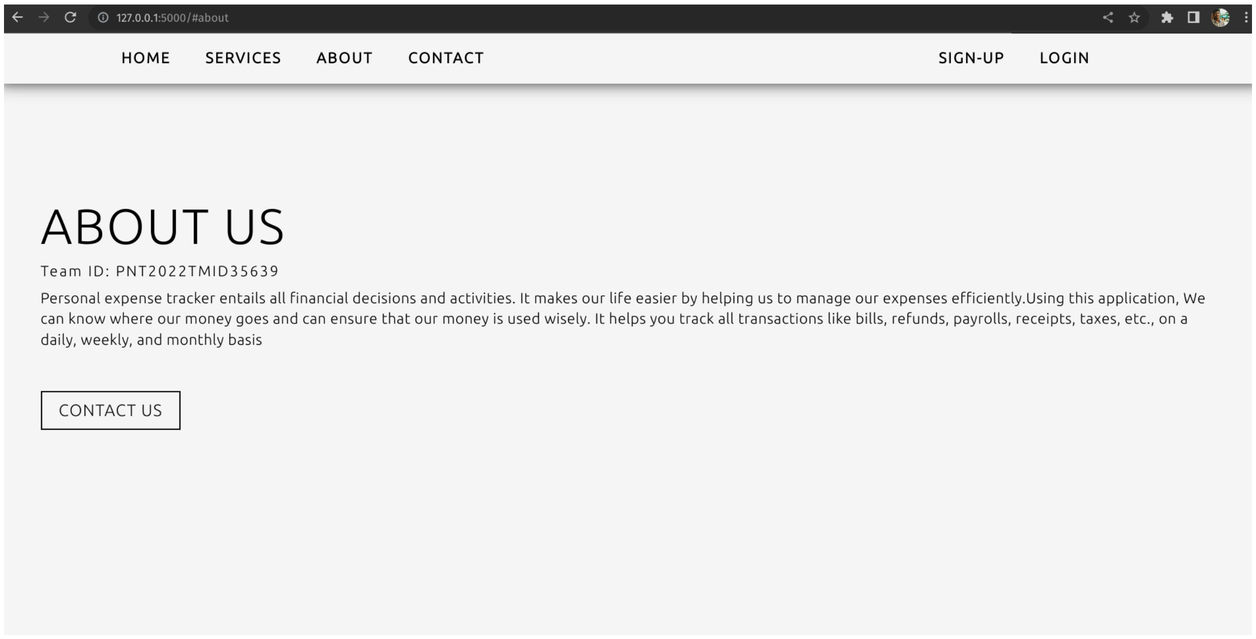
Expenses can be added, edited and viewed. Expenses can also be included in different categories to track spending habits in each.

EMAIL ALERTS

Email Alerts are sent to the respective users' email when they exceed their budget limit. Budget limits can be modified to the users' need.

DOWNLOAD REPORTS

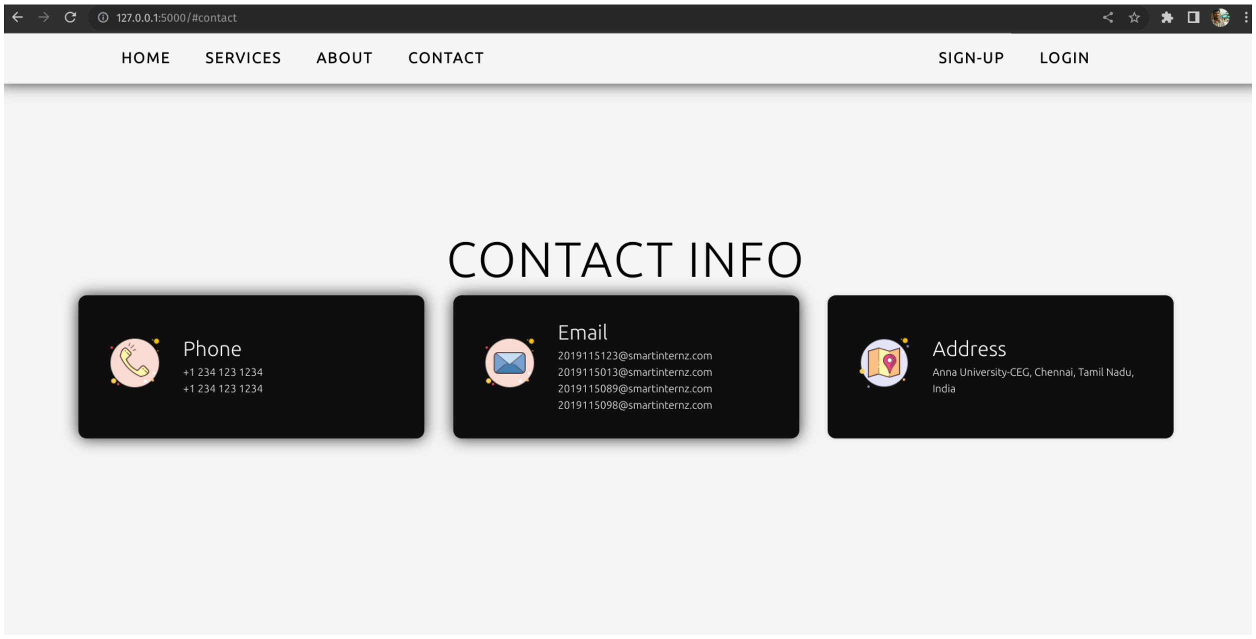
Reports can be generated for each user either weekly or monthly. Generated Reports can be either downloaded manually or sent through email.



ABOUT US

Team ID: PNT2022TMID35639

Personal expense tracker entails all financial decisions and activities. It makes our life easier by helping us to manage our expenses efficiently.Using this application, We can know where our money goes and can ensure that our money is used wisely. It helps you track all transactions like bills, refunds, payrolls, receipts, taxes, etc., on a daily, weekly, and monthly basis

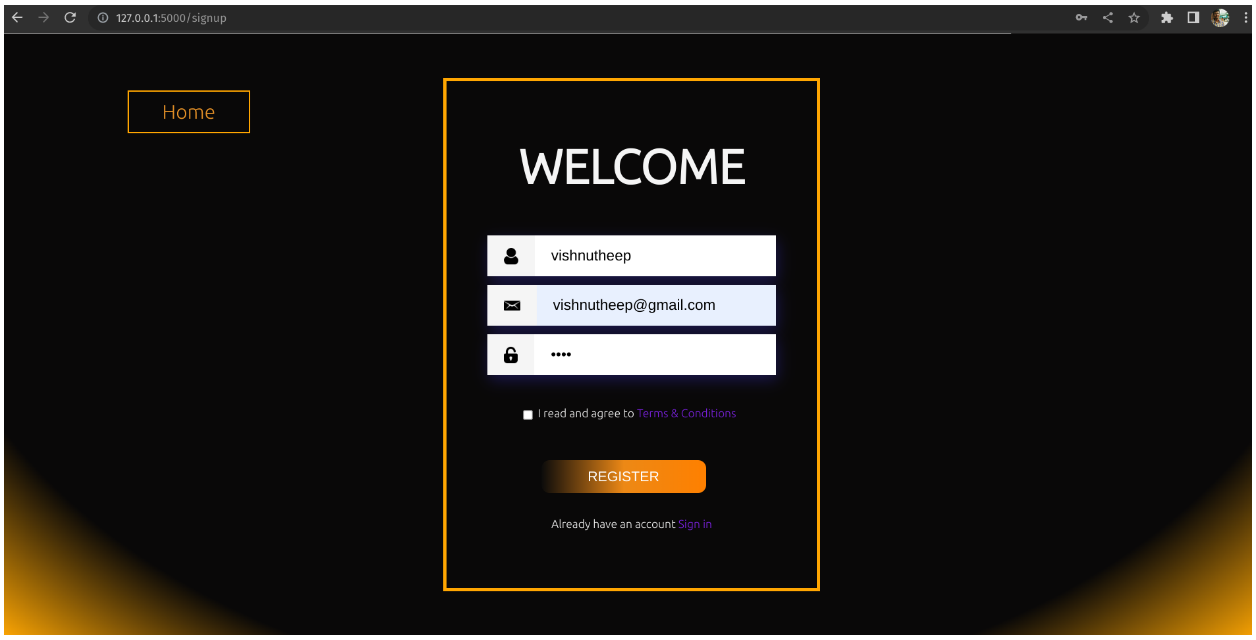


CONTACT INFO

Phone
+1 234 123 1234
+1 234 123 1234

Email
2019115123@smartinternz.com
2019115013@smartinternz.com
2019115089@smartinternz.com
2019115098@smartinternz.com

Address
Anna University-CEG, Chennai, Tamil Nadu, India


127.0.0.1:5000/signup

Home

WELCOME

vishnutheep

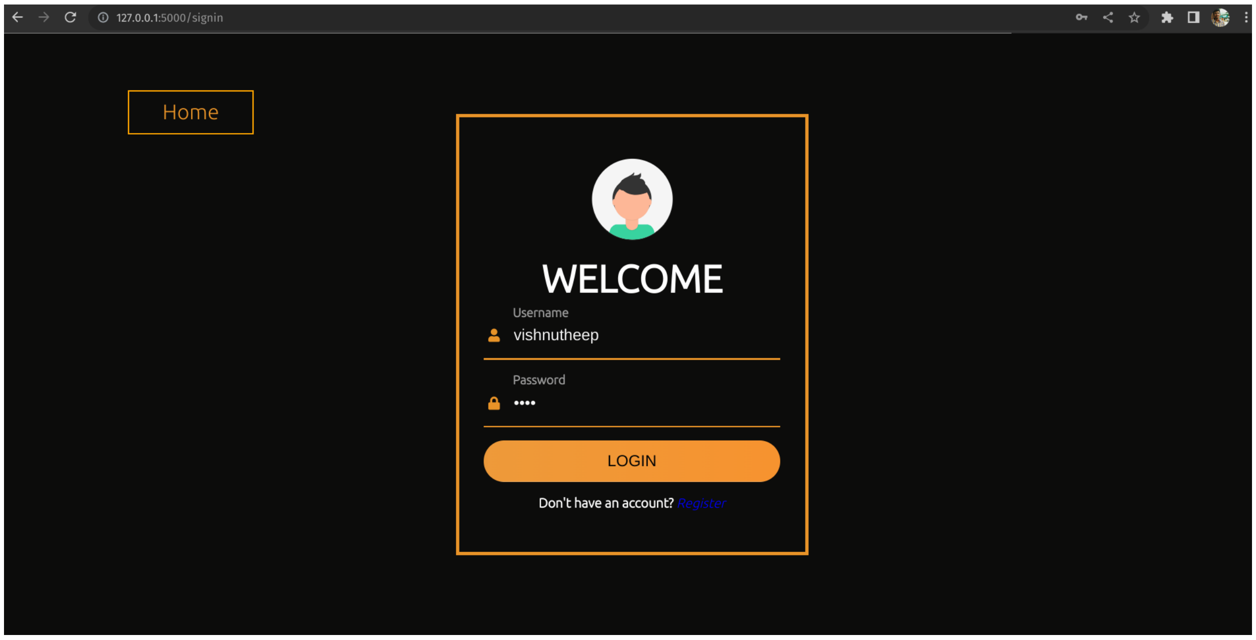
vishnutheep@gmail.com

••••

I read and agree to Terms & Conditions

REGISTER

Already have an account Sign in


127.0.0.1:5000/signin

Home

WELCOME

Username

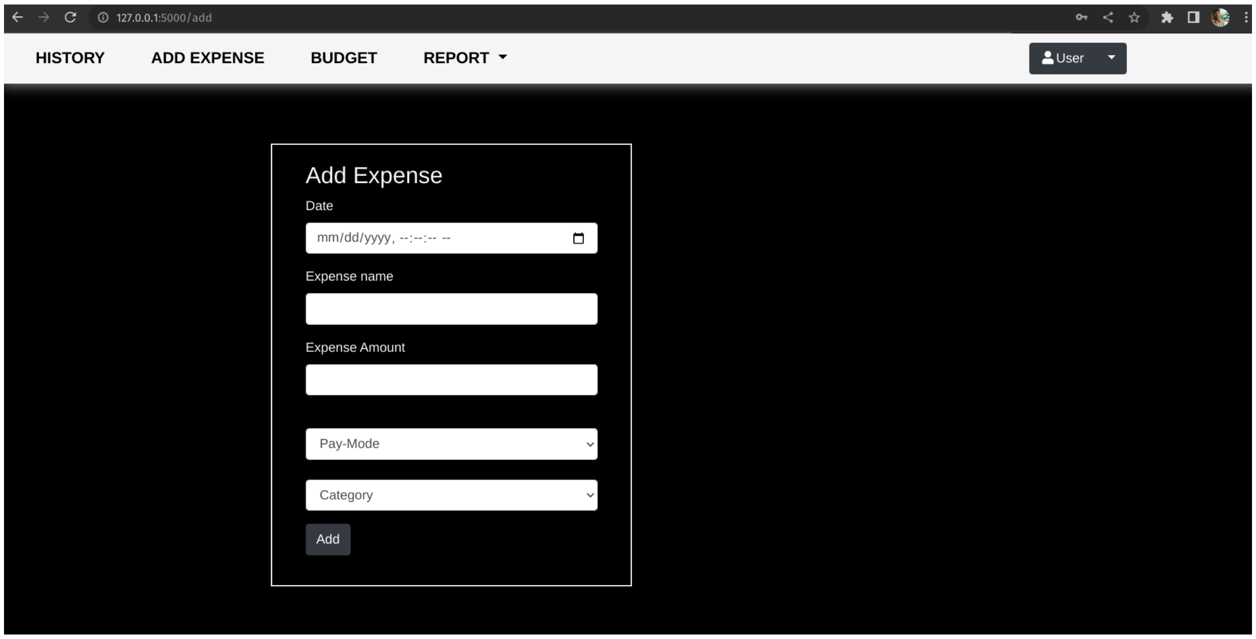
vishnutheep

Password

••••

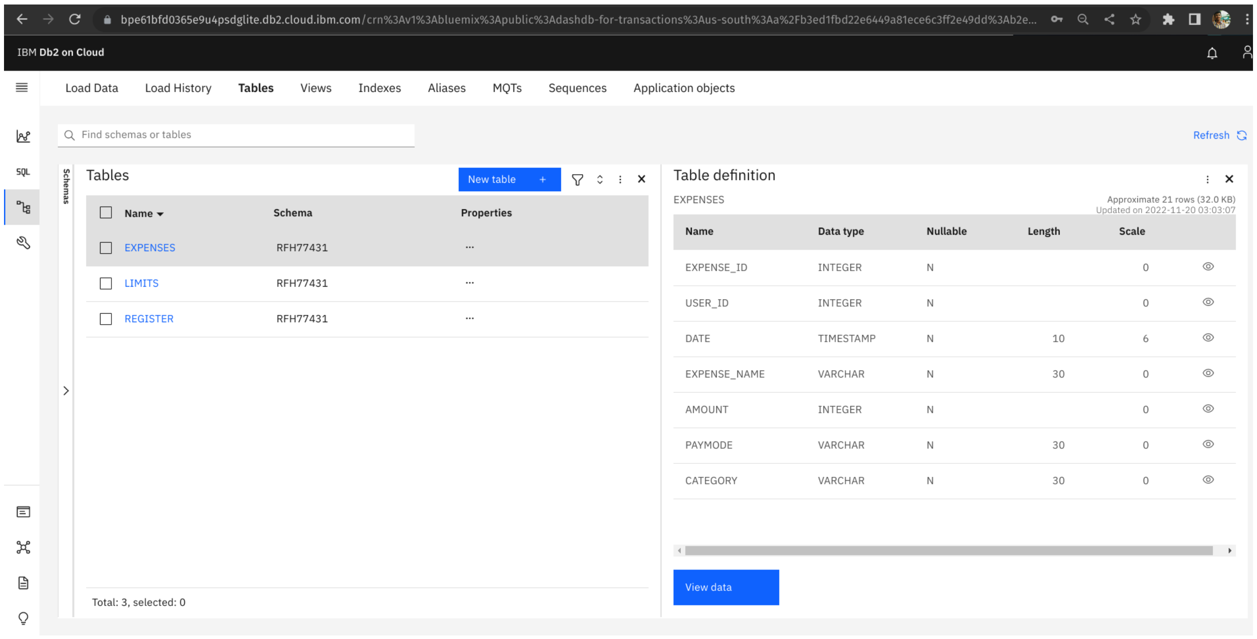
LOGIN

Don't have an account? Register



Add Expense

Date	mm/dd/yyyy, --:--:-- --
Expense name	
Expense Amount	
	Pay-Mode
	Category


bpe61bfd0365e9u4psdglite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aus-south%3Aa%2Fb3ed1fbd22e6449a81ece6c3ff2e49dd%3Ab2e...
IBM Db2 on Cloud
Load Data
Load History
Tables
Views
Indexes
Aliases
MQTs
Sequences
Application objects
Find schemas or tables
Refresh
Schemas
New table
View data

Tables

Name	Schema	Properties
EXPENSES	RFH77431	...
LIMITS	RFH77431	...
REGISTER	RFH77431	...

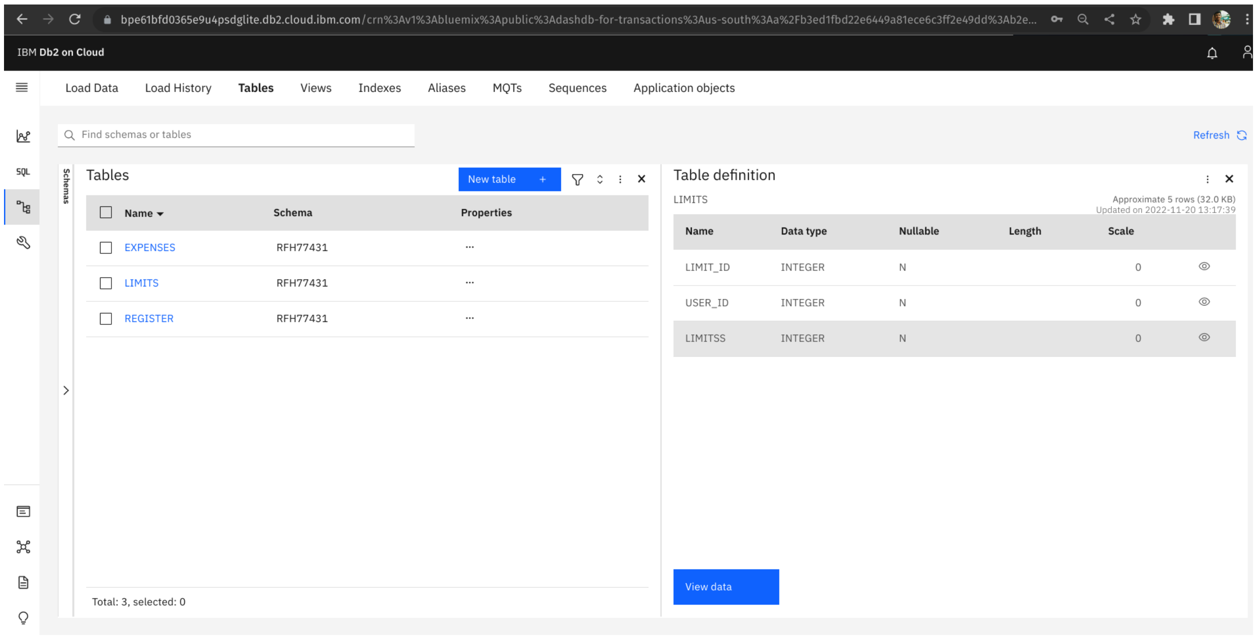
Total: 3, selected: 0

Table definition

EXPENSES

Approximate 21 rows (32.0 KB)
Updated on 2022-11-20 03:03:07

Name	Data type	Nullable	Length	Scale
EXPENSE_ID	INTEGER	N		0
USER_ID	INTEGER	N		0
DATE	TIMESTAMP	N	10	6
EXPENSE_NAME	VARCHAR	N	30	0
AMOUNT	INTEGER	N		0
PAYMODE	VARCHAR	N	30	0
CATEGORY	VARCHAR	N	30	0



Tables

Name	Schema	Properties
EXPENSES	RFH77431	···
LIMITS	RFH77431	···
REGISTER	RFH77431	···

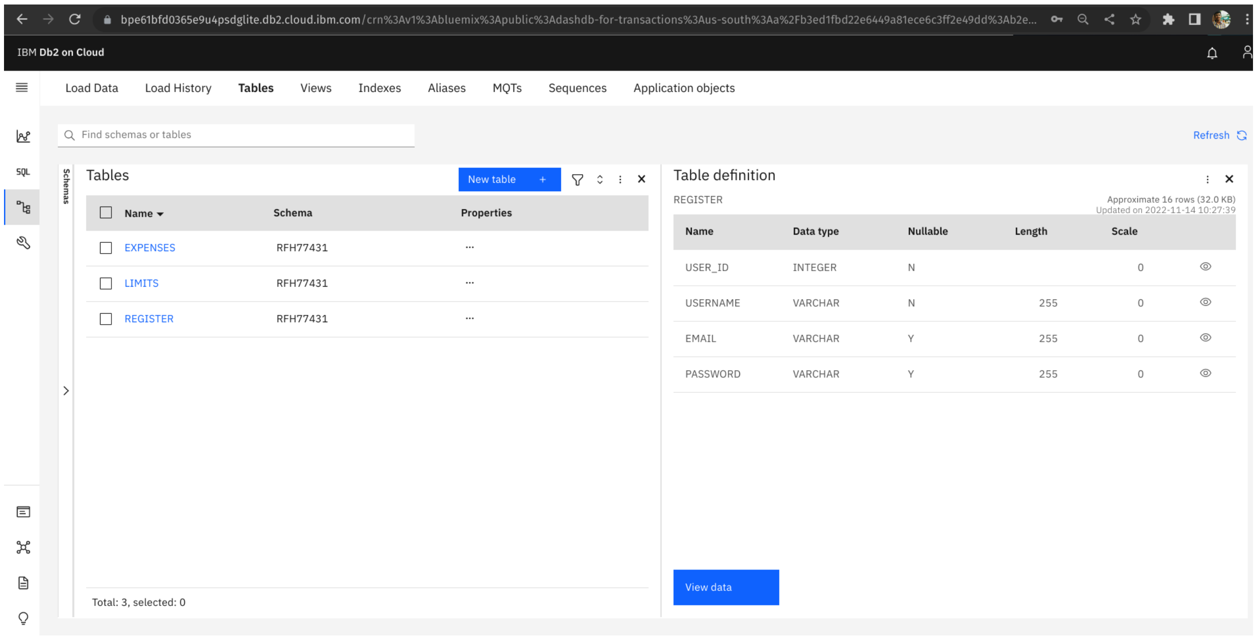
Total: 3, selected: 0

Table definition

LIMITS

Approximate 5 rows (32.0 KB)
Updated on 2022-11-20 13:17:39

Name	Data type	Nullable	Length	Scale
LIMIT_ID	INTEGER	N		0
USER_ID	INTEGER	N		0
LIMITSS	INTEGER	N		0



Tables

Name	Schema	Properties
EXPENSES	RFH77431	···
LIMITS	RFH77431	···
REGISTER	RFH77431	···

Total: 3, selected: 0

Table definition

REGISTER

Approximate 16 rows (32.0 KB)
Updated on 2022-11-14 10:27:39

Name	Data type	Nullable	Length	Scale
USER_ID	INTEGER	N		0
USERNAME	VARCHAR	N	255	0
EMAIL	VARCHAR	Y	255	0
PASSWORD	VARCHAR	Y	255	0

CODE:

```
from flask import Flask, render_template, request, redirect, session ,
make_response, url_for, json, flash
import re, ibm_db
import ibm_db_dbi ,pandas as pd
```

```
from flask_mail import Mail, Message
import os, datetime
from pandas import Timestamp
from pretty_html_table import build_table
import pdfkit


app = Flask(__name__)

app.secret_key = 'SECRET_KEY'



conn = ibm_db.connect("DATABASE=bludb;
HOSTNAME=fbd88901-ebdb-4a4f-a32e-9822b9fb237b.c1ogj3sd0tgtu0lqde00.da
tabases.appdomain.cloud; PORT=32731; SECURITY=SSL;
SSLServerCertificate=DigiCertGlobalRootCA.crt;
UID=rfh77431;PWD=1WClhcJWgdCoeAk5","","")
pd_conn = ibm_db_dbi.Connection(conn)

#HOME--PAGE
@app.route("/home")
def home():
   return render_template("homepage.html")

@app.route("/")
def add():
   return render_template("home.html")



#SIGN--UP--OR--REGISTER
@app.route("/signup")
def signup():
   return render_template("signup.html")
```

```
@app.route('/register', methods =['GET', 'POST'])
def register():
    msg = ''
    if request.method == 'POST' :
        username = request.form['username']
        email = request.form['email']
        password = request.form['password']

        sql = 'SELECT * from REGISTER WHERE USERNAME = ?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print(account)
        if account:
            msg = 'Account already exists !'
        elif not re.match(r'[^@]+@[^@]+\.[^@]+', email):
            msg = 'Invalid email address !'
        elif not re.match(r'[A-Za-z0-9]+', username):
            msg = 'name must contain only characters and numbers !'
        else:
            sql = "INSERT INTO
REGISTER(USER_ID,USERNAME,EMAIL,PASSWORD)
VALUES(DEFAULT,?,?,?)"
            stmt=ibm_db.prepare(conn,sql)
            ibm_db.bind_param(stmt,1,username)
            ibm_db.bind_param(stmt,2,email)
            ibm_db.bind_param(stmt,3,password)
            ibm_db.execute(stmt)

            msg = 'You have successfully registered !'
        return render_template('signup.html', msg = msg)
```

```
#LOGIN--PAGE
@app.route("/signin")
def signin():
    return render_template("login.html")

@app.route('/login',methods =['GET', 'POST'])
def login():
    global userid
    msg = ''

    if request.method == 'POST' :
        username = request.form['username']
        password = request.form['password']

        sql = 'SELECT * from REGISTER WHERE USERNAME = ? AND
PASSWORD = ?'
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)

        account = ibm_db.fetch_assoc(stmt)

        print (account)
        if account:
            session['loggedin'] = True
            session['id'] = account['USER_ID']
            userid =  account['USER_ID']
            session['username'] = account['USERNAME']

            return redirect('/add')
        else:
            msg = 'Incorrect username / password !'
    return render_template('login.html', msg = msg)
```

```
#ADDING----DATA
@app.route("/add")
def adding():

    return render_template('add.html')



@app.route('/addexpense',methods=['GET', 'POST'])
def addexpense():
    return redirect("/display")



#DISPLAY---graph
@app.route("/display")
def display():
    if session.get("id")== None or session.get("username") == None:
        return redirect('/')

    print(session["username"],session['id'])

    id = str(session['id'])

    sql = 'SELECT * FROM EXPENSES WHERE USER_ID = {} ORDER BY DATE
DESC'.format(id)
    df = pd.read_sql(sql,pd_conn)
    expense = df.values.tolist()

    print(expense)

    return render_template('display.html' ,expense = expense)


 #limit
@app.route("/limit" )
```

```
def limit():
      return redirect('/limitn')

@app.route("/limitnum" , methods = ['POST' ])
def limitnum():
       return redirect('/limitn')


@app.route("/limitn")
def limitn():

    return render_template("limit.html",
type="Monthly",expense_data=monthly_expense, y=s)


#log-out
@app.route('/logout')
def logout():
   session.pop('loggedin', None)
   session.pop('id', None)
   session.pop('username', None)
   return render_template('home.html')

if __name__ == "__main__":
    app.run(debug=True)
```