Prototype Report:

The SSH Recipe Prototype is a Python-based project that emulates an intelligent recipe suggestion system. It captures data from the SSH camera of a simulated smart fridge and suggests personalized recipes to minimize food wastage and simplify meal planning. The project was developed collaboratively, with each team member contributing key functionalities to ensure the success of this prototype.

Features and Functionality

The project focuses on the following key features:

- **Ingredient Detection:** Simulates ingredient detection in a fridge environment based on user-selected scenarios.
- **Recipe Suggestion Engine:** Suggests recipes based on the detected ingredients and user preferences.
- **Graphical User Interface (GUI):** Provides a graphical interface for selecting fridge scenarios, detecting ingredients, and displaying recipe suggestions.
- **Handling Missing Ingredients:** Interactively identifies and displays missing ingredients for suggested recipes, enhancing usability.

Each component plays a crucial role in ensuring the system is reliable and user-friendly.

Implementation and Testing

Modular Design

The prototype was implemented modularly in Python to facilitate easy understanding and maintenance. Each key component: ingredient detection, recipe matching, and user interface interactions was assigned to separate modules. This modular approach allows focused development and simplifies debugging and future improvements.

Ingredient Detection

ingredient_detection.py simulates ingredient detection based on predefined fridge scenarios. For instance:

- **"**Breakfast" Scenario: Returns ingredients like eggs, toast, and cheese.
- "Empty Fridge" Scenario**:** Returns an empty list, indicating no available ingredients.

This module categorizes items into groups based on common fridge scenarios, enabling the system to accurately identify available ingredients.

Recipe Suggestion Engine

recipe_suggestion_engine.py implements the logic for matching available ingredients to recipes, filtering by meal type, and identifying missing ingredients.
The suggest_recipes function efficiently searches the modular recipe database to find suitable

matches. It not only highlights recipes that can be fully made but also indicates missing ingredients for partially complete recipes, making the program robust and user-friendly.

Graphical User Interface

gui.py contains the graphical interface logic using Tkinter. The GUI is intuitive and features the following:

- **Scenario Selection:** A combo box allows users to select fridge scenarios such as "Breakfast" and "Dessert."
- **Detect Ingredients Button:** Initiates the ingredient detection process and displays recipe suggestions.
- **Scrollable Text Widget:** Displays detected ingredients, suggested recipes, missing items, and cooking instructions.

Example Functionality:

- **"Breakfast" Scenario:** Displays detected ingredients, highlights missing items, and provides preparation instructions.
- **"Empty Fridge" Scenario:** Informs users that no recipes are available due to the lack of ingredients.

Testing

Comprehensive unit tests were developed to validate the critical functions of each module, ensuring the system's reliability across various scenarios.

test_functions.py

This script tests the ingredient detection and recipe suggestion capabilities by importing the detect_ingredients and suggest_recipes functions. It verifies that ingredient detection works correctly for various fridge scenarios and that recipe suggestions are appropriate based on the detected ingredients.

tests/test_suggestion_engine.py

This unit test file extensively tests the suggest_recipes function using Python's unittest framework. Key test cases include:

- Full Match Test: Verifies that recipes with all required ingredients are suggested without listing missing items.
- Partial Match Test: Ensures that recipes missing some ingredients are suggested, accurately identifying the missing items.
- No Ingredients Test: Confirms that even with no available ingredients, recipes are suggested with all ingredients marked as missing.

- No Recipes Available Test: Checks that an empty list is returned when no recipes exist in the database.
- All Recipes Available Test: Ensures that all recipes are suggested when all ingredients are available.

These tests are invaluable in maintaining the system's reliability and accuracy.

Documentation

A comprehensive README was developed to guide users through cloning the repository, running the application, and understanding its functionalities. The documentation includes:

- Cloning Instructions: Guides users on how to clone and set up the project.
- System Functionality Overview: Explains fridge content simulation and recipe database integration.
- Minimal Dependencies: Highlights that only Python 3.x and Tkinter are required.
- Usage Instructions: Provides step-by-step guidance on using the GUI and running tests.

Continuous Integration

Collaboration and code quality were significantly enhanced through version control and a continuous integration (CI) workflow using GitHub Actions. The CI pipeline automatically runs tests across multiple Python versions whenever new changes are introduced. This ensures that the codebase remains stable and compatible across different environments.

Project Structure

- main.py: Entry point of the application. Runs the GUI and orchestrates the interaction between ingredient detection and recipe suggestion.
- gui.py: Contains the graphical interface logic using Tkinter. Handles user interactions like scenario selection and displays suggested recipes.
- ingredient_detection.py: Simulates ingredient detection based on the chosen scenario and returns a corresponding list of ingredients.
- recipe_suggestion_engine.py: Implements the logic for matching available ingredients to recipes, filtering by meal type, and identifying missing ingredients.
- recipe_database.py: Provides a predefined list of recipes and their required ingredients.
- tests/
  Contains unit tests that verify the functionality of the recipe suggestion engine, ingredient detection, and other components. Running these tests ensures that recent changes haven’t introduced regressions.

- .github/workflows/ci.yml: Defines the GitHub Actions workflow for continuous integration, running tests automatically across multiple Python versions.
- README.md: Provides comprehensive documentation on the project's features, usage, and structure**.**

Conclusion

The SSH Recipe Prototype exemplifies effective collaboration and proper division of labour within a development team. Through extensive testing of its functionality, the GUI and suggestion engine were enhanced for better usability and accuracy. The foundational work on ingredient detection, recipe database management, and the main application logic ensured a robust and reliable system. The combined efforts resulted in a functional prototype that addresses food wastage by simplifying meal planning. This project not only meets its initial goals but also lays a solid foundation for future improvements and scalability**.**