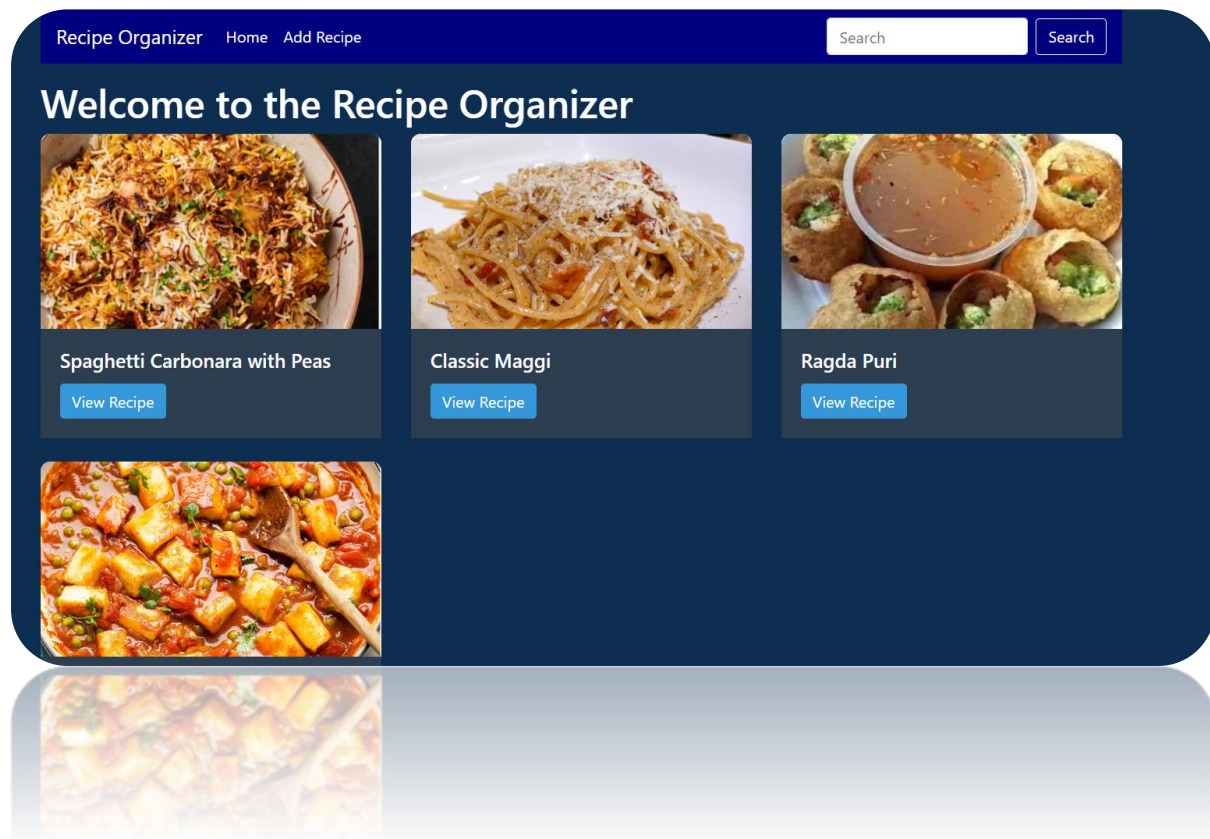


Recipe Organizer Web Application



By – Gaurav Upadhyay – 22BH1A0577

Irfan Sajid Mohammed - 22BH1A0587

Under The Guidance Of – Srinivasa Reddy Sir

St Mary's Engineering College, Hyderabad – 2024

[Affiliated To JNTU , Hyderabad Deshmukhi (v), Hayath
Nagar (M) R.R (D) -508284]

(2022-2024)



UNDERTAKING

We declare that the work presented in this project titled “Recipe Organizer” is done by Gaurav Upadhyay and Mohammad Faiz. We have not plagiarized or submitted the same work for the award of any other examination. In case this undertaking is found

incorrect, we accept that our certificates may be
unconditionally withdrawn

Date: June 2024

Place: St Mary's Engineering College

CERTIFICATE

We certified that the work contained in the project titled “Recipe Organizer”, by Gaurav Upadhyay and Mohammad Sajid Fiaz has been carried out under my supervision and that this work has not been submitted elsewhere for any other exams or projects.

Srinivasa Reddy

(Guidance Lecturer)

(St Mary’s Engineering College-HYD)

Malla Reddy

(Head Of The Department)

(St Mary’s Engineering College-HYD)

Acknowledgement

We would like to thank our hod Malla Reddy Sir,
Srinivas Reddy Sir Guidance Lecture of St Mary's
College For giving an opportunity of making this Project
.We own our sincere gratitude towards St Mary's
Engineering College

Our heartfelt thanks to Srinivasa Reddy Sir for guiding
on this project.

We further thank to all the staff members of St Mary's
Engineering College- Hyderabad

We also express our deepest gratitude to our parents.

Finally , We would like to wind up by paying our
heartfelt thanks to all our near and dear ones.

1)Gaurav Upadhyay

2) Irfan Sajid Mohammed

Index

- Abstract Of The Project
- Introduction Of The Project
- Problem Statement
- System Requirements
- Output Of The Project
- Conclusion Of The Project

ABSTRACT

The Recipe Organizer is a Web Application For Organizing and storing your Food Recipe Safely. It has many features like Adding Recipies, Viewing Recipes with Better UI and UX Design making it more attractive and easy to use. Helping Individual to store there Granny's Recipes Safely without being fear of messing up and losing Stuff.

It's has features like adding images of your cooked food and adding name ,proper instruction and ingridents making it much better to organize and locate for future reference .

Since It's a web application you can share the link with you close one's

Efficient Search Bar help's to Search and locate the recipes in an much efficient and easy manner.

Introduction To Project

- This project uses technologies like python, html, CSS, flask, Werkzeug.
- This web application can run on your local System(Computer, Laptop)
- It is Designed to help to share , organize and view recipes helping individuals to store and organize there recipe safely.
- The User Interface of the project is simple and beautiful making it easy to use enhancing the experience of user's.

- This project is simple with clear and clean folder structure making it easy for individuals to start their journey as developers.
- You can simply follow the instructions, copy the code and it is easy if you want to play with the UX and UI part; you are free to do.
- You can also add many functionalities according to your convenience.
- By starting this project you would gain ideas about the folder structure, basics of Python, HTML, CSS and many more.

Problem Statement

PROBLEM STATEMENT: Precious and delicious ways of cooking food are being lost.

USER PERSONA: Many successful experiments has been done in the past with food to make it taste and healthy. Our Grannies' food taste are lost. why ? Because those recipes were with them only and as they went away there recipes went along with them . Only that remained with us were those taste which was left with us. Few recipies were written but were torn out with time making it like a game to find the missing puzzle.

OBJECTIVE: To SafeQuard our recipes , retain and retrieve them when ever we want.

SOLUTION: An web application that can Organize the recipes very effectively making it more User friendly with Enriched UX and UI.

STAKE HOLDERS: 1) Person who want to see recipes

2) Person who want to add recipes

CHOICES : 1) Add Recipes

2) View Recipes

VALUE PROPOSITION: Making people to Organize There recipes in structured and safe way. Easy for navigation and upload the recipes.

PARAMETERS: Recipe Name

Recipe Ingridient

Recipe Instruction

MVP VISUALIZATION: As many as recipe in a day along with best UI AND UX make it shareable and safe.

System Requirements

- 1) Laptop or Computer along with operating system
- 2) Python 3.8 version installed in the System.

Software Environment

- 1) Operating System: Windows, Linux, macop , etc.
- 2) Integrated Development Environment(IDE): VS
CODE, PyCharm
- 3) Python Package Manager(PIP): For python
packages , dependencies and libraries
- 4) Version Control System(GIT): For real collaboration
and storing it in Cloud .

STEPS TO STAGE

- 1) Download VS CODE from web browser
- 2) Install python 3.8 by clicking on extension button on the right side of screen and type python 3.8

You are ready to go

Folder structure

recipe_organizer

app.py

static

images

style.css

templates

base.html

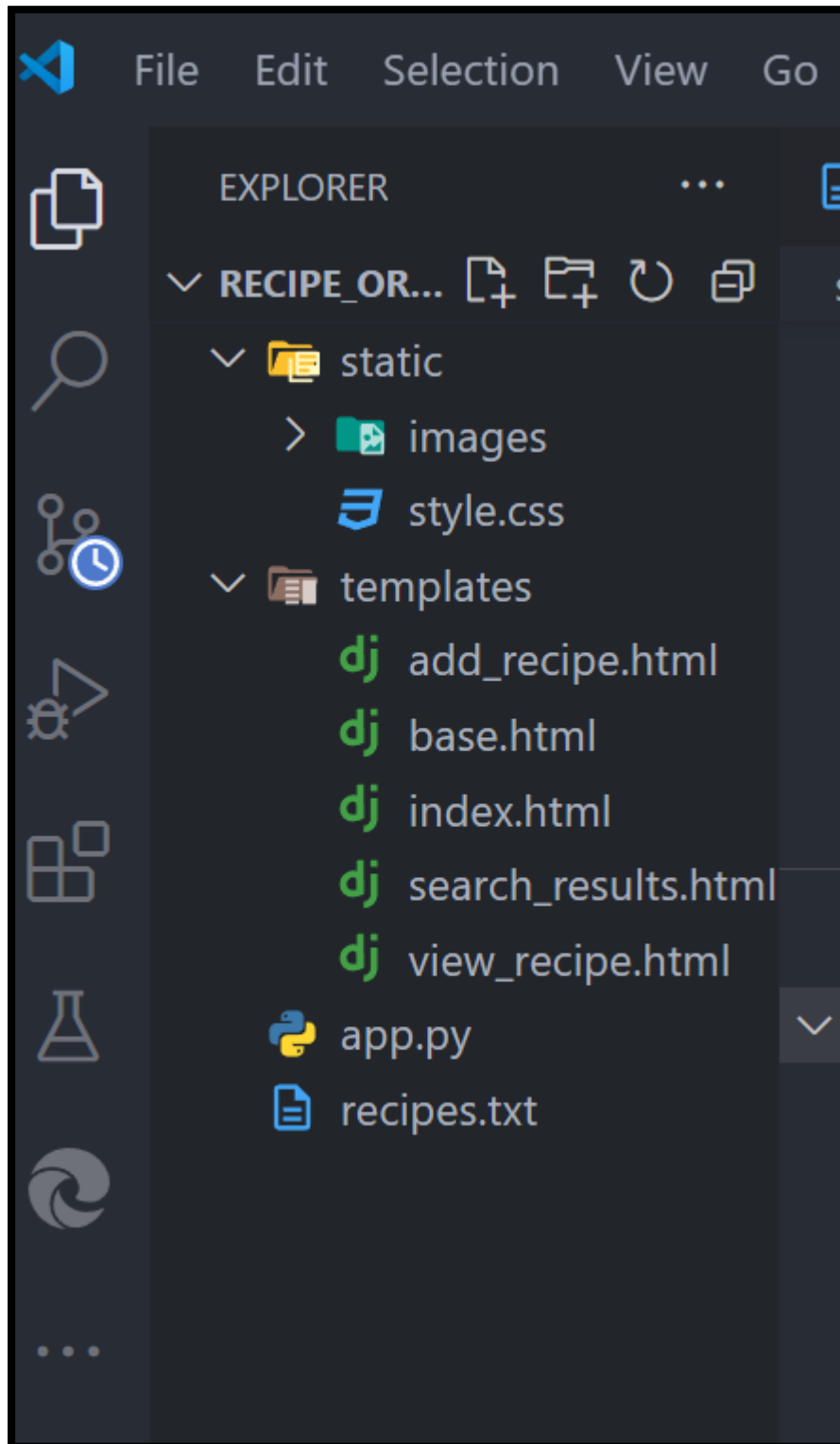
index.html

add_recipe.html

search_results.html

view_recipe.html

recipes.txt



CODING PART

app.py

```
from flask import Flask, request, render_template, redirect, url_for, flash
import os
from werkzeug.utils import secure_filename

app = Flask(__name__)
app.secret_key = 'your_secret_key' # Replace with a random secret key

UPLOAD_FOLDER = 'static/images'
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg', 'gif'}

app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER

if not os.path.exists(UPLOAD_FOLDER):
    os.makedirs(UPLOAD_FOLDER)

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in
ALLOWED_EXTENSIONS

@app.route('/')
def index():
    recipes = []
    if os.path.exists('recipes.txt'):
        with open('recipes.txt', 'r') as file:
            lines = file.readlines()
            recipe = {}
```

```

        for line in lines:
            if line.strip() == "":
                recipes.append(recipe)
                recipe = {}
            else:
                key, value = line.split(": ", 1)
                recipe[key.strip()] = value.strip()
        if recipe:
            recipes.append(recipe)
    return render_template('index.html', recipes=recipes)

@app.route('/add', methods=['GET', 'POST'])
def add_recipe():
    if request.method == 'POST':
        name = request.form['name']
        ingredients = request.form['ingredients']
        instructions = request.form['instructions']
        file = request.files['image']

        if not name or not ingredients or not instructions or not file:
            flash('All fields are required!')
            return redirect(url_for('add_recipe'))

        if file and allowed_file(file.filename):
            filename = secure_filename(file.filename)
            file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
        else:
            flash('Invalid file type!')
            return redirect(url_for('add_recipe'))

        with open('recipes.txt', 'a') as file:
            file.write(f>Name: {name}\nIngredients: {'\n'.join(ingredients.split(',') )}\nInstructions: {instructions}\nImage: {filename}\n\n")

        flash('Recipe added successfully!')
        return redirect(url_for('index'))

```



```

        return render_template('add_recipe.html')

@app.route('/view/<name>')
def view_recipe(name):
    if os.path.exists('recipes.txt'):
        with open('recipes.txt', 'r') as file:
            lines = file.readlines()
            recipe = {}
            found = False
            for line in lines:
                if line.startswith("Name: "):
                    if found:
                        break
                    if line.split(": ", 1)[1].strip() == name:
                        found = True
            if found:
                if line.strip() == "":
                    break
                key, value = line.split(": ", 1)
                recipe[key.strip()] = value.strip()
            if found:
                return render_template('view_recipe.html', recipe=recipe)
    flash('Recipe not found.')
    return redirect(url_for('index'))

@app.route('/search', methods=['POST'])
def search_recipe():
    keyword = request.form['keyword']
    results = []

    if os.path.exists('recipes.txt'):
        with open('recipes.txt', 'r') as file:
            lines = file.readlines()
            recipe = {}
            for line in lines:
                if line.strip() == "":
                    if any(keyword.lower() in v.lower() for v in
recipe.values()):

```

```

        results.append(recipe)
        recipe = {}
    else:
        key, value = line.split(": ", 1)
        recipe[key.strip()] = value.strip()
        if recipe and any(keyword.lower() in v.lower() for v in
recipe.values()):
            results.append(recipe)

    if results:
        return render_template('search_results.html', results=results)
    else:
        flash('No matching recipes found.')
        return redirect(url_for('index'))
else:
    flash('No recipes found.')
    return redirect(url_for('index'))

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

```

add_recipe.html

```

{% extends "base.html" %}

{% block content %}
<h1>Add a New Recipe</h1>
<form method="POST" enctype="multipart/form-data">
    <div class="form-group">
        <label for="name">Recipe Name</label>
        <input type="text" class="form-control" id="name" name="name"
required>
    </div>
    <div class="form-group">
        <label for="ingredients">Ingredients (separated by commas)</label>

```

```

        <input type="text" class="form-control" id="ingredients"
name="ingredients" required>
    </div>
    <div class="form-group">
        <label for="instructions">Cooking Instructions</label>
        <textarea class="form-control" id="instructions" name="instructions"
rows="3" required></textarea>
    </div>
    <div class="form-group">
        <label for="image">Recipe Image</label>
        <input type="file" class="form-control" id="image" name="image"
required>
    </div>
    <button type="submit" class="btn btn-primary">Add Recipe</button>
</form>
{% endblock %}

```

base.html

```

<!doctype html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title>Recipe Organizer</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
">
    <link rel="stylesheet" href="{{ url_for('static', filename='style.css')
}}">
    <style>
        .navbar {
            background-color: #000080; /* Navy Blue */
        }
        .navbar-brand, .nav-link, .btn, .alert {
            color: #ffffff !important; /* White */
        }
        .btn-outline-success {

```

```

        color: #ffffff;
        border-color: #ffffff;
    }
    .btn-outline-success:hover {
        background-color: #ffffff;
        color: #000080;
    }
</style>
</head>
<body>
    <div class="container">
        <nav class="navbar navbar-expand-lg navbar-dark">
            <a class="navbar-brand" href="/">Recipe Organizer</a>
            <button class="navbar-toggler" type="button" data-
toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-
expanded="false" aria-label="Toggle navigation">
                <span class="navbar-toggler-icon"></span>
            </button>
            <div class="collapse navbar-collapse" id="navbarNav">
                <ul class="navbar-nav mr-auto">
                    <li class="nav-item">
                        <a class="nav-link" href="/">Home</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="/add">Add Recipe</a>
                    </li>
                </ul>
                <form action="/search" method="POST" class="form-inline my-2
my-lg-0">
                    <input class="form-control mr-sm-2" type="search"
name="keyword" placeholder="Search" aria-label="Search">
                    <button class="btn btn-outline-success my-2 my-sm-0"
type="submit">Search</button>
                </form>
            </div>
        </nav>
        <div class="content mt-3">
            {% with messages = get_flashed_messages() %}

```

```

        {% if messages %}
        <div class="alert alert-warning" role="alert">
            {{ messages[0] }}
        </div>
        {% endif %}
    {% endwith %}
    {% block content %}{% endblock %}
</div>
</div>
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js
"></script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"><
/script>
</body>
</html>

```

Index.html

```

{% extends "base.html" %}

{% block content %}
<h1>Welcome to the Recipe Organizer</h1>
<div class="row">
    {% for recipe in recipes %}
    <div class="col-md-4">
        <div class="card mb-4 shadow-sm">
            
            <div class="card-body">
                <h5 class="card-title">{{ recipe['Name'] }}</h5>
                <a href="{{ url_for('view_recipe', name=recipe['Name']) }}"
class="btn btn-primary">View Recipe</a>
            </div>
        </div>
    </div>
    {% endfor %}
</div>

```

```
    </div>
    {% endfor %}
</div>
{% endblock %}
```

Search results.html

```
{% extends "base.html" %}

{% block content %}
<h1>Search Results</h1>
<div class="row">
    {% for recipe in results %}
        <div class="col-md-4">
            <div class="card mb-4 shadow-sm">
                
                <div class="card-body">
                    <h5 class="card-title">{{ recipe['Name'] }}</h5>
                    <a href="{{ url_for('view_recipe', name=recipe['Name']) }}"
class="btn btn-primary">View Recipe</a>
                </div>
            </div>
        </div>
    {% endfor %}
</div>
{% endblock %}
```

view recipe.html

```
{% extends "base.html" %}

{% block content %}
<h1>{{ recipe['Name'] }}</h1>

<h2>Ingredients</h2>
```

```
<p>{{ recipe['Ingredients'] }}</p>
<h2>Instructions</h2>
<p>{{ recipe['Instructions'] }}</p>
<a href="/" class="btn btn-secondary">Back to Home</a>
{% endblock %}
```

style.css

```
body {
    background-color: #0d2e50; /* Dark blue background */
    color: #ffffff; /* White text */
}

.container {
    margin-top: 20px;
}

.card {
    border: none;
    border-radius: 10px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

.card-title {
    color: #ffffff; /* White text */
}

.card-body {
    background-color: #2c3e50; /* Dark blue */
}

.card-img-top {
    border-top-left-radius: 10px;
    border-top-right-radius: 10px;
    object-fit: cover; /* Ensures the image covers the entire space without stretching */
    height: 200px; /* Adjust the height as per your design needs */
}
```

```
}

.btn-primary {
  background-color: #3498db; /* Light blue */
  border: none;
}

.btn-primary:hover {
  background-color: #2980b9; /* Darker light blue */
}

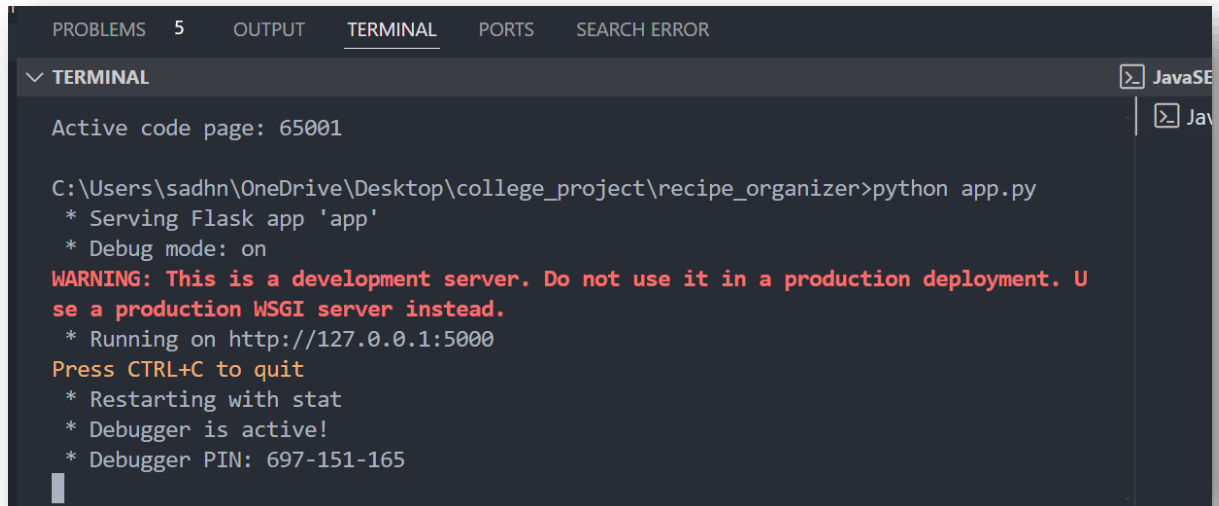
.alert-warning {
  background-color: #f39c12; /* Bootstrap warning orange */
  color: #ffffff; /* White text */
  border: none;
  border-radius: 5px;
}
```

Now let's install the dependencies

pip install Flask Werkzeug : Type in terminal (view -
Terminal)

Then run : **python app.py**

OUTPUT




The screenshot shows a VS Code terminal window with the following content:

```
PROBLEMS 5 OUTPUT TERMINAL PORTS SEARCH ERROR
✓ TERMINAL
Active code page: 65001
C:\Users\sadhn\OneDrive\Desktop\college_project\recipe_organizer>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 697-151-165
```

Press ctrl + link(<http://127.0.0.1:5000>)


[Recipe Organizer](#) [Home](#) [Add Recipe](#)

Welcome to the Recipe Organizer




Spaghetti Carbonara with Peas

[View Recipe](#)




Classic Maggi

[View Recipe](#)



Ragda Puri


[View Recipe](#)





[Recipe Organizer](#) [Home](#) [Add Recipe](#)

Spaghetti Carbonara with Peas

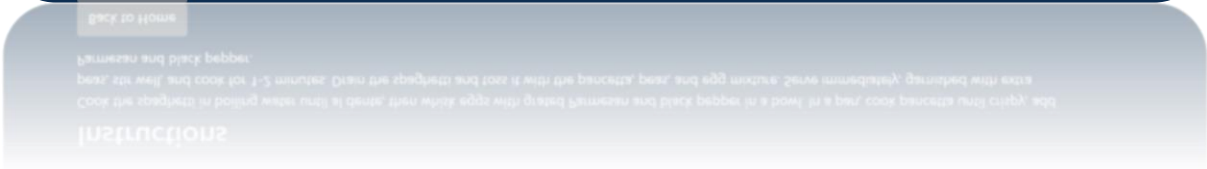


Ingredients

spaghetti, eggs, pancetta, Parmesan, black pepper, peas

Instructions

Cook the spaghetti in boiling water until al dente, then whisk eggs with grated Parmesan and black pepper in a bowl. In a pan, cook pancetta until crispy, add peas, stir well, and cook for 1-2 minutes. Drain the spaghetti and toss it with the pancetta, peas, and egg mixture. Serve immediately, garnished with extra Parmesan and black pepper.

[Back to Home](#)

Recipe Organizer

Home

Add Recipe

Search

Search

Add a New Recipe

Recipe Name

Ingredients (separated by commas)

Cooking Instructions

Recipe Image

Choose File

No file chosen

Add Recipe



Recipe Organizer

Home


Add Recipe

matter

X

Search

Search Results



Matter Paneer Sabji

View Recipe

CONCLUSION

The web application recipe organizer, developed by us offers a straight forward solution for efficiently handling and organizing recipes.

This project demonstrates Python capability to create a particle web application for managing recipes.

Key Features and Benefits:

- 1) Adding Recipes
- 2) Viewing recipes
- 3) Advanced Search Bar
- 4) Simple and local storage System
- 5) User-Friendly
- 6) Simple AND Efficient UX AND UI

In conclusion, this python- based application illustrates how programming can enhance Web Application practically. Changes in user interface like colors can be easy done as comment as passed more functionality can be also added accordingly.

Thank you.

