**TABLE OF CONTENTS**

Acknowledgement I

Abstract II

Declaration III

**Chapter No. Chapter Name Page No**

1 Introduction 01

2 Literature Survey 02

3 System Requirement and Specification 03

4 Proposed System 04

5 Implementation 05-13

6 System Testing 14

7 Result And Analysis 15-18

8 Conclusion 19

References 20

**CHAPTER 1**

**INTRODUCTION**

A daily Expense Tracker is a particular form of a digital diary that aids in keeping track of all of our cash transitions and moreover offers daily, weekly, monthly, and yearly reports on all financial activities. The user receives notifications to enter expenses and incomes that aid the application’s tracking mechanism. All data is kept in offline mode for access at any time and in any location. This application allows the user to enter their income to determine their daily expenses and the results are saved for each user. That the stakeholders in the expense processes can use. Any firm will be able to manage all of its spending more effectively with the use of the expense tracker. Utilizing expenditure management software will aid in reducing wasteful spending.

This project is built on a system for tracking expenses and income. The goal of this project is to provide a simple, quick, and seamless tracking system between expenses and revenues. Additionally, this technology provides opportunities like a digital automated dairy that will enable the user to maintain all financial operations. Therefore, we created our project, which will greatly assist the users, in order to create a better spending monitoring system. The majority of individuals struggle with money problems because they are unable to keep track of their income and expenses. In this situation, a daily expense tracker can assist people in keeping track of their income and expenses on a daily basis and reducing stress. The use of a daily expense tracker aids the user in avoiding unforeseen costs and difficult financial circumstances. Time will be saved and a responsible lifestyle will be offered through this project. This system was developed, is being managed by professionals, and is working well for the user. SQLite database, XML, and Java.

**CHAPTER 2**

**LITERATURE SURVEY**

**Paper 1:** Expenditure Management System

**Authors:** Dr.V.Geetha, G.Nikitha, H.Sri Lasya, Dr.C.K.Gomathy

**Year:** 2022

**Abstract:** In the modern world, consumers have a wide range of options and useful alternatives to pick from. As a result, compared to a decade ago, people’s cost have surged and the expense of living keeps increasing daily. Therefore, keeping a tight eye on expenses will be required in order to live a good lifestyle within the appropriate price range. With the help of this software, users may monitor their prices in the form of bar graphs, pie charts, and summaries on a daily, weekly, monthly, or even yearly basis. The provided tool is a fully-featured expense tracker that is both web-based and mobile, designed to not only assist users in keeping track of their costs, but also to help them save money, but also reduce unrequited fees, which will assist to promote a responsible way of living.

**Paper 2:** Expenses Tracker

**Authors:** Mr. Shekhar Singh, Aman Garg, Mukul Goel, Sagar Mittal

**Year:** 2021

**Abstract:** . The present price tracker and the current tracker applications were compared in an evaluation. The purpose of the mobile application is to assist users in keeping track of their spending and determining whether they are staying within their set spending limits. Potential users had to input the necessary data, including the pricing amount, service provider, category, and date, when creating the expense. Subclasses and other rate remarks are optional records that can also be entered. A monitoring programme that includes the one being added is offered. The expense tracker penny.

**Paper 3:** Expense tracker application

**Authors:** Velmurugan.R, Mrs.P.Usha

**Year:** 2021

**Abstract:** Expense tracker is an android based application. This application allows the user to maintain a computerized dairy. Expense tracker application which will keep a track of expenses of a user on a day-to-day basis. This application keeps a record of your expenses and also will give you a category wise distribution of your expenses. With the help of this application user can track their daily/weekly/monthly expenses. This application will also have a feature which will help you stay on budget because you know your expenses. Expense tracker application will generate report at the end of month to show Expense via a graphical representation. We also have added a special feature which will distributes your expenses in different categories suitable for the user. An expense history will also be provided in application.

**Paper 4:** Daily Expense Tracker

**Authors:** Shivam Mehra, Prabhat Parashar

**Year:** 2021

**Abstract:** This paper represents a Daily Expense Tracker is a tool that resides on a remote server and is accessible via browser. As the name suggest Daily Expense Tracker is a web application that allows you to track the daily expense of the user and help them to better manage their resources. It creates a digital record of the income and expense of the user. It input from the user a income, source of this income and the date of earning that income and creates a transaction entry under income category sums to the total amount of income and making real time changes. The various sources of income can be added and thus the distribution of yours income is also illustrated by real time functioning charts that will keep updating as per your transactions. Similarly, it will also have an expense category where you can make similar transaction about the source of your expense, amount and date. On creating such transaction a different chart for distribution of expense will also be made in real time. The web application will also be voice powered and all the functionalities can be used with voice commands. The application will be accessible and computable to all the devices.

**Paper 5:** A Novel Expense Tracker using Statistical Analysis

**Authors:** Muskan Sharma, Ayush Bansal, Dr. Raju Ranjan, Shivam Sethi

**Year:** 2021

**Abstract:** expense Tracker is used to maintain and manage data of daily expenditure in a more precise way it can give profound knowledge of their expenses. User can choose the kind of spending they wanted to do, even the amount etc, and all these details is going to be saved by the internal database storage. In this system user can actually have a knowledge about their expenditure on their daily basis, weekly as well as monthly basis. This systematic way of storing your information related to your expenses would help you to keep a track of your expenditure and further you do not have to do the manual stuff. Some statistical analysis has to be done to be able to give users correct information on their expenses and help them spend better. This helps the society to prevent the issues like bankruptcy and save time from manual calculations. User can provide his/her income to calculate the total expense per day and the results will be stored for each individual user. People when usually go for trips with friends, can use this tracker to maintain their expense.

**CHAPTER 3**

**SYSTEM REQUIREMENTS SPECIFICATION**

**HARDWARE REQUIREMENTS:**

* Processor : CORE i3
* Ram : 512 Mb
* Input device : Standard Keyboard and mouse

**SOFTWARE REQUIREMENTS:**

* **Operating system** : Windows 10
* **Language Used** : Python
* **Web Browser :** Mozilla, Google Chrome, IE8,OPERA
* **Environment :** Anaconda Framework
* **IDE :** Jupyter Notebook
* **Anaconda:** In order to make package management and deployment easier, Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning, large-scale data processing, predictive analysis, etc.). The distribution comes with a number of Windows, Linux and Mac OS packages. Using Jupyter as a notebook an open-source web tool called the Jupyter Notebook enables you to create and share documents with real-time code, equations, visuals, and text. Examples of applications include data processing and cleaning, numerical simulation, statistical modelling, data visualisation and many more.

**CHAPTER 4**

**PROPOSED SYSTEM METHODOLOGY**

**4.1 Existing System**

There are a rising number of e-news websites, therefore many approaches for extracting and categorising the e-news article text contents of e-news websites have been presented. This part highlights the significance and originality of this research by comparing it to previous works and implementing solutions. And thoroughly examines the advantages and disadvantages of the present strategies, identifies the gaps in the strategies now in use, and designs this system to address the problems with the current strategies. The extraction procedure is divide into two steps. To gather e-news pages, e-news websites must first be crawled. Then, information about e-news article contents is taken from e-news web pages. The e-news websites are made up of a variety of web pages, including shopping pages, blog pages, and advertisement pages, etc. Therefore, it is difficult to isolate solely e-news items. E-news sites typically don’t have a set layout, instead, they alter it dynamically throughout time. The majority of the existing work uses machine learning algorithms like SVM, Random Forest, KNN, and other similar ones. There are numerous alternative methods that have been employed in the existing work. The proposed method establishes a baseline using the previous work. Previous text mining research has been conducted in a variety of methods, with each method including different levels of analysis. As a result, text mining is crucial for solving the issue. We consider this and address it in our suggested system.

**4.2 Limitations of Existing System**

The current system of BBC News Articles is categorized and forecasted for multiple classifications, but the category-wise accuracy prediction for different machine learning methods is not taken into consideration, which is also important for text mining. The classification of a relatively small number of classifiers is evaluated. The current system just classifies the predictions and its classification is validated using machine learning algorithms, it lacks prediction using various classification algorithms and category-wise accuracy prediction for various machine machine learning methods

**4.3 Proposed System**

Investigated is the suggested news text classification approach. Text pre-processing, feature selection using TF-IDF, and text classification using RandomForest, Logistic Regression, K Nearest Neighbours, and Decision Tree classifiers are the three processes in this method

**CHAPTER 5**

**IMPLEMENTATION**

1. Models.py

from django.db import models

from django.utils.timezone import now

from django.contrib.auth.models import User

from django.conf import settings

from django.db.models.signals import post\_save

from django.dispatch import receiver

from django.db.models import Sum

#Create your models here.

SELECT\_CATEGORY\_CHOICES = [

("Food","Food"),

("Travel","Travel"),

("Shopping","Shopping"),

("Necessities","Necessities"),

("Entertainment","Entertainment"),

("Other","Other")

]

ADD\_EXPENSE\_CHOICES = [

("Expense","Expense"),

("Income","Income")

]

PROFESSION\_CHOICES =[

("Employee","Employee"),

("Business","Business"),

("Student","Student"),

("Other","Other")

]

**class** Addmoney\_info(models.Model):

user = models.ForeignKey(User,default = 1, on\_delete=models.CASCADE)

add\_money = models.CharField(max\_length = 10 , choices = ADD\_EXPENSE\_CHOICES )

quantity = models.BigIntegerField()

Date = models.DateField(default = now)

Category = models.CharField( max\_length = 20, choices = SELECT\_CATEGORY\_CHOICES , default ='Food')

**class** Meta:

db\_table:'addmoney'

**class** UserProfile(models.Model):

user = models.OneToOneField(User,on\_delete=models.CASCADE)

profession = models.CharField(max\_length = 10, choices=PROFESSION\_CHOICES)

Savings = models.IntegerField( **null**=**True**, blank=**True**)

income = models.BigIntegerField(**null**=**True**, blank=**True**)

image = models.ImageField(upload\_to='profile\_image',blank=**True**)

**def** \_\_str\_\_(self):

**return** self.user.username

AD

2. Admin.py

It will help register the tables in the database.

# Register your models here.

from .models import Addmoney\_info

From django.contrib import admin

**class** Addmoney\_infoAdmin(admin.ModelAdmin):

list\_display=("user","quantity","Date","Category","add\_money")

admin.site.register(Addmoney\_info,Addmoney\_infoAdmin)

from django.contrib.sessions.models import Session

admin.site.register(Session)

from .models import UserProfile

admin.site.register(UserProfile)

3. Urls.py

from django.contrib import admin

from django.urls import path

from django.urls import include

from . import views

from django.contrib.auth import views as auth\_views

urlpatterns = [

path('', views.home, name='home'),

path('index/', views.index, name='index'),

path('register/',views.register,name='register'),

path('handleSignup/',views.handleSignup,name='handleSignup'),

path('handlelogin/',views.handlelogin,name='handlelogin'),

path('handleLogout/',views.handleLogout,name='handleLogout'),

path('reset\_password/',auth\_views.PasswordResetView.as\_view(template\_name = "home/reset\_password.html"),name='reset\_password'),

path('reset\_password\_sent/',auth\_views.PasswordResetDoneView.as\_view(template\_name="home/reset\_password\_sent.html"),name='password\_reset\_done'),

path('reset/<uidb64>/<token>/',auth\_views.PasswordResetConfirmView.as\_view(template\_name ="home/password\_reset\_form.html"),name='password\_reset\_confirm'),

path('reset\_password\_complete/',auth\_views.PasswordResetView.as\_view(template\_name ="home/password\_reset\_done.html"),name='password\_reset\_complete'),

path('addmoney/',views.addmoney,name='addmoney'),

path('addmoney\_submission/',views.addmoney\_submission,name='addmoney\_submission'),

path('charts/',views.charts,name='charts'),

path('tables/',views.tables,name='tables'),

path('expense\_edit/<int:id>',views.expense\_edit,name='expense\_edit'),

path('<int:id>/addmoney\_update/', views.addmoney\_update, name="addmoney\_update") ,

path('expense\_delete/<int:id>',views.expense\_delete,name='expense\_delete'),

path('profile/',views.profile,name = 'profile'),

path('expense\_month/',views.expense\_month, name = 'expense\_month'),

path('stats/',views.stats, name = 'stats'),

path('expense\_week/',views.expense\_week, name = 'expense\_week'),

path('weekly/',views.weekly, name = 'weekly'),

path('check/',views.check,name="check"),

path('search/',views.search,name="search"),

path('<int:id>/profile\_edit/',views.profile\_edit,name="profile\_edit"),

path('<int:id>/profile\_update/',views.profile\_update,name="profile\_update"),

path('info/',views.info,name="info"),

path('info\_year/',views.info\_year,name="info\_year"),

]

4. Views.py

**a. Importing modules**

from django.shortcuts import render,HttpResponse,redirect

from django.contrib import messages

from django.contrib.auth import authenticate ,logout

from django.contrib.auth import login as dj\_login

from django.contrib.auth.models import User

from .models import Addmoney\_info,UserProfile

from django.contrib.sessions.models import Session

from django.core.paginator import Paginator, EmptyPage , PageNotAnInteger

from django.db.models import Sum

from django.http import JsonResponse

import datetime

from django.utils import timezone

**b. Login and Index function**

AD

**def** home(request):

**if** request.session.has\_key('is\_logged'):

**return** redirect('/index')

**return** render(request,'home/login.html')

# return HttpResponse('This is home')

**def** index(request):

**if** request.session.has\_key('is\_logged'):

user\_id = request.session["user\_id"]

user = User.objects.get(id=user\_id)

addmoney\_info = Addmoney\_info.objects.filter(user=user).order\_by('-Date')

paginator = Paginator(addmoney\_info , 4)

page\_number = request.GET.get('page')

page\_obj = Paginator.get\_page(paginator,page\_number)

context = {

# 'add\_info' : addmoney\_info,

'page\_obj' : page\_obj

}

#if request.session.has\_key('is\_logged'):

**return** render(request,'home/index.html',context)

**return** redirect('home')

**c. Other Functions**

**def** addmoney(request):

**return** render(request,'home/addmoney.html')

**def** profile(request):

**if** request.session.has\_key('is\_logged'):

**return** render(request,'home/profile.html')

**return** redirect('/home')

**def** profile\_edit(request,id):

**if** request.session.has\_key('is\_logged'):

add = User.objects.get(id=id)

**return** render(request,'home/profile\_edit.html',{'add':add})

**return** redirect("/home")

AD

**d. Updating Profile**

**def** profile\_update(request,id):

**if** request.session.has\_key('is\_logged'):

**if** request.method == "POST":

user = User.objects.get(id=id)

user.first\_name = request.POST["fname"]

user.last\_name = request.POST["lname"]

user.email = request.POST["email"]

user.userprofile.Savings = request.POST["Savings"]

user.userprofile.income = request.POST["income"]

user.userprofile.profession = request.POST["profession"]

user.userprofile.save()

user.save()

**return** redirect("/profile")

**return** redirect("/home")

**e. Signup, Login, and Logout backend:**

**def** handleSignup(request):

**if** request.method =='POST':

# get the post parameters

uname = request.POST["uname"]

fname=request.POST["fname"]

lname=request.POST["lname"]

email = request.POST["email"]

profession = request.POST['profession']

Savings = request.POST['Savings']

income = request.POST['income']

pass1 = request.POST["pass1"]

pass2 = request.POST["pass2"]

profile = UserProfile(Savings = Savings,profession=profession,income=income)

# check for errors in input

**if** request.method == 'POST':

**try**:

user\_exists = User.objects.get(username=request.POST['uname'])

messages.error(request," Username already taken, Try something else!!!")

**return** redirect("/register")

**except** User.DoesNotExist:

**if** len(uname)>15:

messages.error(request," Username must be max 15 characters, Please try again")

**return** redirect("/register")

**if** not uname.isalnum():

messages.error(request," Username should only contain letters and numbers, Please try again")

**return** redirect("/register")

**if** pass1 != pass2:

messages.error(request," Password do not match, Please try again")

**return** redirect("/register")

# create the user

user = User.objects.create\_user(uname, email, pass1)

user.first\_name=fname

user.last\_name=lname

user.email = email

# profile = UserProfile.objects.all()

user.save()

# p1=profile.save(commit=False)

profile.user = user

profile.save()

messages.success(request," Your account has been successfully created")

**return** redirect("/")

**else**:

**return** HttpResponse('404 - NOT FOUND ')

**return** redirect('/login')

**def** handlelogin(request):

**if** request.method =='POST':

# get the post parameters

loginuname = request.POST["loginuname"]

loginpassword1=request.POST["loginpassword1"]

user = authenticate(username=loginuname, password=loginpassword1)

**if** user is not None:

dj\_login(request, user)

request.session['is\_logged'] = **True**

user = request.user.id

request.session["user\_id"] = user

messages.success(request, " Successfully logged in")

**return** redirect('/index')

**else**:

messages.error(request," Invalid Credentials, Please try again")

**return** redirect("/")

**return** HttpResponse('404-not found')

**def** handleLogout(request):

del request.session['is\_logged']

del request.session["user\_id"]

logout(request)

messages.success(request, " Successfully logged out")

**return** redirect('home')

**f. Add Money Form and Add Money Update Backend:**

**def** addmoney\_submission(request):

**if** request.session.has\_key('is\_logged'):

**if** request.method == "POST":

user\_id = request.session["user\_id"]

user1 = User.objects.get(id=user\_id)

addmoney\_info1 = Addmoney\_info.objects.filter(user=user1).order\_by('-Date')

add\_money = request.POST["add\_money"]

quantity = request.POST["quantity"]

Date = request.POST["Date"]

Category = request.POST["Category"]

add = Addmoney\_info(user = user1,add\_money=add\_money,quantity=quantity,Date = Date,Category= Category)

add.save()

paginator = Paginator(addmoney\_info1, 4)

page\_number = request.GET.get('page')

page\_obj = Paginator.get\_page(paginator,page\_number)

context = {

'page\_obj' : page\_obj

}

**return** render(request,'home/index.html',context)

**return** redirect('/index')

**def** addmoney\_update(request,id):

**if** request.session.has\_key('is\_logged'):

**if** request.method == "POST":

add = Addmoney\_info.objects.get(id=id)

add .add\_money = request.POST["add\_money"]

add.quantity = request.POST["quantity"]

add.Date = request.POST["Date"]

add.Category = request.POST["Category"]

add .save()

**return** redirect("/index")

**return** redirect("/home")

**g. Expense Edit and Expense Delete Backend:**

**def** expense\_edit(request,id):

**if** request.session.has\_key('is\_logged'):

addmoney\_info = Addmoney\_info.objects.get(id=id)

user\_id = request.session["user\_id"]

user1 = User.objects.get(id=user\_id)

**return** render(request,'home/expense\_edit.html',{'addmoney\_info':addmoney\_info})

**return** redirect("/home")

**def** expense\_delete(request,id):

**if** request.session.has\_key('is\_logged'):

addmoney\_info = Addmoney\_info.objects.get(id=id)

addmoney\_info.delete()

**return** redirect("/index")

**return** redirect("/home")

**h. Monthly, weekly , yearly expense Backend**

**def** expense\_month(request):

todays\_date = datetime.date.today()

one\_month\_ago = todays\_date-datetime.timedelta(days=30)

user\_id = request.session["user\_id"]

user1 = User.objects.get(id=user\_id)

addmoney = Addmoney\_info.objects.filter(user = user1,Date\_\_gte=one\_month\_ago,Date\_\_lte=todays\_date)

finalrep ={}

**def** get\_Category(addmoney\_info):

# if addmoney\_info.add\_money=="Expense":

**return** addmoney\_info.Category

Category\_list = list(set(map(get\_Category,addmoney)))

**def** get\_expense\_category\_amount(Category,add\_money):

quantity = 0

filtered\_by\_category = addmoney.filter(Category = Category,add\_money="Expense")

**for** item **in** filtered\_by\_category:

quantity+=item.quantity

**return** quantity

**for** x **in** addmoney:

**for** y **in** Category\_list:

finalrep[y]= get\_expense\_category\_amount(y,"Expense")

**return** JsonResponse({'expense\_category\_data': finalrep}, safe=**False**)

**def** stats(request):

**if** request.session.has\_key('is\_logged') :

todays\_date = datetime.date.today()

one\_month\_ago = todays\_date-datetime.timedelta(days=30)

user\_id = request.session["user\_id"]

user1 = User.objects.get(id=user\_id)

addmoney\_info = Addmoney\_info.objects.filter(user = user1,Date\_\_gte=one\_month\_ago,Date\_\_lte=todays\_date)

sum = 0

**for** i **in** addmoney\_info:

**if** i.add\_money == 'Expense':

sum=sum+i.quantity

addmoney\_info.sum = sum

sum1 = 0

**for** i **in** addmoney\_info:

**if** i.add\_money == 'Income':

sum1 =sum1+i.quantity

addmoney\_info.sum1 = sum1

x= user1.userprofile.Savings+addmoney\_info.sum1 - addmoney\_info.sum

y= user1.userprofile.Savings+addmoney\_info.sum1 - addmoney\_info.sum

**if** x<0:

messages.warning(request,'Your expenses exceeded your savings')

x = 0

**if** x>0:

y = 0

addmoney\_info.x = abs(x)

addmoney\_info.y = abs(y)

**return** render(request,'home/stats.html',{'addmoney':addmoney\_info})

**def** expense\_week(request):

todays\_date = datetime.date.today()

one\_week\_ago = todays\_date-datetime.timedelta(days=7)

user\_id = request.session["user\_id"]

user1 = User.objects.get(id=user\_id)

addmoney = Addmoney\_info.objects.filter(user = user1,Date\_\_gte=one\_week\_ago,Date\_\_lte=todays\_date)

finalrep ={}

**def** get\_Category(addmoney\_info):

**return** addmoney\_info.Category

Category\_list = list(set(map(get\_Category,addmoney)))

**def** get\_expense\_category\_amount(Category,add\_money):

quantity = 0

filtered\_by\_category = addmoney.filter(Category = Category,add\_money="Expense")

**for** item **in** filtered\_by\_category:

quantity+=item.quantity

**return** quantity

**for** x **in** addmoney:

**for** y **in** Category\_list:

finalrep[y]= get\_expense\_category\_amount(y,"Expense")

**return** JsonResponse({'expense\_category\_data': finalrep}, safe=**False**)

**def** weekly(request):

**if** request.session.has\_key('is\_logged') :

todays\_date = datetime.date.today()

one\_week\_ago = todays\_date-datetime.timedelta(days=7)

user\_id = request.session["user\_id"]

user1 = User.objects.get(id=user\_id)

addmoney\_info = Addmoney\_info.objects.filter(user = user1,Date\_\_gte=one\_week\_ago,Date\_\_lte=todays\_date)

sum = 0

**for** i **in** addmoney\_info:

**if** i.add\_money == 'Expense':

sum=sum+i.quantity

addmoney\_info.sum = sum

sum1 = 0

**for** i **in** addmoney\_info:

**if** i.add\_money == 'Income':

sum1 =sum1+i.quantity

addmoney\_info.sum1 = sum1

x= user1.userprofile.Savings+addmoney\_info.sum1 - addmoney\_info.sum

y= user1.userprofile.Savings+addmoney\_info.sum1 - addmoney\_info.sum

**if** x<0:

messages.warning(request,'Your expenses exceeded your savings')

x = 0

**if** x>0:

y = 0

addmoney\_info.x = abs(x)

addmoney\_info.y = abs(y)

**return** render(request,'home/weekly.html',{'addmoney\_info':addmoney\_info})

**def** check(request):

**if** request.method == 'POST':

user\_exists = User.objects.filter(email=request.POST['email'])

messages.error(request,"Email not registered, TRY AGAIN!!!")

**return** redirect("/reset\_password")

**def** info\_year(request):

todays\_date = datetime.date.today()

one\_week\_ago = todays\_date-datetime.timedelta(days=30\*12)

user\_id = request.session["user\_id"]

user1 = User.objects.get(id=user\_id)

addmoney = Addmoney\_info.objects.filter(user = user1,Date\_\_gte=one\_week\_ago,Date\_\_lte=todays\_date)

finalrep ={}

**def** get\_Category(addmoney\_info):

**return** addmoney\_info.Category

Category\_list = list(set(map(get\_Category,addmoney)))

**def** get\_expense\_category\_amount(Category,add\_money):

quantity = 0

filtered\_by\_category = addmoney.filter(Category = Category,add\_money="Expense")

**for** item **in** filtered\_by\_category:

quantity+=item.quantity

**return** quantity

**for** x **in** addmoney:

**for** y **in** Category\_list:

finalrep[y]= get\_expense\_category\_amount(y,"Expense")

**return** JsonResponse({'expense\_category\_data': finalrep}, safe=**False**)

**def** info(request):

**return** render(request,'home/info.html')

**MODELING AND DESCRIPTION**

**Diagram

Description automatically generated**

**CHAPTER 6**

**SOFTWARE TESTING**

**UNIT TESTING**

The goal is to confirm that each piece of software operates as intended. The smallest software component that can be tested is a unit. Individual software modules or components are tested as part of the level of software testing known as unit testing. The unit test will typically create some kind of artificial environment before invoking the unit’s methods. The returned findings are then compared to a predetermined value. The system can be tested as a hole once the modules are put together using the same procedures. It typically has one or more inputs and one output.

**FUNCTIONAL TESTING**

Functional testing is a type of testing that is used to check all of the aspects of a system or piece of software, including failure routes and boundary cases.

**INTEGRATION TESTING**

After unit testing is complete, the units or modules must be integrated, which prompts integration testing done to ensure that the integrated modules are reliable, performant, and functional. While integrated, they rarely make mistakes.

**SYSTEM TESTING**

Software and hardware systems are tested as a whole, integrated system to see whether they adhere to the requirements that have been set forth. To see if the required results are produced, the hardware and software units are tested separately and then combined.

**PERFORMANCE TESTING**

Performance testing is a non-functional testing method used to assess the system’s stability and responsiveness under varying workloads. Performance testing evaluates the system’s qualities including scalability, dependability, and resource utilisation when an application is being run on the system.

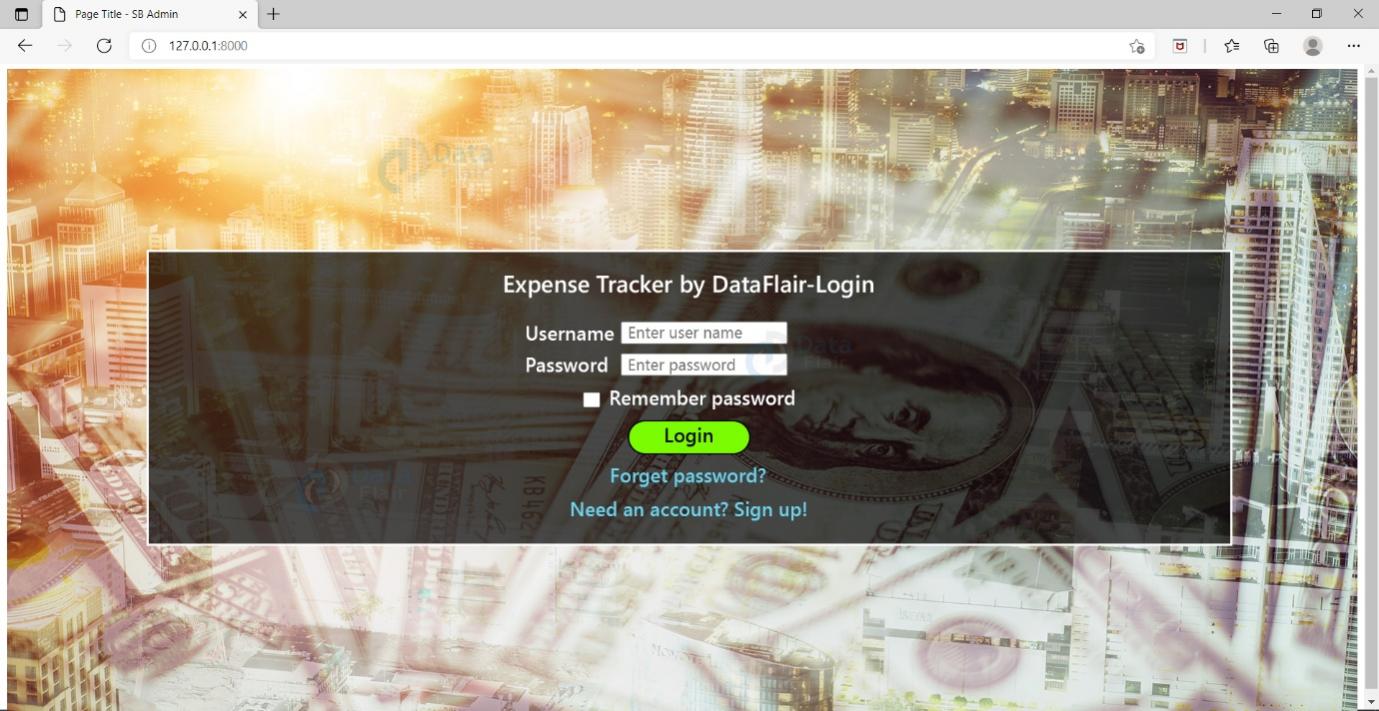
**CHAPTER 7**

**RESULT AND ANALYSIS**

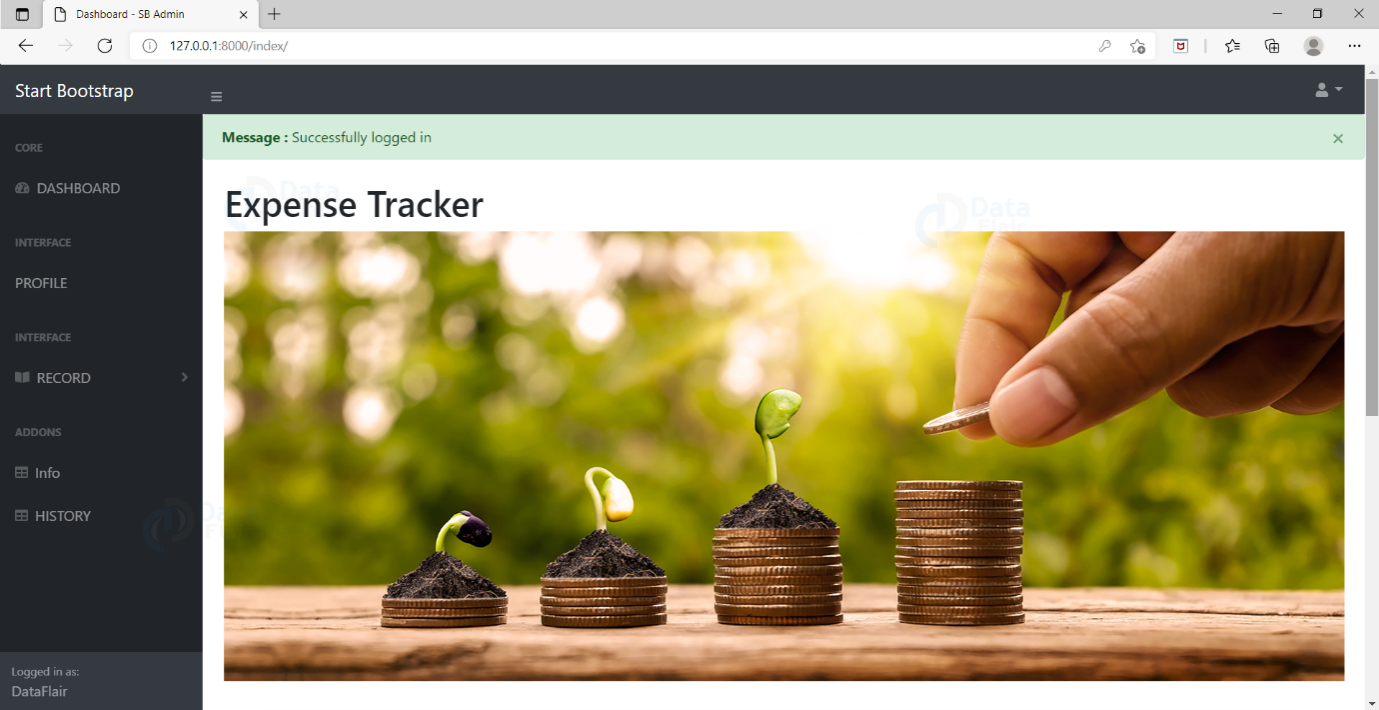
**DESIGN AND TEST CASE SCENARIOS**

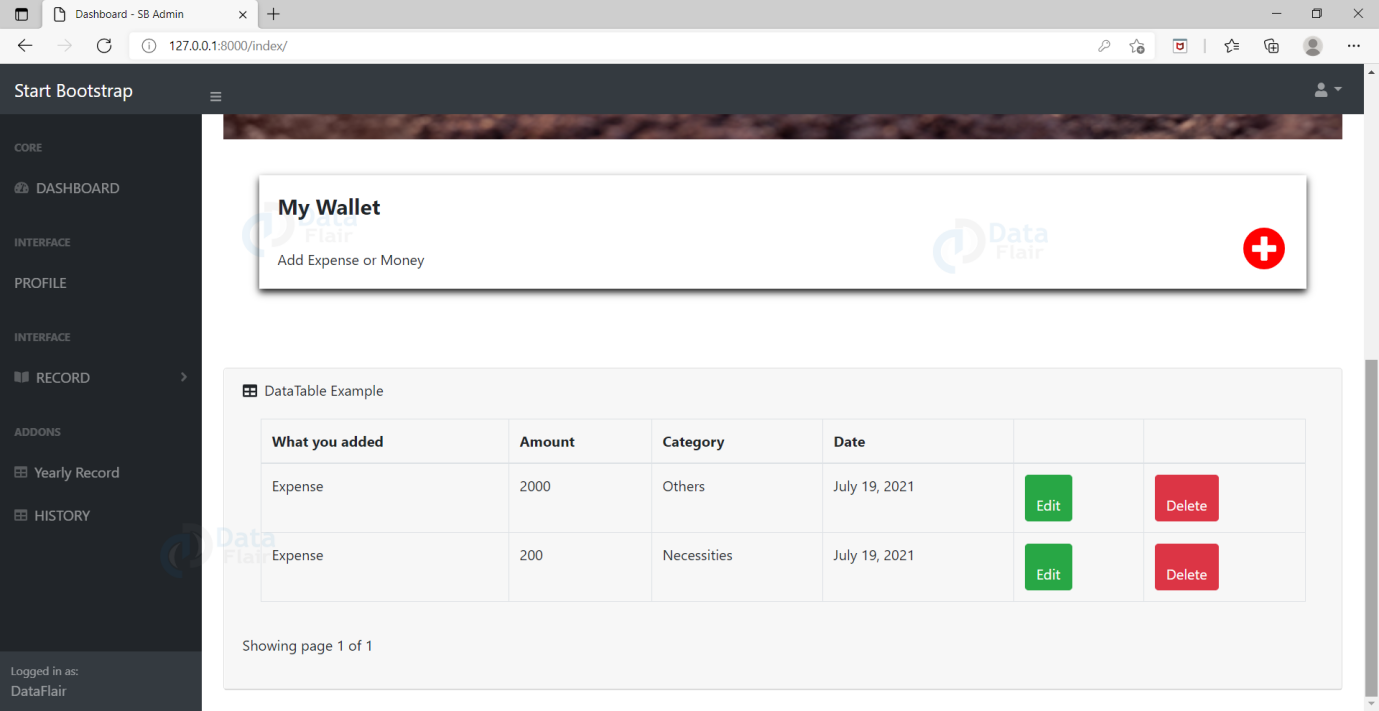
|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Input** | **Description** | **Test Case Result** |
| 1. | Admin Login:  Username: admin  Password: admin | System аllоws ассess | Pass |
| 2. | Registration:  Username : Rachana  Password : rach123  Mobile number : Empty  Email: Rach46@gmail.com | Mobile Number  Соlumn should not be  empty | Pass |
| 3. | User Login  Username: Kalyan  Password : kalian | System аllоws ассess | Pass |
| 4. | Username : Not Registered Username  Password :73043256 | Invalid username or password | Pass |
| 5. | User Login  Username: Rachana  Password : rach123 | Invalid password | Pass |

**Login Form:**

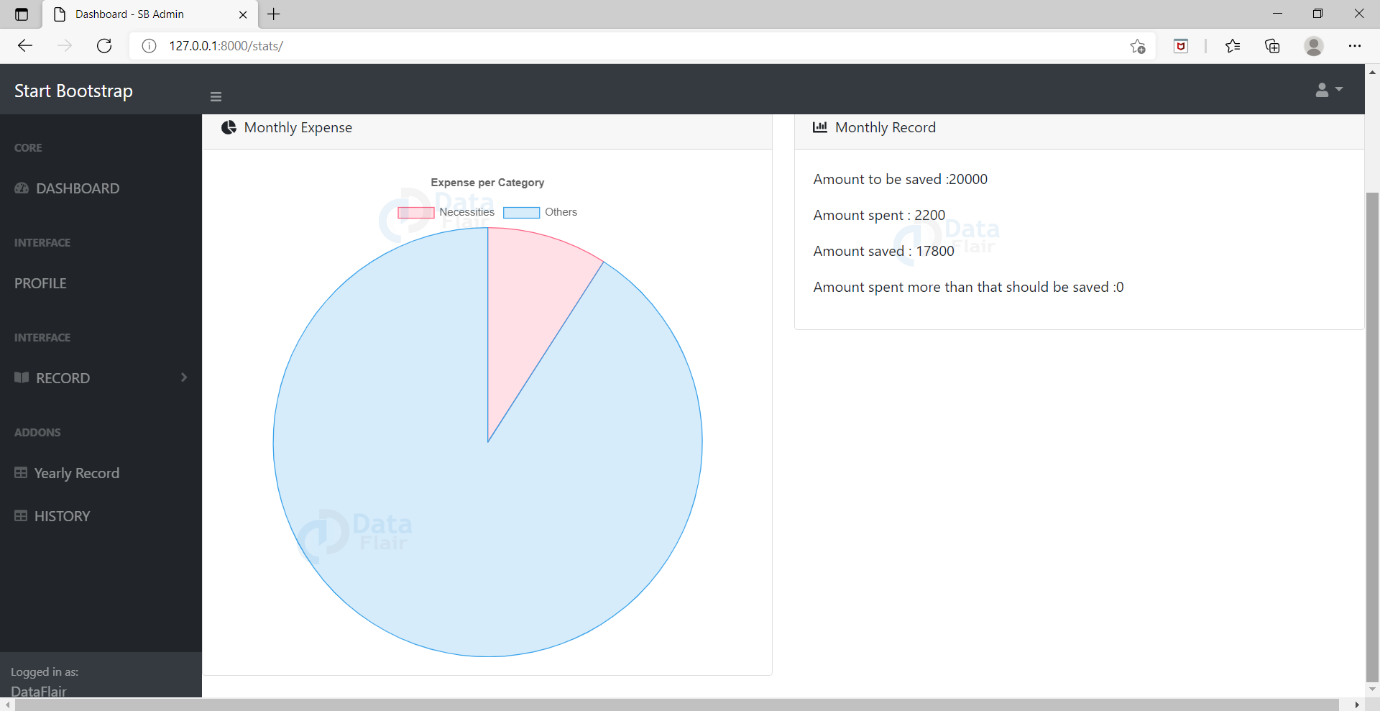
[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2021/07/login-page.png)

**Dashboard:**

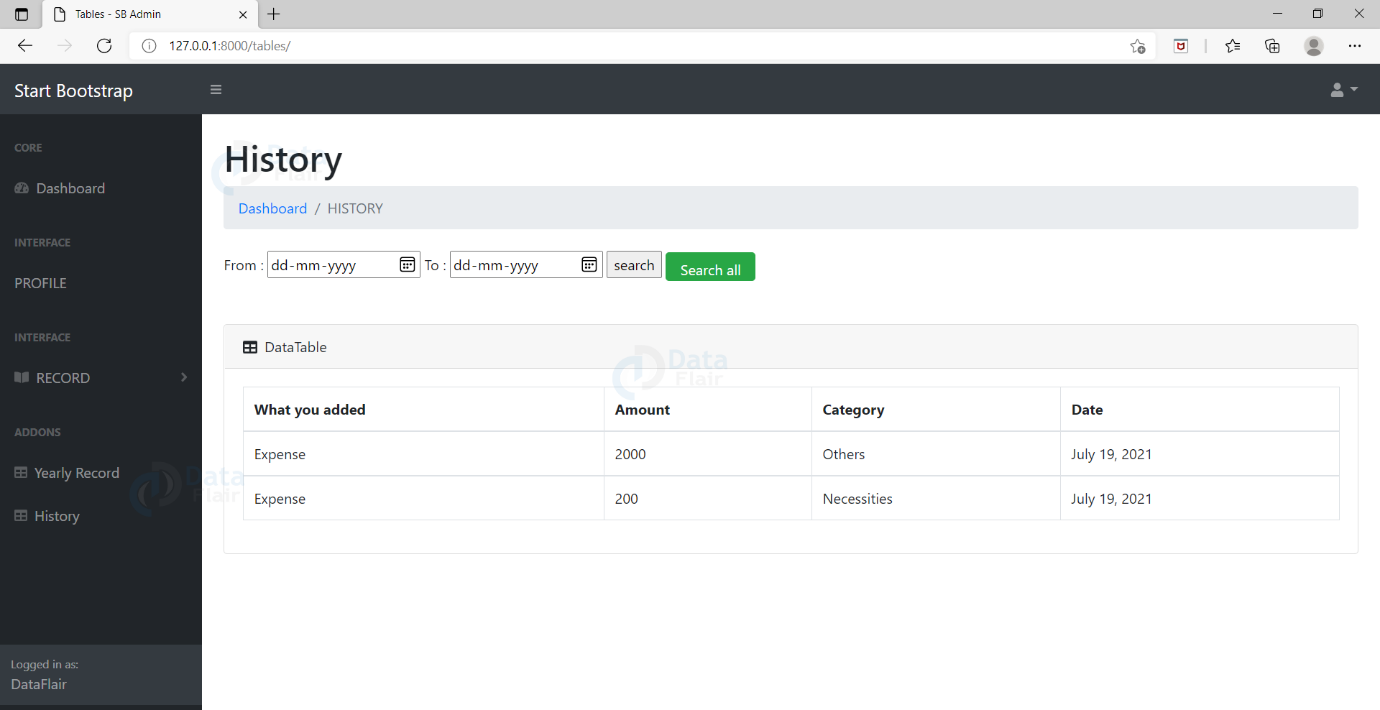
[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2021/07/expense-tracker-dashboard.png)

[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2021/07/dashboard.png)

**Monthly Expense Page:**

[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2021/07/monthly-expense.png)

**History Page:**

[](https://data-flair.training/blogs/wp-content/uploads/sites/2/2021/07/history.png)

**CONCLUSION AND FUTURE SCOPE**

**8.1 Conclusion**

A mobile application called Daily Expression Tracker runs on Android. We developed this program to enable users to precisely estimate their daily expenses. With the help of this application, the user can see how much money he makes and how much he spends. A daily notice will be delivered to the user’s phone detailing how much money he may make and spend, and the user can see his report at any time. This makes the application crucial for the user.

**8.2 Future Scope**

**Scope of Further Developments:** Almost all features are currently covered in our application, but we want to add more features in the future. Here are the features. Support for multiple accounts, currency converter is included, Reports are generated using any category-based format.

**REFERENCES**

[1] “Expense Manager”https://play.google.com/store/search?q=expense%20manager&c=apps// [last accessed: August 27, 2019 at 07.00 pm].

[2] “Expense Tracker” https://play.google.com/store/search?q=my%20daily%20expense%20diary&c=apps // [last accessed: August 27, 2019 at 07.00 pm].

[3] “Monefy Money Manager” https://play.google.com/store/apps/details?id=com.monefy.app.lite // [last accessed: August 27, 2019 at 07.00 pm].

[4] “Android” https://developer.android.com// [last accessed: September 02, 2019 at 07.00 pm].

[5] “Java” https:// https://www.java.com/en// [last accessed: Jun 27, 2019 at 07.00 pm]. [6] “Java” https://sqlitebrowser.org/ [last accessed: Jun 27, 2019 at 07.00 pm]