EXPENSE TRACKER

-Riya Shet,11B

National Public School

23-24

ACKNOWLEDGEMENT

I would like to thank my teachers for this opportunity to learn about Python, and for helping me throughout my project. I would also like to thank my parents for providing the necessary resources.

-Riya Shet,11B

INDEX

PYTHON

Python, a versatile and widely-used programming language, traces its origins back to 1991 when Guido van Rossum conceived it as a language prioritizing readability and simplicity. The name "Python" was inspired by the British comedy group Monty Python, showcasing the language's quirky and humorous nature. Python's development has been guided by the Python Enhancement Proposal (PEP) process, fostering community-driven decision-making. The transition from Python 2 to Python 3, completed in 2008, marked a significant evolution, streamlining the language and introducing modern features. Python boasts a clean and easy-to-understand syntax, facilitating rapid development and reducing the likelihood of errors. Its readability is further enhanced by the use of whitespace indentation instead of braces. Python's design philosophy, often summarized as "The Zen of Python," emphasizes clarity, simplicity, and a focus on explicitness.

Beyond its syntax, Python's strengths lie in its extensive standard library and a vast ecosystem of third-party packages and frameworks. This ecosystem includes Django for web development, NumPy and Pandas for data manipulation, TensorFlow and PyTorch for machine learning, and Flask for building web applications, among many others. Python's adaptability is underscored by its use in various domains, ranging from scripting and automation to complex scientific and research applications.

In summary, this project harnesses Python's rich history, emphasizing its commitment to simplicity and readability, and leverages its diverse features and extensive community support to deliver a versatile and effective solution.

SYNOPSIS

The Expense Tracker Website, developed with Flask and Python, stands as an indispensable tool for users seeking efficient and user-friendly financial management. Boasting a rich interface, the website provides an intuitive platform for users to seamlessly add, categorize, and search for expenses. With an emphasis on user experience, the interface ensures easy navigation, while the expense logging feature enables real-time tracking of financial transactions. The website's advanced search functionality enhances accessibility, allowing users to retrieve specific expense records effortlessly. Furthermore, the inclusion of customizable categories and detailed reporting empowers users with insights into their spending patterns. In essence, the Expense Tracker Website redefines expense management, offering a comprehensive and streamlined approach to financial tracking.

SYSTEM REQUIREMENTS

Hardware-

A Computer

Software-

- Browser
- Terminal
- Code Editor

Instructions-

Run the flask application on localhost with the help of terminal/powershell.You must have the necessary modules installed (mentioned ahead) to successfully launch the website.Set up a flask development server with the help of the following commands-

- First navigate to the project directory
- export(mac and linux)/set(windows) FLASK_APP=name of main python file
- flask run

MODULES AND FUNCTIONS

1.flask-
- Flask
- render_template
- url_for
- flash
- redirect
- request

2.re
3.datetime
4.flask_sqlalchemy-
- SQLAlchemy

5.flask_wtf-
- FlaskForm

6.wtforms-

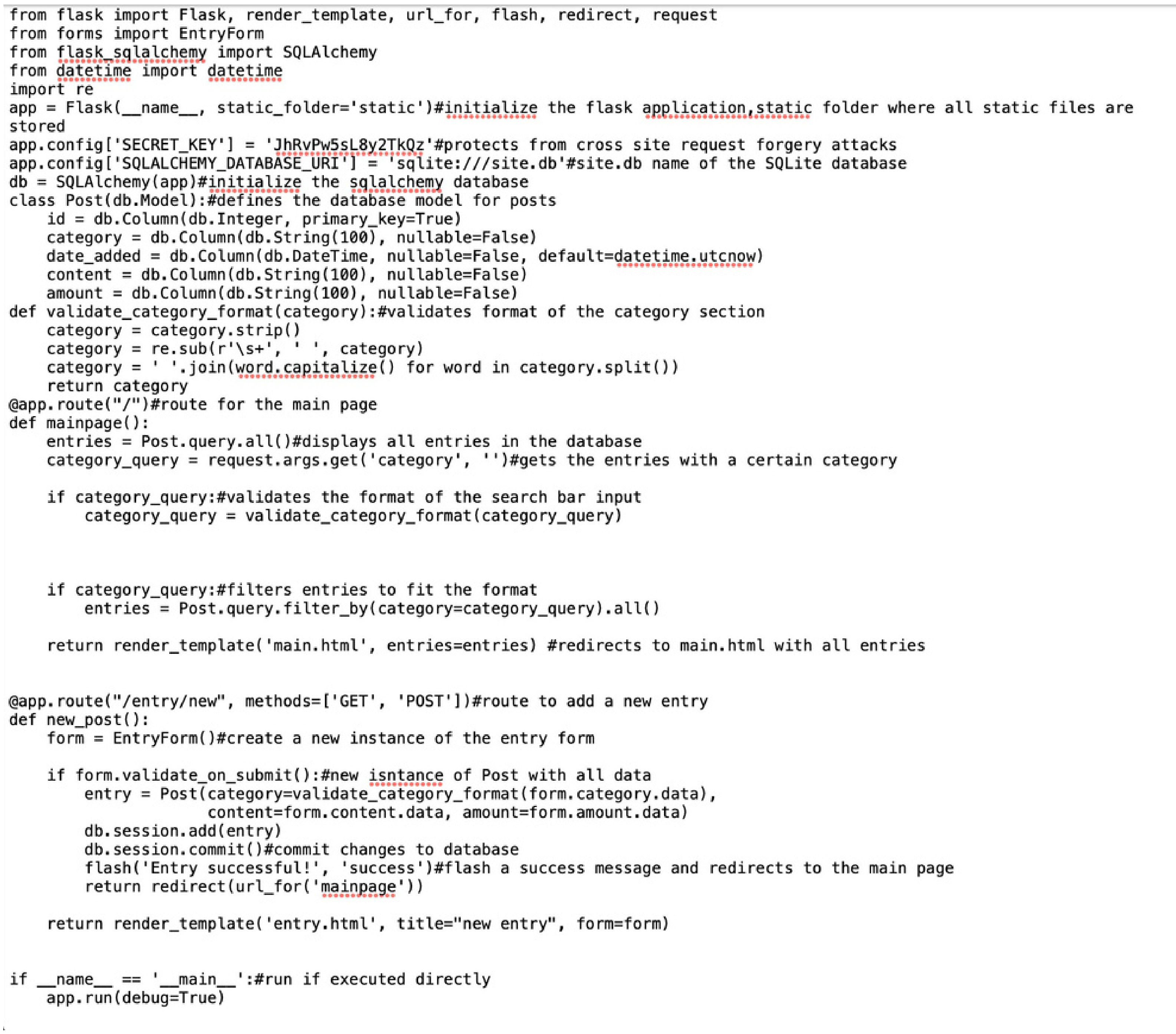
- StringField
- SubmitField
- TextAreaField
- SelectField

7.wtforms.validators-

- DataRequired

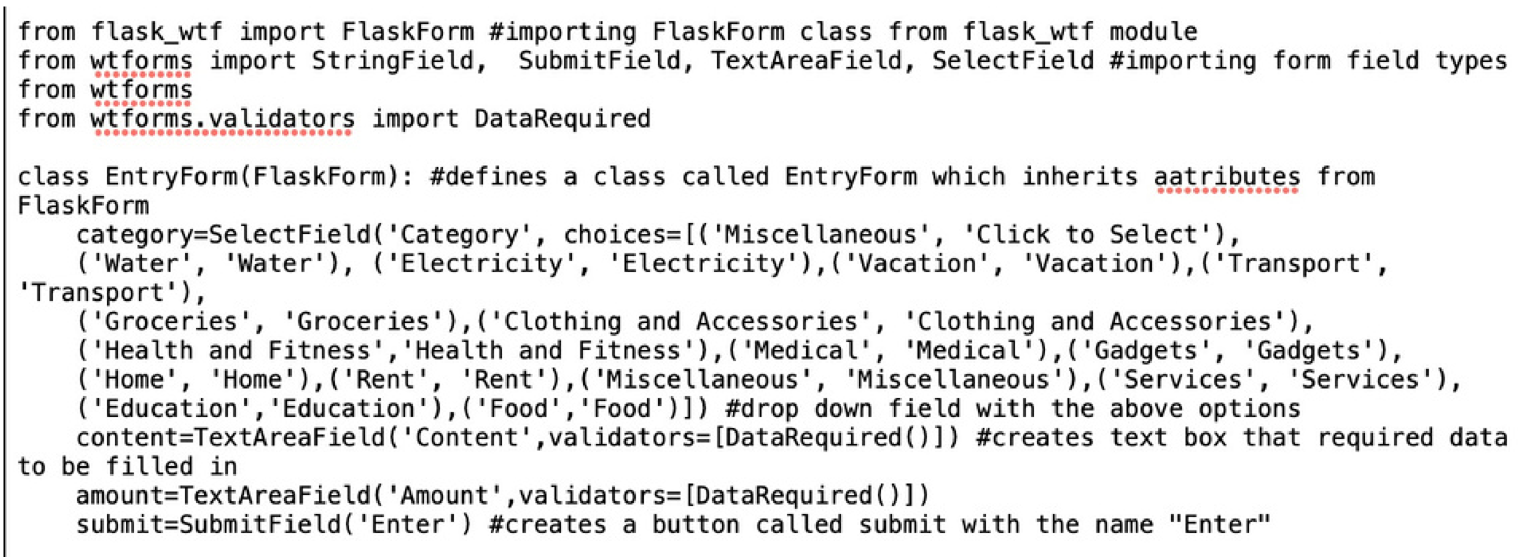
PROGRAM CODE

flaskblog.py



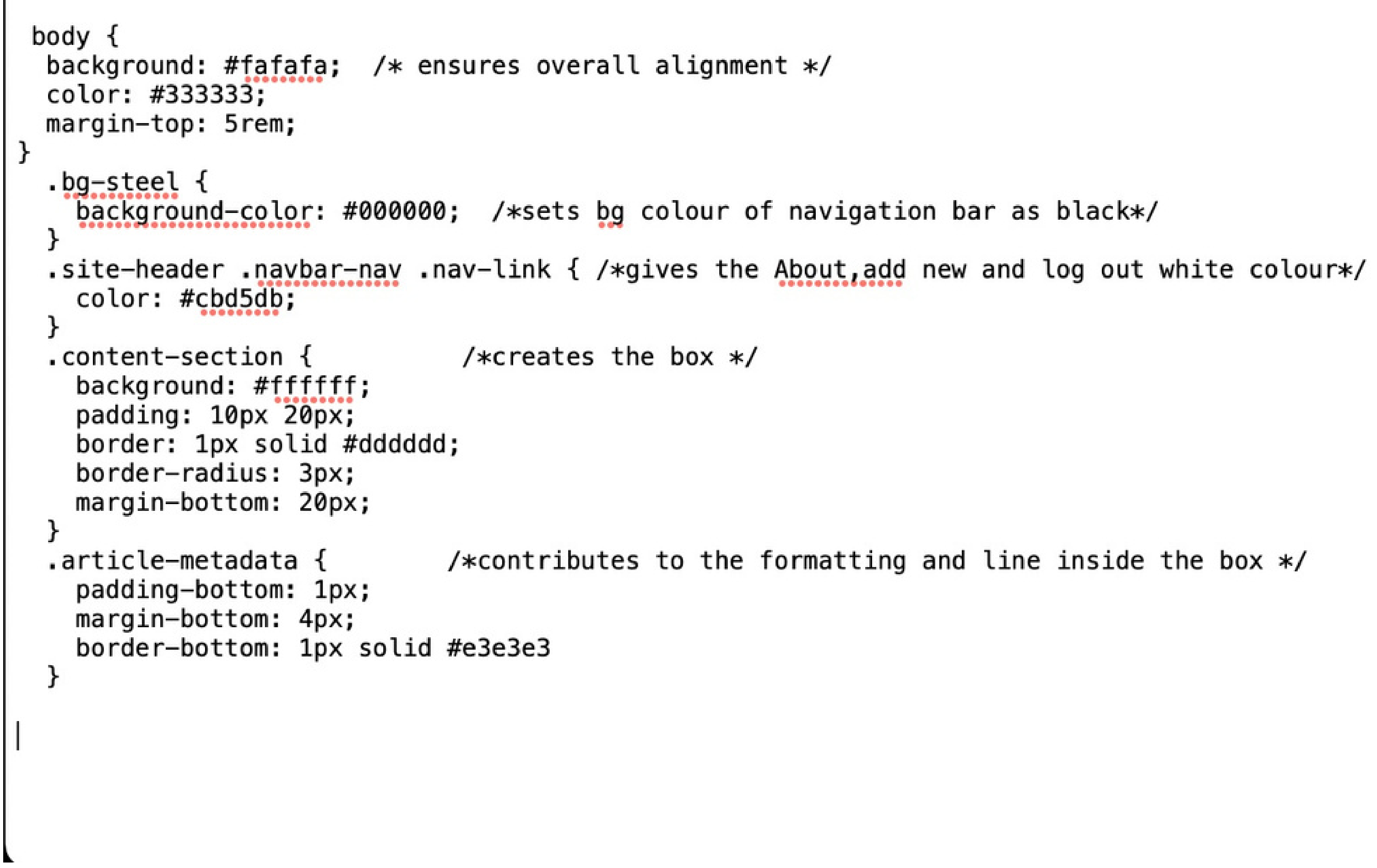
```
from flask import Flask, render_template, url_for, flash, redirect, request
from forms import EntryForm
from flask_sqlalchemy import SQLAlchemy
from datetime import datetime
import re
app = Flask(__name__, static_folder='static')#initialize the flask application,static folder where all static files are
stored
app.config['SECRET_KEY'] = 'JhRvPw5sL8y2TkQz'#protects from cross site request forgery attacks
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///site.db'#site.db name of the SQLite database
db = SQLAlchemy(app)#initialize the sqlalchemy database
class Post(db.Model):#defines the database model for posts
    id = db.Column(db.Integer, primary_key=True)
    category = db.Column(db.String(100), nullable=False)
    date_added = db.Column(db.DateTime, nullable=False, default=datetime.utcnow)
    content = db.Column(db.String(100), nullable=False)
    amount = db.Column(db.String(100), nullable=False)
def validate_category_format(category):#validates format of the category section
    category = category.strip()
    category = re.sub(r'\s+', ' ', category)
    category = ' '.join(word.capitalize() for word in category.split())
    return category
@app.route("/")#route for the main page
def mainpage():
    entries = Post.query.all()#displays all entries in the database
    category_query = request.args.get('category', '')#gets the entries with a certain category

    if category_query:#validates the format of the search bar input
        category_query = validate_category_format(category_query)



    if category_query:#filters entries to fit the format
        entries = Post.query.filter_by(category=category_query).all()

    return render_template('main.html', entries=entries) #redirects to main.html with all entries


@app.route("/entry/new", methods=['GET', 'POST'])#route to add a new entry
def new_post():
    form = EntryForm()#create a new instance of the entry form

    if form.validate_on_submit():#new isntance of Post with all data
        entry = Post(category=validate_category_format(form.category.data),
                     content=form.content.data, amount=form.amount.data)
        db.session.add(entry)
        db.session.commit()#commit changes to database
        flash('Entry successful!', 'success')#flash a success message and redirects to the main page
        return redirect(url_for('mainpage'))

    return render_template('entry.html', title="new entry", form=form)


if __name__ == '__main__':#run if executed directly
    app.run(debug=True)
```

forms.py



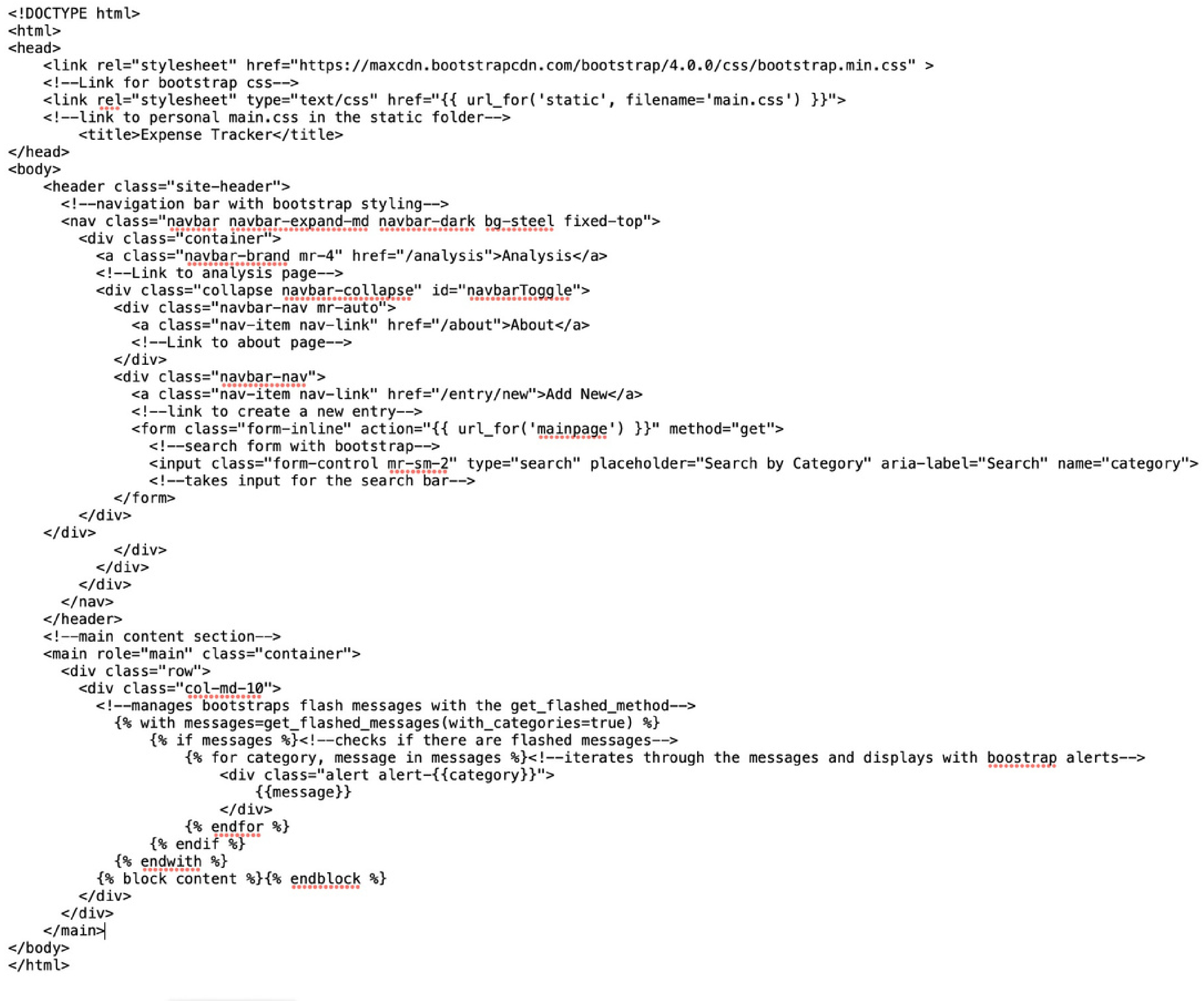
```python
from flask_wtf import FlaskForm #importing FlaskForm class from flask_wtf module
from wtforms import StringField,  SubmitField, TextAreaField, SelectField #importing form field types
from wtforms
from wtforms.validators import DataRequired

class EntryForm(FlaskForm): #defines a class called EntryForm which inherits aatributes from
FlaskForm
    category=SelectField('Category', choices=[('Miscellaneous', 'Click to Select'),
    ('Water', 'Water'), ('Electricity', 'Electricity'),('Vacation', 'Vacation'),('Transport',
'Transport'),
    ('Groceries', 'Groceries'),('Clothing and Accessories', 'Clothing and Accessories'),
    ('Health and Fitness','Health and Fitness'),('Medical', 'Medical'),('Gadgets', 'Gadgets'),
    ('Home', 'Home'),('Rent', 'Rent'),('Miscellaneous', 'Miscellaneous'),('Services', 'Services'),
    ('Education','Education'),('Food','Food')]) #drop down field with the above options
    content=TextAreaField('Content',validators=[DataRequired()]) #creates text box that required data
to be filled in
    amount=TextAreaField('Amount',validators=[DataRequired()])
    submit=SubmitField('Enter') #creates a button called submit with the name "Enter"
```

main.css



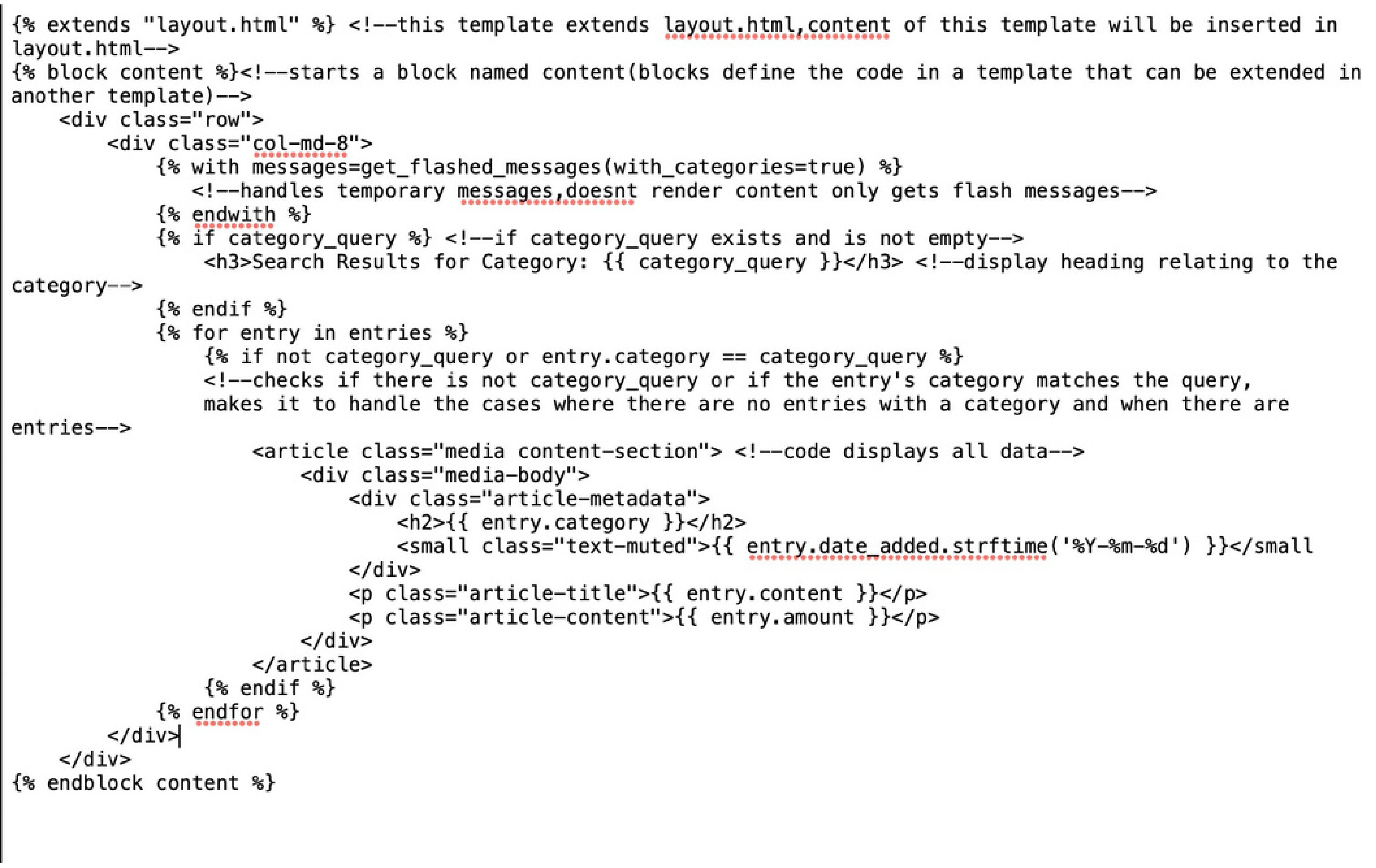
```css
 body {
  background: #fafafa;  /* ensures overall alignment */
  color: #333333;
  margin-top: 5rem;
}
  .bg-steel {
    background-color: #000000;  /*sets bg colour of navigation bar as black*/
  }
  .site-header .navbar-nav .nav-link { /*gives the About,add new and log out white colour*/
    color: #cbd5db;
  }
  .content-section {          /*creates the box */
    background: #ffffff;
    padding: 10px 20px;
    border: 1px solid #dddddd;
    border-radius: 3px;
    margin-bottom: 20px;
  }
  .article-metadata {         /*contributes to the formatting and line inside the box */
    padding-bottom: 1px;
    margin-bottom: 4px;
    border-bottom: 1px solid #e3e3e3
  }
```

layout.html



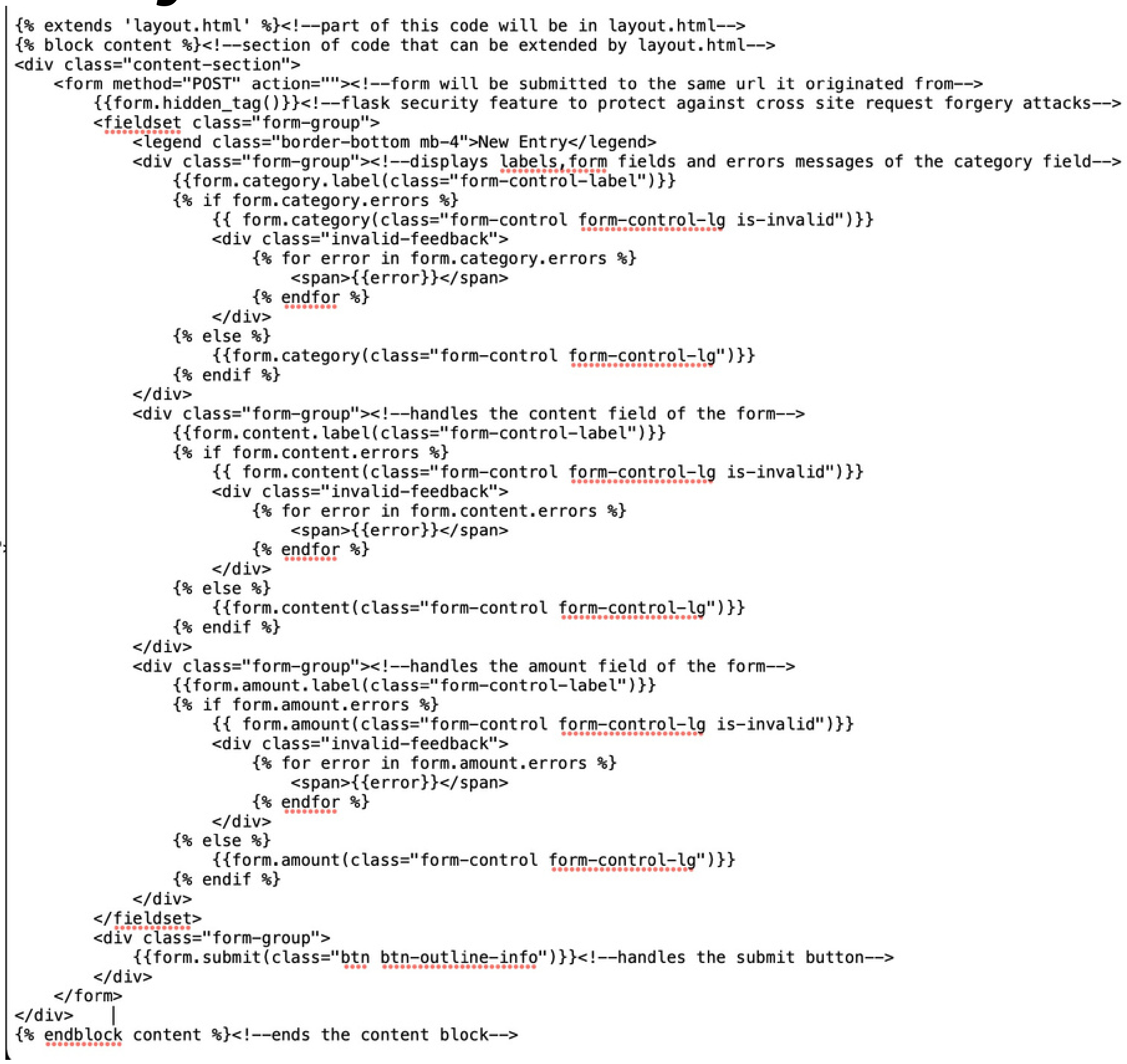
```
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" >
    <!--Link for bootstrap css-->
    <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='main.css') }}">
    <!--link to personal main.css in the static folder-->
        <title>Expense Tracker</title>
</head>
<body>
    <header class="site-header">
      <!--navigation bar with bootstrap styling-->
      <nav class="navbar navbar-expand-md navbar-dark bg-steel fixed-top">
        <div class="container">
          <a class="navbar-brand mr-4" href="/analysis">Analysis</a>
          <!--Link to analysis page-->
          <div class="collapse navbar-collapse" id="navbarToggle">
            <div class="navbar-nav mr-auto">
              <a class="nav-item nav-link" href="/about">About</a>
              <!--Link to about page-->
            </div>
            <div class="navbar-nav">
              <a class="nav-item nav-link" href="/entry/new">Add New</a>
              <!--link to create a new entry-->
              <form class="form-inline" action="{{ url_for('mainpage') }}" method="get">
                <!--search form with bootstrap-->
                <input class="form-control mr-sm-2" type="search" placeholder="Search by Category" aria-label="Search" name="category">
                <!--takes input for the search bar-->
            </form>
        </div>
    </div>
            </div>
          </div>
        </div>
      </nav>
    </header>
    <!--main content section-->
    <main role="main" class="container">
      <div class="row">
        <div class="col-md-10">
          <!--manages bootstraps flash messages with the get_flashed_method-->
            {% with messages=get_flashed_messages(with_categories=true) %}
                {% if messages %}<!--checks if there are flashed messages-->
                    {% for category, message in messages %}<!--iterates through the messages and displays with boostrap alerts-->
                        <div class="alert alert-{{category}}">
                            {{message}}
                        </div>
                    {% endfor %}
                {% endif %}
            {% endwith %}
          {% block content %}{% endblock %}
        </div>
      </div>
    </main>
</body>
</html>
```

main.html



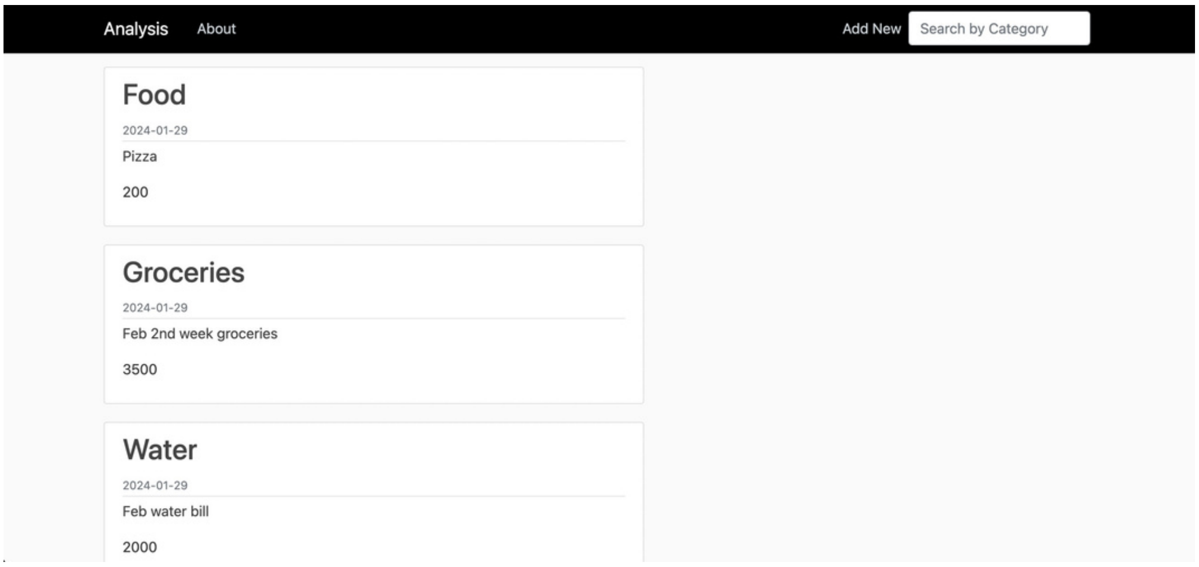
```
{% extends "layout.html" %} <!--this template extends layout.html,content of this template will be inserted in
layout.html-->
{% block content %}<!--starts a block named content(blocks define the code in a template that can be extended in
another template)-->
    <div class="row">
        <div class="col-md-8">
            {% with messages=get_flashed_messages(with_categories=true) %}
               <!--handles temporary messages,doesnt render content only gets flash messages-->
            {% endwith %}
            {% if category_query %} <!--if category_query exists and is not empty-->
                <h3>Search Results for Category: {{ category_query }}</h3> <!--display heading relating to the
category-->
            {% endif %}
            {% for entry in entries %}
                {% if not category_query or entry.category == category_query %}
                <!--checks if there is not category_query or if the entry's category matches the query,
                makes it to handle the cases where there are no entries with a category and when there are
entries-->
                    <article class="media content-section"> <!--code displays all data-->
                        <div class="media-body">
                            <div class="article-metadata">
                                <h2>{{ entry.category }}</h2>
                                <small class="text-muted">{{ entry.date_added.strftime('%Y-%m-%d') }}</small
                            </div>
                            <p class="article-title">{{ entry.content }}</p>
                            <p class="article-content">{{ entry.amount }}</p>
                        </div>
                    </article>
                {% endif %}
            {% endfor %}
        </div>
    </div>
{% endblock content %}
```

entry.html



```
{% extends 'layout.html' %}<!--part of this code will be in layout.html-->
{% block content %}<!--section of code that can be extended by layout.html-->
<div class="content-section">
    <form method="POST" action=""><!--form will be submitted to the same url it originated from-->
        {{form.hidden_tag()}}<!--flask security feature to protect against cross site request forgery attacks-->
        <fieldset class="form-group">
            <legend class="border-bottom mb-4">New Entry</legend>
            <div class="form-group"><!--displays labels,form fields and errors messages of the category field-->
                {{form.category.label(class="form-control-label")}}
                {% if form.category.errors %}
                    {{ form.category(class="form-control form-control-lg is-invalid")}}
                    <div class="invalid-feedback">
                        {% for error in form.category.errors %}
                            <span>{{error}}</span>
                        {% endfor %}
                    </div>
                {% else %}
                    {{form.category(class="form-control form-control-lg")}}
                {% endif %}
            </div>
            <div class="form-group"><!--handles the content field of the form-->
                {{form.content.label(class="form-control-label")}}
                {% if form.content.errors %}
                    {{ form.content(class="form-control form-control-lg is-invalid")}}
                    <div class="invalid-feedback">
                        {% for error in form.content.errors %}
                            <span>{{error}}</span>
                        {% endfor %}
                    </div>
                {% else %}
                    {{form.content(class="form-control form-control-lg")}}
                {% endif %}
            </div>
            <div class="form-group"><!--handles the amount field of the form-->
                {{form.amount.label(class="form-control-label")}}
                {% if form.amount.errors %}
                    {{ form.amount(class="form-control form-control-lg is-invalid")}}
                    <div class="invalid-feedback">
                        {% for error in form.amount.errors %}
                            <span>{{error}}</span>
                        {% endfor %}
                    </div>
                {% else %}
                    {{form.amount(class="form-control form-control-lg")}}
                {% endif %}
            </div>
        </fieldset>
        <div class="form-group">
            {{form.submit(class="btn btn-outline-info")}}<!--handles the submit button-->
        </div>
    </form>
</div>
{% endblock content %}<!--ends the content block-->
```

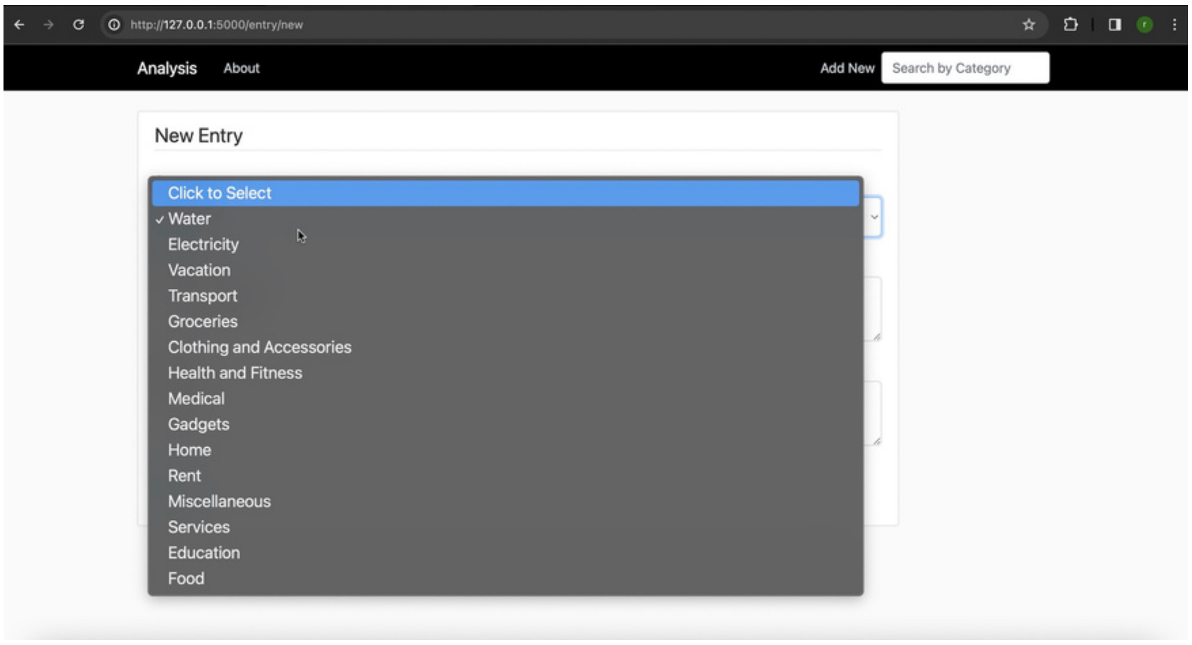
PROGRAM OUTPUT


Analysis About Add New Search by Category

Food
2024-01-29
Pizza
200

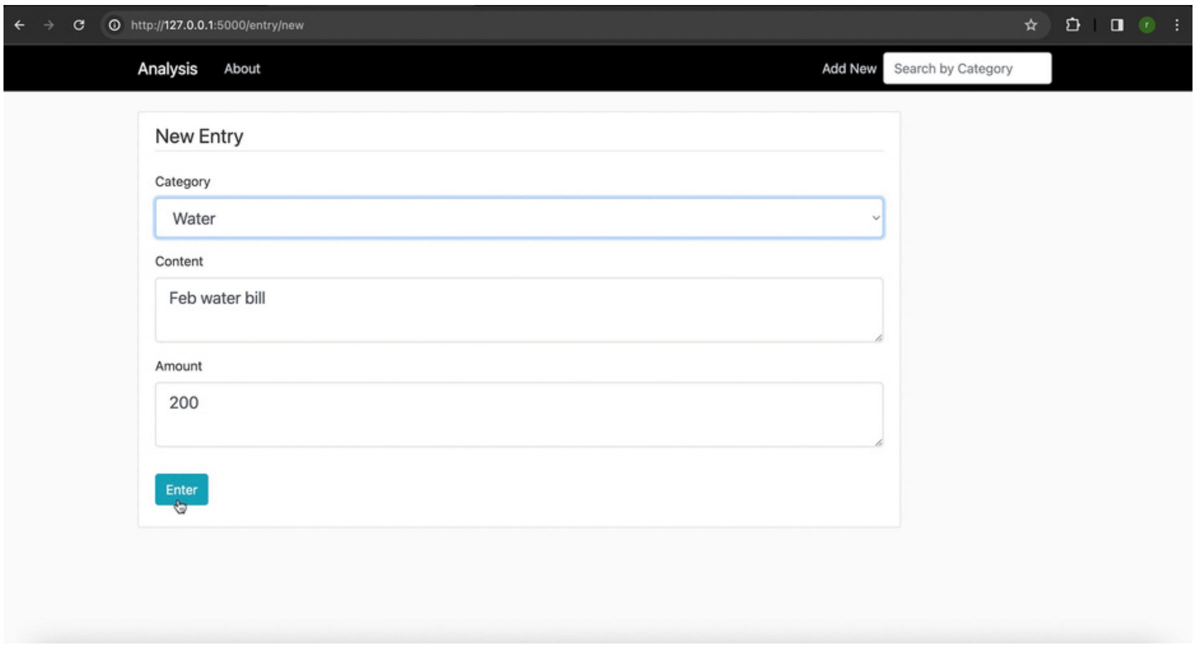
Groceries
2024-01-29
Feb 2nd week groceries
3500

Water
2024-01-29
Feb water bill
2000


http://127.0.0.1:5000/entry/new

Analysis About Add New Search by Category

New Entry



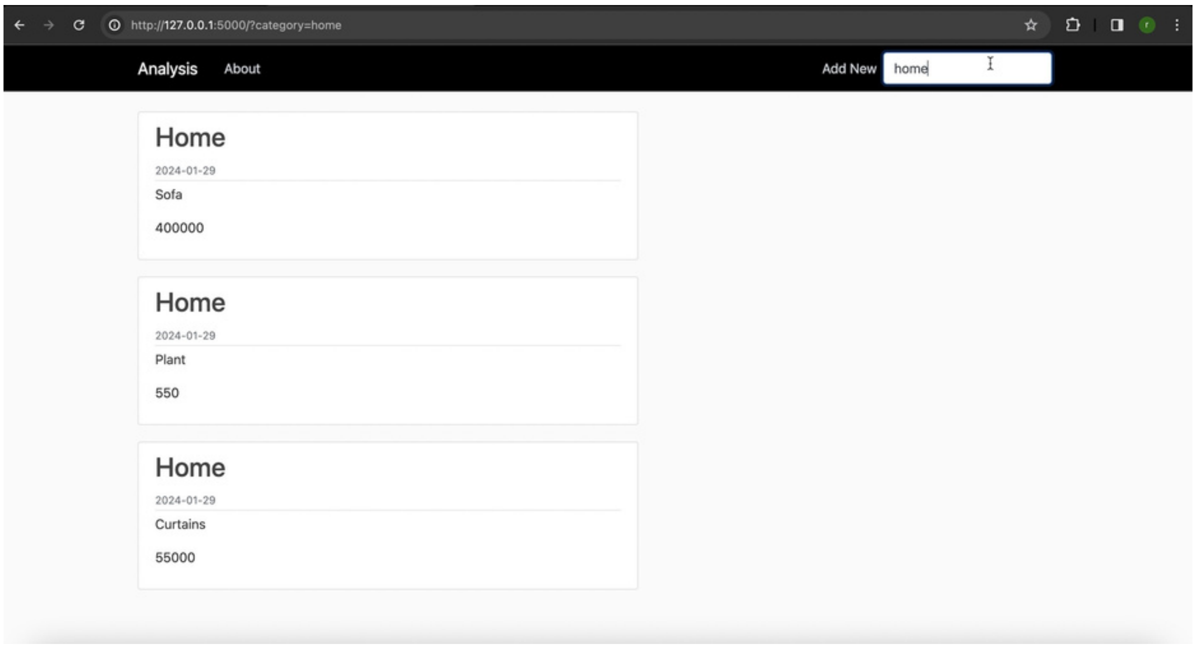
http://127.0.0.1:5000/entry/new

New Entry

Category: Water

Content: Feb water bill

Amount: 200



http://127.0.0.1:5000/?category=home

Home
2024-01-29
Sofa
400000

Home
2024-01-29
Plant
550

Home
2024-01-29
Curtains
55000

LIMITATIONS

Developments to be made in the future-

- **A Login Page**
- **A Register Page**
- **Expense Analysis with the help of mathplotlib**
- **An Edit/Delete option for the entries**

BIBLIOGRAPHY

- https://chat.openai.com/
- https://github.com/
- https://stackoverflow.com/
- https://flask.palletsprojects.com/en/3.0.x/
- https://www.w3schools.com/python/
- https://www.w3schools.com/css/