Software Requirements Specification (SRS)

TasteBud Recipes - Recipe Sharing Platform

Table of Contants

---

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document describes the functional and non-functional requirements for the TasteBud Recipes platform - a web-based recipe sharing application that allows users to create, share, discover, and rate cooking recipes.

1.2 Scope

The TasteBud Recipes platform will provide:

- A web-based interface for recipe management and browsing
- RESTful API for future mobile application integration
- User authentication and profile management
- Recipe creation, editing, and sharing capabilities
- Advanced search functionality by recipe name and ingredients
- Community-driven rating system
- Administrative tools for content moderation

1.3 Definitions and Acronyms

- **API:** Application Programming Interface
- **CRUD:** Create, Read, Update, Delete
- **ERD:** Entity Relationship Diagram
- **MVC:** Model-View-Controller
- **SRS:** Software Requirements Specification
- **UI/UX:** User Interface/User Experience
- **Recipe:** A set of cooking instructions with ingredients and metadata
- **Rating:** User feedback on recipe quality (1-5 stars)

2. Overall Description

2.1 Product Perspective

TasteBud Recipes is a standalone web application designed to compete with existing recipe sharing platforms like "CookShare." The platform serves as a digital cookbook where cooking enthusiasts can contribute and discover recipes, similar to how traditional recipe cards were shared among family and friends.

2.2 User Classes and Characteristics

2.2.1 Guest Users (Unregistered)

- **Characteristics:** Casual visitors, potential users evaluating the platform
- **Technical Expertise:** Basic computer/smartphone users
- **Permissions:** Browse and search recipes, view recipe details, access public content

2.2.2 Registered Users

- **Characteristics:** Cooking enthusiasts, home chefs, recipe collectors
- **Technical Expertise:** Basic to intermediate computer/smartphone users
- **Permissions:** All guest permissions plus create/edit/delete own recipes, rate recipes, manage profile

2.2.3 Administrators

- **Characteristics:** Platform moderators, technical staff
- **Technical Expertise:** Advanced users with platform management knowledge
- **Permissions:** All user permissions plus manage all content, moderate users, access admin dashboard

2.3 Operating Environment

- **Client-side:** Modern web browsers (Chrome, Firefox, Safari, Edge)
- **Server-side:** Windows/Linux server with .NET Framework
- **Database:** SQL Server or SQL Server Express
- **Deployment:** Standard web hosting environment
- **Devices:** Desktop computers, tablets, smartphones (responsive design)

2.4 Assumptions and Dependencies

2.4.1 Assumptions

- Users have basic internet connectivity
- Users are familiar with web browsing
- Initial user base will be 100-500 users in the first month
- Content moderation will be minimal initially (trust-based approach)
- Images for recipes will be implemented in future phases

2.4.2 Dependencies

- ASP.NET MVC Framework availability
- Entity Framework for data access
- ASP.NET Identity for authentication
- SQL Server for data storage
- Standard web hosting services

---

3. Functional Requirements

3.1 User Management (Registration, Login, Profile)

3.1.1 User Registration (F001)

Description: Allow new users to create accounts
Priority: High
Inputs: Email, password, confirm password, display name
Processing: Validate input, check email uniqueness, create user account
Outputs: User account created, confirmation message
Error Conditions: Invalid email format, weak password, email already exists

3.1.2 User Authentication (F002)

Description: Allow registered users to log in and out
Priority: High
Inputs: Email/username, password
Processing: Validate credentials, create session
Outputs: User logged in, redirected to dashboard
Error Conditions: Invalid credentials, account locked

3.1.3 User Profile Management (F003)

Description: Allow users to view and edit their profiles
Priority: Medium

Inputs: Display name, bio, preferences
Processing: Update user information in database
Outputs: Profile updated confirmation
Error Conditions: Invalid data format

3.2 Recipe Management (CRUD operations)

3.2.1 Create Recipe (F004)

Description: Allow authenticated users to create new recipes
Priority: High
Inputs: Title, description, ingredients, instructions, prep time, cook time, servings, difficulty, category
Processing: Validate input, save to database, associate with user
Outputs: Recipe created, redirected to recipe details
Error Conditions: Missing required fields, invalid data format

3.2.2 View Recipe Details (F005)

Description: Display complete recipe information to all users
Priority: High
Inputs: Recipe ID
Processing: Retrieve recipe from database, calculate average rating
Outputs: Recipe details page with all information
Error Conditions: Recipe not found, database error

3.2.3 Edit Recipe (F006)

Description: Allow recipe owners to modify their recipes
Priority: High
Inputs: Updated recipe fields
Processing: Validate ownership, update database
Outputs: Recipe updated confirmation
Error Conditions: User not owner, invalid data

3.2.4 Delete Recipe (F007)

Description: Allow recipe owners and admins to delete recipes
Priority: Medium
Inputs: Recipe ID, confirmation
Processing: Validate ownership/admin rights, remove from database
Outputs: Recipe deleted confirmation
Error Conditions: User not authorized, recipe has dependencies

3.3 Browse and Search

3.3.1 Browse All Recipes (F008)

Description: Display paginated list of all recipes
Priority: High
Inputs: Page number, sort criteria
Processing: Retrieve recipes from database, apply sorting
Outputs: List of recipes with basic information
Error Conditions: Database connection issues

3.3.2 Search Recipes (F009)

Description: Find recipes by title or ingredients
Priority: High
Inputs: Search term
Processing: Query database for matching recipes
Outputs: Filtered list of matching recipes
Error Conditions: No results found, invalid search term

3.3.3 Filter by Category (F010)

Description: Display recipes filtered by category
Priority: Medium
Inputs: Category selection
Processing: Filter recipes by selected category
Outputs: Category-specific recipe list
Error Conditions: Category not found

3.4 Rating System

3.4.1 Rate Recipe (F011)

Description: Allow authenticated users to rate recipes
Priority: High
Inputs: Recipe ID, rating (1-5 stars)
Processing: Validate user authentication, save/update rating
Outputs: Rating saved confirmation
Error Conditions: User not authenticated, invalid rating value

3.4.2 Display Average Rating (F012)

Description: Show average rating for each

recipe
Priority: High
Inputs: Recipe ID
Processing: Calculate average from all ratings

Outputs: Average rating displayed
Error Conditions: No ratings available

3.5 Administrative Features

3.5.1 Admin Dashboard (F013)

Description: Provide overview of platform statistics
Priority: Medium
Inputs: Admin authentication
Processing: Retrieve platform statistics
Outputs: Dashboard with user and recipe counts
Error Conditions: User not admin

3.5.2 Manage Users (F014)

Description: Allow admins to view and manage user accounts
Priority: Medium
Inputs: Admin authentication
Processing: Retrieve user list, allow status changes
Outputs: User management interface
Error Conditions: User not admin

3.5.3 Manage Recipes (F015)

Description: Allow admins to moderate recipe content
Priority: Medium
Inputs: Admin authentication
Processing: Retrieve all recipes, allow deletion
Outputs: Recipe management interface
Error Conditions: User not admin

---

4. Non-Functional Requirements

4.1 Performance

4.1.1 Response Time (NF001)

- Page load time should not exceed 3 seconds under normal conditions
- API responses should be delivered within 1 second
- Search results should appear within 2 seconds

4.1.2 Throughput (NF002)

- System should support 100 concurrent users without performance degradation

- Database should handle 1000 recipe views per hour • Support for 50 new recipe submissions per day

4.1.3 Capacity (NF003)

- Initial capacity for 500 users and 5000 recipes
- Database should support growth to 10,000 users
- Recipe storage should accommodate detailed instructions (up to 5000 characters)

4.2 Security

4.2.1 Authentication (NF004)

- Secure password storage using industry-standard hashing (bcrypt/PBKDF2) • Session management with secure cookies
- Password complexity requirements (minimum 8 characters, mixed case, numbers)

4.2.2 Authorization (NF005)

- Role-based access control (Guest, User, Admin) • Users can only modify their own recipes
- Administrative functions restricted to admin users

4.2.3 Data Protection (NF006)

- HTTPS encryption for all data transmission
- SQL injection prevention through parameterized queries
- Cross-site scripting (XSS) protection

4.3 Usability

4.3.1 User Interface (NF007)

- Intuitive navigation suitable for users of all technical levels
- Large, easily clickable buttons for mobile users
- Clear visual hierarchy and readable typography • Consistent design across all pages

4.3.2 Accessibility (NF008)

- Responsive design for desktop, tablet, and mobile devices
- Support for users with visual impairments (proper contrast ratios)
- Keyboard navigation support

4.3.3 Learnability (NF009)

- New users should be able to create their first recipe within 10 minutes
- Search functionality should be immediately apparent
- Error messages should be clear and actionable

4.4 Reliability

4.4.1 Availability (NF010)

- System uptime of 99.5% (approximately 4 hours downtime per month)
- Graceful handling of server errors with user-friendly messages
- Database backup and recovery procedures

4.4.2 Error Handling (NF011)

- Comprehensive input validation on both client and server sides
- Graceful degradation when services are unavailable
- Detailed error logging for debugging purposes

---

5. Use Cases (minimum required):

5.1 Use Case: Register New User

Use Case ID: UC001
Use Case Name: Register New User
Primary Actor: Guest User
Goal: Create a new user account on the platform
Preconditions: User has internet access and is on registration page
Postconditions: User account is created and user is logged in

Main Success Scenario:

1. User navigates to registration page
2. User enters email address
3. User enters desired password

4. User confirms password
5. User enters display name
6. User clicks "Register" button
7. System validates input data
8. System creates new user account
9. System logs user in automatically
10. System redirects user to dashboard

Alternative Flows:

- **A1:** Email already exists
  - System displays error message
  - User returns to step 2
- **A2:** Password too weak
  - System displays password requirements
  - User returns to step 3
- **A3:** Passwords don't match
  - System displays error message
  - User returns to step 4

5.2 Use Case: Create Recipe

Use Case ID: UC002
Use Case Name: Create Recipe
Primary Actor: Registered User
Goal: Add a new recipe to the platform
Preconditions: User is logged in
Postconditions: New recipe is published on the platform

Main Success Scenario:

1. User clicks "Create Recipe" button
2. System displays recipe creation form
3. User enters recipe title
4. User enters recipe description
5. User enters ingredients list
6. User enters cooking instructions
7. User sets preparation time

8. User sets cooking time
9. User sets number of servings
10. User selects difficulty level
11. User selects category
12. User clicks "Publish Recipe"
13. System validates all required fields
14. System saves recipe to database
15. System displays success message
16. System redirects to recipe details page

Alternative Flows:

- **A1:** Required fields missing
  - System highlights missing fields
  - User returns to step 3-11
- **A2:** Invalid time values
  - System displays error message
  - User corrects time values

5.3 Use Case: Search Recipes

Use Case ID: UC003
Use Case Name: Search Recipes
Primary Actor: Any User (Guest or Registered)
Goal: Find recipes based on search criteria
Preconditions: User is on any page with search functionality
Postconditions: User sees relevant search results

Main Success Scenario:

1. User enters search term in search box
2. User clicks search button or presses Enter
3. System processes search query
4. System searches recipe titles and ingredients
5. System displays matching recipes
6. User reviews search results
7. User clicks on desired recipe to view details

Alternative Flows:

- **A1:** No results found
  - System displays "No recipes found" message
  - System suggests alternative search terms
- **A2:** Empty search term
  - System displays all recipes (browse mode)

5.4 Use Case: Rate Recipe

Use Case ID: UC004
Use Case Name: Rate Recipe
Primary Actor: Registered User
Goal: Provide feedback on recipe quality
Preconditions: User is logged in and viewing recipe details
Postconditions: Recipe rating is updated

Main Success Scenario:

1. User views recipe details page
2. User sees rating section (1-5 stars)
3. User clicks on desired star rating
4. System validates user is logged in
5. System saves/updates user's rating
6. System recalculates average rating
7. System displays updated average rating
8. System shows confirmation message

Alternative Flows:

- **A1:** User not logged in
  - System prompts user to log in
  - User is redirected to login page
- **A2:** User already rated this recipe
  - System updates existing rating
  - System displays "Rating updated" message

5.5 Use Case: Admin Manage Content

Use Case ID: UC005
Use Case Name: Admin Manage Content
Primary Actor: Administrator
Goal: Moderate platform content and users

Preconditions: User is logged in with admin privileges
Postconditions: Content is moderated according to admin actions

Main Success Scenario:

1. Admin logs into admin dashboard
2. System displays admin interface
3. Admin selects content management option
4. System displays list of recipes/users
5. Admin reviews content for policy violations
6. Admin selects inappropriate content
7. Admin chooses action (delete/warning)
8. System executes admin action
9. System logs admin activity
10. System displays confirmation message

Alternative Flows:

- **A1:** Non-admin user tries to access
  - System denies access
  - System redirects to login page

5.6 Use Case: Login User

Use Case ID: UC006
Use Case Name: Login User
Primary Actor: Registered User
Goal: Authenticate and access user-specific features
Preconditions: User has existing account
Postconditions: User is authenticated and can access protected features

Main Success Scenario:

1. User clicks "Login" button
2. System displays login form
3. User enters email/username
4. User enters password
5. User clicks "Login" button
6. System validates credentials
7. System creates user session
8. System redirects to dashboard

or previous page

Alternative Flows:

- **A1: Invalid credentials**
    - System displays error message: *"Invalid email or password."*
    - User remains on the login form
    - User may retry login
- **A2: Account locked (too many failed attempts)**
    - System displays error message: *"Account locked due to multiple failed attempts. Please try again later or reset your password."*
    - User is prompted to use password reset
- **A3: Missing fields**
    - System highlights missing email/password fields
    - User fills in the missing fields and retries
- **A4: Session hijack or reuse attempt**
    - System invalidates session
    - User is logged out forcibly
    - User is redirected to login screen with security notice