**Front Office (ASP.Net Razor Pages):**

1. **Unregistered Users Functionality:**
   * Registration Page: Create a form for users to register with.
   * Search Functionality: Implement a search feature using multiple fields.
   * Recipe Details Page: Display detailed information about each recipe.
2. **Registered Users Functionality:**
   * All functionalities for unregistered users, excluding registration.
   * Insert New Recipes: Create a form for users to submit new recipes (pending approval).
   * Comments: Implement a comment section for registered users to leave comments on recipes.
   * Rating: Allow registered users to rate recipes.
   * Favorites: Allow users to mark recipes as favorites and manage their favorites list.
   * Login/Logout: Implement user authentication and authorization.
   * Personal Area: Create a page where users can manage their personal information.

**Back Office (ASP.Net Razor Pages):**

1. **Administrator Functionality:**
   * User Management: Implement functionality to lock/unlock user accounts.
   * Recipe Approval: Allow administrators to approve pending recipes.
   * Ingredient Management: Add, edit, and delete ingredients.
   * Category Management: Add, edit, and delete recipe categories.
   * Difficulty Level Management: Add, edit, and delete difficulty levels.

**DAL (Data Access Layer):**

1. **Database Connectivity:**
   * Implement data access methods to connect to the SQL database.
   * CRUD Operations: Implement methods to perform CRUD (Create, Read, Update, Delete) operations on database entities.

**Database (SQL):**

1. **Database Schema:**
   * Create tables for users, recipes, comments, ingredients, categories, etc., as per the project requirements.
   * Define relationships between tables (e.g., one-to-many relationship between recipes and comments).
   * Set up appropriate constraints and indexes for data integrity and performance.
2. **Stored Procedures or ORM:**
   * Optionally, you can use stored procedures or an Object-Relational Mapping (ORM) framework like Entity Framework to interact with the database.

**General Recommendations:**

* **Separation of Concerns:** Ensure proper separation of concerns by dividing your application into layers (presentation, business logic, data access).
* **Validation:** Implement validation at both client-side and server-side to ensure data integrity and security.
* **Security:** Implement authentication and authorization mechanisms to protect sensitive functionalities.
* **Testing:** Test your application thoroughly to ensure it meets all requirements and works correctly in different scenarios.
* **Documentation:** Document your code, especially complex functionalities and database schema.
* **Error Handling:** Implement proper error handling to provide meaningful error messages to users and log errors for debugging purposes.

Classes:

using System;

using System.Collections.Generic;

public class Recipe

{

public int Id { get; set; }

public string Title { get; set; }

public string Instructions { get; set; }

public string ImageSource { get; set; }

public int MinutesToCook { get; set; }

public bool IsApproved { get; set; }

public int UserId { get; set; }

public int DifficultyId { get; set; }

public DateTime CreatedAt { get; set; }

// Navigation properties

public User User { get; set; }

public Difficulty Difficulty { get; set; }

public ICollection<RecipeCategory> RecipeCategories { get; set; }

public ICollection<Rating> Ratings { get; set; }

public ICollection<Comment> Comments { get; set; }

public ICollection<RecipeIngredient> RecipeIngredients { get; set; }

}

public class RecipeCategory

{

public int CategoryId { get; set; }

public int RecipeId { get; set; }

// Navigation properties

public Category Category { get; set; }

public Recipe Recipe { get; set; }

}

public class Category

{

public int Id { get; set; }

public string Name { get; set; }

// Navigation properties

public ICollection<RecipeCategory> RecipeCategories { get; set; }

}

public class Comment

{

public int Id { get; set; }

public string Body { get; set; }

public int RecipeId { get; set; }

public int UserId { get; set; }

public DateTime CreatedAt { get; set; }

// Navigation properties

public Recipe Recipe { get; set; }

public User User { get; set; }

}

public class Difficulty

{

public int Id { get; set; }

public string Name { get; set; }

// Navigation properties

public ICollection<Recipe> Recipes { get; set; }

}

public class Rating

{

public int Id { get; set; }

public int Value { get; set; }

public int RecipeId { get; set; }

// Navigation properties

public Recipe Recipe { get; set; }

}

public class RecipeIngredient

{

public int RecipeId { get; set; }

public int IngredientId { get; set; }

// Navigation properties

public Recipe Recipe { get; set; }

public Ingredient Ingredient { get; set; }

}

public class Ingredient

{

public int Id { get; set; }

public string Name { get; set; }

public int Amount { get; set; }

public int UnitId { get; set; }

// Navigation properties

public Unit Unit { get; set; }

public ICollection<RecipeIngredient> RecipeIngredients { get; set; }

}

public class Unit

{

public int Id { get; set; }

public string Name { get; set; }

// Navigation properties

public ICollection<Ingredient> Ingredients { get; set; }

}

public class User

{

public int Id { get; set; }

public string Username { get; set; }

public string Password { get; set; }

public string Email { get; set; }

public string FirstName { get; set; }

public string LastName { get; set; }

public string ContentBio { get; set; }

public string ImageSource { get; set; }

public bool IsAdmin { get; set; }

public bool IsBlocked { get; set; }

public DateTime CreatedAt { get; set; }

// Navigation properties

public ICollection<Comment> Comments { get; set; }

public ICollection<Recipe> Recipes { get; set; }

public ICollection<UserFavoriteRecipe> UserFavoriteRecipes { get; set; }

}

public class UserFavoriteRecipe

{

public int RecipeId { get; set; }

public int UserId { get; set; }

// Navigation properties

public Recipe Recipe { get; set; }

public User User { get; set; }

}