Software Requirements Specification (SRS)

TasteBud Recipe Sharing Platform

Table of Contents

[1. Introduction 3](#_Toc201526398)

[1.1 Purpose 3](#_Toc201526399)

[1.2 Scope 3](#_Toc201526400)

[1.3 Definitions and Acronyms 3](#_Toc201526401)

[2. Overall Description 4](#_Toc201526402)

[2.1 Product Perspective 4](#_Toc201526403)

[2.2 User Classes 4](#_Toc201526404)

[2.3 Operating Environment 4](#_Toc201526405)

[2.4 Assumptions and Dependencies 5](#_Toc201526406)

[3. System Features 5](#_Toc201526407)

[3.1 User Authentication and Registration 5](#_Toc201526408)

[3.1.1 User Registration 5](#_Toc201526409)

[3.1.2 User Login 6](#_Toc201526410)

[3.2 Recipe Management 6](#_Toc201526411)

[3.2.1 Recipe Creation 6](#_Toc201526412)

[3.2.2 Recipe Viewing and Browsing 7](#_Toc201526413)

[3.2.3 Recipe Editing and Deletion 7](#_Toc201526414)

[3.3 Search and Discovery 8](#_Toc201526415)

[3.3.1 Recipe Search 8](#_Toc201526416)

[3.3.2 Recipe Categorization 9](#_Toc201526417)

[3.4 Rating System 9](#_Toc201526418)

[3.4.1 Recipe Rating 9](#_Toc201526419)

[3.4.2 Rating Display and Calculation 10](#_Toc201526420)

[3.5 User Dashboard and Profile 10](#_Toc201526421)

[3.5.1 User Dashboard 10](#_Toc201526422)

[3.5.2 My Recipes Management 11](#_Toc201526423)

[3.6 Administrative Features 11](#_Toc201526424)

[3.6.1 User Management 11](#_Toc201526425)

[3.6.2 Content Moderation 12](#_Toc201526426)

[4. External Interface Requirements 12](#_Toc201526427)

[4.1 User Interface Requirements 12](#_Toc201526428)

[4.2 Hardware Interfaces 12](#_Toc201526429)

[4.3 Software Interfaces 13](#_Toc201526430)

[5. Non-Functional Requirements 13](#_Toc201526431)

[5.1 Performance Requirements 13](#_Toc201526432)

[5.2 Security Requirements 13](#_Toc201526433)

[5.3 Reliability Requirements 13](#_Toc201526434)

[5.4 Usability Requirements 14](#_Toc201526435)

[5.5 Compatibility Requirements 14](#_Toc201526436)

[6. Database Requirements 14](#_Toc201526437)

[6.1 Core Tables Structure 14](#_Toc201526438)

[6.2 Relationships 16](#_Toc201526439)

[6.3 Indexes 16](#_Toc201526440)

[7. User Stories 16](#_Toc201526441)

[7.1 Anonymous User Stories 16](#_Toc201526442)

[7.2 Registered User Stories 17](#_Toc201526443)

[7.3 Administrator Stories 17](#_Toc201526444)

[8. Acceptance Criteria Summary 18](#_Toc201526445)

[8.1 Must Have Features (MVP) 18](#_Toc201526446)

[8.2 Should Have Features 18](#_Toc201526447)

[8.3 Could Have Features (Future Enhancements) 18](#_Toc201526448)

[8.4 Success Criteria 19](#_Toc201526449)

[9. Technical Implementation Notes 19](#_Toc201526450)

[9.1 Architecture Requirements 19](#_Toc201526451)

[9.2 Development Timeline 20](#_Toc201526452)

[9.3 Quality Assurance 20](#_Toc201526453)

# 1. Introduction

## 1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for TasteBud Recipe Sharing Platform, a web-based application that allows home cooking enthusiasts to share, discover, and rate recipes. The system will provide both a traditional web interface and a modern API for future mobile app development.

## 1.2 Scope

TasteBud Recipe Sharing Platform is a web application that will:

* Allow users to register and create personal profiles
* Enable users to create, edit, and manage their own recipes
* Provide a platform for browsing and searching recipes by various criteria
* Implement a rating system for recipes (1-5 stars)
* Offer administrative tools for content management
* Provide both MVC web interface and RESTful API endpoints

## 1.3 Definitions and Acronyms

* **Recipe Creator**: A registered user who creates and shares recipes on the platform
* **Recipe Browser**: Any user (registered or anonymous) who views and searches recipes
* **Admin**: System administrator with full platform oversight and management capabilities
* **API**: Application Programming Interface for mobile app integration
* **MVC**: Model-View-Controller architecture pattern
* **CRUD**: Create, Read, Update, Delete operations

# 2. Overall Description

## 2.1 Product Perspective

TasteBud Recipe Sharing Platform is a standalone web application built using ASP.NET MVC with a 3-tier architecture. The system includes both a traditional web interface and a RESTful API to support future mobile application development.

## 2.2 User Classes

**1. System Administrator**

* Full system access and control
* User and recipe management capabilities
* Platform oversight and content moderation

**2. Registered User**

* Can create, edit, and delete their own recipes
* Can rate recipes created by other users
* Can browse and search all recipes
* Access to personal dashboard and recipe management

**3. Anonymous User**

* Can browse and search all published recipes
* Can view recipe details and ratings
* Cannot create recipes or rate recipes

## 2.3 Operating Environment

* **Server**: Windows Server with IIS
* **Database**: SQL Server with Entity Framework Code First
* **Framework**: ASP.NET MVC and ASP.NET Web API
* **Authentication**: ASP.NET Identity
* **Frontend**: HTML5, CSS3, JavaScript (responsive design)
* **Browser Support**: All modern web browsers (Chrome, Firefox, Edge, Safari)

## 2.4 Assumptions and Dependencies

* Users will have access to modern web browsers
* Internet connectivity is available for all users
* Standard web server hosting environment
* Initial user base of 100-500 users in first month
* No image upload functionality in initial version (text-only recipes)
* Email verification system will be implemented in future versions

# 3. System Features

## 3.1 User Authentication and Registration

### 3.1.1 User Registration

**Description**: New users can register to create and manage recipes  
**Priority**: High

**Functional Requirements**:

* System shall provide user registration form with following fields:
  + Email Address (unique, required)
  + Password (required, minimum 8 characters)
  + Password Confirmation (required)
  + First Name (required)
  + Last Name (required)
  + Display Name/Username (optional)
* System shall validate email uniqueness across all users
* System shall enforce password strength requirements (minimum 8 characters, mixed case, numbers)
* System shall hash and store passwords securely using ASP.NET Identity
* System shall create user profile automatically upon successful registration
* System shall redirect users to login page after successful registration

### 3.1.2 User Login

**Description**: Registered users can securely authenticate and access the platform  
**Priority**: High

**Functional Requirements**:

* System shall authenticate users using email and password combination
* System shall provide "Remember Me" functionality for extended sessions
* System shall provide "Forgot Password" functionality (future enhancement)
* System shall redirect authenticated users to their dashboard
* System shall maintain secure session management
* System shall provide logout functionality
* System shall prevent unauthorized access to protected pages

## 3.2 Recipe Management

### 3.2.1 Recipe Creation

**Description**: Registered users can create new recipes with comprehensive details  
**Priority**: High

**Functional Requirements**:

* Recipe creation form shall include following required fields:
  + Recipe Title (required, maximum 100 characters)
  + Description (required, maximum 500 characters)
  + Ingredients List (required, text area)
  + Step-by-step Instructions (required, text area)
  + Preparation Time in minutes (required, numeric)
  + Cooking Time in minutes (required, numeric)
  + Number of Servings (required, numeric)
  + Difficulty Level (required, dropdown: Easy/Medium/Hard)
  + Category (required, dropdown: Breakfast/Lunch/Dinner/Dessert/Snacks)
* System shall validate all required fields before saving
* System shall associate recipe with authenticated user
* System shall automatically set creation date and time
* System shall provide save and cancel options
* System shall redirect to recipe details page after successful creation

### 3.2.2 Recipe Viewing and Browsing

**Description**: All users can browse and view recipe details  
**Priority**: High

**Functional Requirements**:

* System shall display recipe list on home page showing:
  + Recipe title and category
  + Creator name
  + Average rating (if any)
  + Preparation and cooking time
  + Difficulty level
  + Creation date
* System shall provide recipe details page displaying:
  + Complete recipe information
  + Creator information
  + Average rating and number of ratings
  + All recipe details formatted for easy reading
* System shall allow anonymous users to view all recipes
* System shall display latest recipes on home page
* System shall highlight top-rated recipes on home page

### 3.2.3 Recipe Editing and Deletion

**Description**: Users can modify and delete their own recipes  
**Priority**: High

**Functional Requirements**:

* System shall allow users to edit only their own recipes
* Edit form shall pre-populate with existing recipe data
* System shall validate all changes before saving
* System shall maintain recipe creation date but update modification date
* System shall provide delete functionality with confirmation dialog
* System shall prevent deletion if user is not the recipe owner
* System shall remove all associated ratings when recipe is deleted
* System shall redirect to user's recipe list after edit/delete operations

## 3.3 Search and Discovery

### 3.3.1 Recipe Search

**Description**: Users can search for recipes using various criteria  
**Priority**: High

**Functional Requirements**:

* System shall provide search functionality by:
  + Recipe title (partial match, case-insensitive)
  + Ingredients (partial match within ingredients list)
  + Recipe description
* System shall provide advanced search filters:
  + Category selection
  + Difficulty level
  + Preparation time range
  + Cooking time range
  + Minimum rating
* System shall display search results with same information as recipe browse
* System shall handle empty search results gracefully
* System shall provide search result count
* System shall maintain search criteria in URL for bookmarking

### 3.3.2 Recipe Categorization

**Description**: Recipes are organized by predefined categories  
**Priority**: Medium

**Functional Requirements**:

* System shall maintain following recipe categories:
  + Breakfast
  + Lunch
  + Dinner
  + Dessert
  + Snacks
* System shall allow browsing recipes by category
* System shall display category-specific recipe counts
* System shall provide category navigation in main menu
* System shall validate category selection during recipe creation

## 3.4 Rating System

### 3.4.1 Recipe Rating

**Description**: Registered users can rate recipes from 1 to 5 stars  
**Priority**: High

**Functional Requirements**:

* System shall allow registered users to rate recipes (1-5 stars)
* System shall prevent users from rating their own recipes
* System shall allow only one rating per user per recipe
* System shall provide star-based rating interface
* System shall update recipe's average rating immediately after rating submission
* System shall display current user's rating if already rated
* System shall allow users to update their existing rating
* System shall track total number of ratings per recipe

### 3.4.2 Rating Display and Calculation

**Description**: System calculates and displays recipe ratings  
**Priority**: High

**Functional Requirements**:

* System shall calculate average rating for each recipe
* System shall display average rating using star visualization
* System shall show number of ratings received
* System shall round average rating to nearest 0.5 stars for display
* System shall sort recipes by rating when requested
* System shall highlight top-rated recipes on home page
* System shall handle recipes with no ratings appropriately

## 3.5 User Dashboard and Profile

### 3.5.1 User Dashboard

**Description**: Registered users have access to personal dashboard  
**Priority**: Medium

**Functional Requirements**:

* Dashboard shall display user's personal information
* Dashboard shall show user's recipe statistics:
  + Total number of recipes created
  + Average rating received across all recipes
  + Most popular recipe (highest rated)
  + Recent recipe activity
* Dashboard shall provide quick links to:
  + Create new recipe
  + View all my recipes
  + Edit profile
* Dashboard shall display recent ratings received

### 3.5.2 My Recipes Management

**Description**: Users can manage all their created recipes  
**Priority**: Medium

**Functional Requirements**:

* System shall provide "My Recipes" page showing all user's recipes
* Page shall display recipe list with:
  + Recipe title and category
  + Creation/modification date
  + Average rating and rating count
  + Quick edit/delete actions
* System shall provide sorting options (by date, rating, title)
* System shall provide filtering by category
* System shall show recipe statistics and performance

## 3.6 Administrative Features

### 3.6.1 User Management

**Description**: Administrators can manage platform users  
**Priority**: Medium

**Functional Requirements**:

* Admin shall view list of all registered users
* Admin interface shall display:
  + User email and name
  + Registration date
  + Number of recipes created
  + Account status (active/inactive)
* Admin shall be able to:
  + Deactivate user accounts
  + View user's recipe list
  + Delete inappropriate content
* Admin shall have access to user activity logs

### 3.6.2 Content Moderation

**Description**: Administrators can moderate recipes and content  
**Priority**: Low

**Functional Requirements**:

* Admin shall view all recipes in the system
* Admin shall be able to delete inappropriate recipes
* Admin shall be able to view recipe reports (future enhancement)
* Admin shall access recipe statistics and analytics
* Admin shall have override permissions for all content management

# 4. External Interface Requirements

## 4.1 User Interface Requirements

* **Responsive Design**: Interface shall work seamlessly on desktop, tablet, and mobile devices
* **Clean Layout**: Simple, uncluttered design focused on recipe readability
* **Large Clickable Elements**: Buttons and links optimized for users with messy hands while cooking
* **Consistent Navigation**: Intuitive navigation structure across all pages
* **Accessibility**: Interface shall follow web accessibility guidelines (WCAG 2.1)
* **Loading States**: Clear feedback during data loading and form submissions

## 4.2 Hardware Interfaces

* Standard web browser interface
* No special hardware requirements
* Touch-screen compatible for mobile devices

## 4.3 Software Interfaces

* **Database**: SQL Server integration using Entity Framework
* **Authentication**: ASP.NET Identity for user management
* **API**: RESTful Web API for future mobile app integration
* **Session Management**: ASP.NET session management
* **Caching**: Application-level caching for performance

# 5. Non-Functional Requirements

## 5.1 Performance Requirements

* **Response Time**: Pages shall load within 3 seconds under normal conditions
* **Concurrent Users**: System shall support up to 500 concurrent users
* **Database Performance**: Database queries shall execute within 2 seconds
* **Search Performance**: Recipe search results shall return within 1 second
* **Scalability**: Architecture shall support growth to 10,000 users

## 5.2 Security Requirements

* **Password Security**: Passwords shall be hashed using ASP.NET Identity
* **Session Management**: Secure session handling with appropriate timeout
* **Input Validation**: All user inputs shall be validated and sanitized
* **SQL Injection Protection**: Parameterized queries shall prevent SQL injection
* **XSS Protection**: Output encoding shall prevent cross-site scripting
* **CSRF Protection**: Anti-forgery tokens shall protect form submissions
* **Authentication**: Secure authentication using ASP.NET Identity

## 5.3 Reliability Requirements

* **Uptime**: System shall maintain 99% uptime during peak hours (6 PM - 10 PM)
* **Data Integrity**: All recipe and user data shall be preserved during system operations
* **Error Handling**: Graceful error handling with user-friendly messages
* **Data Backup**: Regular automated database backups
* **Recovery**: System shall recover from failures within 1 hour

## 5.4 Usability Requirements

* **User Experience**: Interface shall be intuitive requiring minimal learning curve
* **Error Messages**: Clear, helpful error messages for all user interactions
* **Help System**: Tooltips and help text for complex features
* **Recipe Readability**: Recipe instructions formatted for easy following while cooking
* **Search Efficiency**: Users shall find desired recipes within 3 clicks

## 5.5 Compatibility Requirements

* **Browser Support**: Chrome, Firefox, Safari, Edge (latest 2 versions)
* **Mobile Compatibility**: Responsive design for iOS and Android browsers
* **API Compatibility**: RESTful API following standard conventions
* **Future Integration**: Architecture supporting future mobile app development

# 6. Database Requirements

## 6.1 Core Tables Structure

**Users Table (ASP.NET Identity)**

* **Id** (Primary Key, string)
* **Email** (Unique, required)
* **EmailConfirmed** (boolean)
* **PasswordHash** (string)
* **UserName** (string)
* **FirstName** (string, required)
* **LastName** (string, required)
* **DisplayName** (string, optional)
* **CreatedDate** (DateTime)
* **LastLoginDate** (DateTime)

**Categories Table**

* **CategoryId** (Primary Key, int)
* **Name** (string, required) - Breakfast, Lunch, Dinner, Dessert, Snacks
* **Description** (string, optional)

**Recipes Table**

* **RecipeId** (Primary Key, int)
* **Title** (string, required, max 100 chars)
* **Description** (string, required, max 500 chars)
* **Ingredients** (text, required)
* **Instructions** (text, required)
* **PrepTimeMinutes** (int, required)
* **CookTimeMinutes** (int, required)
* **Servings** (int, required)
* **Difficulty** (enum: Easy, Medium, Hard)
* **UserId** (Foreign Key to Users, required)
* **CategoryId** (Foreign Key to Categories, required)
* **CreatedDate** (DateTime)
* **ModifiedDate** (DateTime)
* **AverageRating** (decimal, calculated)
* **RatingCount** (int, calculated)

**Ratings Table**

* **RatingId** (Primary Key, int)
* **RecipeId** (Foreign Key to Recipes, required)
* **UserId** (Foreign Key to Users, required)
* **Stars** (int, 1-5, required)
* **CreatedDate** (DateTime)
* **Feedback** (string, max 500 chars)

## 6.2 Relationships

* **Users → Recipes**: One-to-Many (One user can create many recipes)
* **Users → Ratings**: One-to-Many (One user can rate many recipes)
* **Recipes → Ratings**: One-to-Many (One recipe can have many ratings)
* **Categories → Recipes**: One-to-Many (One category can have many recipes)
* **Unique Constraint**: Users can rate each recipe only once (UserId + RecipeId)

## 6.3 Indexes

* **Primary Keys**: All tables have clustered primary key indexes
* **Foreign Keys**: Non-clustered indexes on all foreign key columns
* **Search Optimization**: Indexes on Recipe.Title, Recipe.Ingredients for search performance
* **User Recipes**: Index on Recipe.UserId for user's recipe queries
* **Category Browsing**: Index on Recipe.CategoryId for category filtering

# 7. User Stories

## 7.1 Anonymous User Stories

1. **As an anonymous user**, I want to browse recipes on the home page so that I can discover new dishes to cook
2. **As an anonymous user**, I want to search for recipes by name or ingredients so that I can find specific recipes I'm looking for
3. **As an anonymous user**, I want to view complete recipe details so that I can follow the cooking instructions
4. **As an anonymous user**, I want to see recipe ratings and reviews so that I can choose highly-rated recipes
5. **As an anonymous user**, I want to filter recipes by category so that I can find recipes for specific meal types

## 7.2 Registered User Stories

1. **As a registered user**, I want to create an account so that I can share my own recipes
2. **As a registered user**, I want to create new recipes with all necessary details so that others can cook my dishes
3. **As a registered user**, I want to edit my existing recipes so that I can improve them or fix mistakes
4. **As a registered user**, I want to delete my recipes so that I can remove ones I no longer want to share
5. **As a registered user**, I want to rate other users' recipes so that I can help the community find good recipes
6. **As a registered user**, I want to view my recipe dashboard so that I can manage all my created recipes
7. **As a registered user**, I want to see how my recipes are performing so that I know which ones are popular
8. **As a registered user**, I want to update my existing rating for a recipe so that I can change my opinion

## 7.3 Administrator Stories

1. **As an administrator**, I want to view all users in the system so that I can monitor platform activity
2. **As an administrator**, I want to deactivate user accounts so that I can remove problematic users
3. **As an administrator**, I want to delete inappropriate recipes so that I can maintain content quality
4. **As an administrator**, I want to view platform statistics so that I can track growth and usage
5. **As an administrator**, I want to access all recipes and ratings so that I can moderate content

# 8. Acceptance Criteria Summary

## 8.1 Must Have Features (MVP)

* ✅ User registration and authentication system
* ✅ Recipe CRUD operations (Create, Read, Update, Delete)
* ✅ Recipe search by title and ingredients
* ✅ Recipe categorization and filtering
* ✅ Recipe rating system (1-5 stars)
* ✅ User dashboard for recipe management
* ✅ Responsive web design
* ✅ Basic admin panel for content management
* ✅ Both MVC web interface and Web API endpoints

## 8.2 Should Have Features

* ✅ Advanced search filters (difficulty, time, rating)
* ✅ Recipe performance analytics for users
* ✅ Top-rated recipes highlighting
* ✅ User profile management
* ✅ Recipe sorting options
* ✅ Input validation and error handling
* ✅ Mobile-optimized interface

## 8.3 Could Have Features (Future Enhancements)

* 📱 Recipe image upload and display
* 📧 Email notifications for new ratings
* 💬 Recipe comments and reviews
* 🔍 Advanced search with multiple criteria
* 📊 Detailed analytics and reporting
* 🌐 Multi-language support
* 📱 Native mobile application
* 👥 Social features (following users, recipe collections)
* 📤 Recipe sharing on social media
* 🍽️ Meal planning features

## 8.4 Success Criteria

The project will be considered successful when:

* Users can register, login, and manage their accounts securely
* Users can create, edit, delete, and view recipes with all required information
* Search functionality works efficiently for recipe discovery
* Rating system allows users to rate recipes and displays accurate averages
* Both web interface and API provide full functionality
* Application is responsive and works on all modern browsers
* Admin can manage users and content effectively
* System handles expected load of 100-500 initial users
* Code follows 3-tier architecture with proper separation of concerns

# 9. Technical Implementation Notes

## 9.1 Architecture Requirements

* **N-Tier Architecture**: Models, Data Access Layer (DAL), Business Logic Layer (BLL)
* **ASP.NET MVC**: Web interface using MVC pattern
* **ASP.NET Web API**: RESTful API for future mobile app
* **Entity Framework Code First**: Database modeling and access
* **ASP.NET Identity**: User authentication and authorization
* **Repository Pattern**: Data access abstraction
* **Dependency Injection**: Loose coupling between layers

## 9.2 Development Timeline

* **Day 1**: Requirements analysis, SRS document, database design, UI sketches
* **Day 2**: Backend implementation (Models, DAL, BLL, API endpoints)
* **Day 3**: Frontend implementation (MVC views, JavaScript API client), testing

## 9.3 Quality Assurance

* **Unit Testing**: Core business logic testing
* **Integration Testing**: API endpoint testing
* **User Acceptance Testing**: Feature functionality verification
* **Performance Testing**: Load testing with expected user base
* **Security Testing**: Authentication and input validation testing