Planning Analysis Sheet: Recipe HUB – Popular Indian Recipes

Website Links: https://www.recipehub.pro/, https://recipehub.pro/
Render URL: https://recipe-hub-1xes.onrender.com/home
Swagger: https://recipehub.pro/apidocs/
Repository: https://github.com/Tanvi-Bagwe/recipe

Recipe HUB is a web application designed to showcase popular Indian recipes. It provides users with a visually appealing interface to explore recipes, view detailed steps, and submit testimonials. This document outlines the design, development process, technical implementation, and challenges faced during the project.

---

1. Design and User Experience

- **Intuitive Navigation:** The website has clearly defined sections: Home, Recipes, Recipe Details, About Us, and Testimonials. Users can navigate easily without confusion.
- **Recipe Grid Layout:** Recipe images and titles are displayed in a responsive grid. Clicking on a recipe opens a detail page with instructions and ingredients.
- **Filter Options:** Users can filter recipes by type (Veg, Non-Veg, All) using query parameters and simple filtering logic.
- **Testimonials Section:** Users can submit feedback via a simple form. All testimonials are displayed dynamically in a clean layout.
- **Responsive Design:** Flexbox and grid layouts ensure that content looks good on desktops, tablets, and mobile devices.

---

2. Key Highlights

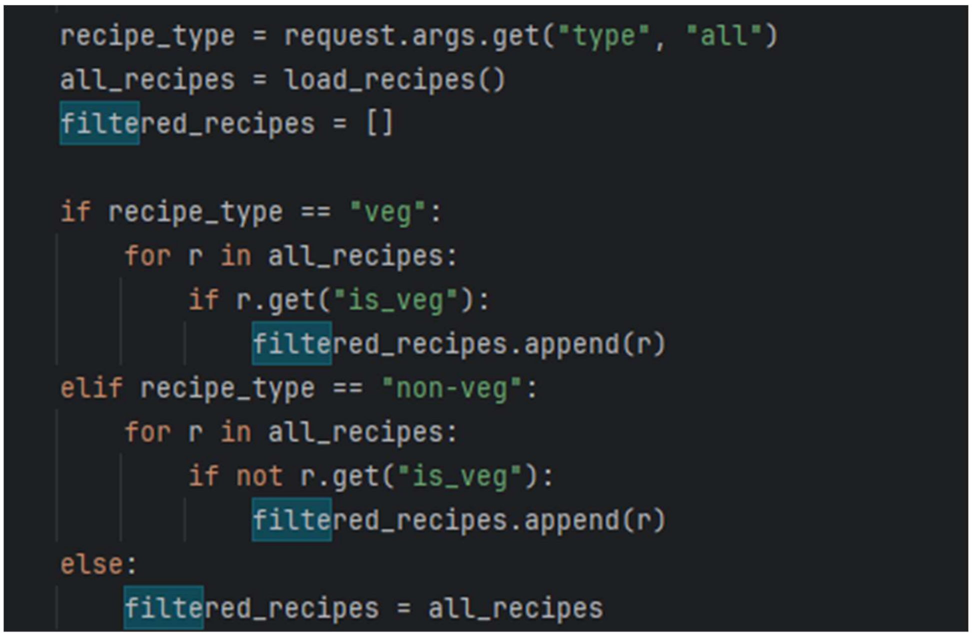
- **Custom Domain -** A custom domain was purchased and configured through Render and GoDaddy to provide a more user-friendly and memorable URL for the website. The DNS settings were updated in GoDaddy by pointing the root domain (@) to Render's IP address using an A record, and the www subdomain was configured with a CNAME record pointing to the Render service. Render automatically provisions and manages SSL certificates for custom domains, ensuring secure HTTPS connections. This setup not only improves accessibility and professionalism but also builds trust with users by enforcing a secure and branded URL.
- **Swagger -** To make the application's REST APIs easy to understand, test, and explore, Swagger has been integrated into Recipe Hub using Python's Flasgger library. This setup provides a live, interactive Swagger UI where developers can view all available endpoints, their request/response formats, and even execute test requests directly from the browser.
- **Scheduler** – On the free plan of Render.com, web services automatically enter a sleep mode after 15 minutes of inactivity. To prevent downtime and ensure continuous availability, a Render Scheduler was configured to run at 5-minute intervals. This scheduler triggers a Python script that uses the requests module to periodically send HTTP requests (pings) to the deployed website, thereby keeping the service active and preventing it from going into sleep mode.

---

3. Development and Implementation

- **Flask Backend:**

    - Each page is a Flask route. Example: /recipes shows the recipe grid, /recipe/<id> shows recipe details.
    - JSON files (recipes.json, testimonials.json) store recipes and feedback.
    - POST and GET routes are used to manage data.
- **Frontend Structure:**
    - HTML templates use **template inheritance** (base.html) for consistent header and footer.
    - CSS is modular: separate files for header, footer, main styles, recipe grid, etc.
    - JavaScript handles dynamic content: form submission, filtering, and updating testimonial lists asynchronously.
- **Data Structures:**
    - Recipes and testimonials are stored as **lists of dictionaries** in JSON files.



```python
recipe_type = request.args.get("type", "all")
all_recipes = load_recipes()
filtered_recipes = []

if recipe_type == "veg":
    for r in all_recipes:
        if r.get("is_veg"):
            filtered_recipes.append(r)
elif recipe_type == "non-veg":
    for r in all_recipes:
        if not r.get("is_veg"):
            filtered_recipes.append(r)
else:
    filtered_recipes = all_recipes
```

- **Interactivity:**
    - Fetch API used for submitting and fetching testimonials without reloading the page.
    - Simple error validation ensures names and feedback meet minimum length requirements.

---

5. Challenges and Solutions

- **Filtering Recipes:** Initially tried list comprehensions but switched to a simple for-loop for clarity and readability.
- **Dynamic Testimonials:** Needed to update the JSON file and frontend dynamically. Using Fetch API with Flask POST/GET routes solved this.
- **Responsive Layouts:** Ensuring recipe cards and testimonials scaled correctly across screen sizes required careful CSS adjustments.
- **Template Redundancy:** Solved repeated header/footer code by using Flask template inheritance.

---

6. Page Overview

1. **Home Page (home.html)**
    - Hero banner with Recipe HUB logo and mission statement.
    - Featured recipes displayed in a responsive grid.
    - Quick links to Recipes, About Us, and Testimonials.
2. **Recipes Page (recipies.html)**
    - Grid display of all recipes with image and title.
    - Filtering by Veg/Non-Veg/All using query parameters.
    - Clicking a recipe opens the details page.
3. **Recipe Details (recipe_detail.html)**
    - Image, ingredients, and preparation instructions.
    - Clear, minimal layout for easy reading.
4. **About Us (about.html)**
    - Describes the goal and mission of Recipe HUB.
    - Simple, centred text layout for clarity.
5. **Testimonials (testimonials.html)**
    - Form to submit feedback with validation.
    - Display of all submitted testimonials fetched dynamically.

---

7. Tools and Technologies

- **Backend:** Python, Flask
- **Frontend:** HTML, CSS, JavaScript
- **Data Storage:** JSON files (recipes.json, testimonials.json)
- **Libraries:** Flasgger (Swagger API documentation)
- **Deployment:** Render.com (free SSL, custom domain)
- **Development Tools:** PyCharm / VS Code, Git & GitHub

---

8. Key Learnings

- How to structure a Flask project with templates and static assets.
- Handling dynamic data from JSON files with Python.
- Using Fetch API for asynchronous GET/POST requests.
- Creating responsive layouts with Flexbox and Grid.
- Implementing clean UI with filters, grids, and forms.

9. References

- **Images:** Unsplash
- **Icons:** Flaticon
- **Wireframes:** Draw.io
- **Text & Recipes:** Curated manually
- **Domain:** https://www.godaddy.com/en-in