Simon Doan - Student ID: 301455974
Yizhang Zhu - Student ID: 3015171258
Nur Ameer Nur Saidy - Student ID: 301575157
Chaitanya Mittal - Student ID: 301584364
Date of Submission: October 18th, 2024

Milestone 1: Project Planning

The Recipe Finder Web App

By Group 17: Sunsets Organization

Finalized 2 APIs

Spoonacular API:

1. Link: https://spoonacular.com/food-api
2. Description:
    a. Large database of diverse recipes, ingredients, and nutritional data.
    b. Supports filtering recipes by nutrients, ingredients, and dietary preferences, aligning with user needs (e.g., nutritionists, chefs, parents).
3. Usage in the Project:
    a. Allows users to search for recipes with specific criteria (e.g., ingredient-based or calorie limits).
    b. Provides detailed nutritional data, helping users track calories and maintain dietary goals.

YouTube Data API

1. Link: https://developers.google.com/youtube/v3
2. Description:
    a. Access to a vast library of video tutorials, including cooking and food preparation techniques.
    b. Helps new users (like students) learn how to cook through video tutorials and playlists.
3. Usage in the Project:
    a. Integrates video tutorials into the app, providing step-by-step guidance for selected recipes.
    b. Allows users to create and save personalized cooking playlists for future reference.

Reason: The original proposal had multiple overlapping features from the APIs (e.g., overlapping meal planning functionality across several APIs). Now, features are more focused, with the Spoonacular API handling recipe searches and meal planning, and the YouTube API managing video tutorials. This avoids redundancy and improves usability.

Features planned to be implemented for each API

Spoonacular API:

1. Search Recipes by Ingredient
    - Users input one or more ingredients, and the API provides a list of recipes matching those ingredients.
    - This feature focuses on minimizing food waste by helping users find recipes based on what they already have.
    - Example: If the user enters “chicken” and “tomato,” the API will return recipes like chicken soup or chicken curry with tomato sauce.
    - It will help to quickly decide what to cook using pantry items. (Busy Parents)
    - Find new dishes using leftover or seasonal ingredients for creative menu items.

2. Search Recipes by Nutritional Value
    - Users search for recipes based on nutritional content such as calories, protein, fat, or carbohydrate levels.
    - The API returns results that meet the specified nutritional requirements (e.g., under 500 calories per serving).
    - Quickly find balanced recipes to recommend to clients.
    - Create meal plans that align with fitness goals like high-protein diets.

3. Meal Planner Feature
    - Users can add selected recipes to a meal planner, organizing them into a structured daily or weekly plan.
    - The API helps streamline this process by allowing users to plan meals according to their dietary goals.
    - Plan menus ahead of time and avoid last minute decisions.

- Stay organized by setting up weekly meal plans to save time on decision-making.

YouTube Data API:

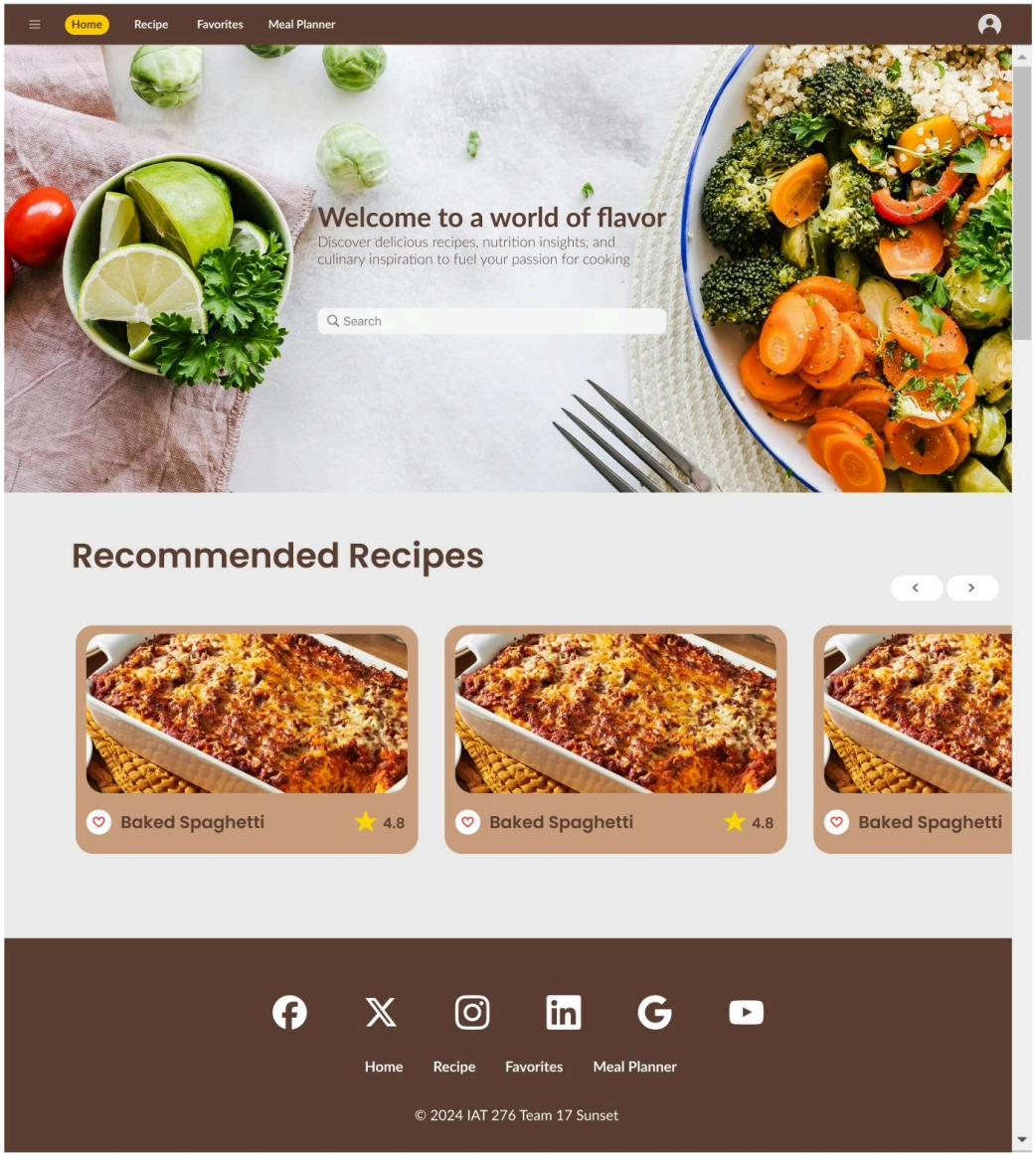
4. Video Search for Cooking Tutorials
    - Users search for video tutorials based on specific recipes or cooking techniques, with the API returning relevant results.
    - Example: Searching for “pasta carbonara tutorial” will retrieve multiple video tutorials showing how to prepare the dish.
    - Learn essential cooking techniques through easy-to-follow tutorials. It is often easier to follow-along rather than a text description with images.
    - Discover new techniques or presentation ideas from expert videos.

5. Create Playlists for Cooking Videos
    - Users can group related videos (e.g., “dessert recipes”) into playlists.
    - Playlists allow users to organize and access all relevant tutorials in one place without needing to search repeatedly.
    - Conveniently revisit grouped tutorials when cooking similar recipes.
    - Easily organize videos based on specific themes or meal types..

6. Saving Favorite Cooking Videos
    - Users can bookmark or save specific tutorials they find useful, allowing quick access later.
    - The API handles saving these videos in a personalized list that the user can revisit at any time.
    - Quickly retrieve frequently-used tutorials without re-searching.

Mid-fidelity prototype

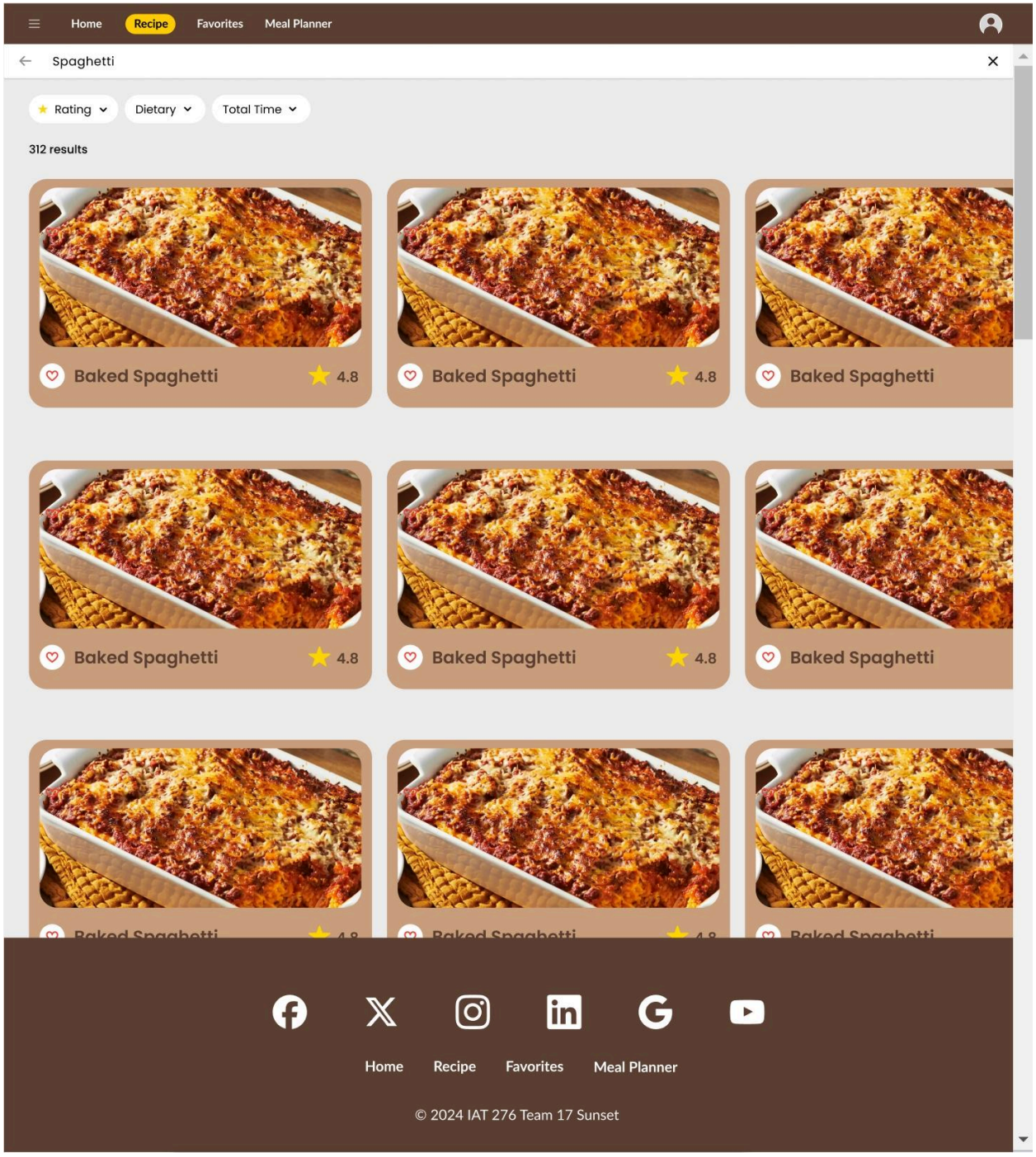
- Home Page



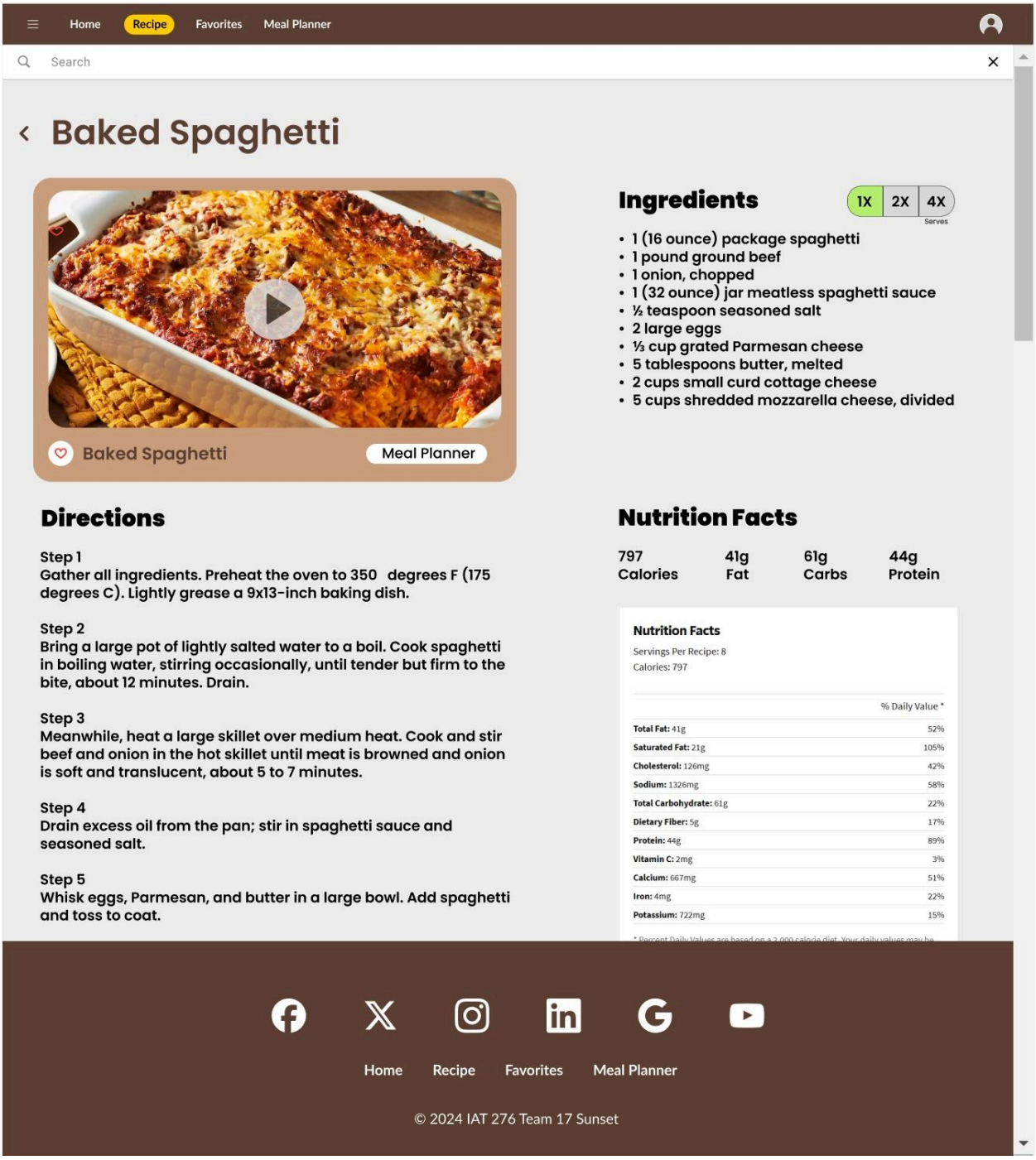
- Menu Bar


Team 17 Sunset
Manage account
Favorites
Meal Planner
Help
Setting
Sign Out

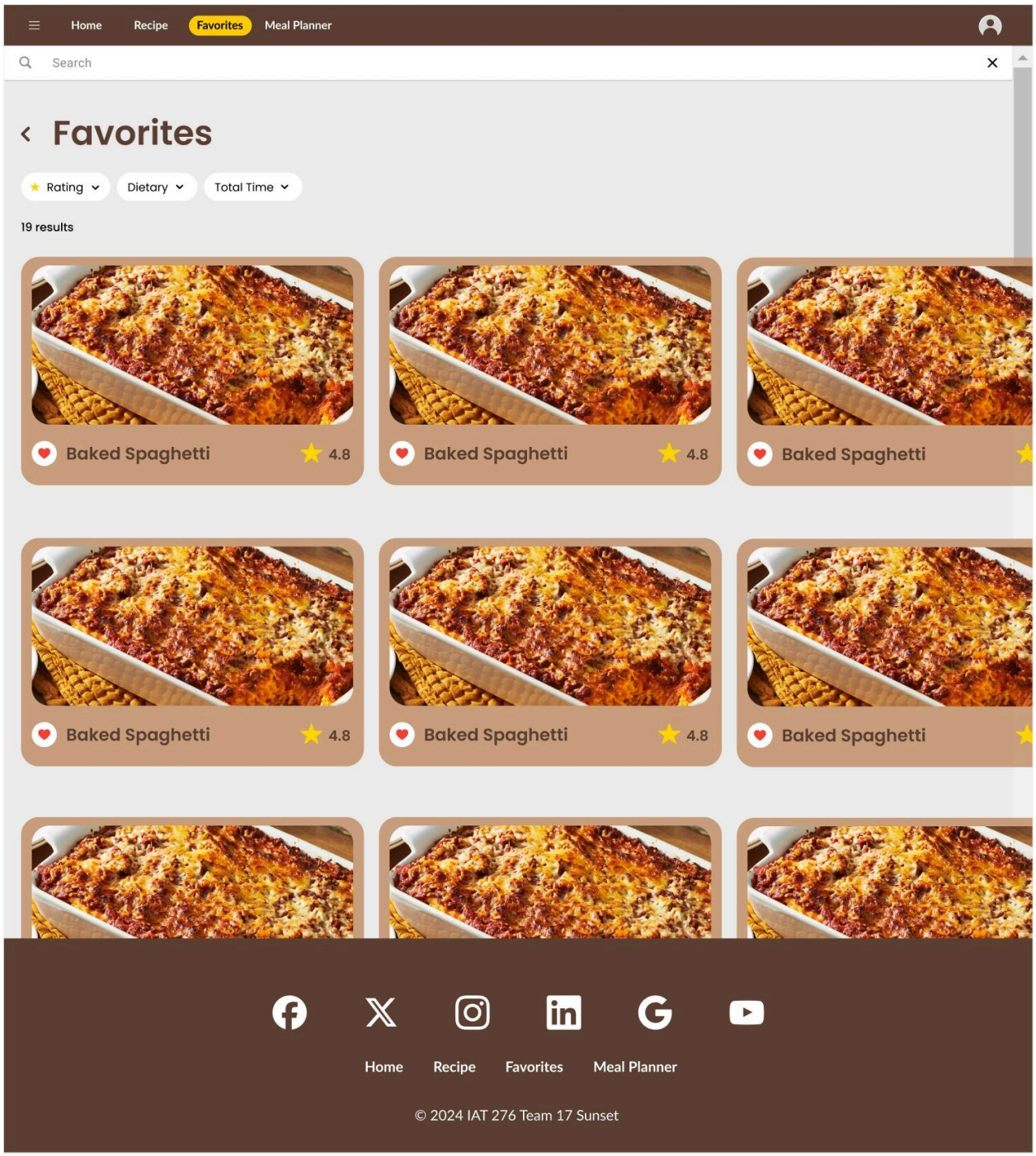
- Search Results Page



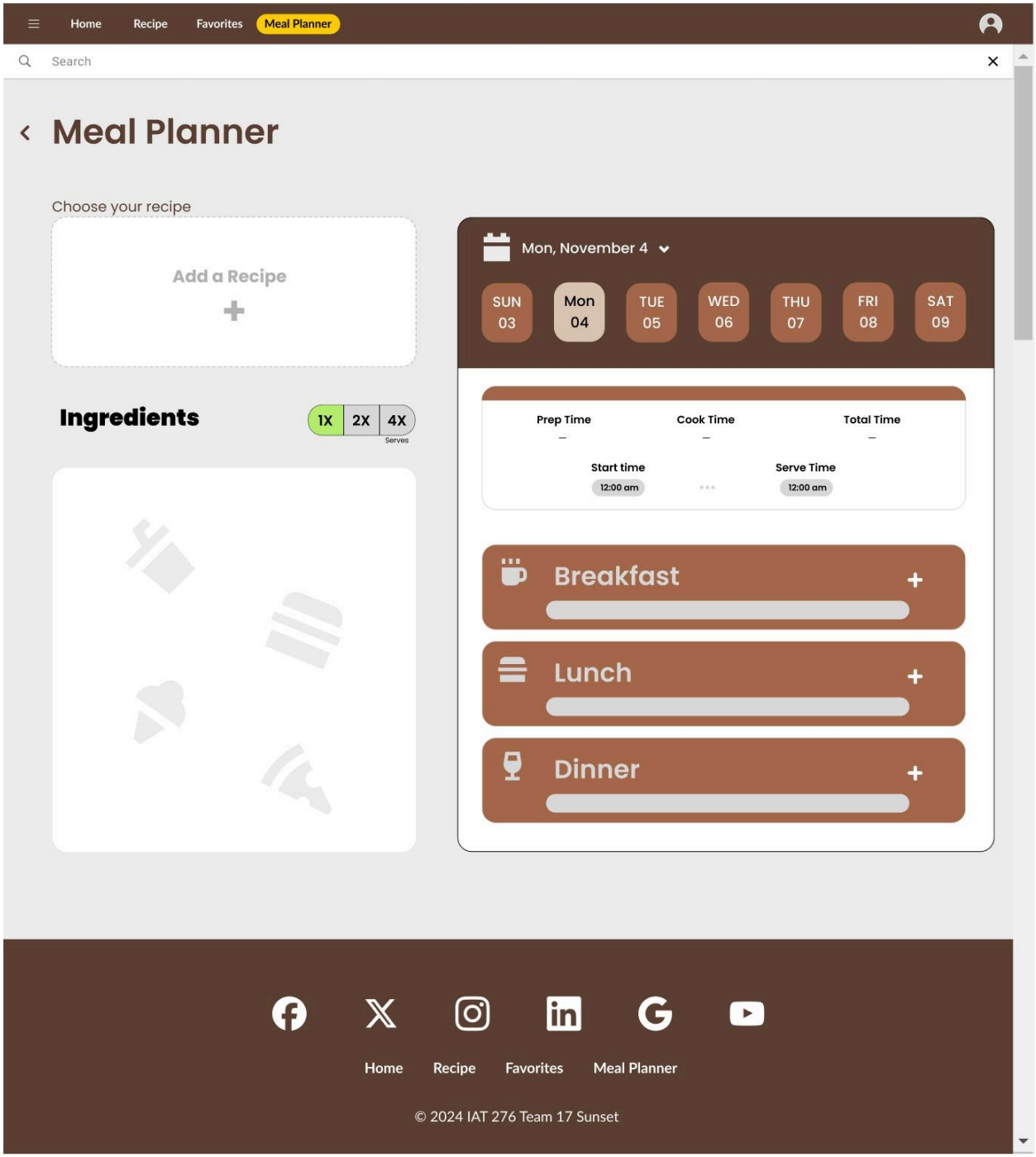
- Recipe Page


Home
Recipe
Favorites
Meal Planner
Search
Baked Spaghetti
Baked Spaghetti
Meal Planner
Ingredients
1X
2X
4X
Serves
• 1 (16 ounce) package spaghetti
• 1 pound ground beef
• 1 onion, chopped
• 1 (32 ounce) jar meatless spaghetti sauce
• ½ teaspoon seasoned salt
• 2 large eggs
• ⅓ cup grated Parmesan cheese
• 5 tablespoons butter, melted
• 2 cups small curd cottage cheese
• 5 cups shredded mozzarella cheese, divided
Directions
Step 1
Gather all ingredients. Preheat the oven to 350 degrees F (175 degrees C). Lightly grease a 9x13-inch baking dish.
Step 2
Bring a large pot of lightly salted water to a boil. Cook spaghetti in boiling water, stirring occasionally, until tender but firm to the bite, about 12 minutes. Drain.
Step 3
Meanwhile, heat a large skillet over medium heat. Cook and stir beef and onion in the hot skillet until meat is browned and onion is soft and translucent, about 5 to 7 minutes.
Step 4
Drain excess oil from the pan; stir in spaghetti sauce and seasoned salt.
Step 5
Whisk eggs, Parmesan, and butter in a large bowl. Add spaghetti and toss to coat.
Nutrition Facts
797 Calories
41g Fat
61g Carbs
44g Protein
Nutrition Facts
Servings Per Recipe: 8
Calories: 797
% Daily Value *
Total Fat: 41g 52%
Saturated Fat: 21g 105%
Cholesterol: 126mg 42%
Sodium: 1326mg 58%
Total Carbohydrate: 61g 22%
Dietary Fiber: 5g 17%
Protein: 44g 89%
Vitamin C: 2mg 3%
Calcium: 667mg 51%
Iron: 4mg 22%
Potassium: 722mg 15%
Home
Recipe
Favorites
Meal Planner
© 2024 IAT 276 Team 17 Sunset

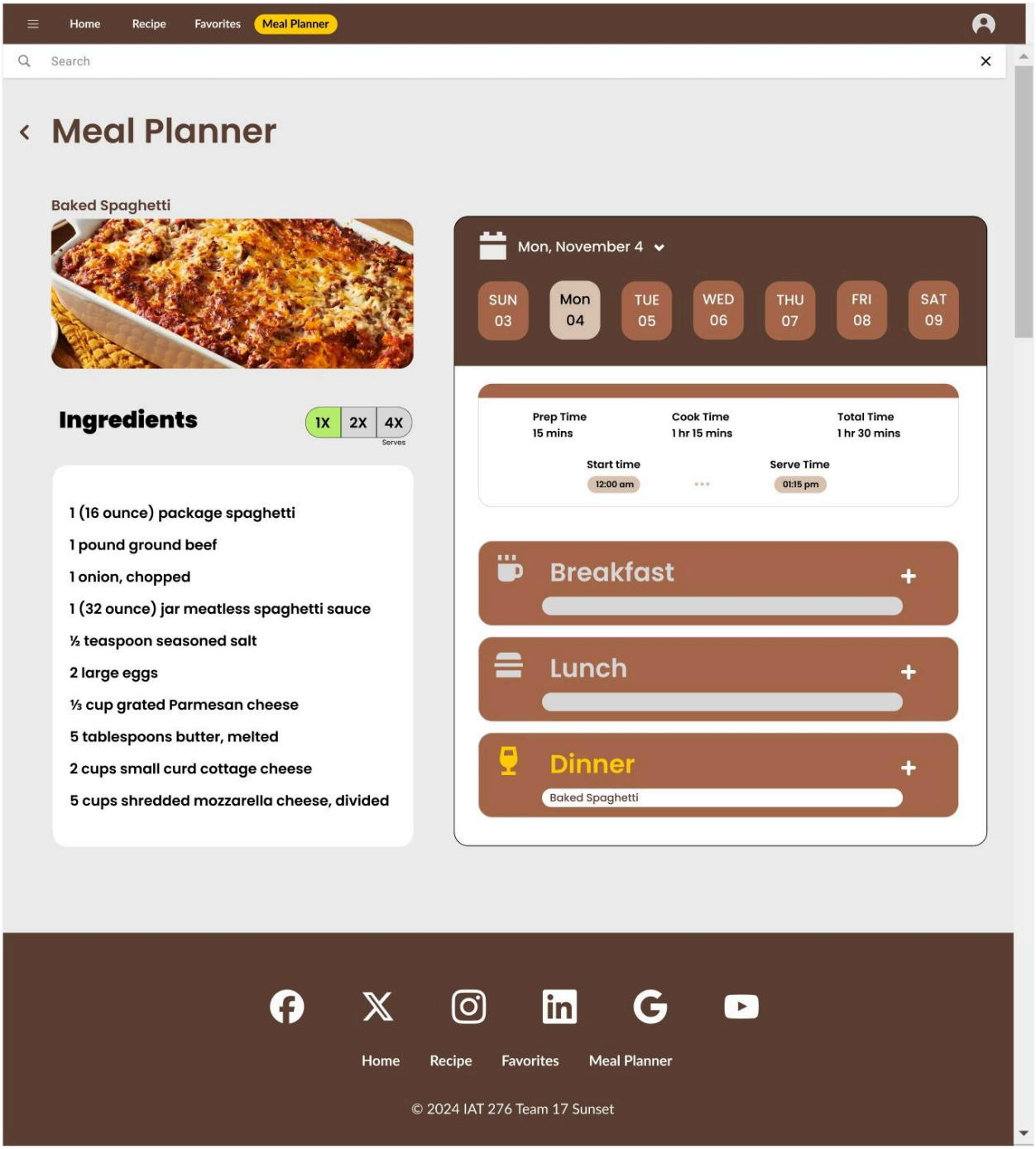
- Favorites Page



- Meal Planner without selected recipe



- Meal Planner with selected recipe



SDLC Model - Agile Model

We will use the Agile model for our recipe finder web application. Using Agile provides our team with flexibility and quick responsiveness to changes, which is especially helpful since this is our first time developing an application. The short development cycles make the process more manageable, and we can continuously gather feedback to improve along the way.

Work Breakdown Structure:

Milestone	Tasks	Sub-Task	Job/Feature	Assign	Dependencies
Project Setup & Foundation	Project Initialization	Set-up github repository		Team	None
		Initialize project environment			None
	HTML & CSS Setup	Basic HTML structure for the website		TBH	Project Initialization
		Basic CSS styling		TBH	HTML setup
	Component Setup	Create reusable React component		TBH	Project Initialization

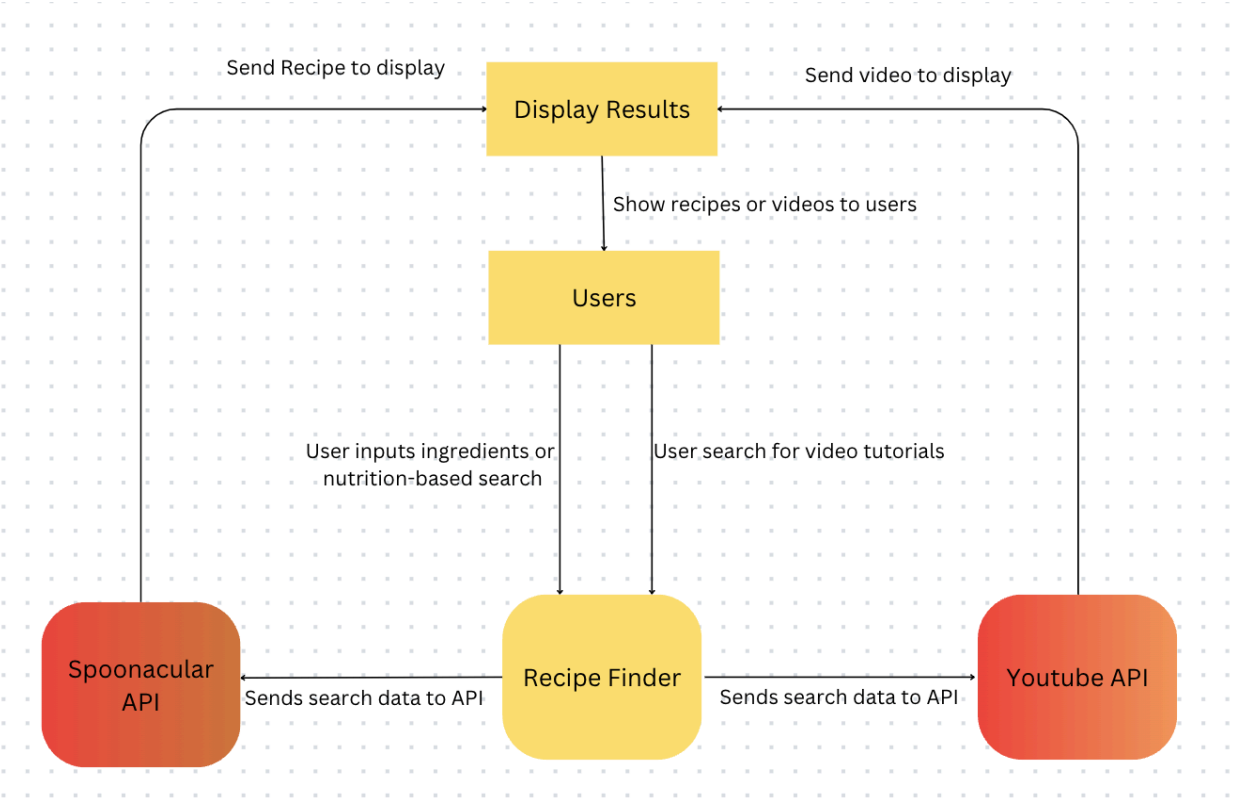
	APIintegration	Connect Spoonacular API	Test API connection	TBH	Project Initialization
		Connect Youtube API	Test API connection	TBH	Project Initialization
Feature and Interface	Spoonacular API	Implement Ingredient Search	Search Recipes by Ingredient	TBH	API integration
			Find recipes with available items	TBH	
		Implement Nutritional Search	Search Recipes by Nutritional Value	TBH	API Integration
		Implement Meal Planner	Organize recipes into meal plans	TBH	API integration
	Youtube Data API	Implement Video Search	Video Search for Cooking Tutorials	TBH	API Integration
		Implement Playlist Creation	Create Playlists for Cooking Videos	TBH	API Integration

		Implement Video Savings	Saving Favorite Cooking Videos	TBH	API Integration
	Models Development	Develop RecipeModel		TBH	API Feature Development
		Develop VideoModel		TBH	API Feature Development
	Views Development	Develop RecipeView		TBH	Models Development
		Develop VideoView		TBH	Models Development
	Controllers Development	Develop RecipeController		TBH	Models Development
		Develop VideoController		TBH	Models Development
	Interface Testing	Regular testing of each feature	Bug fixing	TBH	Feature and Interface Completion
	Deployment Research	Look into hosting options	Github Page	TBH	
Testing and	Code	Finalize	All features	TBH	Feature and

Deployment	Finalization	Spoonacular API features	should work properly		Interface Completion
		Finalize Youtube API	All features should work	TBH	Feature and Interface Completion
		Finalize UI components		TBH	Code Finalization
	Project Report	Write report on project scope		TBH	
	Website Deployment	Deploy the website	Test accessibility and performance	TBH	
	Regular Testing	Ongoing testing during deployment		Team	
Documentation	Complete Documentation	Write user guide, API integration guide and final report		Team	
		Prepare project presentation slides		TBH	

		Review and edit documentation		TBH	
		Submit final documentation and presentation materials		TBH	

Data FLow Diagram:



Data FlowOverview:

1. First, Users will send their requests through the Recipe Finder.
2. Recipe Finder then sends the request to either the Spoonacular API for recipe data or the Youtube Data API for video tutorials.
3. The 2 API will process the request and return relevant recipe data or video tutorial data to Display Results, which formats it for viewing.
4. Finally, Display Results shows the recipes or video tutorials to the Users.

Project Schedule:

- November 2-10
    - Setup project (React, Express, NodeJs)
    - Make the build blocks
        - HTML, CSS, Component Setup
    - Get APIs working
        - Setup basic structure
- November 11-15
    - API features should be almost completed/completed
        - Implement/Finalize:
            - RecipeModel & VideoModel
            - RecipeView & VideoView
            - RecipeController & VideoController
    - The interface should be nearly complete
    - Regular testing of features
    - Start looking at how we can deploy the website
- November 16-25
    - Ensure we have finalized code

        - APIS
        - Components
        - UI
    - Write a report on our project
    - Deploy the website
    - Work on a presentation
    - Continue regular testing of the features

Risk Assessment:

- Low Risk:
    - Running out of API requests
        - Store data into local storage so we do not need to make as many API calls
    - Inconsistent styles on different screens
        - Users with different screen sizes may see the UI differently. Use appropriate CSS that can handle changes in screen sizes
    - Users experience a UI bug
        - Do regular testing to ensure no bugs go through
    - Slow load times
        - Ensure we are using efficient algorithms and optimizing API calls
    - Missing features
        - Due to time constraints, we may not be able to implement all features. Ensure we are doing the most important features first in order to have a functional website.
- Medium Risk:
    - API key leak
        - Make sure to add our API keys to .gitignore. Before pushing to a branch or doing a pull request do multiple checks to ensure API keys are not being leaked.
    - Needing to change APIs
        - Test out the features to see if they still work. Some APIs remove some of their features, which can happen midway through the

project. Have a backup API with similar functionality so we can easily replace it.

    - Limited testing
        - Ensure we keep up regular testing whenever we create new features.
- High Risk:
    - Falling behind schedule
        - This will impact the overall project. Use Agile methods to ensure flexibility
    - Scope Creep
        - Make sure we have our requirements set in stone. Prioritize the main features

MVC Model:

1. Model:
    - The Model will be used to handle our data and interact with the APIs. We will be storing and retrieving data through local storage.
        - Ex. RecipeModel
            - saveRecipe
                - Saves a recipe to local storage
            - getRecipes
                - Gets all recipes from local storage
            - getNutritionalInfo
                - Gets nutritional info for a recipe from local storage
        - Ex. VideoModel
            - getVideos
                - Gets videos from local storage
            - saveVideos
                - Saves a video to local storage
2. View:

- The View will be used to render a display for the user based on the data from the Model.
    - Ex. RecipeList
        - Displays the list of recipes
    - Ex. RecipeInfo
        - Display the recipe information
    - Ex. VideoList
        - Displays the list of saved videos
    - Ex. Video
        - Displays a single video
    - Ex. Search bar
        - Displays a search bar so users can find recipes

3. Controller:
    - The Controller will handle user interaction and the communication between the Model and View.
        - Ex. RecipeController
            - searchRecipe
                - When uses the search bar from view RecipeController will call the getRecipes function from RecipeModel
            - saveRecipeButton
                - When user clicks the save recipe button in the View it will call the saveRecipe function in the Model
        - Ex. VideoController
            - searchVideo
                - When users use the search bar for video from the view it will call the getVideos from the VideoModel
            - saveVideoButton
                - When users click the save video button in the View it will call the saveVideo function in the Model.