MMOM

Version <1.0>

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 30/Oct/24 | 1.0 | Initial Version | Tran Minh Duc |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

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. Flow of Events—Design 4

3. Derived Requirements 7

# Introduction

## Purpose

This Use-Case Realization Specification describes the design model for implementing the "Create Own Recipe" use case in the "Make Your Own Meals" application. It provides detailed information about the interaction between objects and the structure of the classes involved in the implementation.

## Scope

This document covers the realization of the Create Own Recipe use case, including:

* Design models following MVC pattern
* Object interactions through sequence diagrams
* Class structure and relationships
* Implementation requirements

## Definitions, Acronyms, and Abbreviations

* MVC: Model-View-Controller
* DTO: Data Transfer Object
* UI: User Interface
* CRUD: Create, Read, Update, Delete
* DB: Database

## References

* Use Case Specification - Create Own Recipe v1.0
* System Requirements Specification v1.0
* MVC Design Pattern Guidelines v1.0.

## Overview

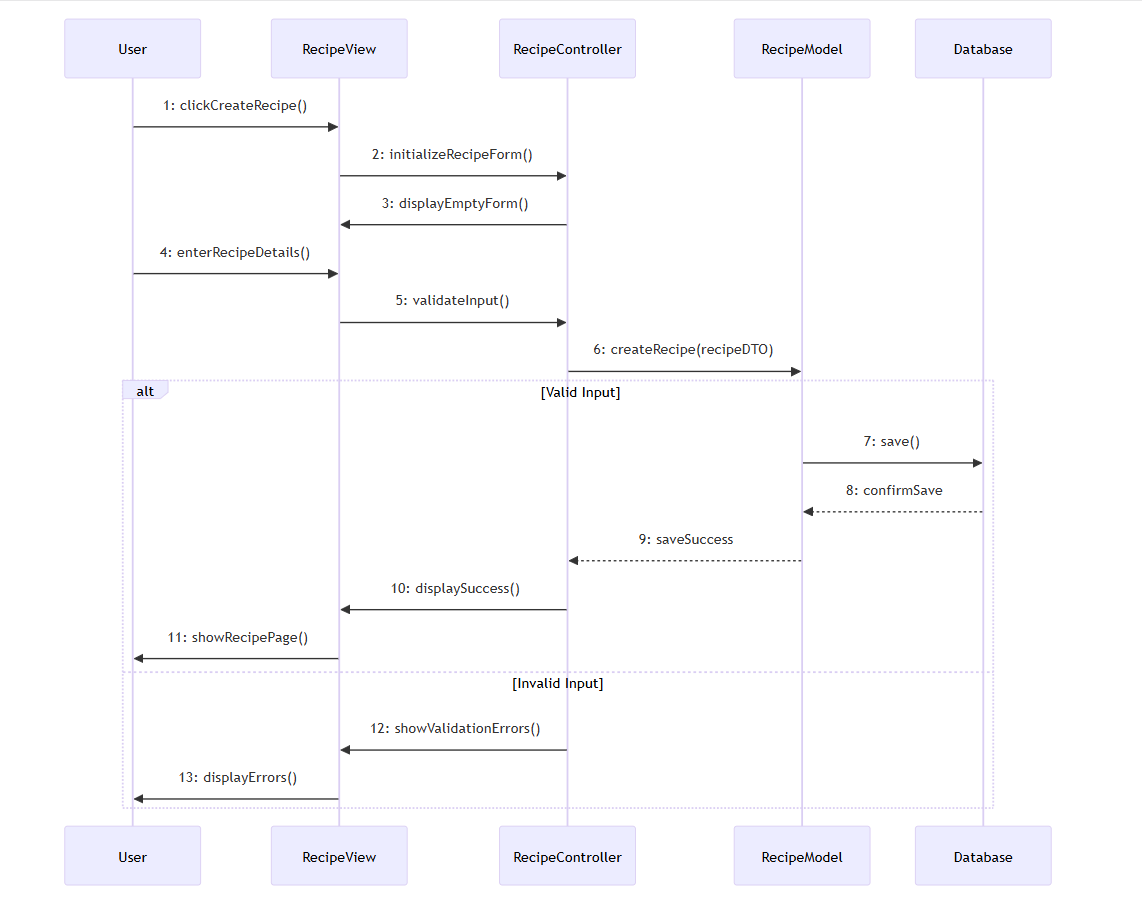
This document is organized into three main sections:

* Introduction - Provides context and background information
* Flow of Events—Design - Details the design model through diagrams
* Derived Requirements - Lists additional implementation requirements

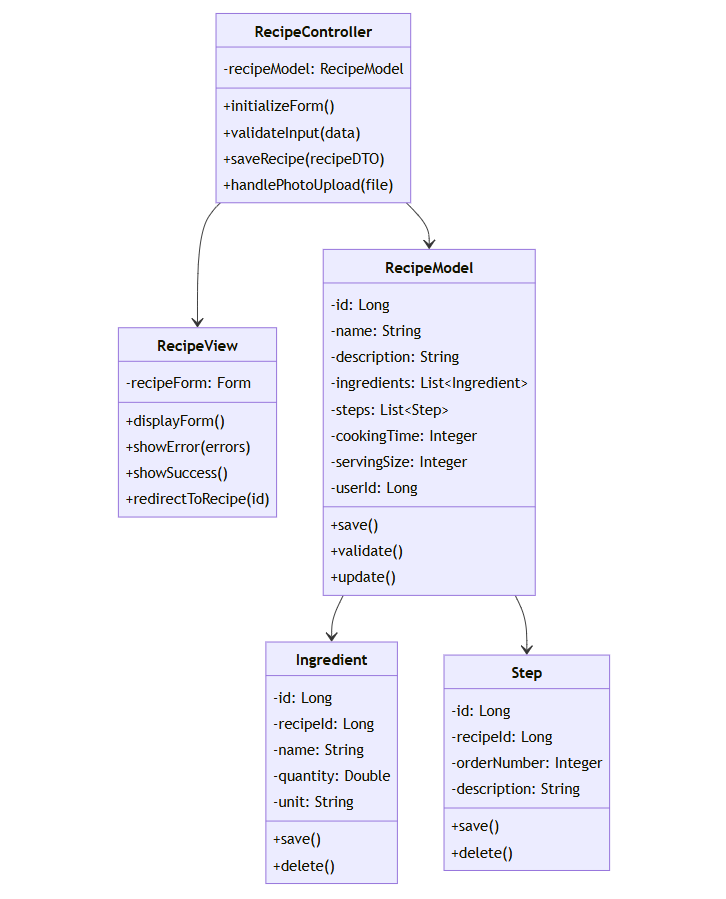
# Flow of Events—Design

[A textual description of how the use case is realized in terms of collaborating objects. Its main purpose is to summarize the diagrams connected to the use case and to explain how they are related.]

## Sequence diagram



## Class diagram



# Derived Requirements

## Performance Requirements

Recipe save operation must complete within 3 seconds

Photo upload and processing must complete within 5 seconds

UI must remain responsive during save operations

## Security Requirements

Input validation must be performed on both client and server side

File upload validation must check file type and size

User authentication must be verified before any database operation

## Database Requirements

Transactions must be used to ensure data consistency

Foreign key constraints must be implemented for recipe relationships

Indexes must be created for frequent query patterns

## Error Handling Requirements

All database errors must be logged with stack traces

User-friendly error messages must be displayed

Validation errors must highlight specific form fields

## UI Requirements

Form must be responsive for mobile devices

Photo upload must show progress indicator

Success/error messages must be clearly visible

Form must preserve data on validation failures