Online Recipe Finder

A Project Report

In the partial fulfillment of the award of the degree of

B. Tech

under

Academy of Skill Development

ASD
Skills For Employment

Submitted by

Soumen Pore,

Souvik Bhaduri, Surajit Pal,

Diptik Mondal

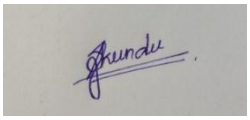
GMIT
it's campuslife

Gargi Memorial Institute of Technology

ASD
Skills For Employment

Certificate from the Mentor

This is to certify that **Soumen Pore** has completed the project titled **DishCovery** under my supervision during the period from **28.10.25** to **20.11.25** which is in partial fulfillment of requirements for the award of the **B. Tech** and submitted to Department **Electronics and Communication Engineering** of **Gargi memorial Institute of Technology.**



_____

Signature of the Mentor

Date:

Acknowledgment

I take this opportunity to express my deep gratitude and sincerest thanks to my project mentor, **Soumili Kundu** for giving the most valuable suggestions, helpful guidance, and encouragement in the execution of this project work.

I would like to give a special mention to my colleagues. Last but not least I am grateful to all the faculty members of the **Academy of Skill Development** for their support.

:: CONTENTS ::

INTRODUCTION

DishCovery was built to make recipe searching faster, easier, and more enjoyable. It offers instant access to global dishes through real-time API data and provides a clean, user-friendly interface for smooth browsing. Overall, it helps users quickly discover new meals and explore diverse cuisines in one simple platform.

The technical stack of **DishCovery** is built entirely on modern front-end technologies, using **HTML**, **CSS**, and **JavaScript** as its core foundation. HTML structures the content, while CSS ensures a clean, responsive, and visually appealing interface across different devices. JavaScript powers the dynamic functionality, enabling interactive features, real-time content updates, and seamless user experience. The project also integrates the **TheMealDB API** using asynchronous JavaScript (Fetch API) to retrieve and display live recipe data, showcasing effective API handling, DOM manipulation, and modular coding practices. This stack makes DishCovery lightweight, fast, and highly efficient for web-based recipe discovery.

DishCovery comes with simple yet helpful features that make exploring recipes feel easy and enjoyable. Users can quickly search for meals, browse by category or region, and view clear cooking instructions with appealing images. The site fetches real-time data from TheMealDB API, so the recipes you see are always fresh and accurate. Behind the scenes, smooth page updates and a responsive layout make everything feel fast and comfortable to use. Altogether, these features create a friendly, interactive space where anyone can discover and enjoy new dishes.

DishCovery includes friendly and easy-to-use features that make discovering recipes a fun experience. You can search for meals, browse by category or region, and check step-by-step instructions with clear, attractive images. The website pulls live data from TheMealDB API, so the recipes always feel fresh and reliable. Thanks to its smooth interactions and responsive design, the site feels fast, simple, and comfortable to use. Altogether, these features create a welcoming space where anyone can explore new dishes and enjoy cooking more.

DishCovery offers a warm and simple way for people to find and enjoy new recipes without any hassle. In the end, it creates a friendly space that encourages users to explore, cook, and enjoy food in a more inspiring way.

OBJECTIVE

The primary objective of **DishCovery** is to design and develop an interactive, user-friendly, and visually appealing recipe finder web application that allows users to explore a wide range of meals from around the world. The website aims to simplify the process of discovering and learning new recipes by integrating real-time data from **TheMealDB API**, while also showcasing practical implementation of **front-end web development skills** such as asynchronous JavaScript, DOM manipulation, and responsive design.

Core Objectives:

- To create a **dynamic web application** that fetches and displays meal information using **TheMealDB API**.
- To provide users with a **simple and responsive interface** for searching and browsing recipes by name, category, or region.
- To display **detailed recipe information**, including ingredients, measurements, and preparation steps, with corresponding meal images.
- To enhance user engagement through **interactive design** and seamless navigation.
- To implement **modern web technologies** (HTML, CSS, and JavaScript) effectively for a professional and efficient user experience.
- To demonstrate the ability to **integrate third-party APIs** and manage data dynamically using asynchronous programming.

In summary, the objective of **DishCovery** is not only to deliver a functional and attractive recipe platform but also to exhibit strong technical understanding of client-side web development and real-world API integration.

SCOPE

DishCovery is a web-based recipe discovery platform that allows users to explore and learn about various cuisines using real-time data from TheMealDB API. The project focuses on building an interactive and responsive front-end with HTML, CSS, and JavaScript, enabling users to search for recipes and view details like ingredients, instructions, and images. The current scope is limited to data retrieval and display, excluding advanced features like user accounts or personalized recommendations.

- **Key Functional Area:**
  1. **User Registration & Login :**
     Secure registration and authentication process for user.
  2. **Recipe Finder:**
     It allows users to search and explore a wide range of meals by name. After giving the input, user will get food image, category, region, cooking time, ingredient , cooking steps and You-tube tutorial video link(If available).
  3. **Cooking Timer:**
     A pop up clock for user where user can input a time. After the time is over user will get another pop up about "Time is Over" and a pop up sound.
  4. **Dietary Profile:**
     This section lets user select a diet type, note allergies or restrictions, and set daily nutrition goals for calories, protein, carbs and fat. Saving the profile enables personalized meal and recommendation and suggestions based on these choices.
  5. **Nutritional Breakdown:**
     This section summarizes a meal's macronutrient composition per serving, showing how balanced it is for protein, carbs and fat. This section will help users quickly judge if a meal aligns with their goals, plan portions and adjust the rest of the day's intake without manual calculation.
  6. **Food Recommendation:**
     This section suggests popular or related dishes to help users discover new meals and enhance their overall recipe exploration experience.

3. SYSTEM ANALYSIS

The **DishCovery** system is a client-side web application that fetches real-time recipe data from **TheMealDB** API and displays it through a responsive interface built with HTML, CSS, and JavaScript. User searches and selections trigger API requests, and the returned data is dynamically shown on the screen. With no backend of its own, the system depends on internet connectivity and focuses on fast performance, simple navigation, and a clean user experience for easy recipe discovery.

3.A IDENTIFICATION OF NEEDS

There is a need for a simple and quick way to find reliable recipes without navigating complex cooking platforms. **DishCovery** fulfils this by offering an easy-to-use interface that provides real-time recipe data, helping users explore dishes, view ingredients, and follow instructions effortlessly.

Key Needs Identified :

- A simple and user-friendly platform for quick recipe discovery.
- Easy access to accurate and real-time recipe information.
- A clean interface to explore ingredients and cooking steps effortlessly.
- A fast and responsive system that works smoothly across devices.
- A convenient way to discover new dishes without navigating complex website.

3.B EASIBILITY STUDY

A feasibility study evaluates whether the **DishCovery** project can be successfully developed, implemented, and operated with available resources. It forces on technical, economic, and operational aspects to ensure the system is practical and achievable.

- ### Technical Feasibility:

  DishCovery is technically feasible as it uses basic web technologies(HTML, CSS, JavaScript) and TheMealDB API, all of which are simple to integrate and require no backend setup. The system works smoothly on any modern browser with minimal technical constraints.

- **Economic Feasibility:**

  The project is cost-effective because it relies on free tools, free hosting (GitHub pages), and free API access. With no server costs or paid resources needed, the financial requirements are extremely low.

- **Operational Feasibility:**

  The System is easy to use, requiring no training or technical knowledge from users. Its clean interface and simple navigation ensure smooth daily operation, making it practical for regular use.

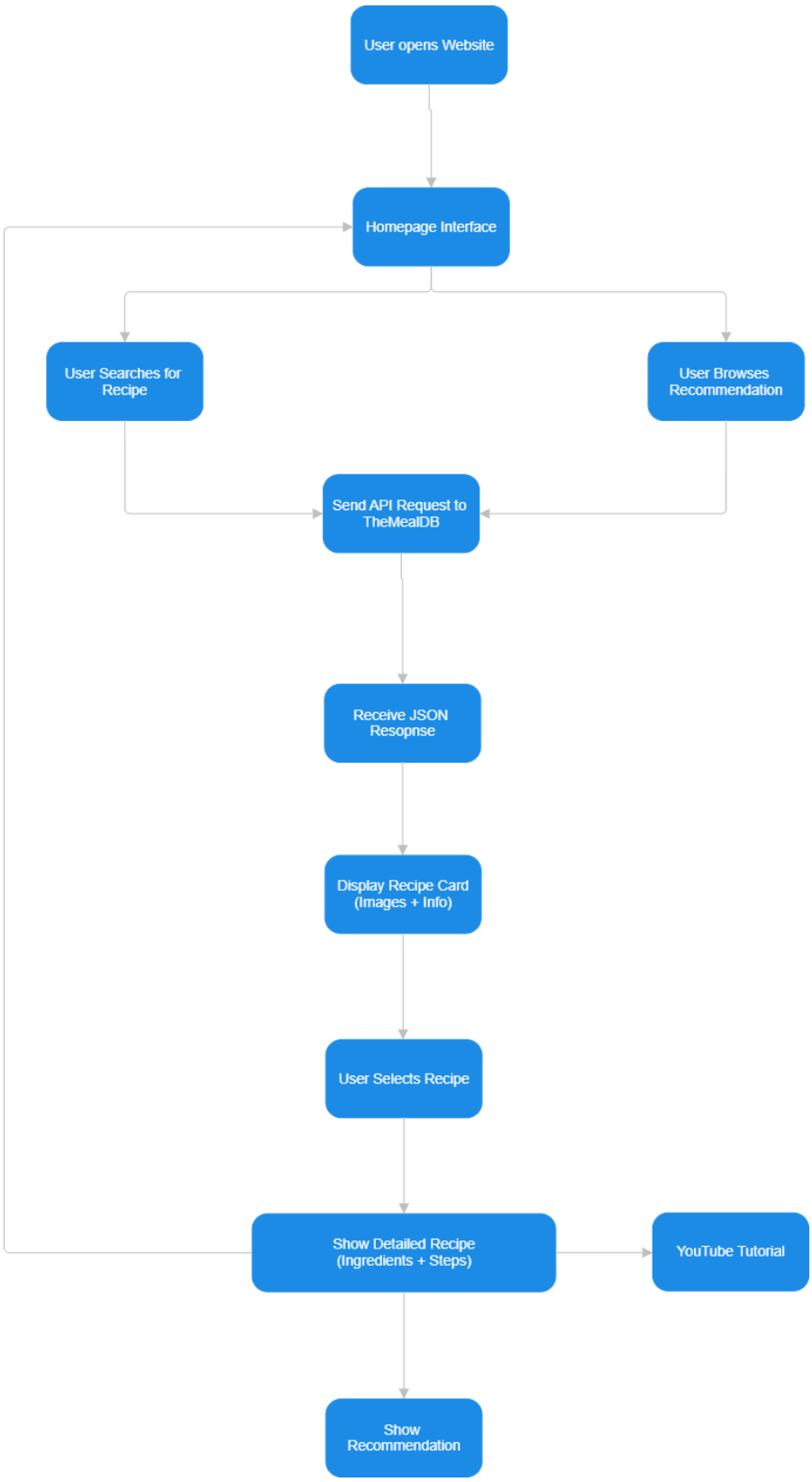
WORKFLOW

The workflow of **Dishcovery** outlines how users interact with the perform and how the system processes recipe data to deliver a smooth and efficient browsing experience.

- **User Launches the Website –** The homepage loads in the browser with all interface elements ready.
- **User Enters Search and Browses categories –** Users search by name or select from **Recommendation** category.
- **System Sends API Request – DishCovery** sends a request to **TheMealDB API** based on the user's input.
- **API Returns Recipe Data --** The API responds with matching recipe in JSON format.
- **System Processes and displays Results –** Recipe cards, images, and information are dynamically shown on the page.
- **User Selects a Recipe --** The user clicks a recipe to view more deails.
- **Detailed Recipe Page Loads –** Ingredients, instructions, and meal images are displayed.
- **User Views Recommendation –** Similar or related dishes may appear for further exploration.
- **User Continues Browsing and searches Again –** The user can explore more recipes or start a new search.

- **Advantages :**
  1. Easy-to-use interface with real-time recipe data.
  2. Fast, lightweight, and works on any device.
- **Disadvantages :**
  1. Requires internet connection to fetch recipes.
  2. Limited to TheMealDB API data.
- **Application :**
  1. Quick Recipe search for users and beginners.
  2. Useful for exploring new dishes and ingrdients.
  3. Ideal as a web development learning.

Workflow Diagram:



.

STUDY OF THE SYSTEM

A system study helps analyse how different components of an application function and support user needs. It provides a detailed understanding of core features, their purpose, and how users interact with them. The following study covers key modules commonly found in a recipe or food-related platform.

1. **Registration :**
   The registration module allows new users to create an account by entering basic details such as name, email, and password. It ensures secure user onboarding and enables personalized features in the system.
2. **Login:**
   The login module verifies user identity using stored credentials and provides secure access to the platform. It maintains user privacy and supports role-based access if required.
3. **User Interface (UI):**
   The UI module focuses on providing a clean, responsive, and user-friendly design. It ensures smooth navigation, clear visual elements, and easy interaction with search, recipe viewing, and other features.
4. **Dietary Profile:**
   He dietary profile module allows users to specify their food preferences, restrictions, or allergies. It helps the system understand user needs and offer more relevant recipe suggestions.
5. **Nutrition Customization:**
   This module enables users to set nutritional goals or adjust recipes based on calorie limits, macros, or health preferences. It supports personalized meal planning and improves the system's health-focused functionally.

INPUTS AND OUTPUTS

The DishCovery system processes various user action and converts them into meaningful recipe data through real-time API interaction. The following inputs and outputs describe how users interact with the platform and what information the system provides in return.

- **Inputs :**
  1. **Search Input:**
     - User types dish name, keyword, or ingredient into search bar.
     - The system uses this input to retrieve matching recipes from TheMealDB API.
  2. **Recipe Selection:**
     - User clicks on a displayed recipe card.
     - This triggers the system to load detailed information for the selected meal.
  3. **Navigation Actions:**
     - User interacts with navigation options(Home, Recipe Finder, Recommendation, Dietary Profile etc.).
     - The system updates the displayed content accordingly.
  4. **Dietary Profile:**
     - User selects option like vegetarian, vegan etc.
     - User inputs allergens such as nuts, egg, seafood etc.
     - User specifies restrictions.
     - User sets daily/ per-meal nutrition goal.
- **Outputs :**
  1. **Search Results Display:**
     - The system shows recipe cards containing images, meal names, and short description.
     - Results are based on API.
  2. **Detailed Recipe Information:**
     - When user selects recipe, the system outputs.
       - Ingredients and measurements.
       - Step-by-step instructions.
       - Meal Image.
       - Category, region and additional info.

        - Estimated Nutrition(per meal).
    - Provides complete guidance for preparing the dish with YouTube tutorial video also.
3. **Error or No-Result Message:**
    - If no recipe matches the search input, the system outputs a message such as “No results found.”
    - Helps users understand when data is unavailable.
4. **Recommendation Recipes:**
    - The system displays related or suggested dishes to encourage exploration.
    - Enhances user experience and keeps engagement high.
5. **Estimated Nutrition:**
    - The system analyse the food item and give estimated nutrition per meal about calories, protein, carbs and fat.

SYSTEM DESIGN

Effective system design is critical for delivering a seamless user experience and ensuring reliable functionality. The design phase in **DishCovery** focuses on structuring interactions between system users and the services offered by the platform. The use case diagram provides a visual representation of how various users interact with different modules of the system.

- **Actors :** User – the primary actor interacting with the website.
- **System - DishCovery** (Recipe Finder Website)

USE CASES

- **UC1: Search Recipes**
  - **Actor:** User.
  - **Description:** User enters food Item name to find recipe.
  - **Precondition:** User is on **Recipe Finder** page.
  - **Postcondition:** Matching recipes are displayed.
- **UC2: View Recipe Details**
  - **Actor:** User.
  - **Description:** User selects a recipe to view ingredients, instructions, and images.
  - **Precondition:** Recipe list or recommendation section is visible.
  - **Postcondition:** Full Recipe page is shown.
- **UC3: View Recommendations**
  - **Actor:** User.
  - **Description:** System automatically displays related or popular dishes to encourage further exploration.
  - **Precondition:** User is viewing a specific recipe or home screen.
  - **Postcondition:** Suggested Meals are shown.
- **UC4: Apply Dietary Preferences**
  - **Actor:** User.
  - **Description:** User selects diet type(veg, low-calorie, etc.) to filter recipes.
  - **Postcondition:** Filtered recipe appear.

DESCRIPTION OF KEY INTERACTION

- **User Searches for a Recipe:**

  The user enters a keyword (e.g. “Chicken”) into search bar. The system sends a request to TheMealDB API and displays all matching meals with images and short descriptions. The user can click any meal to see full detail.

- **User Views Full Recipe Details**

  After selecting a meal, the system displays ingredients, step-by-step instructions, cooking method and YouTube tutorial link (If available). The user can return back or view additional suggestions.

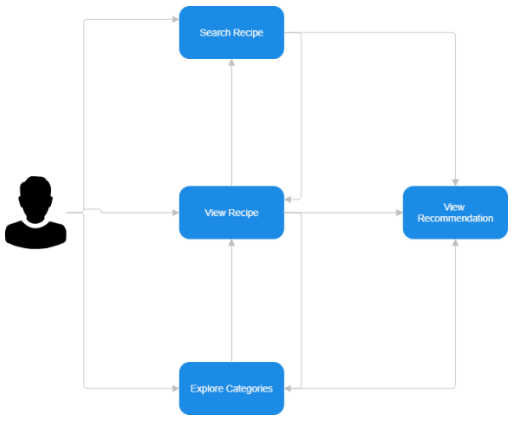
- **User Receives Recommendation**

  While viewing any dish, the system automatically displays other meals that are similar or commonly explored. This helps users discover new dishes quickly.

- **User Applies Dietary or Nutrition Filters**

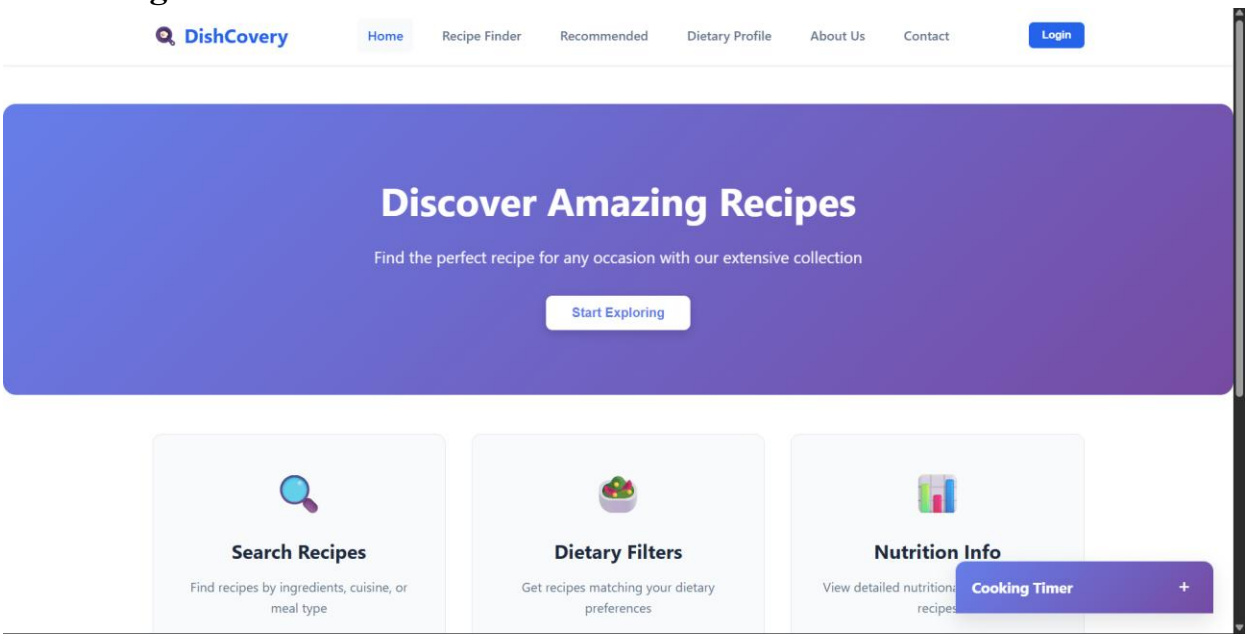
  The user sets a dietary preference or nutrition criteria. The system processes these filters and displays only recipes that match these requirements.

USE CASE DIAGRAM

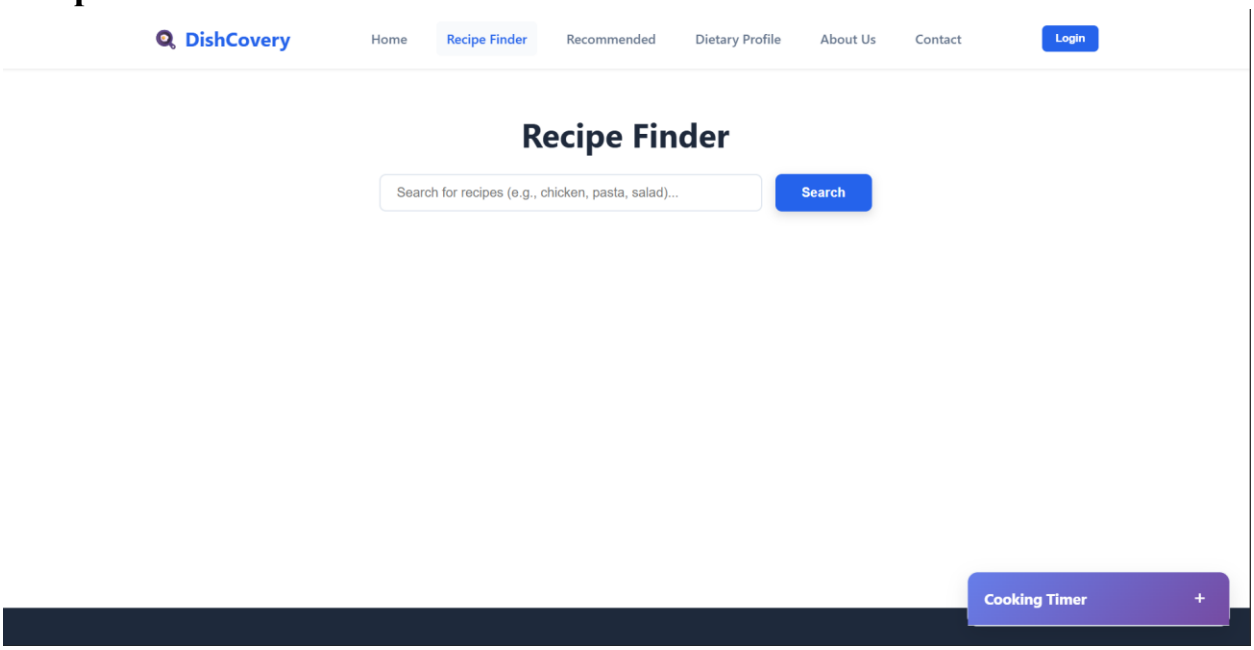


UI SNAPSHOTS:

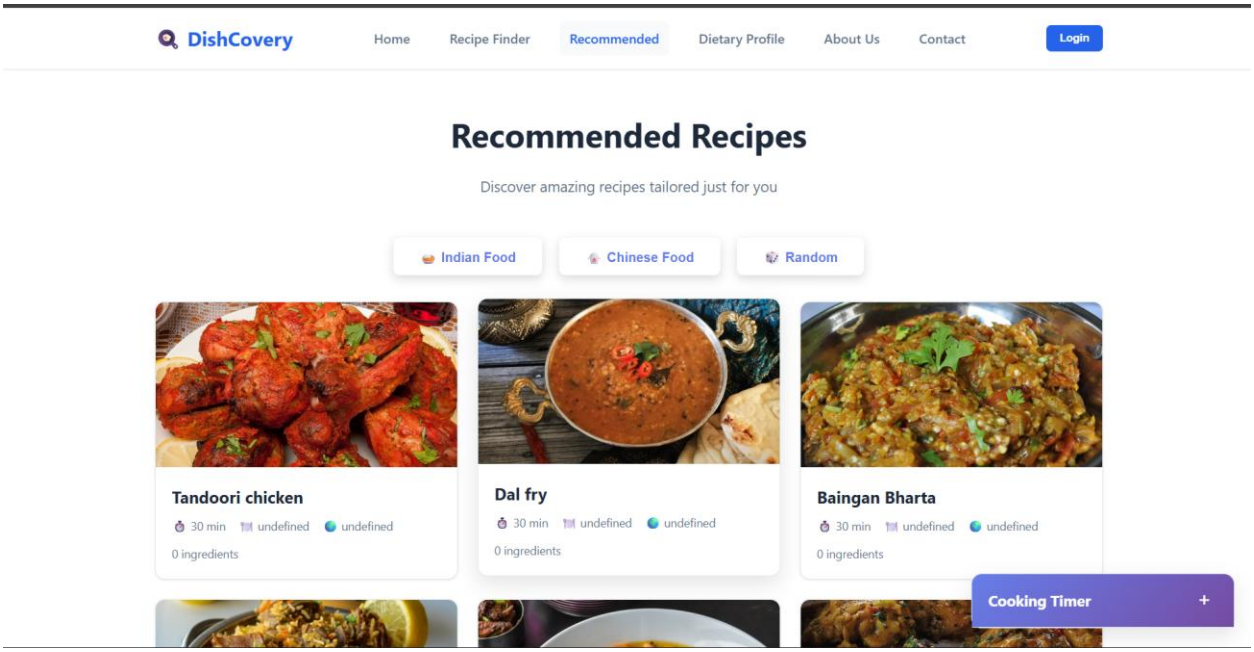
❖ Home Page



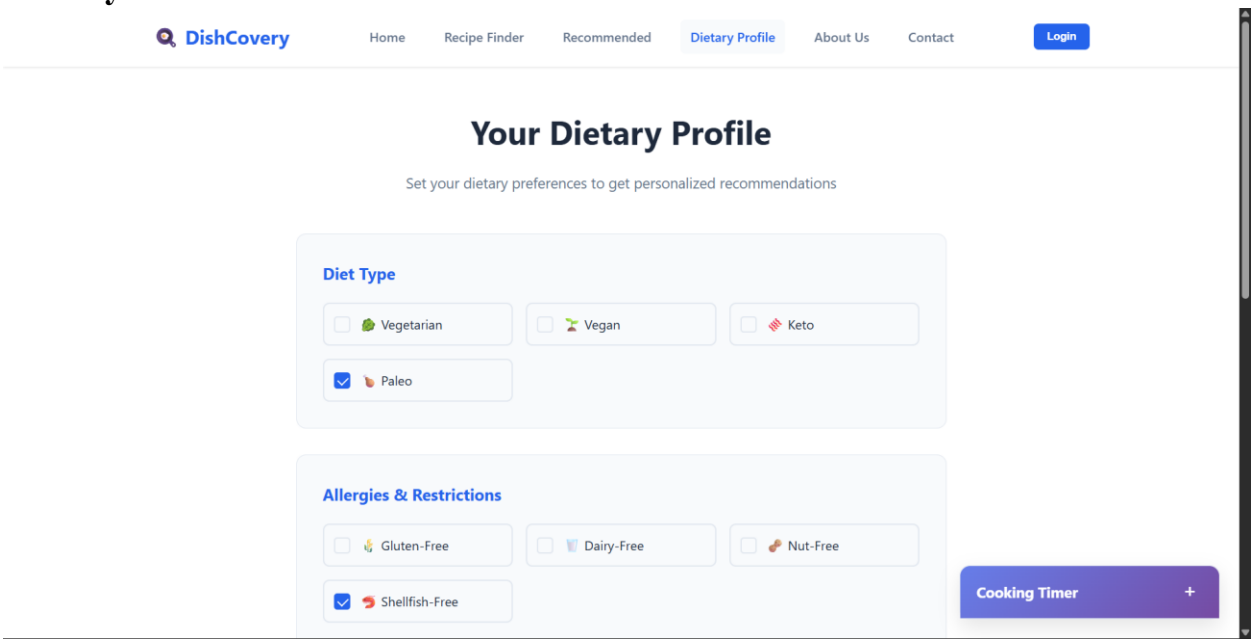
❖ Recipe Finder



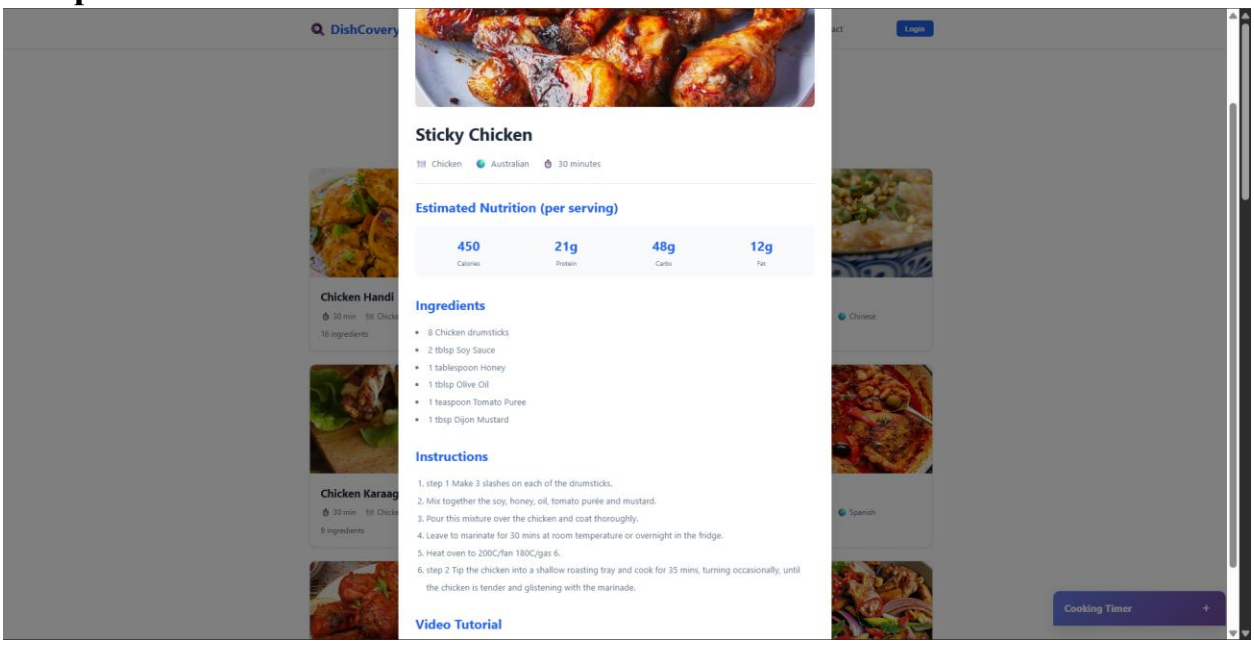
❖ Recommended



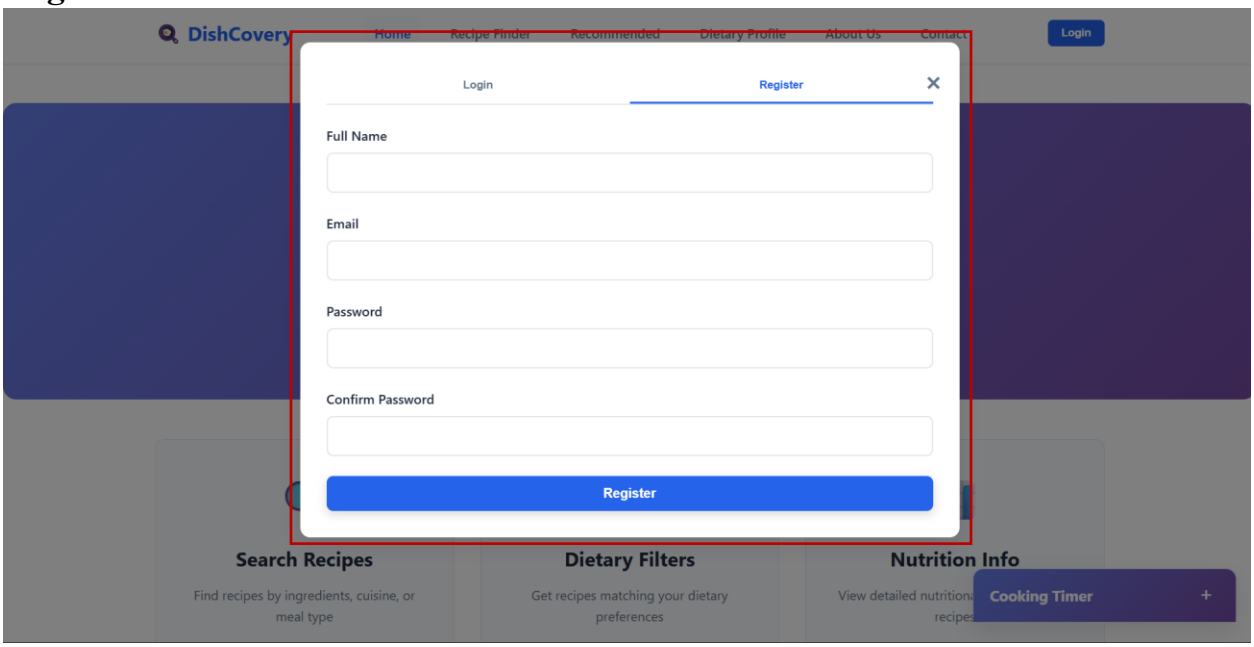
❖ Dietary Profile



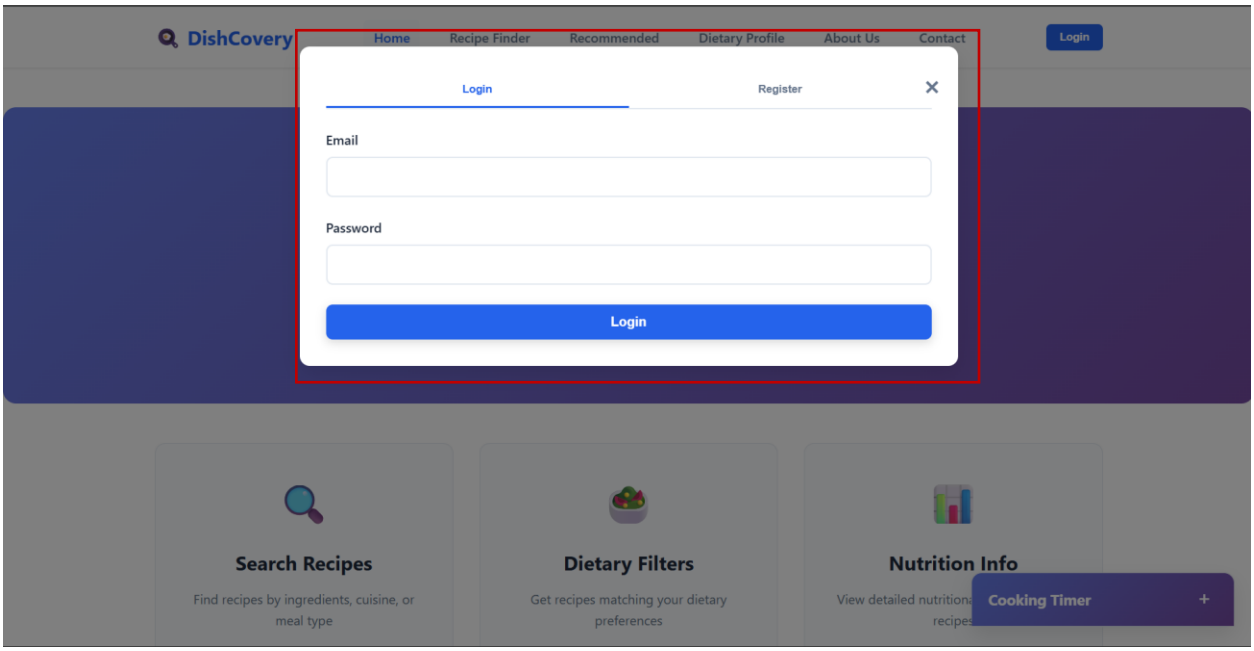
❖ Recipe



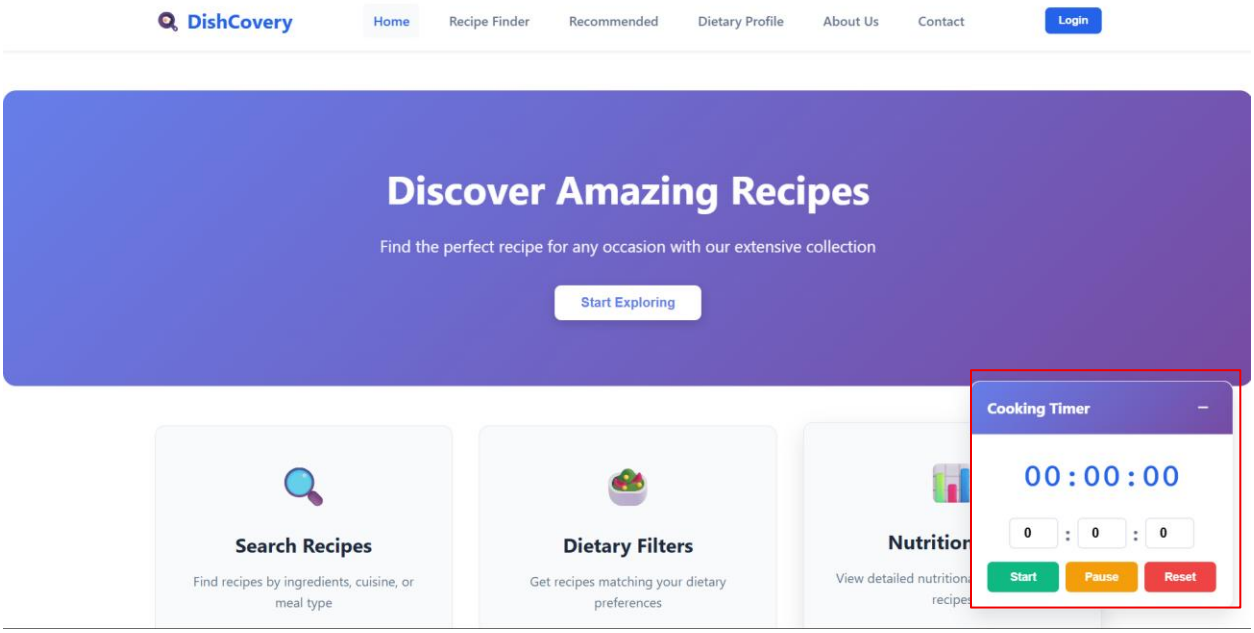
❖ Register



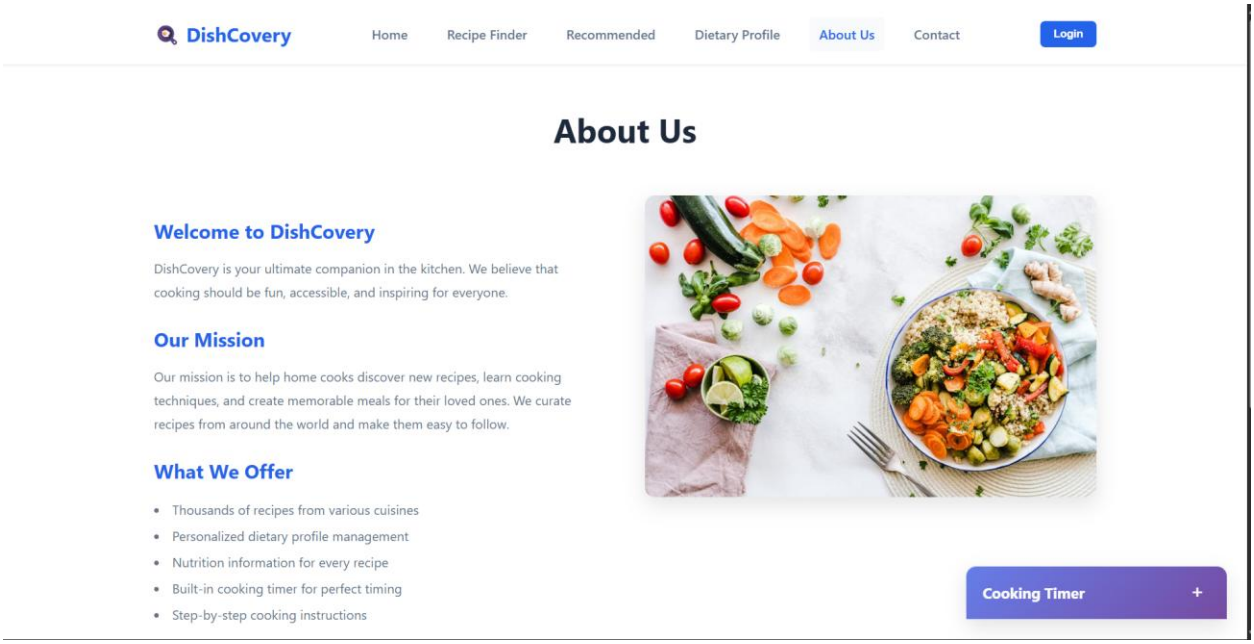
❖ Login



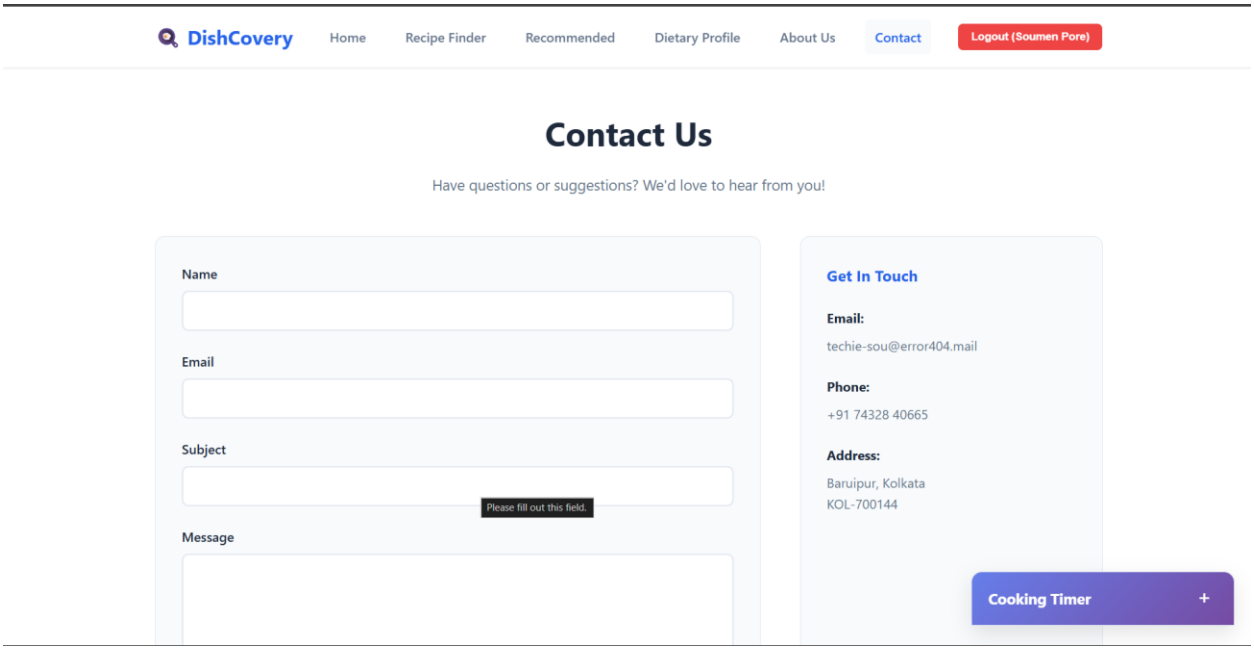
❖ Cooking Timer



❖ About



❖ Contact



Source Code

❖ Home Page

```
<main id="main-content">
    <section id="home" class="page active">
     <div class="hero">
      <div class="container">
       <h2 class="hero-title">Discover Amazing Recipes</h2>
       <p class="hero-subtitle">Find the perfect recipe for any occasion with our extensive collection</p>
       <button class="btn-primary" onclick="navigateTo('finder')">Start Exploring</button>
      </div>
     </div>

     <div class="container">
      <div class="features">
       <div class="feature-card">
        <div class="feature-icon">🔍</div>
        <h3>Search Recipes</h3>
        <p>Find recipes by ingredients, cuisine, or meal type</p>
       </div>
       <div class="feature-card">
        <div class="feature-icon">🥗</div>
        <h3>Dietary Filters</h3>
        <p>Get recipes matching your dietary preferences</p>
       </div>
       <div class="feature-card">
        <div class="feature-icon">📊</div>
        <h3>Nutrition Info</h3>
        <p>View detailed nutritional information for recipes</p>
       </div>
      </div>
     </div>
    </section>
```

❖ Recipe Finder

```
<section id="finder" class="page">
      <div class="container">
       <h2 class="page-title">Recipe Finder</h2>

       <div class="search-section">
        <input type="text" id="search-input" class="search-input" placeholder="Search for recipes (e.g., chicken, pasta, salad)..." />
        <button class="btn-primary" onclick="searchRecipes()">Search</button>
       </div>

       <div id="loading" class="loading" style="display: none;">
        <div class="spinner"></div>
        <p>Loading recipes...</p>
       </div>

       <div id="recipes-grid" class="recipes-grid"></div>
      </div>
     </section>
```

❖ Recommended

```
<section id="recommended" class="page">
```

```
    <div class="container">
      <h2 class="page-title">Recommended Recipes</h2>
      <p class="page-subtitle">Discover amazing recipes tailored just for you</p>

      <div class="cuisine-buttons" style="text-align: center; margin-bottom: 32px;">
        <button class="btn-primary" onclick="loadCuisineRecommendations('Indian')" style="margin: 0 8px;">🍛 Indian Food</button>
        <button class="btn-primary" onclick="loadCuisineRecommendations('Chinese')" style="margin: 0 8px;">🥡 Chinese Food</button>
        <button class="btn-primary" onclick="loadRecommendedRecipes()" style="margin: 0 8px;">🎲 Random</button>
      </div>

      <div id="recommended-grid" class="recipes-grid"></div>
    </div>
  </section>
```

❖ Dietary Profile

```
<section id="dietary" class="page">
    <div class="container">
      <h2 class="page-title">Your Dietary Profile</h2>
      <p class="page-subtitle">Set your dietary preferences to get personalized recommendations</p>

      <div class="dietary-container">
        <div class="dietary-section">
          <h3>Diet Type</h3>
          <div class="diet-options">
            <label class="diet-checkbox">
              <input type="checkbox" name="diet" value="vegetarian" onchange="updateDietaryProfile()">
              <span class="checkmark"></span>
              🥦 Vegetarian
            </label>
            <label class="diet-checkbox">
              <input type="checkbox" name="diet" value="vegan" onchange="updateDietaryProfile()">
              <span class="checkmark"></span>
              🌱 Vegan
            </label>
            <label class="diet-checkbox">
              <input type="checkbox" name="diet" value="keto" onchange="updateDietaryProfile()">
              <span class="checkmark"></span>
              🥓 Keto
            </label>
            <label class="diet-checkbox">
              <input type="checkbox" name="diet" value="paleo" onchange="updateDietaryProfile()">
              <span class="checkmark"></span>
              🍗 Paleo
            </label>
          </div>
        </div>

        <div class="dietary-section">
          <h3>Allergies & Restrictions</h3>
          <div class="allergy-options">
            <label class="diet-checkbox">
              <input type="checkbox" name="allergy" value="gluten-free" onchange="updateDietaryProfile()">
              <span class="checkmark"></span>
              🌾 Gluten-Free
            </label>
            <label class="diet-checkbox">
              <input type="checkbox" name="allergy" value="dairy-free" onchange="updateDietaryProfile()">
              <span class="checkmark"></span>
              🥛 Dairy-Free
            </label>
            <label class="diet-checkbox">
```

```
                <input type="checkbox" name="allergy" value="nut-free" onchange="updateDietaryProfile()">
                <span class="checkmark"></span>
                🥜 Nut-Free
              </label>
              <label class="diet-checkbox">
                <input type="checkbox" name="allergy" value="shellfish-free" onchange="updateDietaryProfile()">
                <span class="checkmark"></span>
                🦐 Shellfish-Free
              </label>
            </div>
          </div>

          <div class="dietary-section">
            <h3>Nutrition Goals</h3>
            <div class="nutrition-goals">
              <div class="goal-item">
                <label for="calorie-target">Daily Calorie Target:</label>
                <input type="number" id="calorie-target" min="1000" max="3500" value="2000" onchange="updateDietaryProfile()">
              </div>
              <div class="goal-item">
                <label for="protein-target">Protein Goal (g):</label>
                <input type="number" id="protein-target" min="0" max="300" value="50" onchange="updateDietaryProfile()">
              </div>
              <div class="goal-item">
                <label for="carbs-target">Carbs Goal (g):</label>
                <input type="number" id="carbs-target" min="0" max="500" value="200" onchange="updateDietaryProfile()">
              </div>
              <div class="goal-item">
                <label for="fat-target">Fat Goal (g):</label>
                <input type="number" id="fat-target" min="0" max="200" value="65" onchange="updateDietaryProfile()">
              </div>
            </div>
          </div>

          <button class="btn-primary btn-save" onclick="saveDietaryProfile()">Save Profile</button>
          <div id="profile-message" class="profile-message"></div>
        </div>
      </div>
    </section>
```

❖ Recipe Finder

```
function displayRecipes(recipes, containerId) {
  const container = document.getElementById(containerId);
  container.innerHTML = '';

  recipes.forEach(recipe => {
    const card = document.createElement('div');
    card.className = 'recipe-card';
    card.onclick = () => showRecipeDetails(recipe);

    const ingredients = getIngredients(recipe);
    const time = '30 min';

    card.innerHTML = `
      <img src="${recipe.strMealThumb}" alt="${recipe.strMeal}" class="recipe-image" />
      <div class="recipe-content">
        <h3 class="recipe-title">${recipe.strMeal}</h3>
        <div class="recipe-meta">
          <span> ⏱ ${time}</span>
```

```
        <span>🍽️ ${recipe.strCategory}</span>
        <span>🌍 ${recipe.strArea}</span>
      </div>
      <p class="recipe-description">${ingredients.length} ingredients</p>
    </div>
  `;

  container.appendChild(card);
 });
}

function getIngredients(recipe) {
 const ingredients = [];
 for (let i = 1; i <= 20; i++) {
  const ingredient = recipe[`strIngredient${i}`];
  const measure = recipe[`strMeasure${i}`];
  if (ingredient && ingredient.trim()) {
   ingredients.push({ ingredient, measure });
  }
 }
 return ingredients;
}

// <CHANGE> Updated to show nutrition info in modal
function showRecipeDetails(recipe) {
 const modal = document.getElementById('recipe-modal');
 const modalBody = document.getElementById('modal-body');

 const ingredients = getIngredients(recipe);

 const ingredientsList = ingredients.map(item =>
  `<li>${item.measure} ${item.ingredient}</li>`
 ).join('');

 const instructions = recipe.strInstructions
  .split('.')
  .filter(step => step.trim().length > 0)
  .map((step, index) => `<li>${step.trim()}.</li>`)
  .join('');

 const profile = JSON.parse(localStorage.getItem('dishcovery_profile') || 'null');
 const estimatedNutrition = calculateEstimatedNutrition(recipe, ingredients);

 modalBody.innerHTML = `
  <img src="${recipe.strMealThumb}" alt="${recipe.strMeal}" class="modal-recipe-image" />
  <h2 class="modal-recipe-title">${recipe.strMeal}</h2>
  <div class="modal-recipe-meta">
   <div class="meta-item">
    <span>🍽️</span>
    <span>${recipe.strCategory}</span>
   </div>
   <div class="meta-item">
    <span>🌍</span>
    <span>${recipe.strArea}</span>
   </div>
   <div class="meta-item">
    <span>⏱️</span>
    <span>30 minutes</span>
   </div>
  </div>

  <div class="modal-section">
   <h3>Estimated Nutrition (per serving)</h3>
   <div class="nutrition-info">
```

```
        <div class="nutrition-item">
          <strong>${estimatedNutrition.calories}</strong>
          <span>Calories</span>
        </div>
        <div class="nutrition-item">
          <strong>${estimatedNutrition.protein}g</strong>
          <span>Protein</span>
        </div>
        <div class="nutrition-item">
          <strong>${estimatedNutrition.carbs}g</strong>
          <span>Carbs</span>
        </div>
        <div class="nutrition-item">
          <strong>${estimatedNutrition.fat}g</strong>
          <span>Fat</span>
        </div>
      </div>
    </div>

    <div class="modal-section">
      <h3>Ingredients</h3>
      <ul>${ingredientsList}</ul>
    </div>

    <div class="modal-section">
      <h3>Instructions</h3>
      <ol>${instructions}</ol>
    </div>

    ${recipe.strYoutube ? `
      <div class="modal-section">
        <h3>Video Tutorial</h3>
        <p><a href="${recipe.strYoutube}" target="_blank" style="color: var(--primary-color);">Watch on YouTube</a></p>
      </div>
    ` : ''}
  `;

  modal.classList.add('active');
}

// <CHANGE> Added estimated nutrition calculation
function calculateEstimatedNutrition(recipe, ingredients) {
  const servings = 4;
  return {
    calories: Math.round(recipe.strIngredient1 ? 450 : 350),
    protein: Math.round(ingredients.length * 3.5),
    carbs: Math.round(ingredients.length * 8),
    fat: Math.round(ingredients.length * 2),
  };
}

function closeModal() {
  const modal = document.getElementById('recipe-modal');
  modal.classList.remove('active');
}

window.onclick = function(event) {
  const modal = document.getElementById('recipe-modal');
  if (event.target === modal) {
    closeModal();
  }

  const authModal = document.getElementById('auth-modal');
  if (event.target === authModal) {
    closeAuthModal();
```

```
  }
};
```

❖ Register Page

- ✓ Register – App.js

```
function handleRegister(e) {
  e.preventDefault();
  const name = document.getElementById('register-name').value;
  const email = document.getElementById('register-email').value;
  const password = document.getElementById('register-password').value;
  const confirm = document.getElementById('register-confirm').value;

  if (password !== confirm) {
    alert('Passwords do not match');
    return;
  }

  const users = JSON.parse(localStorage.getItem('dishcovery_users') || '[]');
  if (users.find(u => u.email === email)) {
    alert('Email already registered');
    return;
  }

  users.push({ name, email, password });
  localStorage.setItem('dishcovery_users', JSON.stringify(users));

  currentUser = { name, email };
  localStorage.setItem('dishcovery_user', JSON.stringify(currentUser));
  updateAuthButton();
  closeAuthModal();
  alert('Registration successful!');
}
```

- ✓ Register – Index.html

```
<form id="register-form" class="auth-form" onsubmit="handleRegister(event)">
        <div class="form-group">
          <label for="register-name">Full Name</label>
          <input type="text" id="register-name" required />
        </div>
        <div class="form-group">
          <label for="register-email">Email</label>
          <input type="email" id="register-email" required />
        </div>
        <div class="form-group">
          <label for="register-password">Password</label>
          <input type="password" id="register-password" required />
        </div>
        <div class="form-group">
          <label for="register-confirm">Confirm Password</label>
          <input type="password" id="register-confirm" required />
        </div>
        <button type="submit" class="btn-primary btn-full">Register</button>
      </form>
```

❖ Login Page

- ✓ Login – App.js

```
function handleLogin(e) {
  e.preventDefault();
  const email = document.getElementById('login-email').value;
  const password = document.getElementById('login-password').value;

  const users = JSON.parse(localStorage.getItem('dishcovery_users') || '[]');
  const user = users.find(u => u.email === email && u.password === password);

  if (user) {
    currentUser = { name: user.name, email: user.email };
    localStorage.setItem('dishcovery_user', JSON.stringify(currentUser));
    updateAuthButton();
    closeAuthModal();
    alert('Login successful!');
  } else {
    alert('Invalid email or password');
  }
}
```

- ✓ Login – Index.html

```
<form id="login-form" class="auth-form active" onsubmit="handleLogin(event)">
        <div class="form-group">
          <label for="login-email">Email</label>
          <input type="email" id="login-email" required />
        </div>
        <div class="form-group">
          <label for="login-password">Password</label>
          <input type="password" id="login-password" required />
        </div>
        <button type="submit" class="btn-primary btn-full">Login</button>
      </form>
```

❖ Cooking Timer

```
function toggleTimer() {
  const widget = document.getElementById('timer-widget');
  const toggleBtn = document.querySelector('.timer-toggle');

  widget.classList.toggle('collapsed');
  toggleBtn.textContent = widget.classList.contains('collapsed') ? '+' : '−';
}

function startTimer() {
  if (isTimerRunning) return;

  const hours = parseInt(document.getElementById('input-hours').value) || 0;
  const minutes = parseInt(document.getElementById('input-minutes').value) || 0;
  const seconds = parseInt(document.getElementById('input-seconds').value) || 0;


  timeRemaining = hours * 3600 + minutes * 60 + seconds;

  if (timeRemaining <= 0) {
    alert('Please set a time greater than 0');
    return;
  }

  isTimerRunning = true;
```

```
  timerInterval = setInterval(() => {
    if (timeRemaining > 0) {
      timeRemaining--;
      updateTimerDisplay();
    } else {
      pauseTimer();
      playTimerAlert();
      alert('Time is up!');
    }
  }, 1000);
}

function pauseTimer() {
  isTimerRunning = false;
  if (timerInterval) {
    clearInterval(timerInterval);
    timerInterval = null;
  }
}

function resetTimer() {
  pauseTimer();
  timeRemaining = 0;
  updateTimerDisplay();
  document.getElementById('input-hours').value = 0;
  document.getElementById('input-minutes').value = 0;
  document.getElementById('input-seconds').value = 0;
}

function updateTimerDisplay() {
  const hours = Math.floor(timeRemaining / 3600);
  const minutes = Math.floor((timeRemaining % 3600) / 60);
  const seconds = timeRemaining % 60;

  document.getElementById('timer-hours').textContent = String(hours).padStart(2, '0');
  document.getElementById('timer-minutes').textContent = String(minutes).padStart(2, '0');
  document.getElementById('timer-seconds').textContent = String(seconds).padStart(2, '0');
}

function playTimerAlert() {
  const context = new (window.AudioContext || window.webkitAudioContext)();
  const oscillator = context.createOscillator();
  const gainNode = context.createGain();

  oscillator.connect(gainNode);
  gainNode.connect(context.destination);

  oscillator.frequency.value = 800;
  oscillator.type = 'sine';

  gainNode.gain.setValueAtTime(0.3, context.currentTime);
  gainNode.gain.exponentialRampToValueAtTime(0.01, context.currentTime + 0.5);

  oscillator.start(context.currentTime);
  oscillator.stop(context.currentTime + 0.5);
}
```

❖ About Page

```
<section id="about" class="page">
      <div class="container">
        <h2 class="page-title">About Us</h2>

        <div class="about-content">
          <div class="about-text">
            <h3>Welcome to DishCovery</h3>
            <p>DishCovery is your ultimate companion in the kitchen. We believe that cooking should be fun, accessible, and inspiring for everyone.</p>

            <h3>Our Mission</h3>
            <p>Our mission is to help home cooks discover new recipes, learn cooking techniques, and create memorable meals for their loved ones. We curate recipes from around the world and make them easy to follow.</p>

            <h3>What We Offer</h3>
            <ul>
              <li>Thousands of recipes from various cuisines</li>
              <li>Personalized dietary profile management</li>
              <li>Nutrition information for every recipe</li>
              <li>Built-in cooking timer for perfect timing</li>
              <li>Step-by-step cooking instructions</li>
            </ul>
          </div>

          <div class="about-image">
              <img  src="https://images.pexels.com/photos/1640777/pexels-photo-1640777.jpeg?auto=compress&cs=tinysrgb&w=800" alt="Cooking" />
          </div>
        </div>
      </div>
    </section>
```

❖ Contact Page

```
    <section id="contact" class="page">
      <div class="container">
        <h2 class="page-title">Contact Us</h2>
        <p class="page-subtitle">Have questions or suggestions? We'd love to hear from you!</p>

        <div class="contact-content">
          <form class="contact-form" onsubmit="handleContactSubmit(event)">
            <div class="form-group">
              <label for="name">Name</label>
              <input type="text" id="name" name="name" required />
            </div>

            <div class="form-group">
              <label for="email">Email</label>
              <input type="email" id="email" name="email" required />
            </div>

            <div class="form-group">
              <label for="subject">Subject</label>
              <input type="text" id="subject" name="subject" required />
            </div>

            <div class="form-group">
              <label for="message">Message</label>
              <textarea id="message" name="message" rows="6" required></textarea>
            </div>

            <button type="submit" class="btn-primary">Send Message</button>
          </form>
```

```
                    <div class="contact-info">
                      <h3>Get In Touch</h3>
                      <div class="info-item">
                        <strong>Email:</strong>
                        <p>techie-sou@error404.mail</p>
                      </div>
                      <div class="info-item">
                        <strong>Phone:</strong>
                        <p>+91 74328 40665</p>
                      </div>
                      <div class="info-item">
                        <strong>Address:</strong>
                        <p>Baruipur, Kolkata <br>KOL-700144</p>
                      </div>
                    </div>
                  </div>
                </div>
              </section>
            </main>
```

- **CSS Styling –** Styles.css

```
* {
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}

:root {
  --primary-color: #2563eb;
  --primary-hover: #1d4ed8;
  --secondary-color: #10b981;
  --background: #ffffff;
  --surface: #f8fafc;
  --text-primary: #1e293b;
  --text-secondary: #64748b;
  --border-color: #e2e8f0;
  --shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
  --shadow-lg: 0 10px 25px rgba(0, 0, 0, 0.1);
}

body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  line-height: 1.6;
  color: var(--text-primary);
  background-color: var(--background);
  margin: 0;
  padding: 0;
}

.container {
  max-width: 1200px;
  margin: 0 auto;
  padding: 0 24px;
}

.header {
  background-color: var(--background);
  box-shadow: var(--shadow);
  position: sticky;
  top: 0;
  z-index: 100;
  padding: 16px 0;
}

.header .container {
  display: flex;
  justify-content: space-between;
  align-items: center;
```

```
}

.logo h1 {
  font-size: 1.5rem;
  color: var(--primary-color);
  margin: 0;
}

.nav {
  display: flex;
  gap: 24px;
}

.nav-link {
  text-decoration: none;
  color: var(--text-secondary);
  font-weight: 500;
  padding: 8px 12px;
  border-radius: 6px;
  transition: all 0.3s ease;
}

.nav-link:hover,
.nav-link.active {
  color: var(--primary-color);
  background-color: var(--surface);
}

/* <CHANGE> Added header-right for auth button styling */
.header-right {
  display: flex;
  align-items: center;
  gap: 16px;
}

.btn-auth {
  background-color: var(--primary-color);
  color: white;
  border: none;
  padding: 8px 16px;
  border-radius: 6px;
  font-weight: 600;
  cursor: pointer;
  transition: all 0.3s ease;
}

.btn-auth:hover {
  background-color: var(--primary-hover);
}

.btn-auth.logout {
  background-color: #ef4444;
}

.btn-auth.logout:hover {
  background-color: #dc2626;
}

.nav-toggle {
  display: none;
}

.hamburger {
  display: block;
  position: relative;
}

.hamburger,
.hamburger::before,
.hamburger::after {
```

```
  background: var(--primary-color);
  width: 2em;
  height: 3px;
  border-radius: 1em;
  transition: transform 250ms ease-in-out;
}

.page {
  display: none;
  padding: 48px 0;
  min-height: calc(100vh - 200px);
}

.page.active {
  display: block;
}

.hero {
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  padding: 80px 0;
  text-align: center;
  border-radius: 16px;
  margin-bottom: 48px;
}

.hero-title {
  font-size: 3rem;
  margin-bottom: 16px;
  font-weight: 700;
}

.hero-subtitle {
  font-size: 1.25rem;
  margin-bottom: 32px;
  opacity: 0.95;
}

.btn-primary {
  background-color: white;
  color: #667eea;
  border: none;
  padding: 12px 32px;
  font-size: 1rem;
  font-weight: 600;
  border-radius: 8px;
  cursor: pointer;
  transition: all 0.3s ease;
  box-shadow: 0 4px 12px rgba(0, 0, 0, 0.15);
}

.btn-primary:hover {
  transform: translateY(-2px);
  box-shadow: 0 6px 16px rgba(0, 0, 0, 0.2);
}

.btn-primary.btn-full {
  width: 100%;
  background-color: var(--primary-color);
  color: white;
}

.btn-primary.btn-full:hover {
  background-color: var(--primary-hover);
}

.btn-primary.btn-save {
  background-color: var(--primary-color);
  color: white;
  margin-top: 24px;
```

```
}

.features {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
  gap: 32px;
  margin-top: 48px;
}

.feature-card {
  background: var(--surface);
  padding: 32px;
  border-radius: 12px;
  text-align: center;
  transition: all 0.3s ease;
  border: 1px solid var(--border-color);
}

.feature-card:hover {
  transform: translateY(-4px);
  box-shadow: var(--shadow-lg);
}

.feature-icon {
  font-size: 3rem;
  margin-bottom: 16px;
}

.feature-card h3 {
  font-size: 1.5rem;
  margin-bottom: 12px;
  color: var(--text-primary);
}

.feature-card p {
  color: var(--text-secondary);
}

.page-title {
  font-size: 2.5rem;
  margin-bottom: 16px;
  color: var(--text-primary);
  text-align: center;
}

.page-subtitle {
  text-align: center;
  color: var(--text-secondary);
  font-size: 1.125rem;
  margin-bottom: 48px;
}

.search-section {
  display: flex;
  gap: 16px;
  max-width: 600px;
  margin: 0 auto 48px;
}

.search-input {
  flex: 1;
  padding: 12px 20px;
  font-size: 1rem;
  border: 2px solid var(--border-color);
  border-radius: 8px;
  transition: border-color 0.3s ease;
}

.search-input:focus {
  outline: none;
```

```
  border-color: var(--primary-color);
}

.search-section .btn-primary {
  background-color: var(--primary-color);
  color: white;
}

.search-section .btn-primary:hover {
  background-color: var(--primary-hover);
}

/* <CHANGE> Added dietary profile styles */
.dietary-container {
  max-width: 800px;
  margin: 0 auto;
}

.dietary-section {
  background: var(--surface);
  padding: 32px;
  border-radius: 12px;
  margin-bottom: 32px;
  border: 1px solid var(--border-color);
}

.dietary-section h3 {
  color: var(--primary-color);
  margin-bottom: 20px;
  font-size: 1.25rem;
}

.diet-options,
.allergy-options {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 16px;
}

.diet-checkbox {
  display: flex;
  align-items: center;
  cursor: pointer;
  padding: 12px;
  border: 2px solid var(--border-color);
  border-radius: 8px;
  transition: all 0.3s ease;
  position: relative;
}

.diet-checkbox:hover {
  background-color: rgba(37, 99, 235, 0.05);
}

.diet-checkbox input {
  display: none;
}

.diet-checkbox input:checked + .checkmark {
  background-color: var(--primary-color);
  border-color: var(--primary-color);
}

.diet-checkbox input:checked + .checkmark::after {
  display: block;
}

.checkmark {
  display: inline-block;
  width: 20px;
```

```
  height: 20px;
  border: 2px solid var(--border-color);
  border-radius: 4px;
  margin-right: 12px;
  position: relative;
  transition: all 0.3s ease;
}

.checkmark::after {
  content: '';
  display: none;
  position: absolute;
  left: 6px;
  top: 2px;
  width: 6px;
  height: 10px;
  border: solid white;
  border-width: 0 2px 2px 0;
  transform: rotate(45deg);
}

.nutrition-goals {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
  gap: 16px;
}

.goal-item {
  display: flex;
  flex-direction: column;
}

.goal-item label {
  font-weight: 600;
  margin-bottom: 8px;
  color: var(--text-primary);
}

.goal-item input {
  padding: 10px;
  border: 2px solid var(--border-color);
  border-radius: 6px;
  font-size: 1rem;
  transition: border-color 0.3s ease;
}

.goal-item input:focus {
  outline: none;
  border-color: var(--primary-color);
}

.profile-message {
  text-align: center;
  margin-top: 16px;
  padding: 12px;
  border-radius: 6px;
  min-height: 20px;
}

.profile-message.success {
  background-color: #d1fae5;
  color: #065f46;
}

.profile-message.error {
  background-color: #fee2e2;
  color: #991b1b;
}

.loading {
```

```
  text-align: center;
  padding: 48px;
}

.spinner {
  border: 4px solid var(--border-color);
  border-top: 4px solid var(--primary-color);
  border-radius: 50%;
  width: 50px;
  height: 50px;
  animation: spin 1s linear infinite;
  margin: 0 auto 16px;
}

@keyframes spin {
  0% { transform: rotate(0deg); }
  100% { transform: rotate(360deg); }
}

.recipes-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(300px, 1fr));
  gap: 24px;
  margin-top: 32px;
}

.recipe-card {
  background: white;
  border-radius: 12px;
  overflow: hidden;
  box-shadow: var(--shadow);
  transition: all 0.3s ease;
  cursor: pointer;
  border: 1px solid var(--border-color);
}

.recipe-card:hover {
  transform: translateY(-4px);
  box-shadow: var(--shadow-lg);
}

.recipe-image {
  width: 100%;
  height: 200px;
  object-fit: cover;
}

.recipe-content {
  padding: 20px;
}

.recipe-title {
  font-size: 1.25rem;
  margin-bottom: 8px;
  color: var(--text-primary);
}

.recipe-meta {
  display: flex;
  gap: 16px;
  color: var(--text-secondary);
  font-size: 0.875rem;
  margin-bottom: 12px;
}

.recipe-description {
  color: var(--text-secondary);
  font-size: 0.875rem;
  line-height: 1.5;
}
```

```
.about-content {
  display: grid;
  grid-template-columns: 1fr 1fr;
  gap: 48px;
  align-items: start;
  margin-top: 48px;
}

.about-text h3 {
  color: var(--primary-color);
  margin: 24px 0 12px;
  font-size: 1.5rem;
}

.about-text p {
  color: var(--text-secondary);
  line-height: 1.8;
  margin-bottom: 16px;
}

.about-text ul {
  list-style-position: inside;
  color: var(--text-secondary);
  line-height: 2;
}

.about-image img {
  width: 100%;
  border-radius: 12px;
  box-shadow: var(--shadow-lg);
}

.contact-content {
  display: grid;
  grid-template-columns: 2fr 1fr;
  gap: 48px;
  margin-top: 48px;
}

.contact-form {
  background: var(--surface);
  padding: 32px;
  border-radius: 12px;
  border: 1px solid var(--border-color);
}

.form-group {
  margin-bottom: 24px;
}

.form-group label {
  display: block;
  margin-bottom: 8px;
  font-weight: 600;
  color: var(--text-primary);
}

.form-group input,
.form-group textarea {
  width: 100%;
  padding: 12px;
  border: 2px solid var(--border-color);
  border-radius: 8px;
  font-size: 1rem;
  font-family: inherit;
  transition: border-color 0.3s ease;
}

.form-group input:focus,
```

```
.form-group textarea:focus {
  outline: none;
  border-color: var(--primary-color);
}

.form-group textarea {
  resize: vertical;
}

.contact-form .btn-primary {
  background-color: var(--primary-color);
  color: white;
  width: 100%;
}

.contact-info {
  background: var(--surface);
  padding: 32px;
  border-radius: 12px;
  border: 1px solid var(--border-color);
}

.contact-info h3 {
  color: var(--primary-color);
  margin-bottom: 24px;
}

.info-item {
  margin-bottom: 24px;
}

.info-item strong {
  display: block;
  margin-bottom: 8px;
  color: var(--text-primary);
}

.info-item p {
  color: var(--text-secondary);
  line-height: 1.6;
}

/* <CHANGE> Added auth modal styles */
.auth-modal-content {
  max-width: 400px;
}

.auth-tabs {
  display: flex;
  gap: 0;
  margin-bottom: 24px;
  border-bottom: 2px solid var(--border-color);
}

.auth-tab-btn {
  flex: 1;
  padding: 12px;
  background: transparent;
  border: none;
  font-weight: 600;
  color: var(--text-secondary);
  cursor: pointer;
  border-bottom: 3px solid transparent;
  transition: all 0.3s ease;
}

.auth-tab-btn.active {
  color: var(--primary-color);
  border-bottom-color: var(--primary-color);
}
```

```
.auth-form {
  display: none;
}

.auth-form.active {
  display: block;
}

.timer-widget {
  position: fixed;
  bottom: 24px;
  right: 24px;
  background: white;
  border-radius: 12px;
  box-shadow: var(--shadow-lg);
  width: 320px;
  z-index: 1000;
  border: 1px solid var(--border-color);
}

.timer-header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 16px 20px;
  border-bottom: 1px solid var(--border-color);
  background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
  color: white;
  border-radius: 12px 12px 0 0;
}

.timer-header h3 {
  margin: 0;
  font-size: 1.125rem;
}

.timer-toggle {
  background: transparent;
  border: none;
  color: white;
  font-size: 1.5rem;
  cursor: pointer;
  width: 32px;
  height: 32px;
  display: flex;
  align-items: center;
  justify-content: center;
  border-radius: 4px;
  transition: background-color 0.3s ease;
}

.timer-toggle:hover {
  background-color: rgba(255, 255, 255, 0.2);
}

.timer-content {
  padding: 20px;
}

.timer-widget.collapsed .timer-content {
  display: none;
}

.timer-display {
  font-size: 2.5rem;
  font-weight: 700;
  text-align: center;
  color: var(--primary-color);
  margin-bottom: 16px;
```

```
  font-family: 'Courier New', monospace;
}

.timer-inputs {
  display: flex;
  justify-content: center;
  align-items: center;
  gap: 8px;
  margin-bottom: 16px;
}

.timer-inputs input {
  width: 60px;
  padding: 8px;
  text-align: center;
  border: 2px solid var(--border-color);
  border-radius: 6px;
  font-size: 1rem;
  font-weight: 600;
}

.timer-inputs span {
  font-size: 1.5rem;
  font-weight: 700;
  color: var(--text-secondary);
}

.timer-controls {
  display: flex;
  gap: 8px;
}

.btn-timer {
  flex: 1;
  padding: 10px;
  border: none;
  border-radius: 6px;
  font-weight: 600;
  cursor: pointer;
  transition: all 0.3s ease;
  font-size: 0.875rem;
}

.btn-start {
  background-color: var(--secondary-color);
  color: white;
}

.btn-start:hover {
  background-color: #059669;
}

.btn-pause {
  background-color: #f59e0b;
  color: white;
}

.btn-pause:hover {
  background-color: #d97706;
}

.btn-reset {
  background-color: #ef4444;
  color: white;
}

.btn-reset:hover {
  background-color: #dc2626;
}
```

```
.modal {
  display: none;
  position: fixed;
  z-index: 2000;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.5);
  overflow-y: auto;
}

.modal.active {
  display: block;
}

.modal-content {
  background-color: white;
  margin: 48px auto;
  padding: 32px;
  border-radius: 12px;
  max-width: 800px;
  position: relative;
  box-shadow: var(--shadow-lg);
}

.close {
  position: absolute;
  right: 20px;
  top: 20px;
  font-size: 2rem;
  font-weight: 700;
  color: var(--text-secondary);
  cursor: pointer;
  transition: color 0.3s ease;
}

.close:hover {
  color: var(--text-primary);
}

.modal-recipe-image {
  width: 100%;
  height: 300px;
  object-fit: cover;
  border-radius: 8px;
  margin-bottom: 24px;
}

.modal-recipe-title {
  font-size: 2rem;
  margin-bottom: 16px;
  color: var(--text-primary);
}

.modal-recipe-meta {
  display: flex;
  gap: 24px;
  margin-bottom: 24px;
  padding-bottom: 24px;
  border-bottom: 2px solid var(--border-color);
}

.meta-item {
  display: flex;
  align-items: center;
  gap: 8px;
  color: var(--text-secondary);
}
```

```
.nutrition-badge {
  display: inline-block;
  background: #dbeafe;
  color: #1e40af;
  padding: 4px 8px;
  border-radius: 4px;
  font-size: 0.75rem;
  font-weight: 600;
  margin-right: 4px;
}

.modal-section {
  margin-bottom: 32px;
}

.modal-section h3 {
  color: var(--primary-color);
  margin-bottom: 16px;
  font-size: 1.5rem;
}

.modal-section ul {
  list-style-position: inside;
  color: var(--text-secondary);
  line-height: 2;
}

.modal-section ol {
  padding-left: 20px;
  color: var(--text-secondary);
  line-height: 2;
}

.nutrition-info {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 16px;
  padding: 16px;
  background: var(--surface);
  border-radius: 8px;
}

.nutrition-item {
  text-align: center;
}

.nutrition-item strong {
  display: block;
  font-size: 1.5rem;
  color: var(--primary-color);
}

.nutrition-item span {
  display: block;
  font-size: 0.75rem;
  color: var(--text-secondary);
  margin-top: 4px;
}

.footer {
  background-color: var(--text-primary);
  color: white;
  padding: 48px 0 24px;
  margin-top: 80px;
}

.footer-content {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
  gap: 48px;
```

```
  margin-bottom: 32px;
}

.footer-section h3 {
  margin-bottom: 16px;
  font-size: 1.25rem;
}

.footer-section h4 {
  margin-bottom: 16px;
  font-size: 1.125rem;
}

.footer-section p {
  color: rgba(255, 255, 255, 0.8);
  line-height: 1.8;
}

.footer-section ul {
  list-style: none;
}

.footer-section ul li {
  margin-bottom: 8px;
}

.footer-section ul li a {
  color: rgba(255, 255, 255, 0.8);
  text-decoration: none;
  transition: color 0.3s ease;
}

.footer-section ul li a:hover {
  color: white;
}

.social-links {
  display: flex;
  gap: 16px;
}

.social-link {
  color: rgba(255, 255, 255, 0.8);
  text-decoration: none;
  transition: color 0.3s ease;
}

.social-link:hover {
  color: white;
}

@media (max-width: 768px) {
  .header .container {
    justify-content: space-between;
  }

  .nav {
    position: fixed;
    background: var(--background);
    color: var(--text-primary);
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    z-index: 100;
    transform: translateX(100%);
    transition: transform 250ms cubic-bezier(0.5, 0, 0.5, 1);
    flex-direction: column;
    justify-content: center;
    align-items: center;
```

```
  gap: 2rem;
}

.nav--visible {
  transform: translateX(0);
}

.nav-toggle {
  display: block;
  cursor: pointer;
  background: transparent;
  border: 0;
  padding: 0.5em;
  position: absolute;
  z-index: 9999;
  right: 1rem;
  top: 1rem;
}

.hamburger::before,
.hamburger::after {
  content: '';
  position: absolute;
  left: 0;
  right: 0;
}

.hamburger::before {
  top: 6px;
}
.hamburger::after {
  bottom: 6px;
}

.hero-title {
  font-size: 2rem;
}

.hero-subtitle {
  font-size: 1rem;
}

.features {
  grid-template-columns: 1fr;
}

.about-content,
.contact-content,
.footer-content {
  grid-template-columns: 1fr;
}

.search-section {
  flex-direction: column;
}

.recipes-grid {
  grid-template-columns: 1fr;
}

.timer-widget {
  width: calc(100% - 32px);
  left: 16px;
  right: 16px;
}

.page-title {
  font-size: 2rem;
}
```

```
  .nutrition-info {
    grid-template-columns: repeat(2, 1fr);
  }

  .diet-options,
  .allergy-options {
    grid-template-columns: 1fr;
  }

  .modal-recipe-meta {
    flex-direction: column;
    gap: 12px;
  }
}
```

Conclusion:

DishCovery was created with a simple idea in mind—making cooking easier and more enjoyable for everyone. Whether someone is trying a new dish or looking for quick meal ideas, the platform offers a friendly space to explore recipes without any complexity. Its clean interface and smooth flow make it easy for users to enjoy discovering food.

Behind the scenes, DishCovery runs on HTML, CSS, and JavaScript, with real-time recipe data coming from TheMealDB API. Each feature, from the search bar to the recipe display, was built with a focus on quick responses and a smooth experience. The modular coding approach ensures that different components work together effortlessly, while still being easy to maintain and expand in the future.

The platform helps users explore new cuisines, experiment with ingredients, and step out of their regular cooking habits. It also supports better meal planning and healthier choices by showing clear ingredient lists and preparation steps. For many users, DishCovery can become more than just a recipe finder, it can be an everyday companion in the kitchen.

In the end, DishCovery brings together simplicity, usefulness, and a pleasant user experience. It makes food discovery fun, accessible, and stress-free. With room for future features like diet-based suggestions or nutrition tools, the platform is well-positioned to grow into an even more helpful cooking assistant.

Future Scope & Enhancement

While this version of DishCovery already meets the essential needs of recipe exploration, it has strong potential to grow further. As user expectations evolve, the platform can adopt smarter, more personalized features to enhance discovery, convenience, and overall cooking experience.

- **Plans to Enhance:**
  - Advanced Dietary Profile – More diet section and categories will be added.
  - AI Power meal suggestion – AI powered meal suggestion will be added (Using Gemini API) with current meal suggestion.
  - Cooking Tutorial in our own website – Pop up tutorial will be added on recipe finder section.
  - Check Previous searches – A new section on profile will be added for finding previous searches.
  - Rating System on Food – A rating system (with negative rating) will be introduced to user.
  - User Community – Few user community(Based on taste and category) will be added where user can post their opinion and advices where other users can view the post.
  - Adding nearby shops for ingredients – Nearby shops and shop location will be added.

BIBLIOGRAPHY

The following resources and references were consulted and utilized during the planning, development, and documentation of the **DishCovery (Online Recipe Finder)** project:

- **Web Development & Programming Resources**
  - W3Schools – HTML, CSS, JavaScript reference
  - MDN Web Docs – JavaScript and DOM reference
  - Stack Overflow – Community solutions and code debugging
  - GeeksforGeeks – Web development tutorials and backend guides
  - Bootstrap Documentation – Styling and responsive layout components
  - Google Fonts – Fonts such as *Baloo 2* for aesthetic design

---

- **Design & UI Inspiration**
  - Canva – Color schemes, UI layout inspiration
  - Dribble – Interface and component ideas
  - Figma – Wireframing and UI prototyping (optional reference)

- **Tools & Platforms Used**
  - Visual Studio Code – Code editor for frontend and HTML development
  - Git & GitHub – Version control and code backup
  - Google Chrome DevTools – For responsive design testing and debugging
  - Localhost & Live Server – For local development and testing
  - GitHub Pages – For hosting globally

- **Domain Research**
  - Articles and blog posts about e-learning trends in performing arts
  - YouTube and online dance academy platforms for understanding UI/UX needs

- Educational platforms such as **Udemy**, **Coursera**, and **Kalamandalam** (Indian classical institutes)

➢ **Personal Contribution**

- UI content, layout, and screenshots were self-developed and implemented as part of the original project.
- Real-world feedback and user experience testing shaped design and navigation decisions.