PERSONAL EXPENSE TRACKER

Using python flask

Developed by: Saloni M P

VMware Tanzu Build-A-Thon

1. INTRODUCTION

In today's busy and expensive life, we are in a great rush to make money. But at the end of the month, we broke off. As we are unknowingly spending money on little and unwanted things. So, we have come over with the idea to track our earnings. Personal Expense Tracker (PET) aims to help everyone who are planning to know their expenses and save from it. PET is an website which users can execute in their mobile phones and update their daily expenses so that they are well known to their expenses. Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spent and also can add some information in additional information to specify the expense. Although this website is focused on new job holders, interns and teenagers, everyone who wants to track their expense can use this app. Personal Expense Tracker System is intended to monitor Income-Expense of a person on an everyday premise. This System separates the Income-based daily expenses.

1.1 Overview

Personal Expense Tracker System is designed to keep a track of Income-Expense of a person on a day-to-day basis. This System divides the Income based on daily expenses. If you exceed day's expense, system will cut it from your income and will provide new daily expense allowed amount. It will let you add the savings amount, which you had saved for some particular days. This System takes Income from person and divides in daily expense allowed. If u exceed that day's expense it will cut if from your income and gives an alert mail for the registered person. Once we start off by tracking our expenses each day, we will be able to get a better idea where you are spending your money, so you stay in control and achieve your goal.

1.2 Purpose

Personal Expense Tracker (PET) aims to help everyone who are planning to know their expenses and save from it. PET is a website in which users can execute in their laptop and update their daily expenses so that they are well known to their expenses. Here user can define their own categories for expense type like food, clothing, rent and bills where they have to enter the money that has been spent and also can add some information in additional information to specify the expense. Although this app is focused on new job holders, interns and teenagers, everyone who wants to track their expense can use this app.

2. LITERATURE SURVEY

Tracking daily expense is not so innovative. Many traditional and technological approach is found to track our expenses and budget with their own functionality. From decades ago, and today we have been writing our expenditure in a register to calculate the profit or saving. Not only this many desktop and mobile applications has been developed for this purpose. Quicken and Microsoft money where the first desktop applications was developed decades ago but was not so familiar with the users. Personal capital and dollar bird application were used to visualize the expenses in chart or graphs with the calendar system. QuickBooks were the application for the small business holder to wrap up their whole business. YNAB and Penny were the latest application which were embedded with AI and applicable for importing expenses automatically. However, Mint was the one which was widely used and trusted. Explaining about the latest application built in this category, YNAB is an expense tracker that gives the automatic tracking of our expense through our bank account or credit cards.

2.1 Existing problem

In existing, we need to maintain the Excel sheets, CSV etc. files for the user daily and monthly expenses. In existing, there is no as such complete solution to keep a track of its daily expenditure easily. To do so a person as to keep a log in a diary or in a computer, also all the calculations need to be done by the user which may sometimes results in errors leading to losses.

2.2 Proposed solution

To reduce manual calculations, we propose an application which is developed by Python flask. This application allows users to maintain a digital automated diary. Each user will be required to register on the system at any time, the user will be provided id, which will be used to maintain the record of each unique user. Personal Expense Tracker application which will keep a track of Income-Expense of a user on a day-to-day basis. This application takes Income from user and divides in daily expense allowed. If u exceed that day's expense it will cut if from your income and give alert mail to the registered user.

3. THEORITICAL ANALYSIS

3.1 Block Diagram

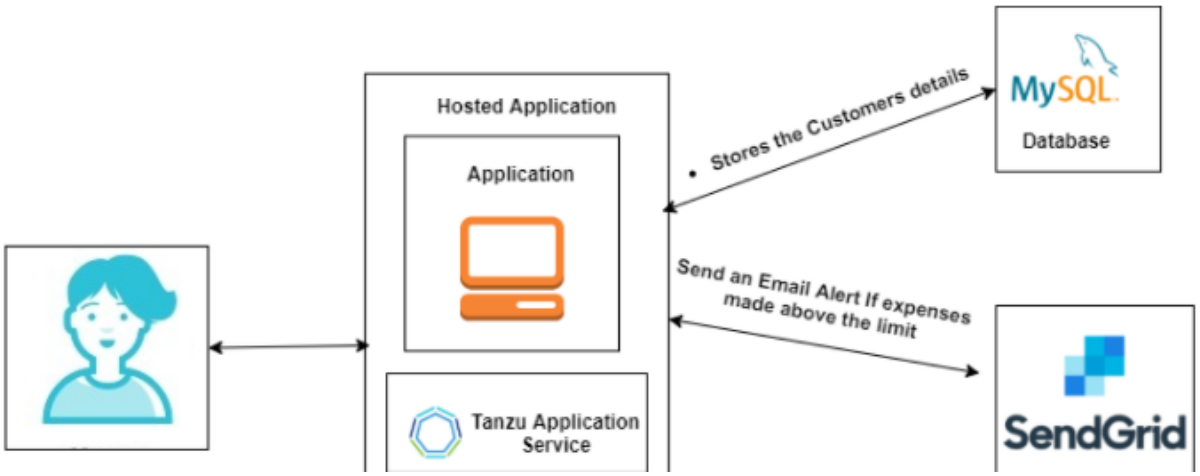

Fig: Block Diagram

3.2 Hardware / Software Designing

3.2.1 Hardware Requirements

- Processor: Intel® Core™ i3-7020 CPU @ 2.30GHz
- RAM: 4GB
- Hard Disk: 1TB

3.2.2 Software Requirements

- Operating System: Windows 10
- Language: Python
- Database: Remote MySQL
- IDE: Spyder
- Browser: Google Chrome

4. FLOWCHART

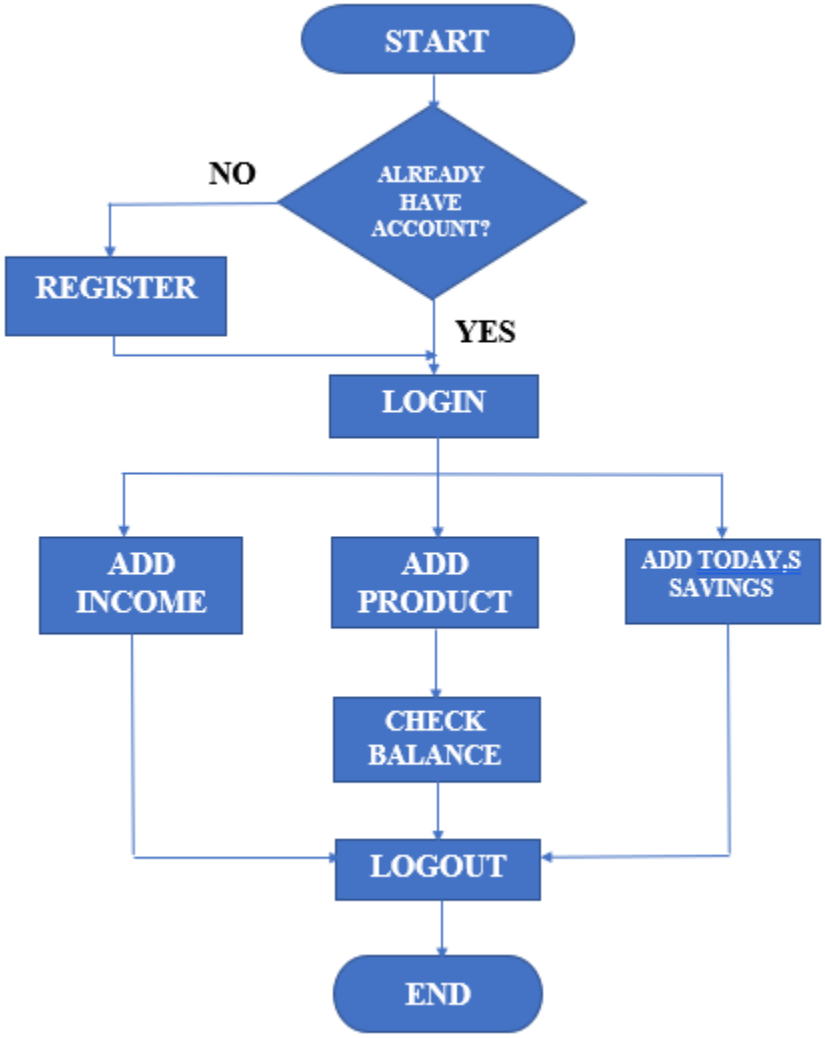

Fig: Flowchart

5. RESULT

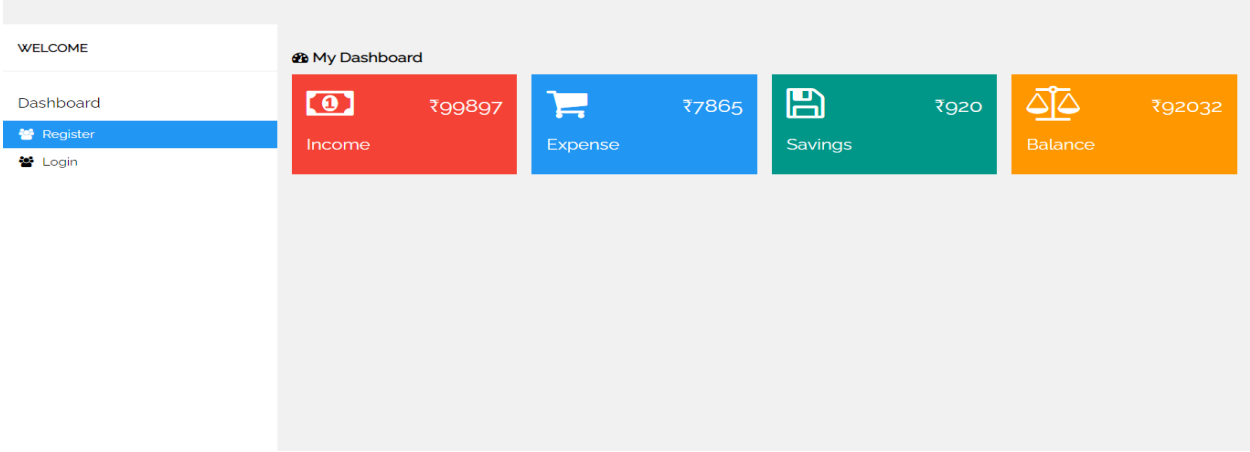

Fig: Home Page

Description: The above figure represents the home page of the project.

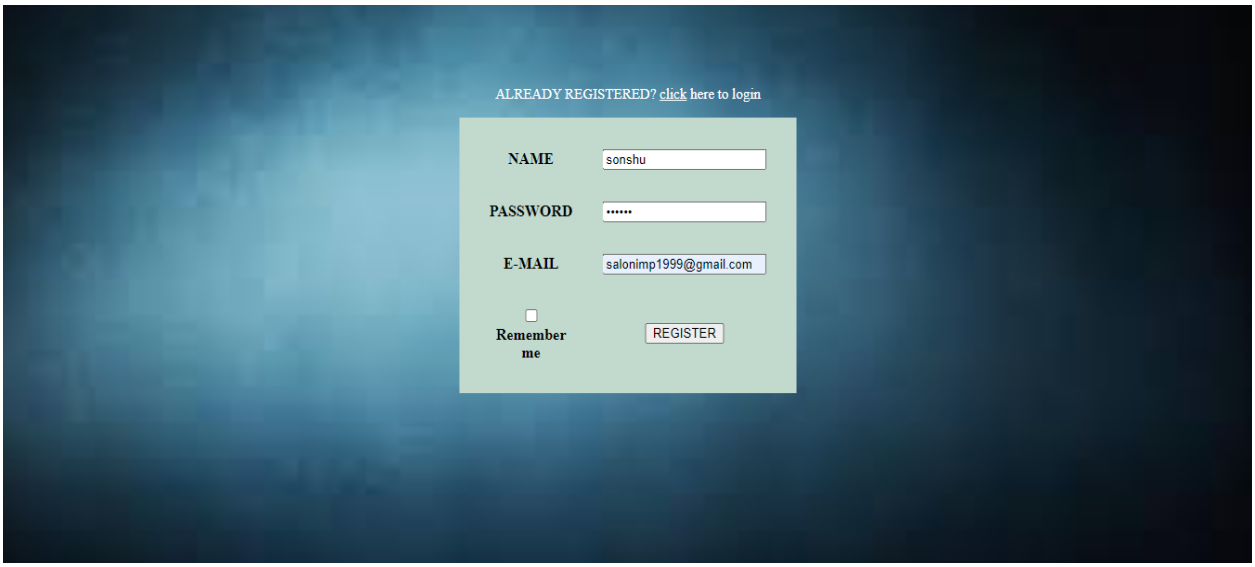

Fig: Register Page

Description: The above figure represents the register page where any person can register for the application if not registered before.

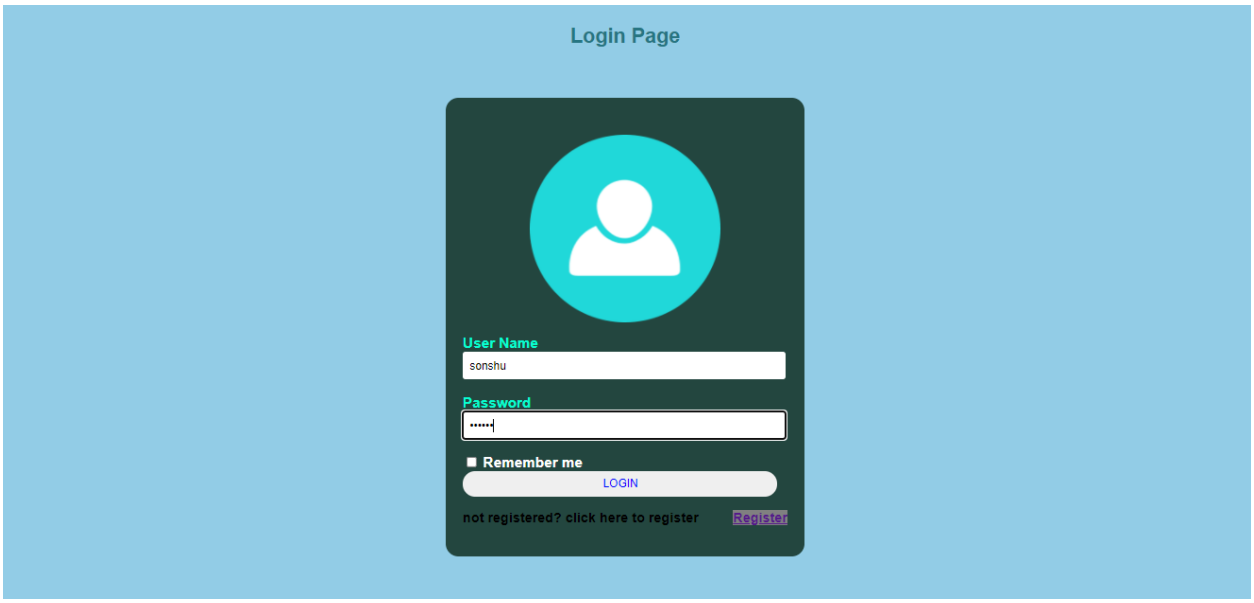

Fig: Login Page

Description: The above page represents the login page where registered user can login.

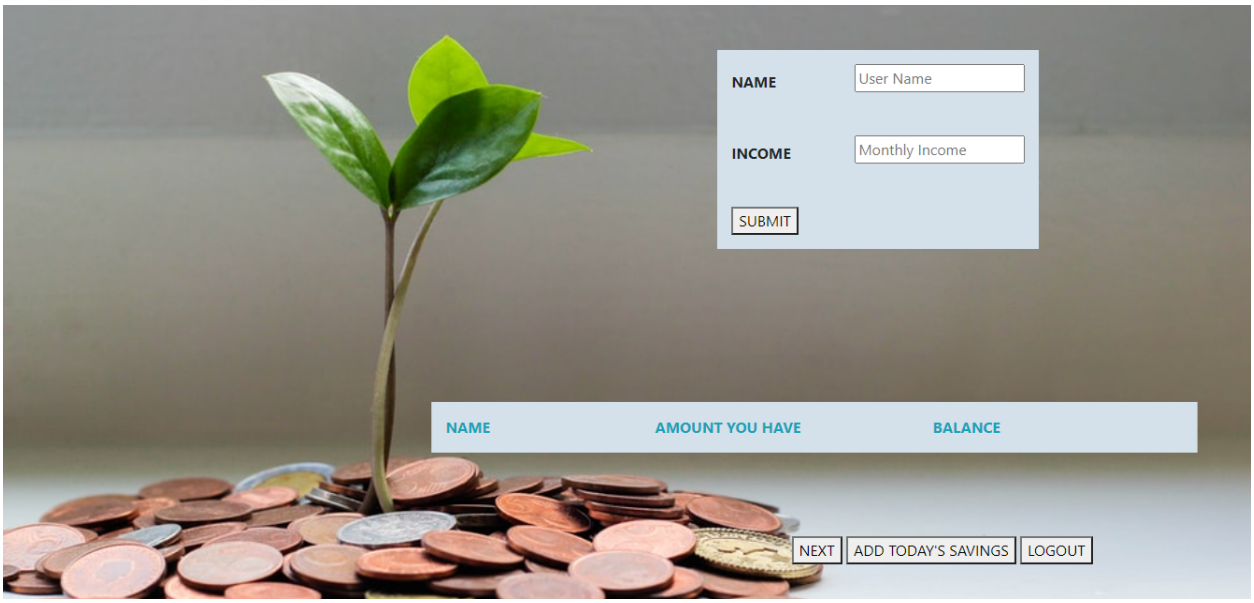

Fig: Income Page

Description: The above page represents the income page where the registered user can add their income and check balance.

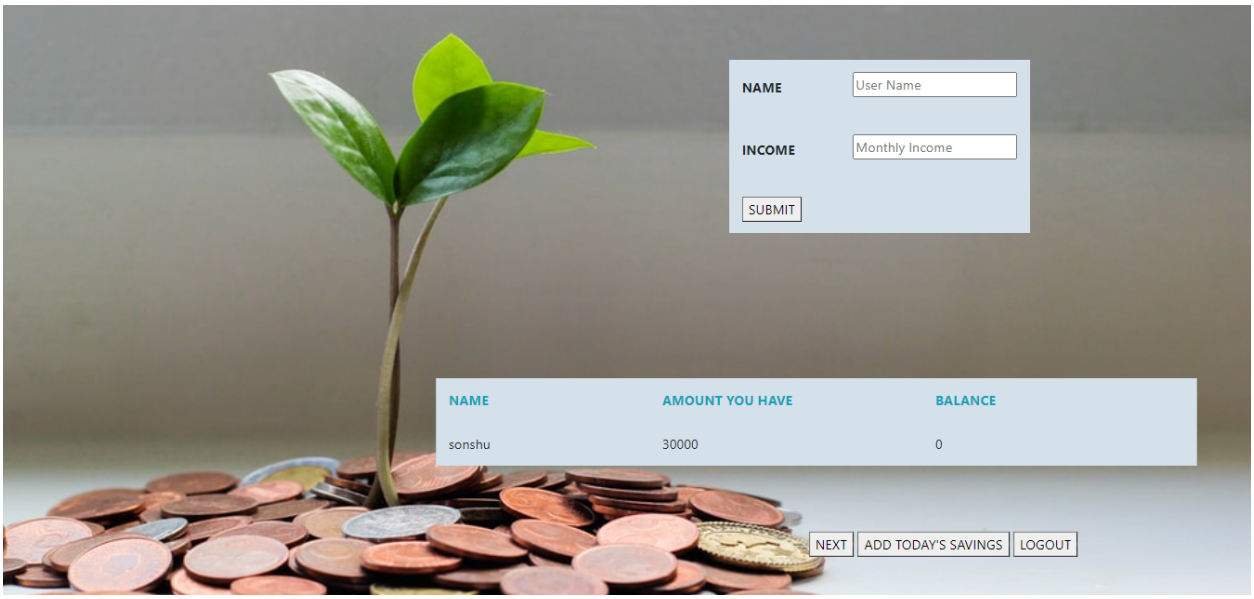

Fig: Income Page after adding income

Description: The above page represents the income page after user add their income.

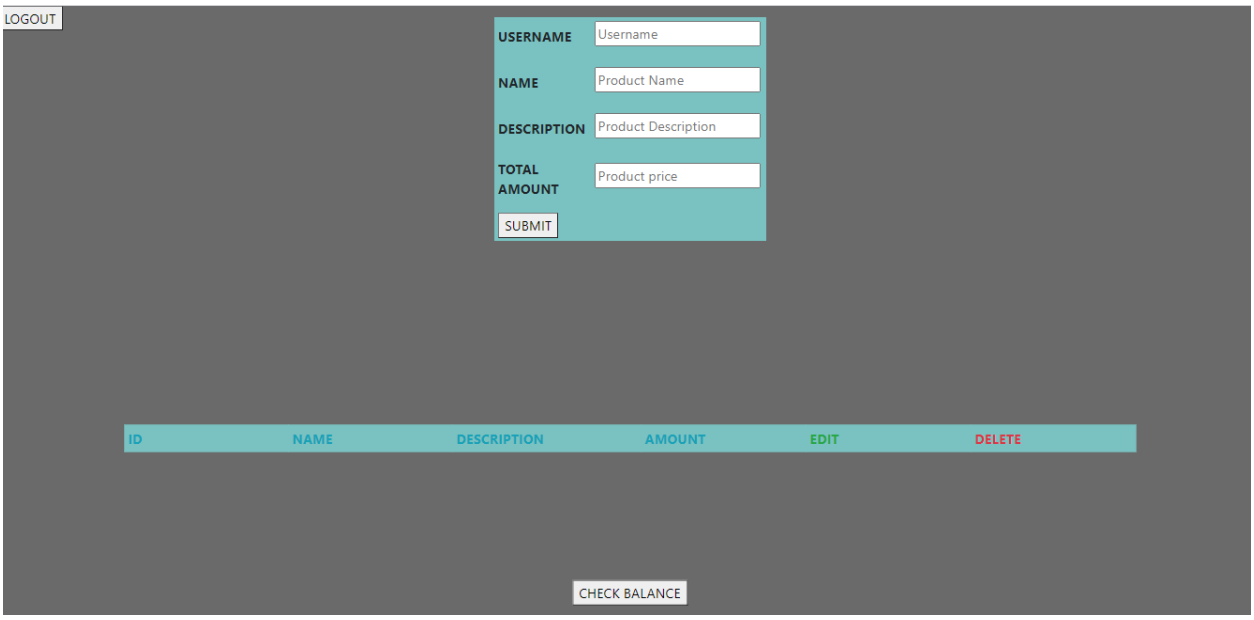

Fig: View Page

Description: The above page represents the view page where user can add their product, description and its amount.

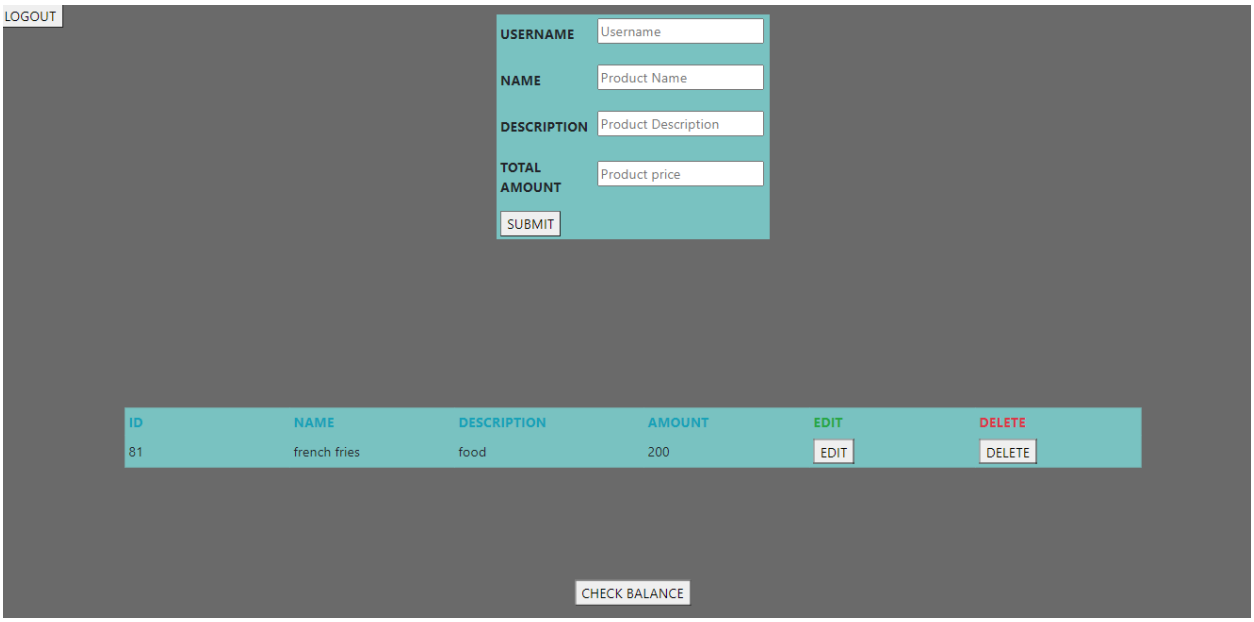

Fig: View Page after adding product

Description: The above page represents the view page after adding the product, description and amount.

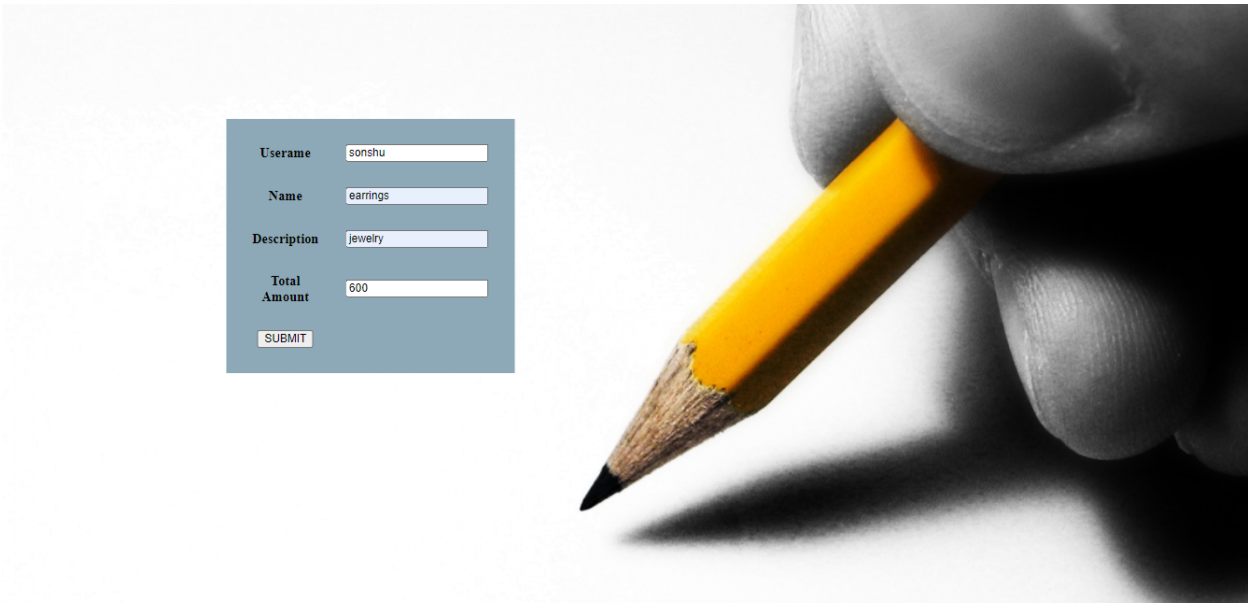

Fig: Edit Page

Description: The above page represents the edit page where user can edit the product name, description and amount if it is wrong.

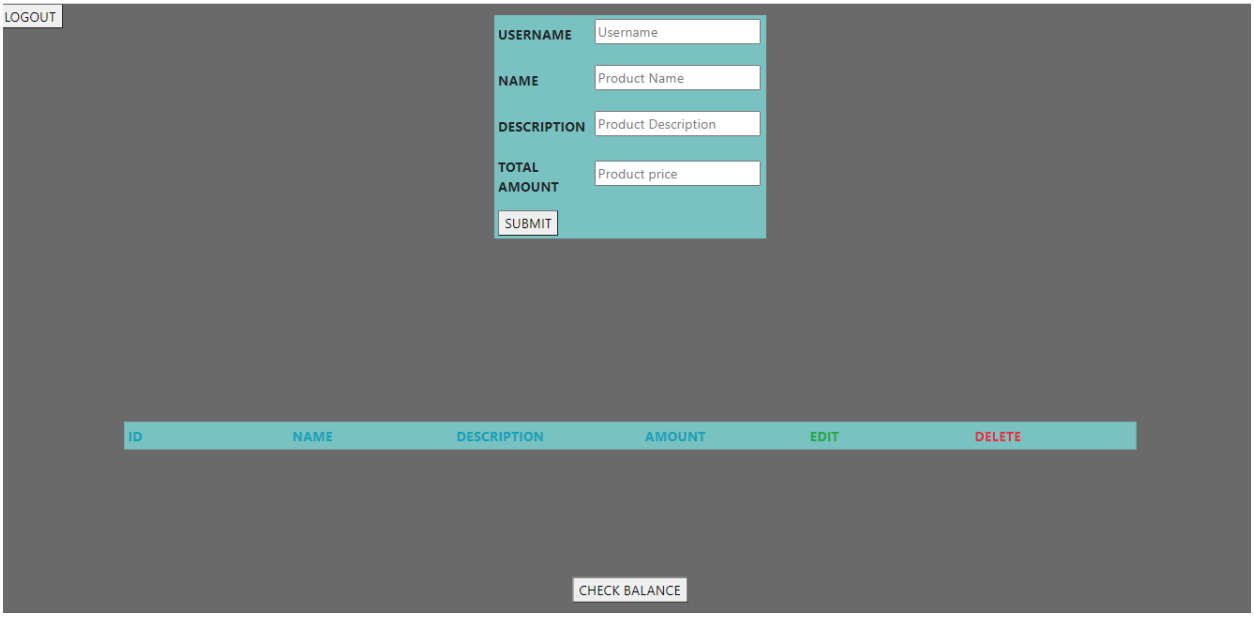

Fig: View Page after deleting the product

Description: The above page represents the view page after deleting the product.

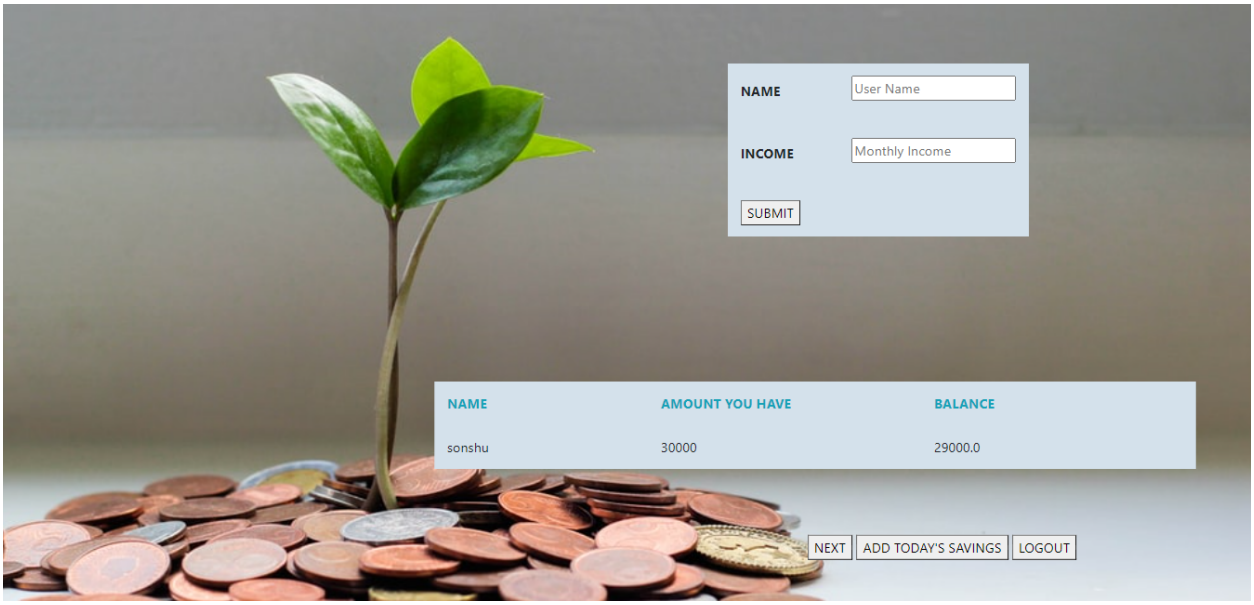

Fig: Income Page after adding product

Description: The above page represents the income page after adding the product. Here, the user can get the updated balance.

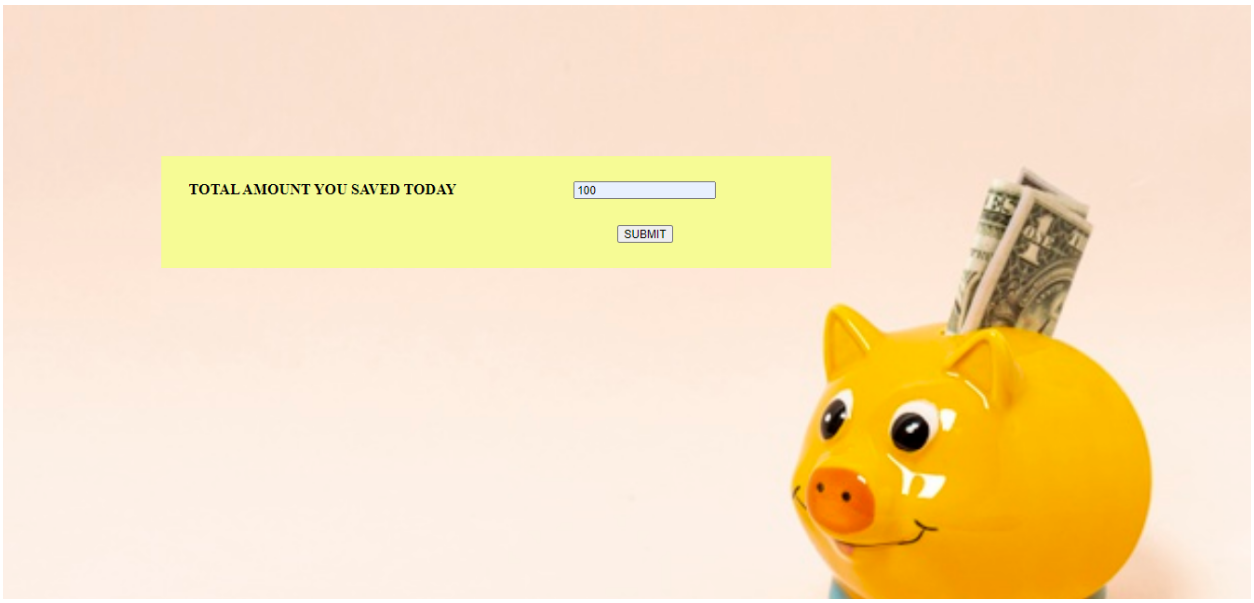

Fig: Savings Page

Description: The above page represents the savings page where user can add their savings.

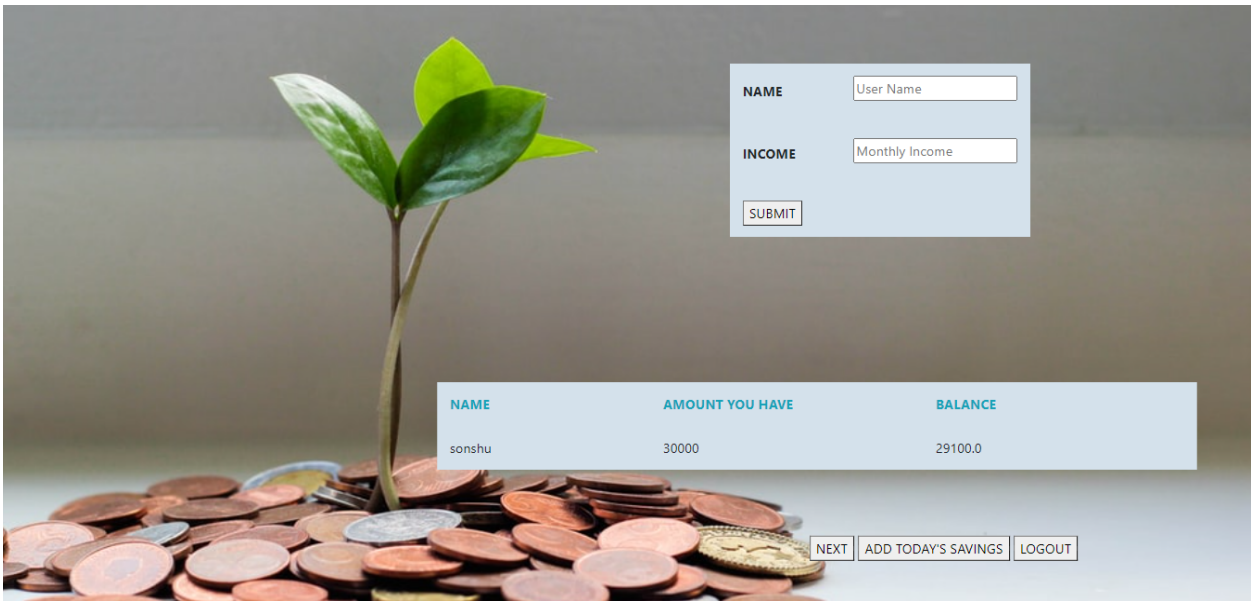

Fig: Income Page after adding savings

Description: The above page represents the income page after adding savings in which the balance will be updated.

6. ADVANTAGES AND DISADVANTAGES

6.1 Advantages

- This system helps a person to reduce their expenses.
- Easy to use.
- You'll have better insight into your spending habits.
- It provides a better overview and comprehensive analysis.
- Saving and Investment.

6.2 Disadvantages

- May get inaccurate results if data is not inserted in correct manner.
- User have to entry every record manually.
- Person who is handling system must have some technical knowledge.

7. CONCLUSION

After making this application we assure that this application will help its users to manage the cost of their daily expenditure. It will guide them and aware them about their daily expenses. It will prove to be helpful for the people who are frustrated with their daily budget management, irritated because of amount of expenses and wishes to manage money and to preserve the record of their daily cost which may be useful to change their way of spending money. In short, this application will help its users to overcome the wastage of money.

8. FUTURE SCOPE

In further days, there will be calendar based reports and pay mode embedded with the app. Also, backup details will be recorded on cloud. Not only can you track your expenses, you can categorise them too. So, you know exactly how much of money you are spending under each of these categories.

9. BIBLIOGRAPHY

- https://www.geeksforgeeks.org/login-and-registration-project-using-flask-and-mysql/
- https://www.w3schools.com/
- https://html.com
- https://www.w3schools.com/icons/fontawesome5_icons_finance.asp
- https://www.filemakr.com/btech-final-year-project-report-daily-expense-tracker

APPENDIX

A. Source Code

```
from flask import Flask,render_template,request,redirect,session
from flaskext.mysql import MySQL
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
mysql = MySQL()
app = Flask(__name__)
app.secret_key="hello"
app.config['MYSQL_DATABASE_USER']='N2xPXoMdDK'
app.config['MYSQL_DATABASE_PASSWORD']='SkC3hQwn5D'
app.config['MYSQL_DATABASE_DB']='N2xPXoMdDK'
app.config['MYSQL_DATABASE_HOST']='remotemysql.com'
mysql = MySQL(app)
@app.route('/')
def root():
    return redirect('/demo')
@app.route("/demo")
```

```
def demo():
    return render_template('demo.html')
@app.route("/admin",methods=['GET','POST'])
def admin():
    if request.method=="POST":
        username=request.form['fname']
        password=request.form['pass']
        session['username']  = username
        conn=mysql.connect()
        cur=conn.cursor()
                        cur.execute("select  `password`,`email`  from  `adminregister`  WHERE `name`=%s",(username))
        data=cur.fetchall()
        emailid=data[0][1]
        session['emailid']=emailid
        if password==data[0][0]:
            return redirect('/income')
        else:
            return " admin login failed"
    else:
        return render_template('login.html')
@app.route("/adminregister",methods=['GET','POST'])
def adminregister():
    if request.method=="POST":
        username=request.form['fname']
        password1=request.form['pass']
        emailid=request.form['emailid']
        session['emailid']=emailid
        session['username']  = username
```

```
        conn=mysql.connect()
        cur=conn.cursor()
                    cur.execute("INSERT INTO `adminregister`(`name`, `password`, `email`) VALUES(%s,%s,%s)",(username,password1,emailid))
        conn.commit()
        return redirect('/income')
    else:
        return render_template('signup.html')
@app.route("/logout")
def logout():
    return render_template('demo.html')
@app.route("/add",methods=['GET','POST'])
def add():
    if request.method=="POST":
        un=request.form['un']
        pn=request.form['pn']
        pd=request.form['pd']
        pr=request.form['pr']
        session['username']=un
        conn=mysql.connect()
        cur=conn.cursor()

                cur.execute("INSERT INTO `expense`(`username`,`expensename`,`expensedesc`, `amount`) VALUES(%s,%s,%s,%s)",(un,pn,pd,pr))
        conn.commit()
        return redirect('view')
    else:
        return render_template('view.html')
@app.route("/addincome",methods=['GET','POST'])
```

```
def addincome():
    if request.method=="POST"
        pn=request.form['pn']
        pr=request.form['pr']
        session['username']=pn
        conn=mysql.connect()
        cur=conn.cursor()
                        cur.execute("INSERT INTO `income`(`username`,`income`,`balance`)
VALUES(%s,%s,%s)",(pn,pr,0))
        conn.commit()
        return redirect('income')
    else:
        return render_template('income.html')
@app.route("/edit",methods=['GET','POST'])
def edit():
    if request.method=="POST":
         id=request.form['id']
        un=request.form['un']
        expensename=request.form['expensename']
        expensedesc=request.form['expensedesc']
        amount=request.form['amount']
        conn=mysql.connect()
        cur=conn.cursor()
                                cur.execute("UPDATE `expense` SET `username`='"+un+"',
`expensename`='"+expensename+"',`expensedesc`='"+expensedesc+"'
,`amount`='"+amount+"'WHERE `id`='"+id+"'")
        conn.commit()
        return redirect('view')
    else:
        id=request.args.get('id')
```

```
        conn=mysql.connect()
        cur=conn.cursor()
        cur.execute("select * from expense where `id`='"+str(id)+"'")
        data=cur.fetchone()
        return render_template('edit.html',product=data)
@app.route("/delete",methods=['GET','POST'])
def delete():
    id=request.args.get('id')
    conn=mysql.connect()
    cur=conn.cursor()
    cur.execute("DELETE FROM `expense` WHERE `id`='"+id+"'")
    conn.commit()
    return redirect('view')
@app.route("/view",methods=['GET','POST'])
def view():
    conn=mysql.connect()
    cur=conn.cursor()
    cur.execute("select * from expense where username=%s" ,(session['username'],))
    data=cur.fetchall()
    return render_template('view.html',data=data)
@app.route("/income",methods=['GET','POST'])
def income():
    conn=mysql.connect()
    cur=conn.cursor()
    cur.execute("select * from income where username=%s" ,(session['username'],))
    data=cur.fetchall()
    return render_template('income.html',data=data)
@app.route("/calculate",methods=['GET','POST'])
def calculate():
```

```
    from flask import session
    conn=mysql.connect()
    cur=conn.cursor()
    cur.execute("select `income` from income where username=%s" ,(session['username'],))
    data=cur.fetchall()
    cur.execute("select sum(amount) from expense where username=%s" ,(session['username'],))
    abc=cur.fetchall()
    threshold=float(data[0][0])*2/100
    if float(abc[0][0])>=threshold:
        message = Mail(from_email='salonimp1999@gmail.com',
                       to_emails=session['emailid'],
                       subject='ALERT MESSAGE TO CUSTOMERS',
                       plain_text_content='you have crossed your daily limit',
                       html_content='<strong>you have crossed your daily limit</strong>')
        try:

sg=SendGridAPIClient('SG.JiV83drFQEkxecSiQK_zw.g_CEvCnVzQwEmyrAG5tNCrbPHjAIi
qZOhxUO7oFLIaY')
            response = sg.send(message)
            print(response.status_code)
            print(response.body)
            print(response.headers)
        except Exception as e:
            print(e)
    result=str(float(data[0][0])-float(abc[0][0]))
         cur.execute("UPDATE `income` SET `balance`='"+str(result)+"' where username=%s"
,(session['username'],))
    conn.commit()
    return redirect('income')
@app.route("/saving",methods=['GET','POST'])
```

```
def saving():
   if request.method=="POST":
        conn=mysql.connect()
        cur=conn.cursor()
        saving=request.form['save']
        cur.execute("select `balance` from income where username=%s" ,(session['username'],))
        data=cur.fetchall()
        result=str(float(data[0][0])+float(saving))
                cur.execute("UPDATE `income` SET `balance`='"+result+"' where username=%s" ,(session['username'],))
        conn.commit()
        return redirect('/income')
    else:
        return render_template('saving.html')
if __name__=="__main__":
    app.run(host='0.0.0.0',debug = True,port=8080)
```