Design of Recipe Finder Full Stack Web Application

Team Members:

Parthiban M 3122215001065

Sathvika V.S. 3122215001097

Y.V.Ojus 3122215001125

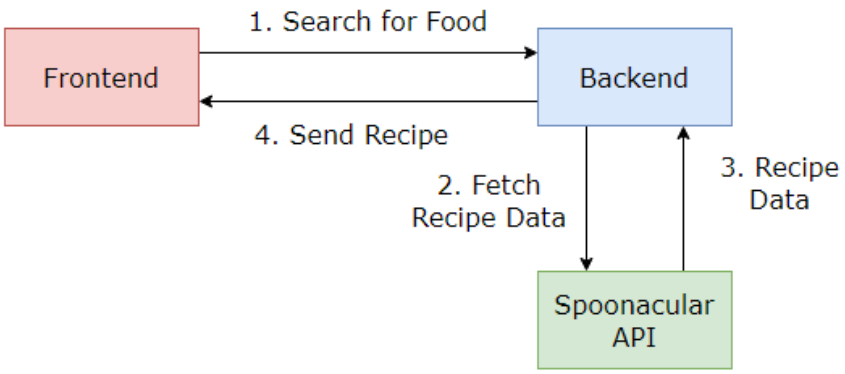
Github Link:

Project

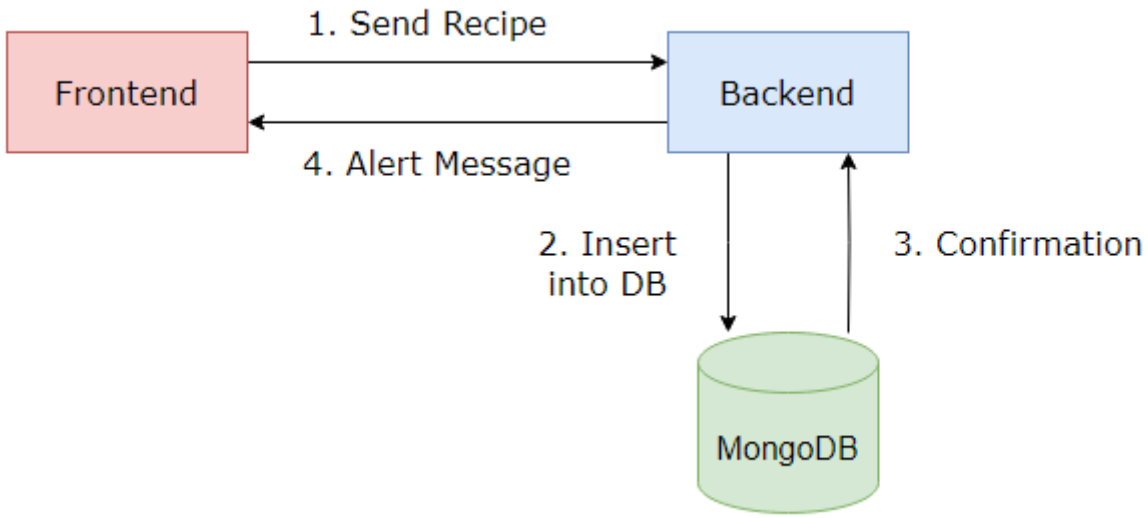
Aim:

Design a full stack Recipe Finder web application which allows users to search for recipes based on the name of the recipe. The app can fetch data from various recipe APIs, display recipes, and provide additional details such as cooking instructions, nutritional information, and user reviews. Prepare a report containing the design, code, output snapshots, best practices used and learning outcomes.

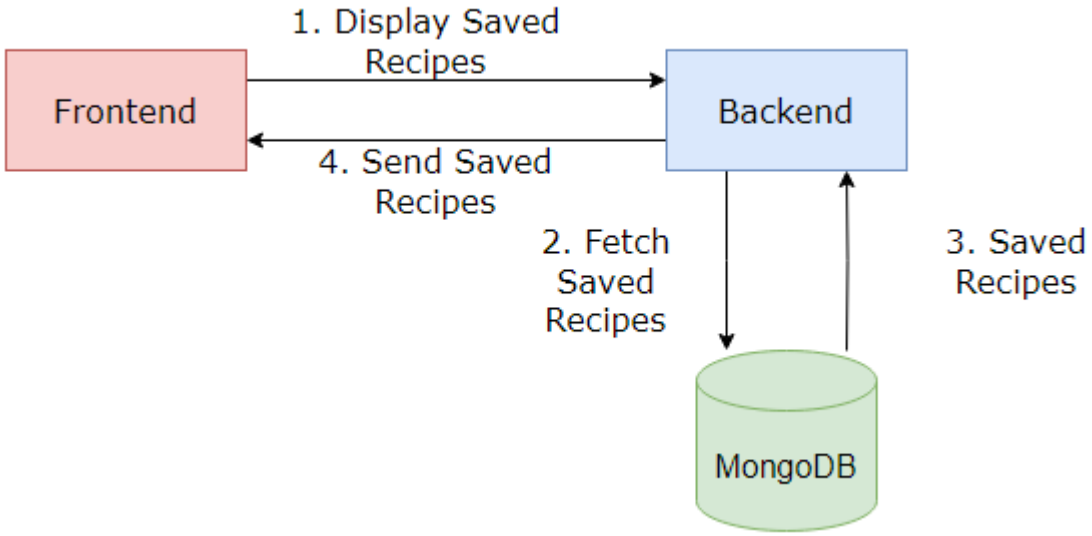
Design Diagram:



Fetching Recipe Data



Saving a Recipe



Fetching a Saved Recipe

Collections Used:

The data is stored on the backend database **MongoDB** where the recipe information such as recipe ID, recipe title and recipe image is stored in the document **recipe** and collection **saved**


```
_id: ObjectId('663cca2c235e4f6c709fe4ff')
id : 665769
title : "Zucchini Pizza Boats"
image : "https://img.spoonacular.com/recipes/665769-312x231.jpg"

_id: ObjectId('663cca8f235e4f6c709fe500')
id : 654857
title : "Pasta On The Border"
image : "https://img.spoonacular.com/recipes/654857-312x231.jpg"
```


The corresponding information such as steps, ingredients required is fetched upon the **API request (spoonacular)** and displayed to the user. Rather than storing the entire recipe, the ID is alone stored and data is fetched when required to display.

API Link

Code:

App.js:

```
import "./App.css";
import Header from "./Components/Header";
import Body from "./Components/Body";
import { useState } from "react";
import axios from "axios";
import Card from "./Components/Card";
```

```
import ReactLoading from "react-loading"
import Recipe from "./Recipe";




function App() {
  let [search, setSearch] = useState("");  //State Management for search query
  let [data, setData] = useState([]);
  const [userdata, setUserdata] = useState([]);
  const [loading, setLoading] = useState(false); // State variable for loading

  const [recipe, setRecipe] = useState();

  function sendSearch() {
    setUserdata([]);
    setLoading(true); // Set loading to true while fetching data
    axios
      .get(`http://127.0.0.1:3001/send?data=${search}`)
      .then((res) => {
        console.log(res.data)
        setData(res.data);
        setLoading(false); // Set loading to false after data fetching completes
      })
      .catch((err) => {
        console.log(err);
        setLoading(false); // Set loading to false in case of error
      });
  }

  async function displayfunc(){
    setData([]);
    setLoading(true); // Set loading to true while fetching data
    try {
      const response = await fetch("http://localhost:3001/display", {
        method: 'POST',
        headers: {
          'content-type': 'application/json',
        },
      });
      const data = await response.json();
      console.log(data.user);
      setUserdata(data.user);
      setLoading(false); // Set loading to false after data fetching completes
```

```
    } catch(err) {
      console.log(err);
      setLoading(false); // Set loading to false in case of error
    }
  }

  function changeRecipe(recipe)
  {
    setRecipe(recipe)
  }


  return (
    <div className="App">
      <Header />
      <Body />

      <div className="search--div">
        <input
          type="text"
          placeholder="Enter a Recipe"
          id="search--input"
          onChange={(e) => setSearch(e.target.value)}
        />
  <button onClick={sendSearch} className="display-button">
        <span className="material-icons">search</span>
      </button>

      </div>
      <center><br/><br/>
      <button onClick={displayfunc} className="display-button">
        Display saved items
      </button>

</center>

      {/* Conditional rendering based on loading state */}
      {loading ? (
          <center><ReactLoading type="bubbles" color="#0000FF" height={200}
width={200} /> </center>
      ) : (
        <div>
          <div className="card--collection">
            {data.map((element, index) => (
```

```
              <Card id={element.id} title={element.title} image={element.image}
key={index} flag={true} />
            ))}
          </div>
          {/* Conditional rendering based on data length */}
          {userdata.length > 0 && (
            <div className="card--collection">
              {userdata.map((user) => (
                <div key={user.id}>
                  <Card id={user.id} title={user.title} image={user.image}
flag={false} change= {changeRecipe} />
                </div>
              ))}
            </div>
          )}
        </div>
      )}
      {recipe && <Recipe recipe={recipe} /> }
    </div>
  );
}

export default App;
```

App.css:

```
@import url('https://fonts.googleapis.com/icon?family=Material+Icons');
*{
  padding: 0;
  margin: 0;
  box-sizing: border-box;
}

/* Fonts */

.pacifico-regular {
  font-family: "Pacifico", cursive;
  font-weight: 400;
  font-style: normal;
}


.concert-one-regular {
```

```
  font-family: "Concert One", sans-serif;
  font-weight: 400;
  font-style: normal;
}

/* App.js  */

.App {
  height: 100vh;
  width: 100vw;
  background-color: blanchedalmond;
  overflow: auto;
}

.card--collection{
  display: flex;
  flex-wrap: wrap;
  overflow-y: auto;
}


/* Header Component  */

.navbar{
  width: 100%;
  background-color: orange;
  height: 3rem;
  display: flex;
  justify-content: center;
  align-items: center;
}

.navbar ul{
  list-style: none;
}


/* Body Component  */

#body--heading{
  text-align: center;
  margin: 2rem;
  font-size: 35px;
}
```

```css
/* Search Component  */

.search--div{
  display: flex;
  justify-content: center;
  align-items: center;
  margin: 5rem;
}

#search--input{
  width: 50%;
  height: 2rem;
  margin-right: 1rem;

  text-align: center;
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
}

::placeholder{
  text-align: center;
}

::-webkit-input-placeholder {
  text-align: center;
}

#search-icon{
  cursor: pointer;
  height: 40px;
  width: 40px;
}

/* Card Component  */

.card{
  display: flex;
  flex-direction: column;
  justify-content: space-around;
  align-items: center;
  padding: 1rem;
  margin: 1rem;
  width: 300px;
```

```
  height: 300px;
  background-color:darkorange;
  border: 1px solid black;
}

.card img{
  height: 100px;
  width: 100px;
}

.card button{
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  padding: 3px;
}


.display-button {
  background-color: #48bdbb;
  color: white;
  font-size: 18px;
  padding: 5px 5px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s;
}

.display-button:hover {
  background-color: #3ca8a6;
}
```

Recipe.js:

```
import React from 'react';
import './Recipe.css';

function Recipe({ recipe }) {
  return (
    <div className="recipe-container">
      <h1 className="recipe-title">{recipe.title}</h1>
```

```
      <center> <img className="recipe-image" src={recipe.image} alt={recipe.title}
/> </center>
      <div className="recipe-details">
        <p><span className="detail-label">Servings:</span> {recipe.servings}</p>
        <p><span className="detail-label">Ready in Minutes:</span>
{recipe.readyInMinutes}</p>
        <p><span className="detail-label">Source:</span> {recipe.sourceName}</p>
        <p><span className="detail-label">Price Per Serving:</span>
${recipe.pricePerServing}</p>
      </div>
      <div className="recipe-ingredients">
        <h2>Ingredients:</h2>
        <ul>
          {recipe.extendedIngredients.map((ingredient, index) => (
            <li key={index} className="ingredient-item">{ingredient.original}</li>
          ))}
        </ul>
      </div>
      <div className="recipe-summary">
        <h2>Summary:</h2>
        <p dangerouslySetInnerHTML={{ __html: recipe.summary }}></p>
      </div>
      <div className="recipe-wine-pairing">
        <h2>Wine Pairing:</h2>
        <p className="wine-pairing-text">{recipe.winePairing.pairingText}</p>
        <div className="product-matches">
          {recipe.winePairing.productMatches.map(product => (
            <div key={product.id} className="product-match">
              <h3>{product.title}</h3>
             <center> <img src={product.imageUrl} alt={product.title}
className="product-image" /> </center>
              <p><span className="detail-label">Description:</span>
{product.description}</p>
              <p><span className="detail-label">Price:</span> {product.price}</p>
              <p><span className="detail-label">Rating:</span>
{product.averageRating} (out of {product.ratingCount} ratings)</p>
              <a href={product.link} className="buy-now-link">Buy Now</a>
            </div>
          ))}
        </div>
      </div>
    </div>
  );
}
```

```
export default Recipe;
```

Recipe.css:

```
.recipe-container {
  background-color:antiquewhite;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
  margin-top: 20px;
}

.recipe-title {
  color: #333;
  font-size: 24px;
  margin-bottom: 10px;
}

.recipe-image {
  width: 100;
  border-radius: 10px;
  margin-bottom: 20px;
  align-items: center;
}

.recipe-details p {
  margin: 5px 0;
}

.detail-label {
  font-weight: bold;
}

.recipe-ingredients {
  margin-bottom: 20px;
}

.ingredient-item {
  list-style-type: disc;
  margin-left: 20px;
}
```

```
.recipe-summary {
  margin-bottom: 20px;
}

.recipe-summary h2 {
  color: #333;
  font-size: 20px;
  margin-bottom: 10px;
}

.recipe-wine-pairing {
  margin-bottom: 20px;
}

.recipe-wine-pairing h2 {
  color: #333;
  font-size: 20px;
  margin-bottom: 10px;
}

.wine-pairing-text {
  margin: 5px 0;
}

.product-match {
  margin-bottom: 20px;
}

.product-match h3 {
  color: #333;
  font-size: 18px;
  margin-bottom: 10px;
}

.product-image {
  width: 100;
  border-radius: 5px;
  margin-bottom: 10px;
}

.buy-now-link {
  color: #007bff;
  text-decoration: none;
```

```
}

.buy-now-link:hover {
  text-decoration: underline;
}
```

Components:

Header.js:

```
import React from "react";
import "../App.css";

export default function Header() {
  return (
    <nav>
      <div className="navbar">
        <ul>
          <li>
            <h1 className="pacifico-regular">Flavour Fusion</h1>
          </li>
        </ul>
      </div>
    </nav>
  );
}
```

Card.js:

```
import { useState } from "react";
// import { Link } from 'react-router-dom';
import "../App.css";
// import Recipe from "../Recipe";

export default function Card({ id, title, image, flag, change}) {

    const [recipe, setRecipe] = useState('');
```

```
    async function sendPOSTmethod({ id, title, image }) {
        try {
            const response = await fetch("http://localhost:3001/save", {
                method: 'POST',
                headers: {
                    'content-type': 'application/json',
                },
                body: JSON.stringify({ id, title, image }),
            });
            const data = await response.json();
            window.alert(data.message);
        } catch (err) {
            console.log(err);
        }
    }

    async function getRecipe(id) {
        try {
            const apiKey = "ae0636d4501d4470af462802baf0528f";
            const response = await
fetch(`https://api.spoonacular.com/recipes/${id}/information?apiKey=${apiKey}`
            , {
                method: 'GET',
                headers: {
                    'content-type': 'application/json',
                },
            });
            const data = await response.json();
            setRecipe(data);
            change(data);

        } catch (err) {
            console.log(err);
        }
    }

    return (
        <div className="card">
            <p><b>Recipe ID : </b>{id}</p>
            <p><b>Recipe Name:</b></p>
            <p>{title}</p>
            <img src={image} alt="Not Found" />
            {
```

```
                flag ? (<button onClick={() => sendPOSTmethod({ id, title, image
})}>Save</button>) :
                (

              <button onClick={() => {getRecipe(id)
                console.log(recipe)}}>Display Recipe</button>

                                )
            }
        </div>
    );
}
```

Body.js:

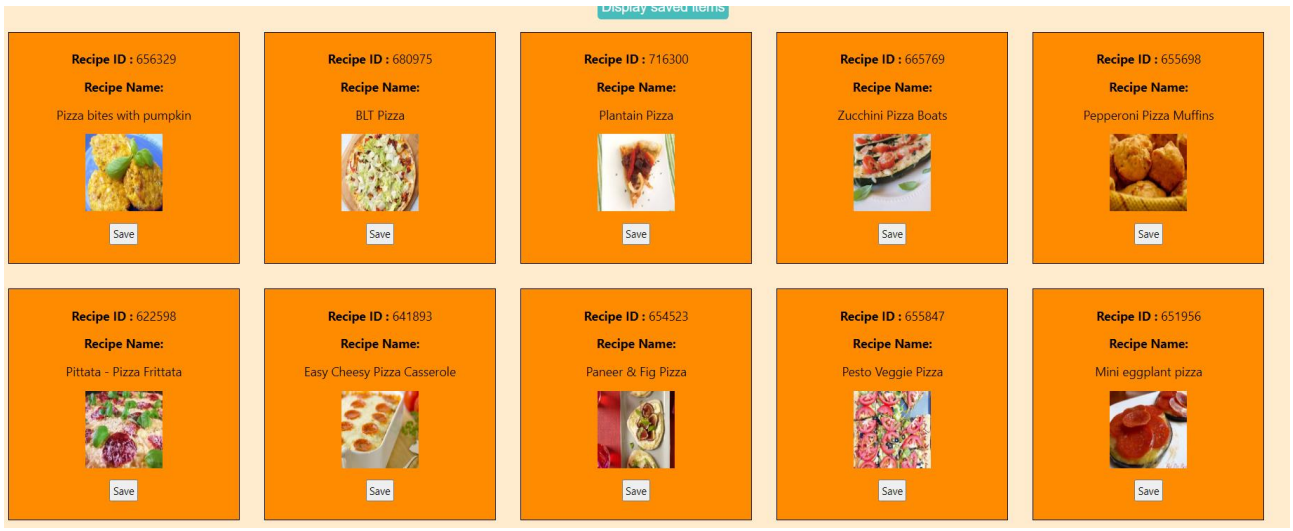
```
import "../App.css"

export default function Body(){
    return(
        <body>
            <h1 className=".concert-one-regular" id="body--heading">😋 Discover
Delicious Recipes for Every Occasion! 💥</h1>
        </body>
    )
}
```

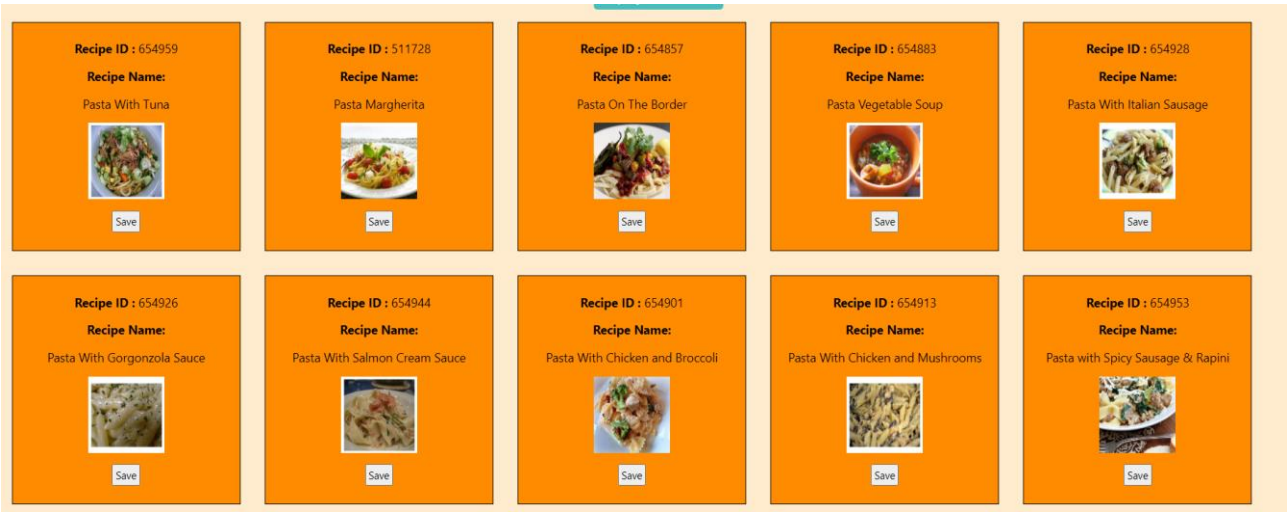
Theory 65, 97, 125
3122 21 5001 125

Output:

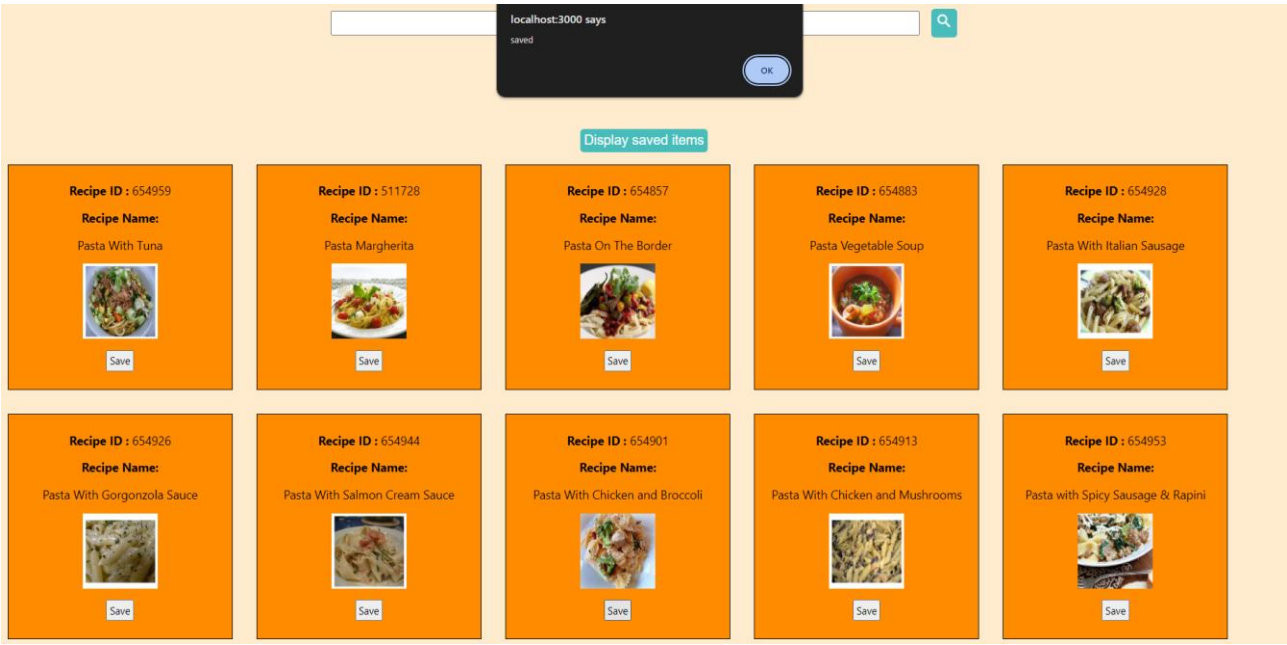
Frontend UI



Pizza Recipes



Pasta Recipes



Saving item Pasta With Chicken and Broccoli

```
_id: ObjectId('663cd8dc24d45af45c0782bc')
id : 654901
title : "Pasta With Chicken and Broccoli"
image : "https://img.spoonacular.com/recipes/654901-312x231.jpg"
```

MongoDB

```
_id: ObjectId('663cd90a24d45af45c0782bd')
id : 511728
title : "Pasta Margherita"
image : "https://img.spoonacular.com/recipes/511728-312x231.jpg"
```

MongoDB

Displaying Saved Items

Theory 65, 97, 125
3122 21 5001 125

Pasta With Chicken and Broccoli

Servings: 4
Ready in Minutes: 45
Source: Foodista
Price Per Serving: $193.38

Ingredients:

- 1/4 cup olive oil
- 1/2 pound boneless chicken breasts cut in ½ in strips
- 1 pinch red pepper flakes
- 1/2 pound bow tie pasta or small lasagna noodles, (cooked)
- 2 garlic cloves, minced
- 1 1/2 cups broccoli florets
- 1 teaspoon dried basil leaves
- 1/4 cup white wine
- 3/4 cup chicken broth
- Grated parmesan cheese

Pasta With Chicken and Broccoli Recipe

Pasta Margherita

Servings: 4
Ready in Minutes: 45
Source: Pick Fresh Foods
Price Per Serving: $274.82

Ingredients:

- ¼ cup fresh basil, thinly sliced
- ¼ cup fresh basil, thinly sliced
- 12 ounces fresh mozzarella cheese, cut into ½-inch cubes
- 1 garlic clove, pressed
- 10 oz grape tomatoes, cut in half lengthwise
- ½ tsp kosher salt
- ¼ cup extra-virgin olive oil
- 1 pound linguine pasta

Summary:

You can never have too many main course recipes, so give Pasta Margheritan a try. One serving contains **809 calories**, **34g of protein**, and **34g of fat**. This recipe serves 4. For **$2.75 per serving**, this recipe **covers 25%** of your daily requirements of vitamins and minerals. It is brought to you by Pick Fresh Foods. 1 person were glad they tried this recipe. A mixture of kosher salt, mozzarella cheese, grape tomatoes, and a handful of other ingredients are all it takes to

Pasta Margheria Recipe

Theory 65, 97, 125
3122 21 5001 125

Contribution:

Parthiban – Displaying Recipe details + Backend (Fetching Recipe Details) + Documentation

Sathvika – Integration of Database + Backend (Saving and Fetching Recipe) + Documentation

Ojus – Frontend (Display Recipes) + Search using API + Documentation