Shelf Smart

Software Architecture Document

Version 1.2

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 09/4/24 | 1.0 | Initial ideas, brainstorming requirements, and definitions. | Ansuman Sharma, Yadhunath Tharakeswaran, Shayaan Mohammed, Kundhana Pragnya. |
| 12/4/24 | 1.1 | Finalizing ideas, Declaring functions, Started with backend, initial codings. | Ansuman Sharma, Yadhunath Tharakeswaran, Shayaan Mohammed, Kundhana Pragnya. |
| 14/4/24 | 1.2 | Created the homepage of the website, working further in the backend, connecting to Restful API. | Ansuman Sharma, Yadhunath Tharakeswaran, Shayaan Mohammed, Kundhana Pragnya. |

Table of Contents

1. Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.4 References 4

1.5 Overview 4

2. Architectural Representation 4

3. Architectural Goals and Constraints 4

4. Use-Case View 4

4.1 Use-Case Realizations 5

5. Logical View 5

5.1 Overview 5

5.2 Architecturally Significant Design Packages 5

6. Interface Description 5

7. Size and Performance 5

8. Quality 5

Software Architecture Document

# Introduction

The Software Architecture Document (SAD) sets the stage for the development of Shelf-Smart, providing a comprehensive blueprint for its architectural design. This document outlines the structure, components, and interactions of the system, aiming to guide developers, stakeholders, and users through the implementation phase. By elucidating the architectural decisions and design rationale, the SAD facilitates a coherent and efficient development process, ensuring alignment with project goals and user requirements

## Purpose

The purpose of this SAD is to delineate the architectural framework of Shelf-Smart, elucidating the organization, behavior, and interdependencies of its constituent parts. It aims to provide a clear understanding of the system's structure and functionality to developers, stakeholders, and users, facilitating effective communication and collaboration throughout the development lifecycle. Additionally, the SAD serves as a reference guide for future maintenance and evolution of the system, fostering scalability and adaptability.

## Scope

This SAD pertains to the architectural design of the Shelf-Smart web application, encompassing its structural components, interfaces, and interactions. It delineates the system architecture, including both functional and non-functional aspects, and influences the development process to ensure coherence with the specified architecture. Moreover, this document informs decisions regarding technology selection, system integration, and performance optimization, guiding the implementation of Shelf-Smart

## Definitions, Acronyms, and Abbreviations

For clarity and consistency, this section provides definitions of key terms, acronyms, and abbreviations pertinent to the Shelf-Smart project. Refer to the project's Glossary for a comprehensive list of defined terms.

Here are some basic definitions, Acronyms and Abbreviations

* Ingredient: A commodity added to the user's pantry.
* Spice (child class of Ingredient): Resembles an ingredient but doesn't require an expiry date.
* Recipe: Consists of the dish name and a brief description.
* User: Any individual utilizing the Shelf-Smart application for pantry management.

## References

* 01 - Project plan Shelf-Smart 2/25/2024
* 02 - Software Requirements Specifications Shelf-Smart 3/24/2024
* Project glossary: comprehensive list of defined terms, acronyms and abbreviations relevant to the project

## Overview

The SAD is organized into distinct sections, each addressing specific aspects of the system architecture. It begins with an Introduction providing an overview of the document, followed by sections detailing the architectural purpose, scope, definitions, and references. Additionally, the SAD provides an overview of the architectural representation, goals, constraints, and quality decisions guiding the development of Shelf-Smart. This structured approach aims to facilitate a comprehensive understanding of the system architecture and its implications for the project

# Architectural Representation

1. **Logical View:**

This viewpoint emphasizes the system's logical organization while abstracting actual implementation details. This includes:

* Inventory Management Module: Provides logical representations of inventory data structures and procedures for adding, modifying, and retrieving inventory items.
* The Ingredient Classification Module defines logical models for categorizing ingredients based on kind, using classification algorithms.
* Recommendation Engine: Demonstrates logical frameworks for analyzing available components and making personalized recipe recommendations.
* User Interface Components: Defines the logical representations of user interface elements such as input forms, inventory displays, and recommendation presentations.

1. **Development View:**

This viewpoint focuses on software development, stressing modular structure and development environment. It includes:

* Module Structure: Describes how the system is divided into components such as inventory management, classification, recommendation, and user interface.
* The development environment: specifies the programming languages (e.g., C++), frameworks, libraries, and tools used in development. It may also specify development processes and version control practices.

1. **Process View:**

This view emphasizes the dynamic behavior of the system, focusing on the interactions between components and processes. It includes:

* Request Processing: Describes the flow of requests and responses between clients and servers, including how user requests are handled and processed.
* Data Flow: Shows the movement of data within the system, including how inventory data is processed by modules like classification and suggestion.

1. **Physical View:**

This view details the physical deployment and distribution of system components across hardware infrastructure. It includes:

* Server Component: Refers to physical servers or cloud instances which host backend components such as databases, application servers, and processing modules.
* Client Devices: Describes the physical devices that interface with the system, such as desktops, laptops, and mobile devices, taking into account various platforms and form factors.

Overall, these views provide a comprehensive representation of the Shelf-Smart architecture, addressing its logical structure, development organization, dynamic behavior, and physical deployment.

# Architectural Goals and Constraints

The architectural goals for Shelf-Smart revolve around crafting a dynamic and user-centric web application tailored for pantry management and recipe recommendations. Key objectives include:

* User-Centric Design: Shelf-Smart aims to prioritize user experience, offering an intuitive interface for efficient pantry organization and seamless recipe discovery.
* Functional Precision: The system is engineered to accurately interpret user inputs, manage ingredient data comprehensively, and deliver precise recipe recommendations based on individual preferences and dietary requirements.
* Scalability and Accessibility: Shelf-Smart is architected with scalability and accessibility in mind, ensuring compatibility across various devices and platforms to cater to a diverse user base.
* Security and Data Privacy: The architecture emphasizes robust security measures to protect user data, implementing encryption protocols and authentication mechanisms to safeguard sensitive information.
* Integration and Adaptability: Shelf-Smart is designed to seamlessly integrate with external APIs and services, facilitating future enhancements and customizations to meet evolving user needs.

Constraints:

* Programming Language: Shelf-Smart is developed using C++, harnessing its performance capabilities and compatibility with web development frameworks.
* Development Environment: The application is developed within the Linux environment, adhering to platform-specific dependencies and requirements.
* Team Collaboration: The development team follows a collaborative approach, with each member contributing to various modules based on their expertise and assigned responsibilities as outlined in the project plan.
* Schedule: Development activities are primarily scheduled on Wednesdays and Fridays, aligning with the availability of team members and project timelines.
* Completion Deadline: The project is targeted for completion by early May, necessitating efficient resource allocation and timely execution of development tasks to meet the established deadline.

By adhering to these architectural goals and constraints, Shelf-Smart endeavors to deliver a sophisticated and user-friendly solution for pantry management and culinary exploration, while ensuring adherence to project objectives and constraints within the specified timeframe.

# Logical View

## Overview

The logical view of the Shelf-Smart architecture outlines the system's decomposition into various packages and layers, highlighting the organization of architecturally significant design modules.

## Architecturally Significant Design Modules or Packages

* Inventory Management Module: This module encapsulates the core functionalities related to managing the user's pantry inventory. It includes classes responsible for handling inventory data structures, such as Ingredients and Spice. The Inventory Management Module facilitates operations for adding, modifying, and retrieving inventory items.
* Ingredient Classification Module: This module encompasses classes involved in categorizing ingredients based on their characteristics. It defines logical models and algorithms for classifying ingredients, ensuring efficient organization and retrieval. The module aids in enhancing the user experience by facilitating intuitive inventory management.
* Recommendation Engine: This module orchestrates the process of analyzing available inventory components and generating personalized recipe recommendations. It employs logical frameworks to interpret user preferences and dietary requirements, delivering tailored suggestions for culinary exploration. The module fosters user engagement by providing relevant and appealing recipe options.
* User Interface Components: This components package comprises classes representing various elements of the graphical user interface (GUI). It includes interfaces for input forms, inventory displays, and recommendation presentations. These components ensure a seamless and intuitive user experience, enabling efficient interaction with Shelf-Smart functionalities.

# Interface Description

1. **Input Interface:**

* Screen Format: Users will interact with Shelf-Smart using a graphical user interface (GUI) on their devices.
* Valid Inputs: Users can enter food items, quantities, and expiration dates into the designated fields. They can also choose from a list of established ingredients or enter their own.
* Resulting Outputs: After entering the relevant information, users will click the submit button to proceed. The system will validate the inputs and indicate any mistakes that exist. After a successful submission, the supplied data will be processed and users will receive confirmation notifications.

1. **Inventory Showcase Interface:**

* Screen Format: The inventory showcase will be provided in a tabular or grid format, displaying food products, quantities, and expiry dates.
* Valid Inputs: Users can access their inventory without providing any direct input. However, users may be able to filter or sort the inventory depending on a variety of parameters such as type, amount, or expiration date.
* Resulting Outputs: The inventory display will give consumers a thorough picture of their pantry necessities, allowing them to swiftly determine which components are accessible.

1. **Personalized Recommendations Interface:**

* Screen Format: After entering their pantry ingredients, customers will receive a list of customized food recommendations.
* Valid Inputs: Users can enter particular ingredients or allow Shelf-Smart to select items at random from their pantry. They may also have the ability to change their choices, such as dietary restrictions or cuisine preferences.
* Resulting Outputs: Shelf-Smart will provide 3-5 dish recommendations, each with a short description. Users can choose to view calorie information for the recommended dishes. Each tip will be clickable, allowing readers to access comprehensive recipes if desired.

These interfaces collectively provide users with a seamless and intuitive experience while using

Shelf-Smart, allowing them to manage their pantry and prepare meals more efficiently.

# Quality

The quality of Shelf-Smart is assured through rigorous testing procedures outlined in the Test Case Document, overseen by TA's and the Professor. With a focus on reliability, the system ensures accurate interpretation of user inputs and seamless functionality, while its modular architecture facilitates future extensibility and adaptability. Additionally, Shelf-Smart's interface accessibility via standard devices ensures portability and user convenience