Proposed Software Architecture for the PantrySense Application

Gray Meredeth

Troy University

Software Engineering I

Mr. Reggie Haseltine

February 15, 2026

Introduction

As the PantrySense software application progresses from requirements definition into detailed design activities, selecting the appropriate architectural style is most critical for ensuring project maintainability, scalability, and overall stakeholder understanding. Software architecture serves as a high-level blueprint that defines the system structure and interaction among components. The following paper presents and justifies the selection of the Model–View–Controller (MVC) architectural style for PantrySense and explains how data is input, processed, stored, and output by the system.

Overview of the Model–View–Controller Architecture

The Model–View–Controller (MVC) architecture separates an application into three distinct components, each with a well-defined responsibility. The Model manages application data and business logic, the View handles user interface presentation, and the Controller processes user input and coordinates interactions between the View and the Model (Pressman & Maxim, 2020).

Justification for Selecting MVC

The MVC architectural style was selected over data-flow and main program/subprogram architectures because it better supports PantrySense's functional and nonfunctional requirements. PantrySense is an event-driven application where the user initiates actions that result in updates to shared application state. MVC allows the UI to evolve independently of business logic and supports agile SCRUM development (Sommerville, 2016).

Architectural Component Responsibilities

In the PantrySense MVC architecture, the View consists of the mobile and web user interfaces through which users interact with the system. The Controller interprets user actions and issues commands to the Model. The Model contains the business logic, manages application data, and interacts with the database.

Data Management

PantrySense inputs data through user interaction such as item entry and barcode scanning. A database is required to store user profiles, inventory items, notifications, and recipes. The Model layer is solely responsible for database access to ensure data integrity.

Conclusion

The Model–View–Controller architectural style provides a clear, scalable, and maintainable structure for the PantrySense application. Its separation of concerns supports incremental development and long-term sustainability.

References

Pressman, R. S., & Maxim, B. R. (2020). Software engineering: A practitioner’s approach (9th ed.). McGraw-Hill Education.