**Recipe Organizer**

**Software Design Specification**

– Ho Chi MinhMinh, May 2023 –

Record of changeS

| Date | A\*  M, D | In charge | Change Description |
| --- | --- | --- | --- |
| June 24 | A | TramLU | Add detail Code Designs 3, 4, 5 |
| June 24 | A | CongCV | Code package added |
| June 26 | M | CongCV | Code package modified and add description |
| June 25 | A | TramLU | Add details to section II Code Designs item number 8, 9, 10 |
| June 25 | M | TramLU | Modified details to section II Code Designs - Sequence diagram of item 6, 7 |
| June 28 | A | PhiDN | Add Code Designs (11, 12, 13, 14) |
| June 30 | M | PhiDN | Modified table description (13, 18, 19, 20) |
| June 30 | A | KhoaND | Add Search, Category. HomePage function |
| July 01 | M | TramLU | Double check & edit errors |
| July 03 | A | CongCV | Add login, user profile function |
| July 03 | M | CongCV | Add description of table in database design |
| July 03 | A | KhoaND | Check All |

\*A - Added M - Modified D - Deleted

# **Table of Contents**

[I. Overview 4](https://docs.google.com/document/d/1WBB_4j834Uwdk1o0y1uniPAjU7eefxSB1dedfeFSzJI/edit#heading=h.30j0zll)

[1.](https://docs.google.com/document/d/1WBB_4j834Uwdk1o0y1uniPAjU7eefxSB1dedfeFSzJI/edit#heading=h.1fob9te) Code Packages [4](https://docs.google.com/document/d/1WBB_4j834Uwdk1o0y1uniPAjU7eefxSB1dedfeFSzJI/edit#heading=h.1fob9te)

[2.](https://docs.google.com/document/d/1WBB_4j834Uwdk1o0y1uniPAjU7eefxSB1dedfeFSzJI/edit#heading=h.3znysh7)Database Design4

a. Database Schema5

b. Table Description 6

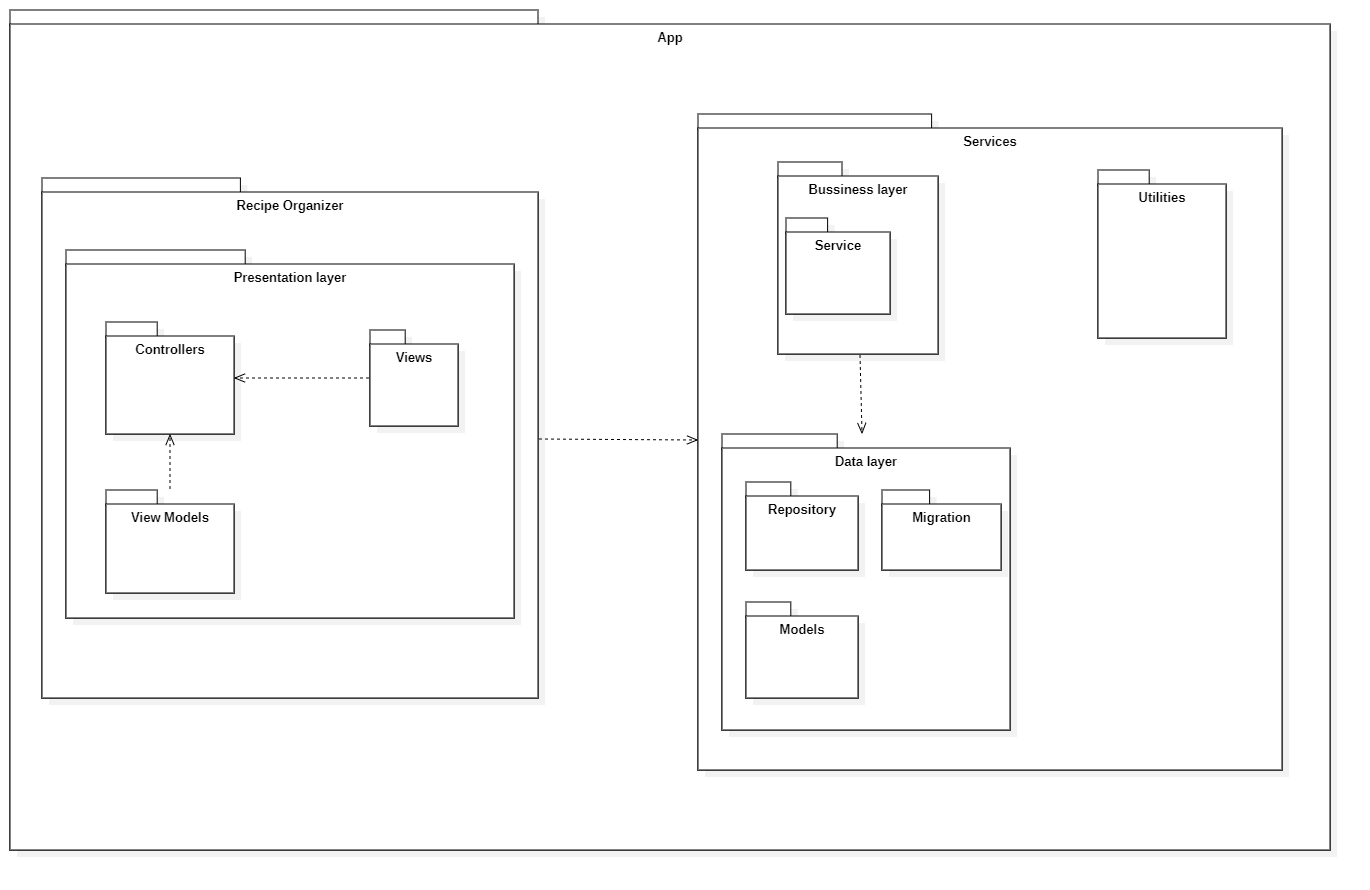
3. Use Case Diagram 8

[II.](https://docs.google.com/document/d/1WBB_4j834Uwdk1o0y1uniPAjU7eefxSB1dedfeFSzJI/edit#heading=h.tyjcwt) Code Designs9

1. Home Page 9
2. Search Page 11
3. Recipe Details 14
4. Add Recipe Page 19
5. Edit Recipe Page 24
6. User Profile Page 29
7. Login Page 32
8. User Collection List Page 36
9. User Recipe Attribute List Page 39
10. User Feedback List Page 43
11. Show Meal Planning 45
12. Add Recipe To Plan 52
13. Remove Recipe From Plan 56
14. Save Plan 58

# **I. Overview**

## **1. Code Packages**

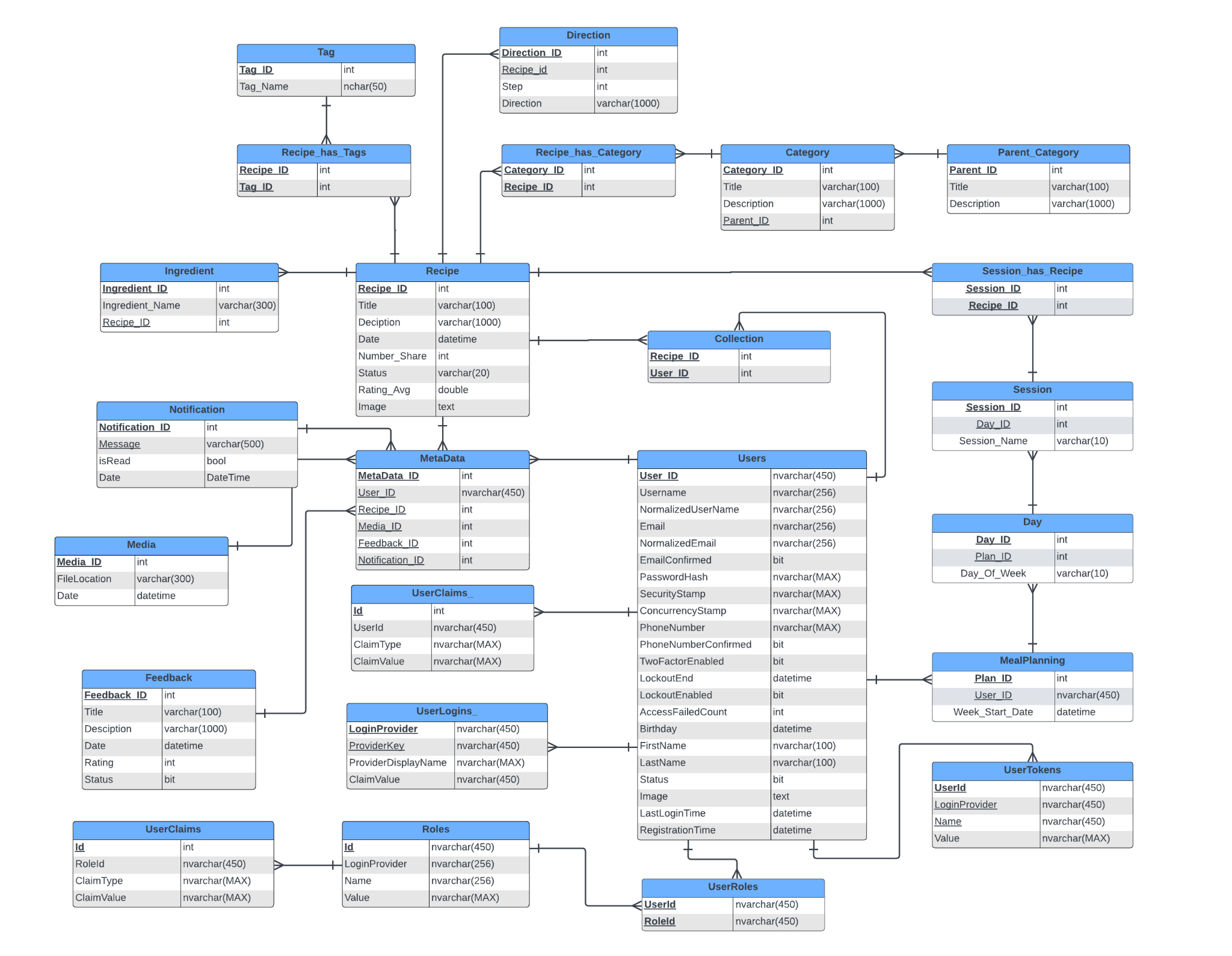
**

***Package descriptions***

| **No** | **Package** | **Description** |
| --- | --- | --- |
| 01 | Controllers | Handles user interactions and invokes appropriate actions. |
| 02 | View Models | Provides data and behavior for the views |
| 03 | Views | Contains the views responsible for rendering the user interface. |
| 04 | Service | Implements the business logic of the application, such as recipe management, user authentication, etc. |
| 05 | Repositories | Provides an interface for accessing and manipulating data from the underlying data source. |
| 06 | Model | Defines the domain models that represent the entities in the application. |
| 07 | Migration | Manages database schema changes using code-first migrations. |
| 08 | Utilities | Contains various utility classes and components that provide generic functionalities and services that can be used throughout the application. |

## **2. Database Design**

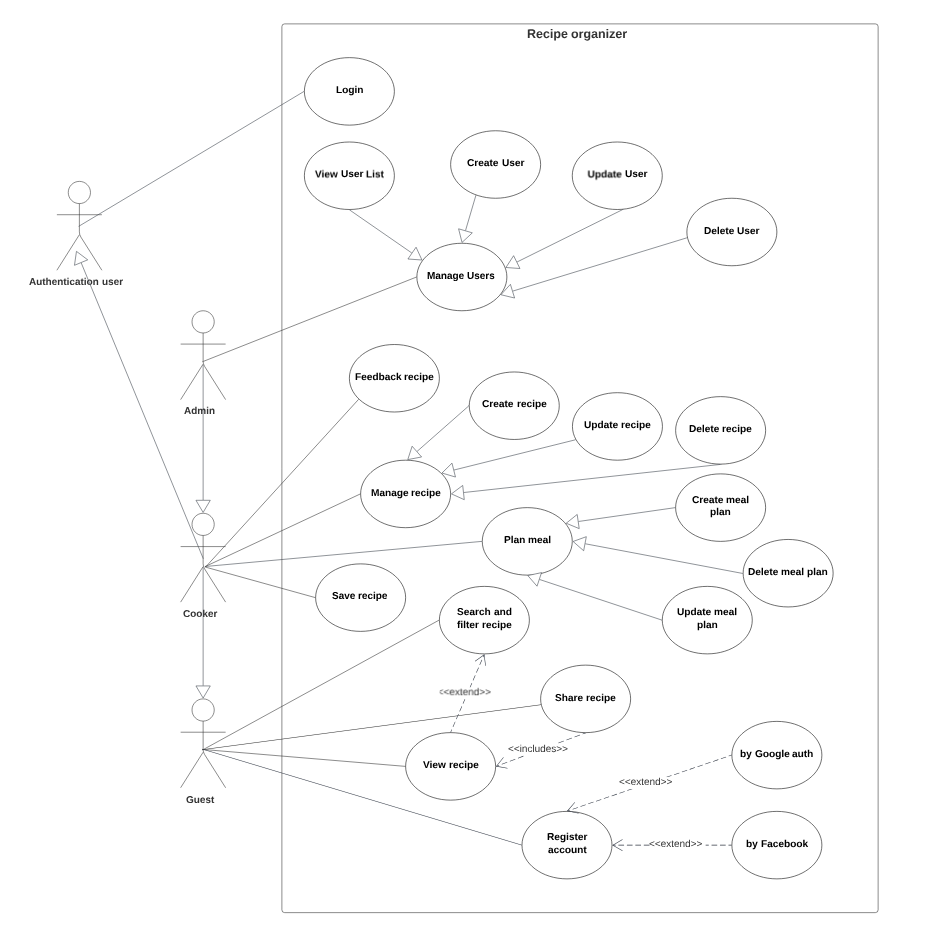
### **a. Database Schema**



### **b. Table Description**

| **No** | **Table** | **Description** |
| --- | --- | --- |
| 01 | AspNetUsers | User-specific information such as user ID, username, email, password hash, security stamp, and other user-related properties. |
| 02 | Role | Role-related information such as role ID and role name. Roles are used to group users with similar permissions or access levels. |
| 03 | UserRole | Establishes a relationship between users and roles. It stores the mapping of user IDs and role IDs, indicating which users belong to which roles. |
| 04 | UserClaims | User claims or assertions about the user's identity. Claims represent specific user attributes, such as name, email, or custom properties. |
| 05 | UserLogins | External logins associated with a user account. It maintains the information about the external provider, such as Google or Facebook, and the user's unique identifier provided by that provider. |
| 06 | UserTokens | Security tokens for password reset and email confirmation processes. It includes fields such as user ID, token type, and token value. |
| 07 | UserRoleClaims | The claims associated with each role. Claims represent additional information or attributes about a role, such as its permissions or any custom claims defined in the application. |
| 08 | Notification | Store notification when user create new recipe and when admin approve or reject recipe |
| 09 | Recipe | Store recipe detail includes Title, description, date, status, number share, avg rating |
| 10 | Category | category include recipe one a type recipe, make user see easy by type |
| 11 | Recipe\_has\_Category | Map 1-n;n-1 from recipe,category, its net n-n |
| 12 | Collection | Store recipes that the user has saved |
| 13 | Feedback | Its is QA, comment form user it is possible vision all viewer |
| 14 | MealPlanning | The specific information related to a meal plan, such as the selected recipes, their quantities, cooking instructions, and any additional notes or preferences provided by the user |
| 15 | Media | Store link (location) of images |
| 16 | Parent\_Category | The parent item of the category of recipe |
| 17 | Day | Store day of plan |
| 18 | Session | Store session of day |
| 19 | SessionHasRecipe | Store recipes of session |
| 20 | MetaData | Map 1-n from recipe-metadata, user-metadata, feedback-metadata, media-metadata, notification-metadata |
| 21 | Tag | It’s store hashtag for recipe |
| 22 | Ingredient | Store ingredients or each recipe |
| 23 | Direction | Store step-by-step direction of recipe |
| 24 | Recipe\_has\_Tags | Map m-m from recipe-tag |

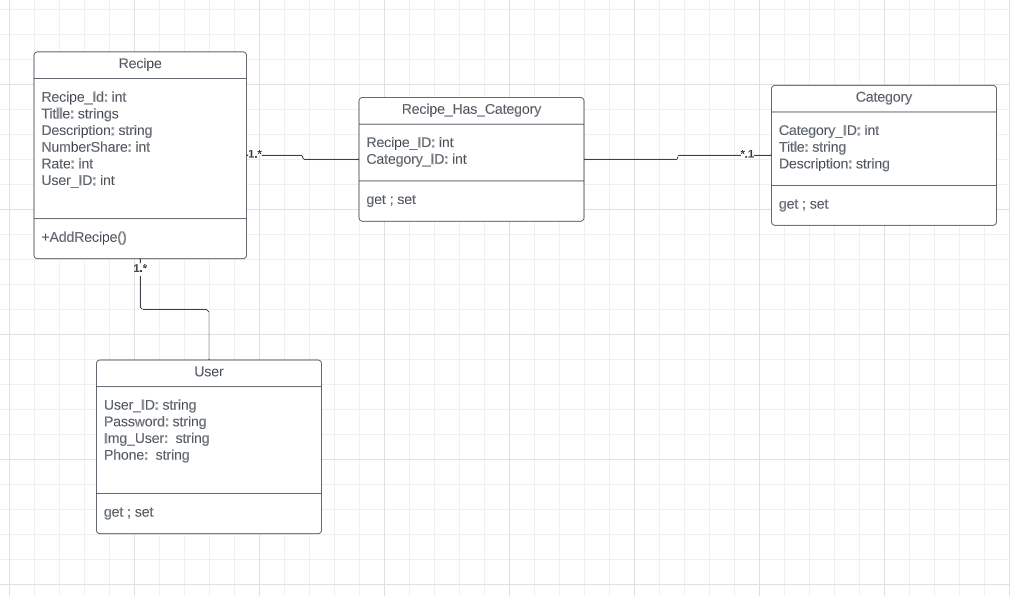
## **3. Use Case Diagram**



# **II. Code Designs**

## **1. Home Page**

### **a. Class Diagram**



### **b. Class Specifications**

Description: Represents the home page of the Recipe Organization application.

#### **Component Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getCategory | this is method get component recipe list in category match, the input is categoryParentID, its call method getListbyCategory in categoryReposive |
| 02 | getNewRecipe | this is method get recipe list new from user by date recent, the void method, its call method getNewlistRecipe in recipeReposive |

#### **RecipeRepositive Class**

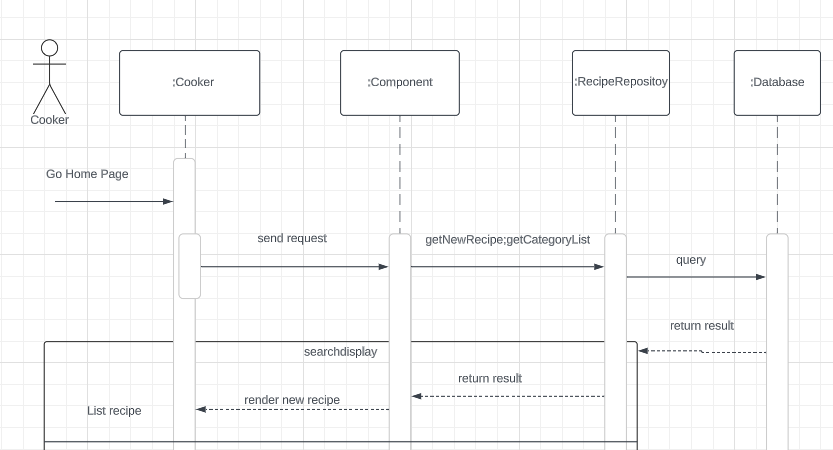
**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRecipebyCategory | this is method select recipe in db, return list recipe by input categoryid |
| 02 | getNewListRecipe | this is method select recipe in db where time recent and status public, return list recipe by method void |

**RepositoryBase Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SearchByProperty(Expression<Func<T, bool>> predicate) | Function used to query data, get model list from database |
| 02 | GetAll() | Function used to query data, get all in model from database |
| 03 | Add(T entity) | Function used to add data to the database |
| 04 | Update(T entity) | Function used to edit data into the database |
| 05 | Delete(T entity) | Function used to delete data into the database |

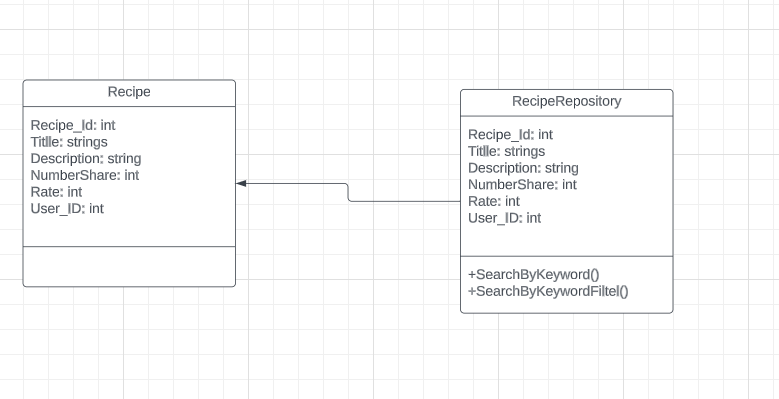
### **c. Sequence Diagram(s)**

**

### 

## **2. Search Page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **SearchController Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | searchKeyWord | method input paging number and keyword, its call getRecipeAllKeyWord in recipeRepository to return list recipe have keyword, the output is a list recipe |
| 02 | searchKeyWordFilter | method same search keyword, diff by have Filter function by A-Z, date, Numbershare, its call getRecipeAllKeyWordFillter in recipeRepository to return list recipe have filter, the output is a list recipe filter |
| 03 | getRecipebyCategory | method input the idcategory return listcategory have categoryId want, its call getCategorybyId in recipeRepository to return list recipe |

#### **RecipeRepository Class**

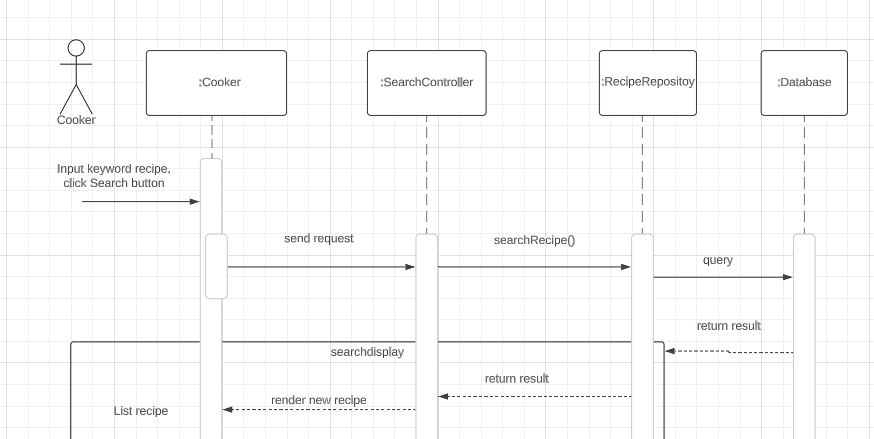
**Class Methods**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getRecipeAllKeyWord(string keyword) | this method will find contain the word same in database recipe table, it is same title recipe in table, the list recipe will return, the out is list recipe have contain keyword |
| 02 | getRecipeAllKeyWordFillter (string keyword, int type) | this method will find contain the word, and verify the type filter want, to save list, it is same title recipe in table database, the list recipe will return, the out are list recipe have contain keyword was filter. |

**RepositoryBase Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SearchByProperty(Expression<Func<T, bool>> predicate) | Function used to query data, get model list from database |
| 02 | GetAll() | Function used to query data, get all in model from database |
| 03 | Add(T entity) | Function used to add data to the database |
| 04 | Update(T entity) | Function used to edit data into the database |
| 05 | Delete(T entity) | Function used to delete data into the database |

### **c. Sequence Diagram(s)**

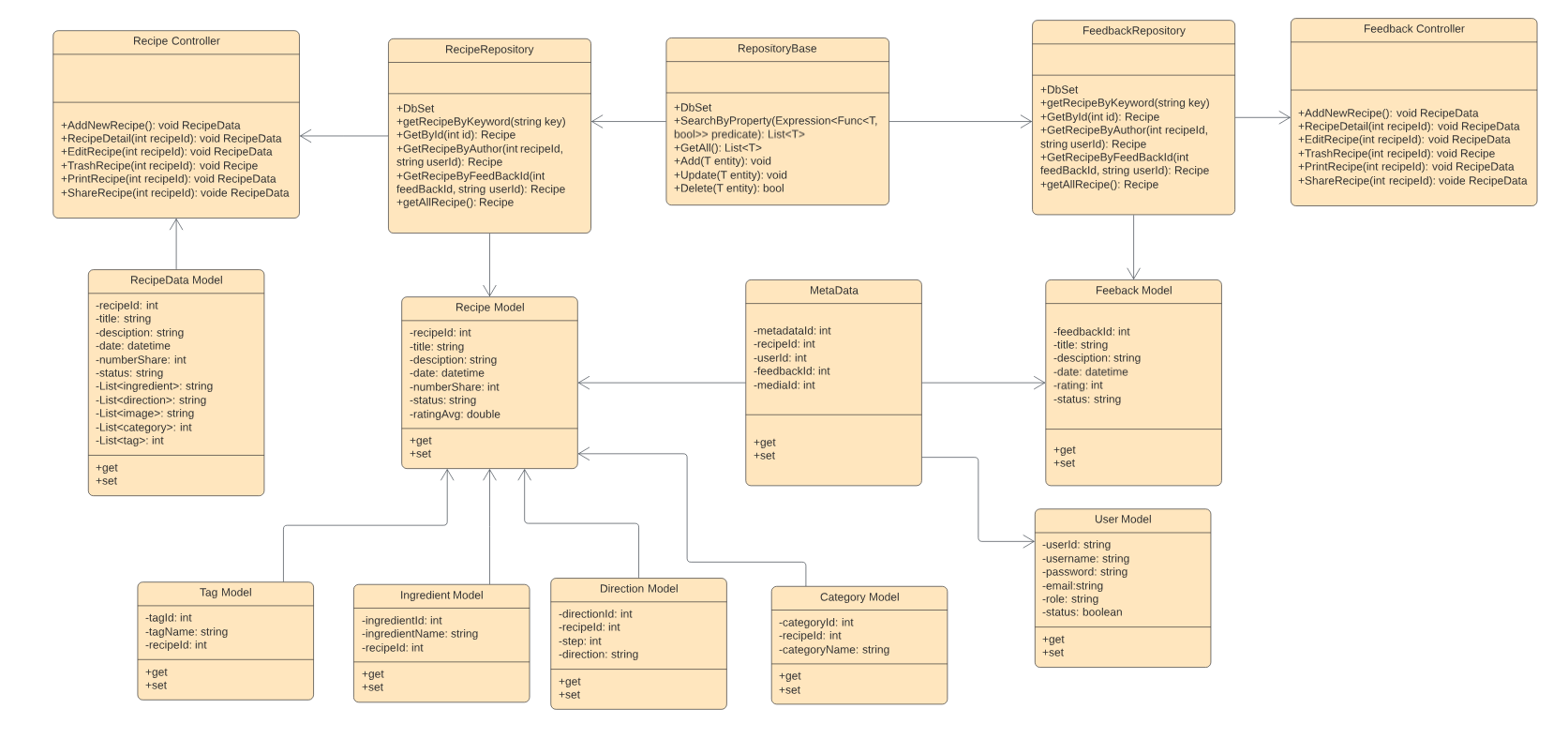
**

### **d. Database Queries**

Select \* From recipe Where title Like ‘%keyword %’

## **3. Recipe Detail Page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **RecipeController Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet RecipeDetail(int id) | The function displays the recipe details of each recipe |
| 02 | doGet ToggleCollection(int recipeId) | Function to save a recipe to the user's favorites |
| 03 | doGet PrintRecipe(int recipeId) | Function to print recipe when user wants to store for themself |
| 04 | doGet ShareRecipe(int recipeId) | Function to share a recipe via email to someone the user wants to share |
| 05 | doGet IncreaseNumberShare(int recipeId) | Function used to receive user input when user shares recipe |
| 06 | doGet AddNewRecipe() | The function is used to display a list that includes a category for the user to choose when adding a recipe |
| 07 | doPost AddNewRecipe(RecipeData recipe, List<IFormFile> files) | Function used to get user input when adding recipe |
| 08 | doGet EditRecipe(int id) | Function used to display user input so that user can be edited |
| 09 | doPost EditRecipe(RecipeData recipe, string Action, List<IFormFile> files) | Function used to get user input when user edits their own recipe data |
| 10 | doGet UsrPendingRecipeNoti(int id, int noti) | The function is used to display the message and the status of the recipe after the user has contributed |

#### **RecipeRepository Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRecipe() | Function used to query data, get recipe list from database |
| 02 | GetById(int id) | The function is used to query data, get a recipe through recipeId from the database |
| 03 | getRecipeByKeyword(string keyword) | The function is used to query data, retrieve recipes through keywords from the database |
| 04 | IncreaseNumberShare(int recipeId) | The function used to change the data to increase the number of shares |
| 05 | GetUserByAuthor(string userId) | Function used to query data, get recipe list with condition userId from database |

#### **FeedbackController Class**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet UserFeedbackList() | Function to display all feedbacks of a user |
| 02 | doGet RecipeFeedbackList() | Function to display all feedbacks of a recipe |
| 03 | AddFeedback(FeedbackViewModel feedbackViewModel) | Function used to add a new feedback of a recipe from a user |

#### **FeedbackRepository Class**

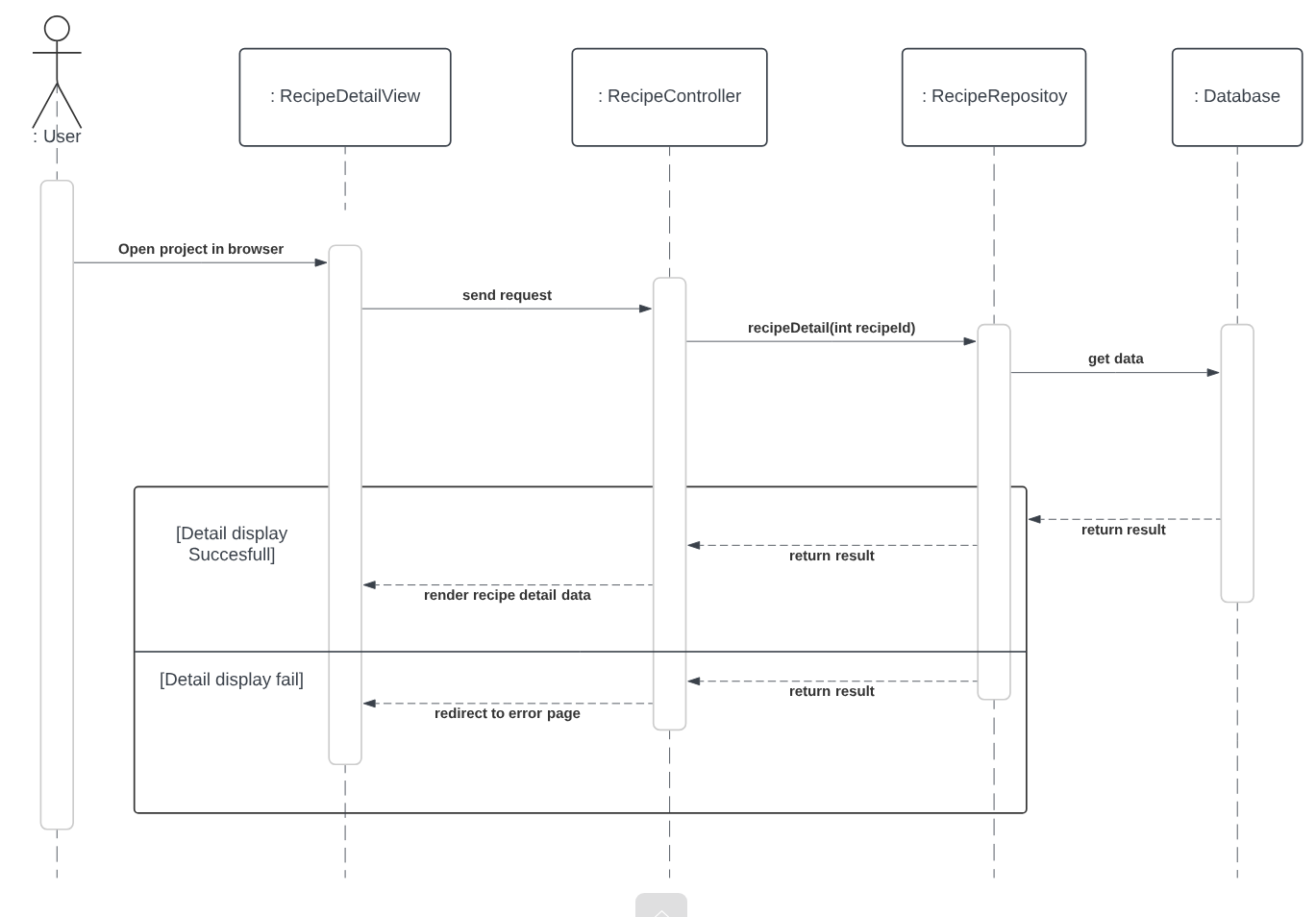
***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | valueAvgRateRecipe(int recipeId) | The function is used to calculate the rating formula of the average value of a recipe and store it in the database |
| 02 | GetByFeedbackByUser(string userId) | The function is used to query the data, get feedbacks through the user id from the database |
| 03 | GetByFeedbackByRecipeId(int recipeId) | Function used to query data, get all feedback through recipe id from the database |

#### **RepositoryBase Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SearchByProperty(Expression<Func<T, bool>> predicate) | Function used to query data, get model list from database |
| 02 | GetAll() | Function used to query data, get all in model from database |
| 03 | Add(T entity) | Function used to add data to the database |
| 04 | Update(T entity) | Function used to edit data into the database |
| 05 | Delete(T entity) | Function used to delete data into the database |

### **c. Sequence Diagram**

**

### **d. Database Queries**

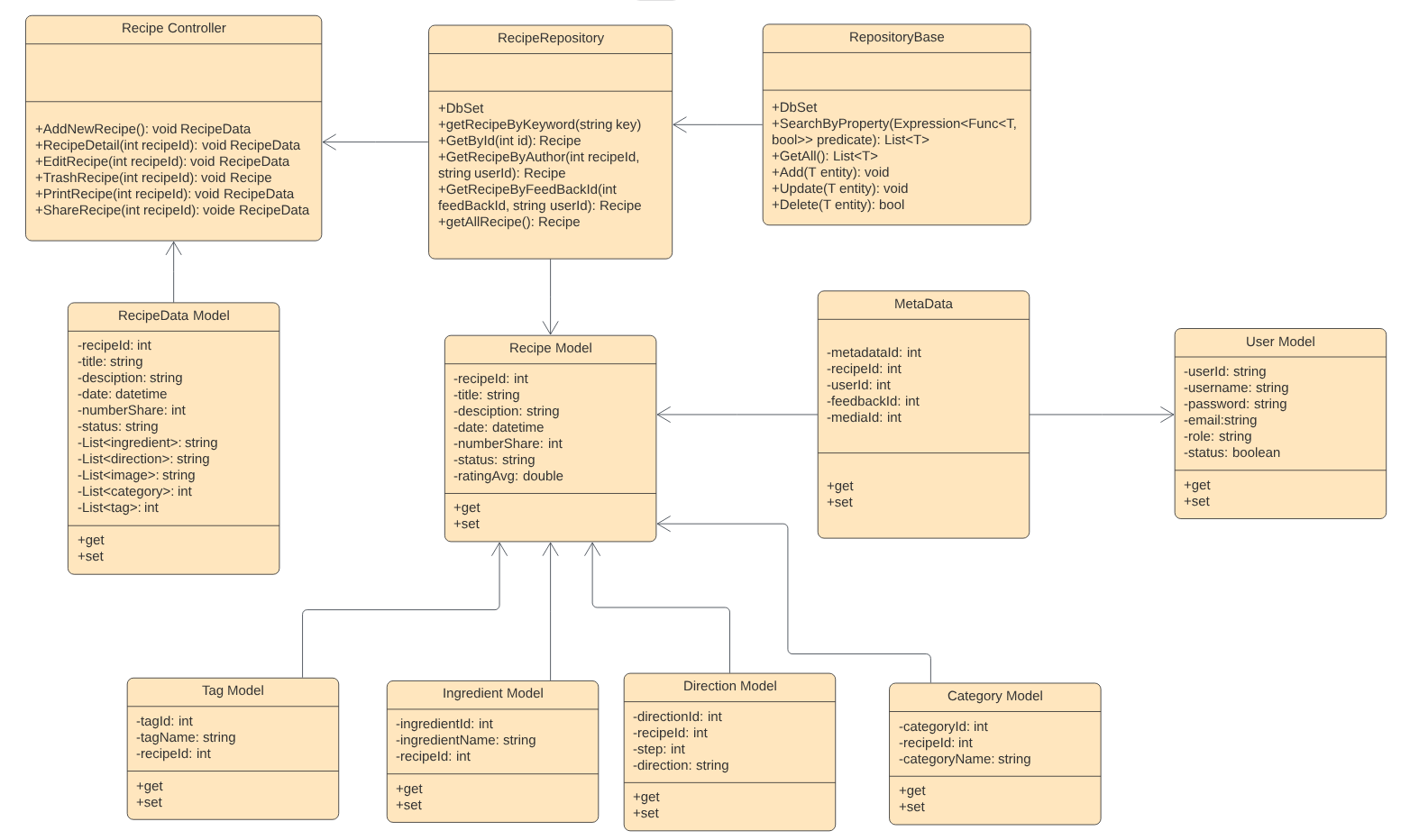
SELECT \*

FROM [dbo].[Recipe]

WHERE [recipe\_id] = <recipeId>;

## **4. Add Recipe Page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **RecipeController Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet RecipeDetail(int id) | The function displays the recipe details of each recipe |
| 02 | doGet ToggleCollection(int recipeId) | Function to save a recipe to the user's favorites |
| 03 | doGet PrintRecipe(int recipeId) | Function to print recipe when user wants to store for themself |
| 04 | doGet ShareRecipe(int recipeId) | Function to share a recipe via email to someone the user wants to share |
| 05 | doGet IncreaseNumberShare(int recipeId) | Function used to receive user input when user shares recipe |
| 06 | doGet AddNewRecipe() | The function is used to display a list that includes a category for the user to choose when adding a recipe |
| 07 | doPost AddNewRecipe(RecipeData recipe, List<IFormFile> files) | Function used to get user input when adding recipe |
| 08 | doGet EditRecipe(int id) | Function used to display user input so that user can be edited |
| 09 | doPost EditRecipe(RecipeData recipe, string Action, List<IFormFile> files) | Function used to get user input when user edits their own recipe data |
| 10 | doGet UsrPendingRecipeNoti(int id, int noti) | The function is used to display the message and the status of the recipe after the user has contributed |

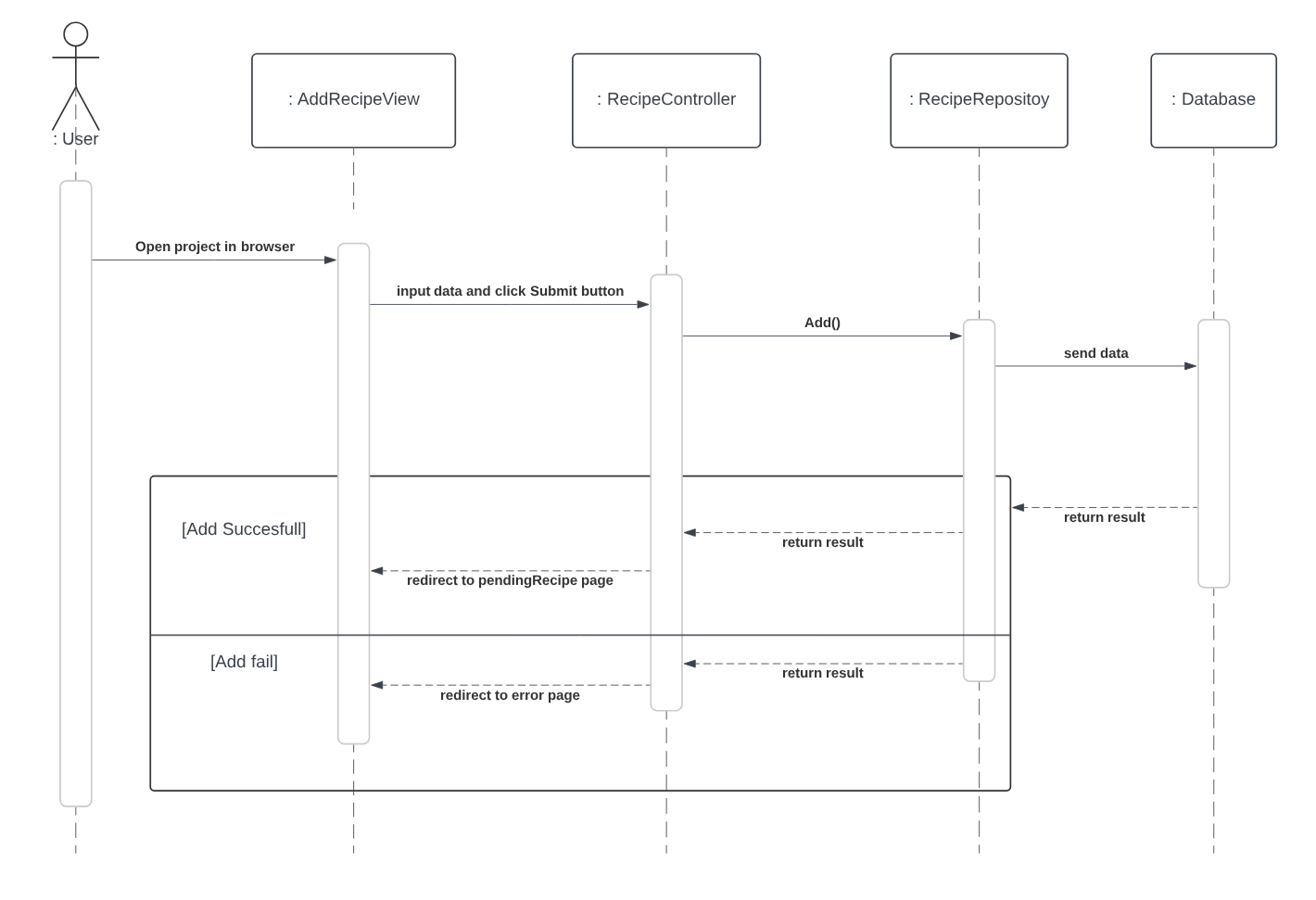
#### **RecipeRepository Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRecipe() | Function used to query data, get recipe list from database |
| 02 | GetById(int id) | The function is used to query data, get a recipe through recipeId from the database |
| 03 | getRecipeByKeyword(string keyword) | The function is used to query data, retrieve recipes through keywords from the database |
| 04 | IncreaseNumberShare(int recipeId) | The function used to change the data to increase the number of shares |
| 05 | GetUserByAuthor(string userId) | Function used to query data, get recipe list with condition userId from database |

#### **RepositoryBase Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SearchByProperty(Expression<Func<T, bool>> predicate) | Function used to query data, get model list from database |
| 02 | GetAll() | Function used to query data, get all in model from database |
| 03 | Add(T entity) | Function used to add data to the database |
| 04 | Update(T entity) | Function used to edit data into the database |
| 05 | Delete(T entity) | Function used to delete data into the database |

### **c. Sequence Diagram(s)**

**

### **d. Database Queries**

*INSERT INTO [dbo].[Recipe]*

*([title],[description],[date],[number\_share],[status],[avg\_rate],[image])*

*VALUES*

*(<title, varchar(100),>,<description, varchar(1000),>,<date, datetime,>,<number\_share, int,>,<status, varchar(20),>,<avg\_rate, float,>,<image, text,>)*

*INSERT INTO [dbo].[Direction]*

*([recipe\_id],[step],[direction])*

*VALUES*

*(<recipe\_id, int,>,<step, int,>,<direction, varchar(1000),>)*

*INSERT INTO [dbo].[Ingredient]*

*([ingredient\_name],[recipe\_id])*

*VALUES*

*(<ingredient\_name, varchar(300),>,<recipe\_id, int,>)*

*INSERT INTO [dbo].[Tag]*

*([tag\_name])*

*VALUES*

*(<tag\_name, nchar(50),>)*

*INSERT INTO [dbo].[Recipe\_has\_Tags]*

*([recipe\_id],[tag\_id])*

*VALUES*

*(<recipe\_id, int,>,<tag\_id, int,>)*

*INSERT INTO [dbo].[Media]*

*([fileLocation],[Date])*

*VALUES*

*(<fileLocation, nvarchar(450),>,<date, datetime,>)*

*INSERT INTO [dbo].[MetaData]*

*([user\_id],[recipe\_id],[media\_id])*

*VALUES*

*(<user\_id, nvarchar(450),>,<recipe\_id, int,>,<media\_id, int,>)*

*INSERT INTO [dbo].[Recipe\_has\_Category]*

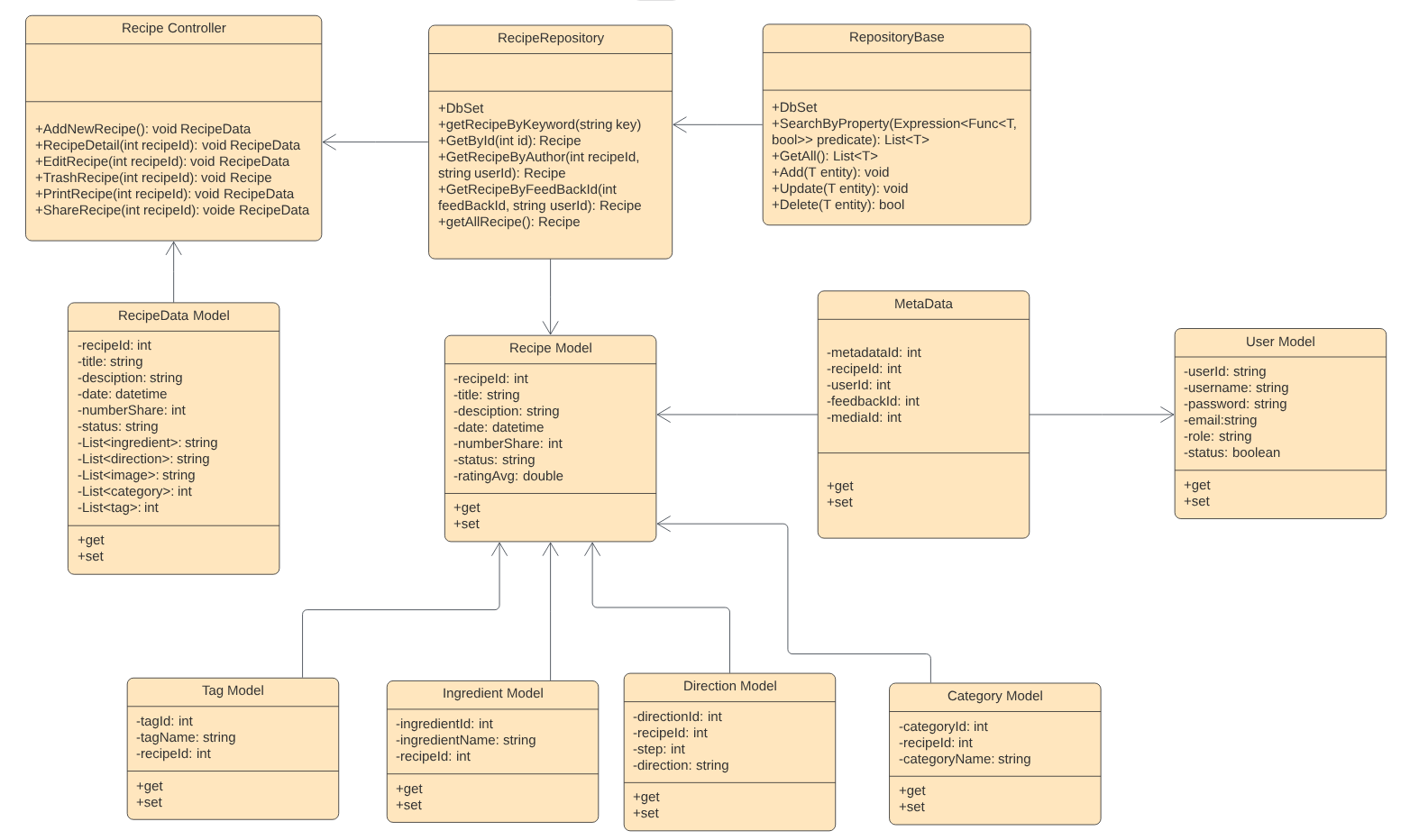
*([category\_id],[recipe\_id])*

*VALUES*

*(<category\_id, int,>,<recipe\_id, int,>)*

## **245. Edit Recipe Page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **RecipeController Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet RecipeDetail(int id) | The function displays the recipe details of each recipe |
| 02 | doGet ToggleCollection(int recipeId) | Function to save a recipe to the user's favorites |
| 03 | doGet PrintRecipe(int recipeId) | Function to print recipe when user wants to store for themself |
| 04 | doGet ShareRecipe(int recipeId) | Function to share a recipe via email to someone the user wants to share |
| 05 | doGet IncreaseNumberShare(int recipeId) | Function used to receive user input when user shares recipe |
| 06 | doGet AddNewRecipe() | The function is used to display a list that includes a category for the user to choose when adding a recipe |
| 07 | doPost AddNewRecipe(RecipeData recipe, List<IFormFile> files) | Function used to get user input when adding recipe |
| 08 | doGet EditRecipe(int id) | Function used to display user input so that user can be edited |
| 09 | doPost EditRecipe(RecipeData recipe, string Action, List<IFormFile> files) | Function used to get user input when user edits their own recipe data |
| 10 | doGet UsrPendingRecipeNoti(int id, int noti) | The function is used to display the message and the status of the recipe after the user has contributed |

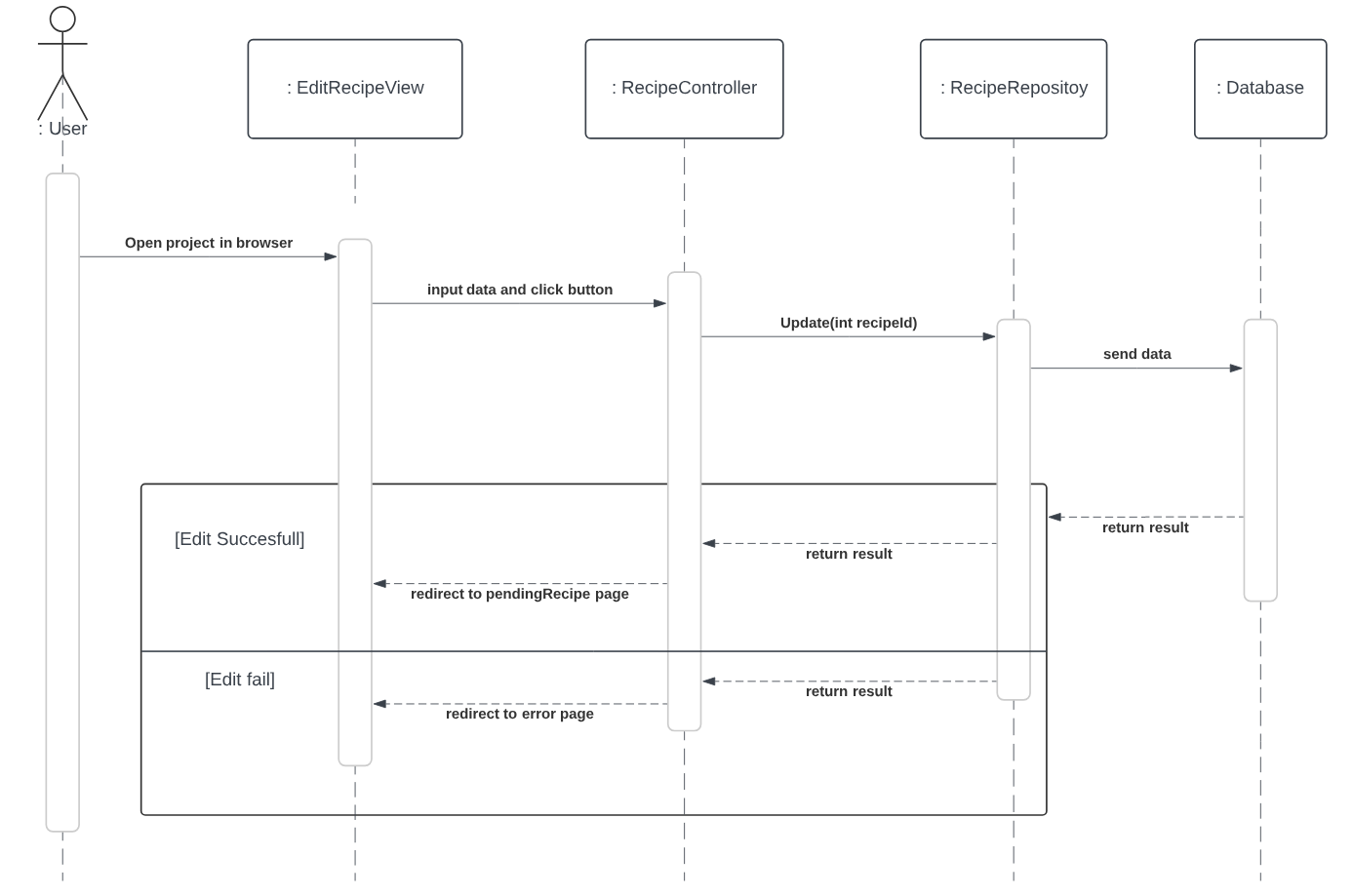
#### **RecipeRepository Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRecipe() | Function used to query data, get recipe list from database |
| 02 | GetById(int id) | The function is used to query data, get a recipe through recipeId from the database |
| 03 | getRecipeByKeyword(string keyword) | The function is used to query data, retrieve recipes through keywords from the database |
| 04 | IncreaseNumberShare(int recipeId) | The function used to change the data to increase the number of shares |
| 05 | GetUserByAuthor(string userId) | Function used to query data, get recipe list with condition userId from database |

#### **RepositoryBase Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SearchByProperty(Expression<Func<T, bool>> predicate) | Function used to query data, get model list from database |
| 02 | GetAll() | Function used to query data, get all in model from database |
| 03 | Add(T entity) | Function used to add data to the database |
| 04 | Update(T entity) | Function used to edit data into the database |
| 05 | Delete(T entity) | Function used to delete data into the database |

### **c. Sequence Diagram(s)**

**

### **d. Database Queries**

-- UPDATE for Recipe table

UPDATE [dbo].[Recipe]

SET [title] = <new\_title>,

[description] = <new\_description>,

[date] = <new\_date>,

[number\_share] = <new\_number\_share>,

[status] = <new\_status>,

[avg\_rate] = <new\_avg\_rate>,

[image] = <new\_image>

WHERE [recipe\_id] = <recipe\_id>;

-- UPDATE for Direction table

UPDATE [dbo].[Direction]

SET [step] = <new\_step>,

[direction] = <new\_direction>

WHERE [recipe\_id] = <recipe\_id> AND [step] = <step>;

-- UPDATE for Ingredient table

UPDATE [dbo].[Ingredient]

SET [ingredient\_name] = <new\_ingredient\_name>

WHERE [recipe\_id] = <recipe\_id> AND [ingredient\_name] = <ingredient\_name>;

-- UPDATE for Tag table

UPDATE [dbo].[Tag]

SET [tag\_name] = <new\_tag\_name>

WHERE [tag\_id] = <tag\_id>;

-- UPDATE for Recipe\_has\_Tags table

UPDATE [dbo].[Recipe\_has\_Tags]

SET [recipe\_id] = <new\_recipe\_id>,

[tag\_id] = <new\_tag\_id>

WHERE [recipe\_id] = <recipe\_id> AND [tag\_id] = <tag\_id>;

-- UPDATE for Media table

UPDATE [dbo].[Media]

SET [fileLocation] = <new\_fileLocation>,

[Date] = <new\_date>

WHERE [media\_id] = <media\_id>;

-- UPDATE for MetaData table

UPDATE [dbo].[MetaData]

SET [user\_id] = <new\_user\_id>,

[recipe\_id] = <new\_recipe\_id>,

[media\_id] = <new\_media\_id>

WHERE [metadata\_id] = <metadata\_id>;

-- UPDATE for Recipe\_has\_Category table

UPDATE [dbo].[Recipe\_has\_Category]

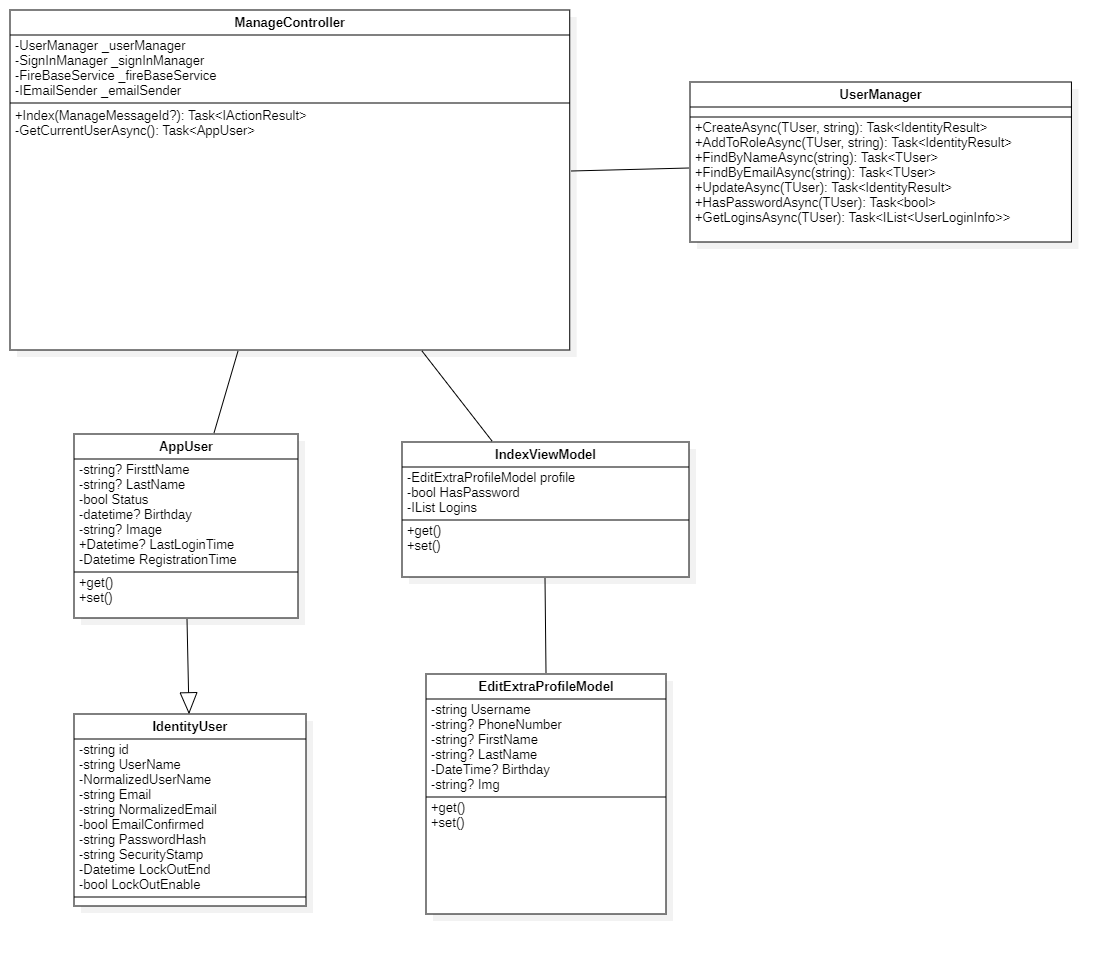
SET [category\_id] = <new\_category\_id>,

[recipe\_id] = <new\_recipe\_id>

WHERE [category\_id] = <category\_id> AND [recipe\_id] = <recipe\_id>;

## **6. User Profile Page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **ManageController**

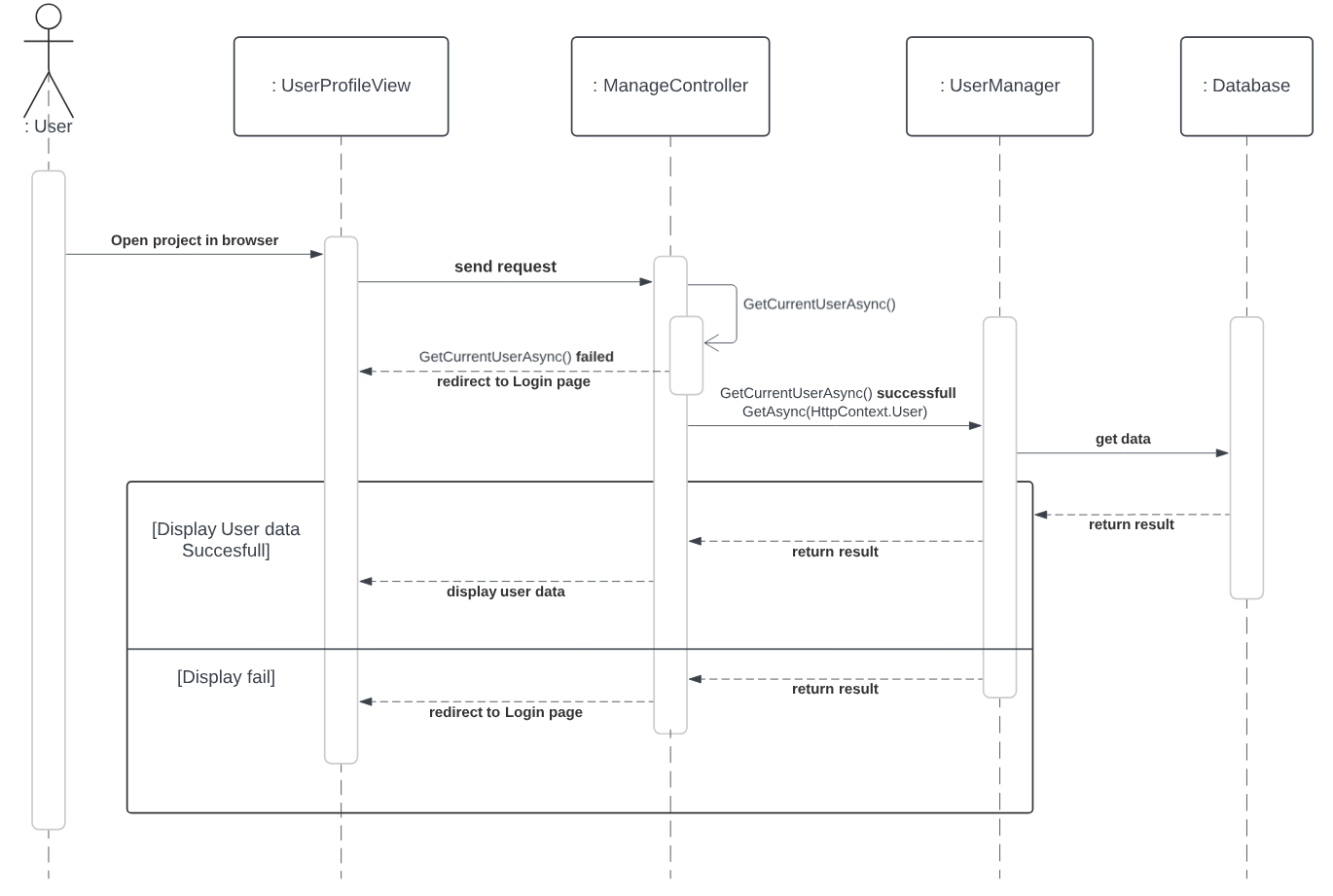
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | Index(in ManageMessageId?): Task<IActionResult> | * HttpGet method * Responsible for displaying the main user profile page. It accepts an optional parameter ManageMessageId of type ManageMessageId enumeration, which is used to display specific success or error messages to the user. |
| 02 | GetCurrentUserAsync(): Task<AppUser> | Retrieves the currently logged-in user from the UserManager in ASP.NET Identity. It returns a Task that represents the asynchronous operation of getting the user. |

#### **UserManager**

***Class method***

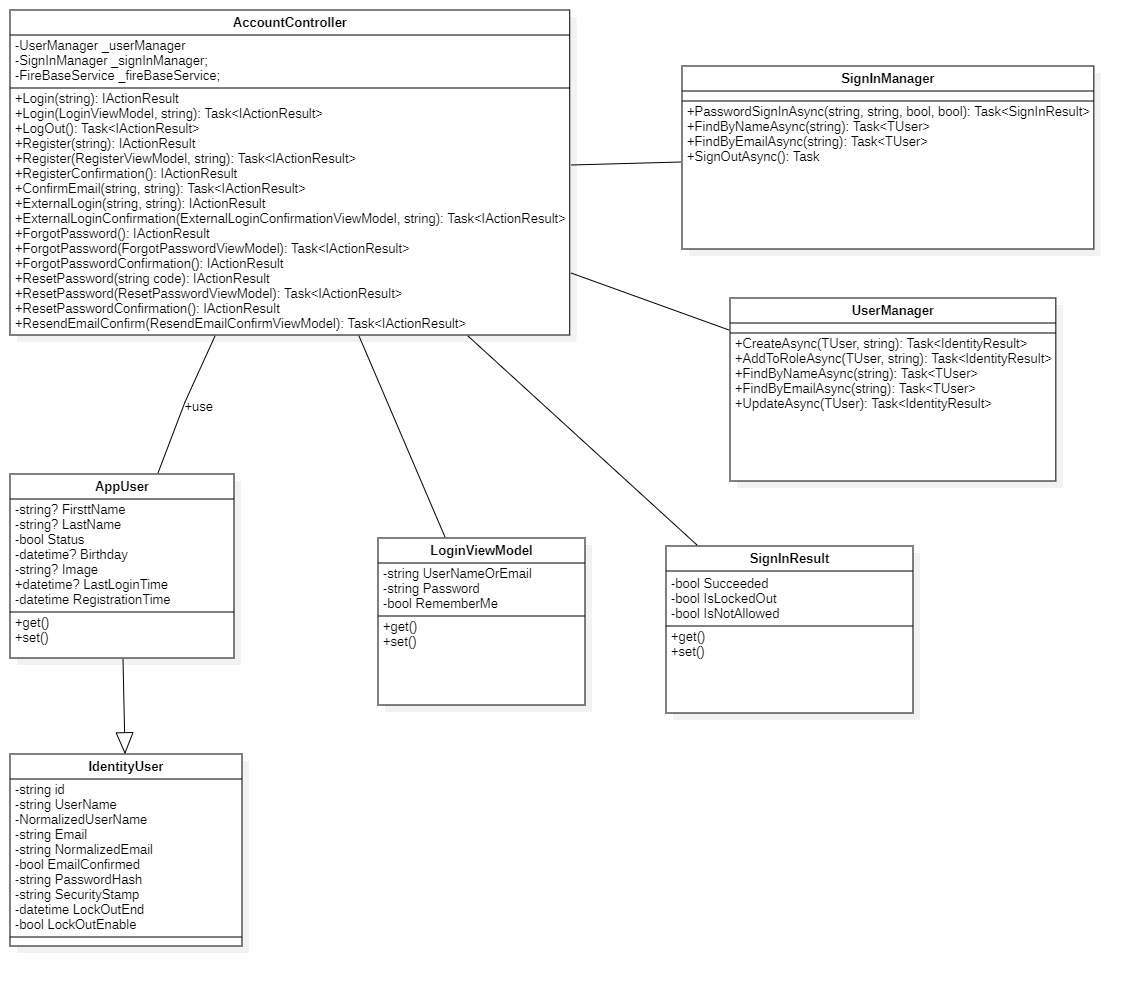
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | FindByNameAsync(string): Task<TUser> | It is used to find a user by their username. |
| 02 | FindByEmailAsync(string): Task<TUser> | It retrieves the user object associated with the provided email address. |
| 03 | UpdateAsync(TUser): Task<IdentityResult> | It is used to update a user's login time in the user store. |
| 04 | CreateAsync(TUser, string): Task<IdentityResult> | Creates a new user with the AppUser object and password. It is used to register a new user in the application. |
| 05 | AddToRoleAsync(TUser, string): Task<IdentityResult> | Adds a user to a specified role(Admin, Cook). It is used to assign a role to a user, granting them access to specific functionalities or resources associated with that role. |
| 06 | HasPasswordAsync(in TUser): Task<bool> | Checks if a user has a password associated with their account in the ASP.NET Identity system. |
| 07 | GetLoginsAsync(in TUser): Task<IList<UserLoginInfo>> | Retrieves the external login(Google, Facebook) information associated with a user in the ASP.NET Identity system. |

### **c. Sequence Diagram(s)**

**

## **7. Login page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **AccountController**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | Login(string): IActionResult | * HttpGet method * This method handles the login functionality for the Recipe Organizer application. It receives a string parameter representing the returnUrl, which is the URL the user should be redirected to after a successful login. The method returns an IActionResult, which represents the result of the login operation. * This will redirect Login page first |
| 02 | Login( LoginViewModel, string): Task<IActionResult> | * HttpPost method * This method handles the login functionality for the Recipe Organizer application. It receives a LoginViewModel object containing the user's login credentials and a string parameter representing the returnUrl, which is the Homepage URL the user should be redirected to after a successful login. |

#### **SignInManager**

***Class method***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | PasswordSignInAsync(string, string, bool, bool): Task<SignInResult> | Used to authenticate a user's credentials (username and password) and sign them into the application. |
| 02 | SignOutAsync(): Task | an asynchronous method that signs the current user out of the application. It clears the user's authentication cookies and any associated authentication information. |

#### **UserManager**

***Class method***

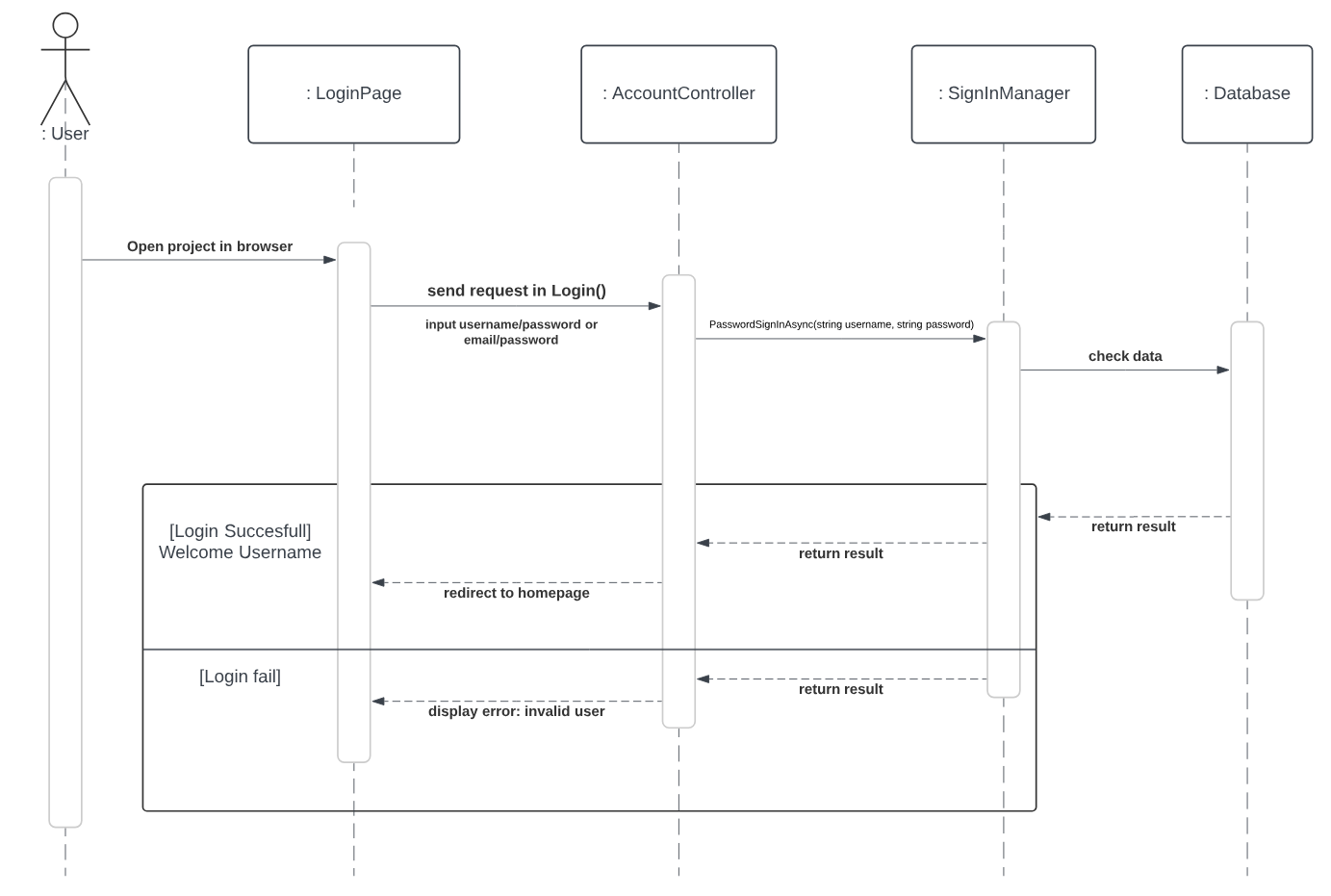
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | FindByNameAsync(string): Task<TUser> | It is used to find a user by their username. |
| 02 | FindByEmailAsync(string): Task<TUser> | It retrieves the user object associated with the provided email address. |
| 03 | UpdateAsync(TUser): Task<IdentityResult> | It is used to update a user's login time in the user store. |
| 04 | CreateAsync(TUser, string): Task<IdentityResult> | Creates a new user with the AppUser object and password. It is used to register a new user in the application. |
| 05 | AddToRoleAsync(TUser, string): Task<IdentityResult> | Adds a user to a specified role(Admin, Cook). It is used to assign a role to a user, granting them access to specific functionalities or resources associated with that role. |

**AppUtilities**

***Class method***

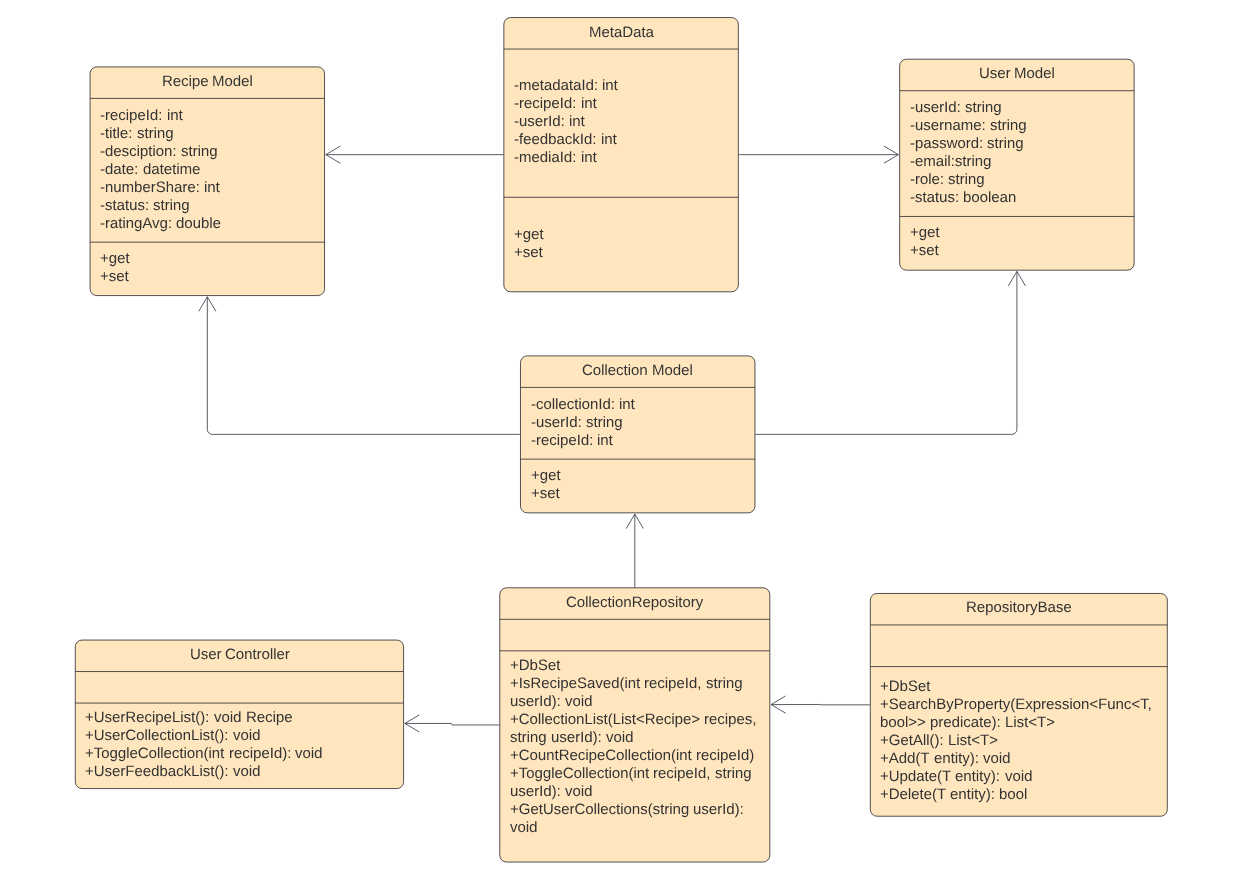
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | IsValidEmail(string): bool | Checks whether a given email address is valid or not. It performs several validations on the email address to ensure it meets the required format and structure. |

### **c. Sequence Diagram(s)**

**

## **8. User Collection List Page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **UserController Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet UserRecipeList() | Function to display all recipes contributed by the user |
| 02 | doGet UserCollectionList() | Function to display all recipes that the user has saved |
| 03 | doGet ToggleCollection(int recipeId) | Function to unsave recipes that the user has saved |
| 04 | doGet UserFeedbackList() | Function to display all the feedbacks that the user has sent |

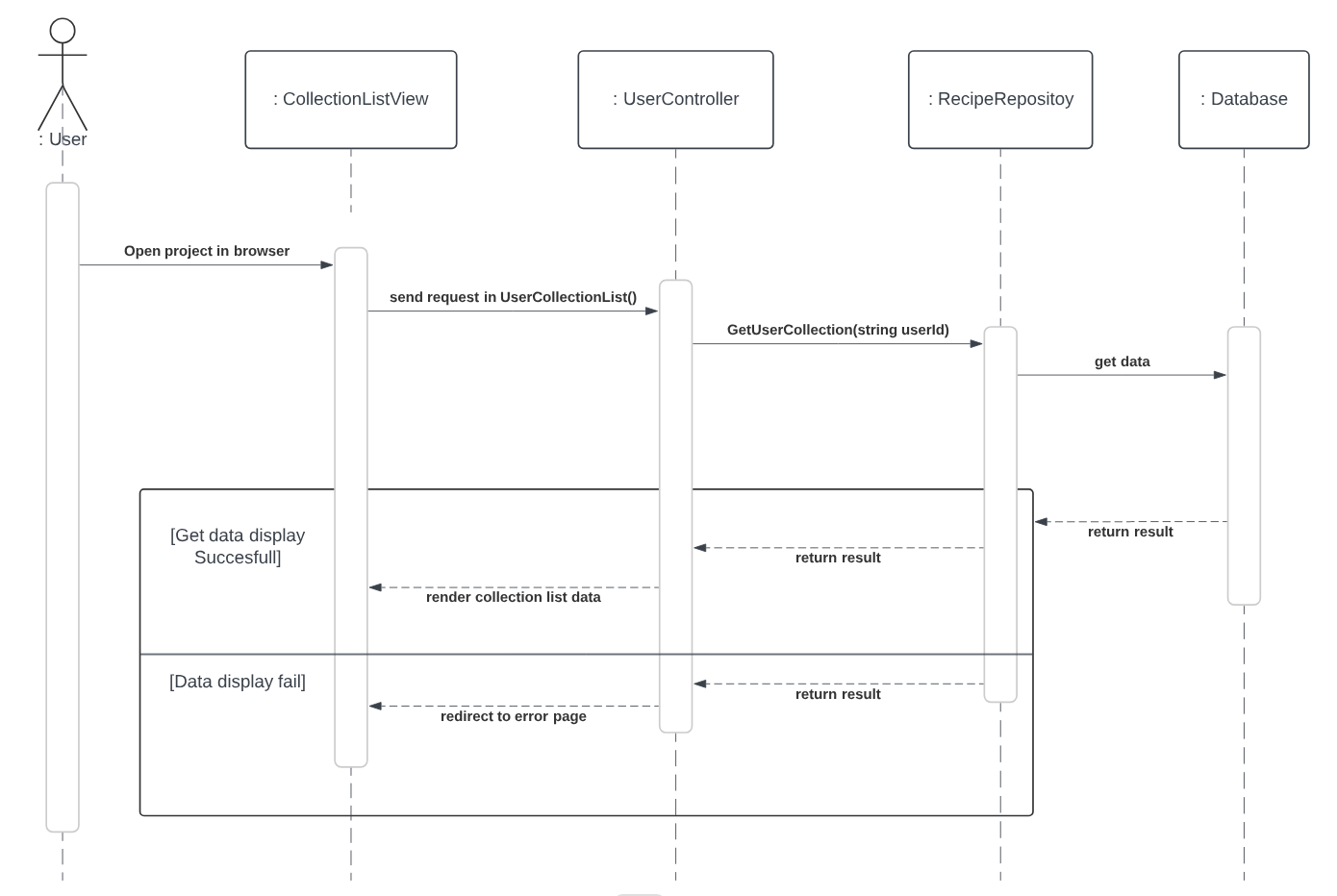
#### **CollectionRepository Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | IsRecipeSaved(int recipeId, string userId) | Function used to query data, check if recipes have been saved by the user |
| 02 | CollectionList(List<Recipe> recipes, string userId) | Function is used to query data, get all recipes that the user has saved |
| 03 | CountRecipeCollection(int recipeId) | Function is used to query data, to count how many users have saved a recipe for a recipe |
| 04 | ToggleCollection(int recipeId, string userId) | Function to save or unsave a recipe to the user's favorites |
| 05 | GetUserCollections(string userId) | Function is used to query the data displays all the recipes that the user has saved |

#### **RepositoryBase Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SearchByProperty(Expression<Func<T, bool>> predicate) | Function used to query data, get model list from database |
| 02 | GetAll() | Function used to query data, get all in model from database |
| 03 | Add(T entity) | Function used to add data to the database |
| 04 | Update(T entity) | Function used to edit data into the database |
| 05 | Delete(T entity) | Function used to delete data into the database |

### **c. Sequence Diagram**

**

### **d. Database Query**

SELECT r.\*

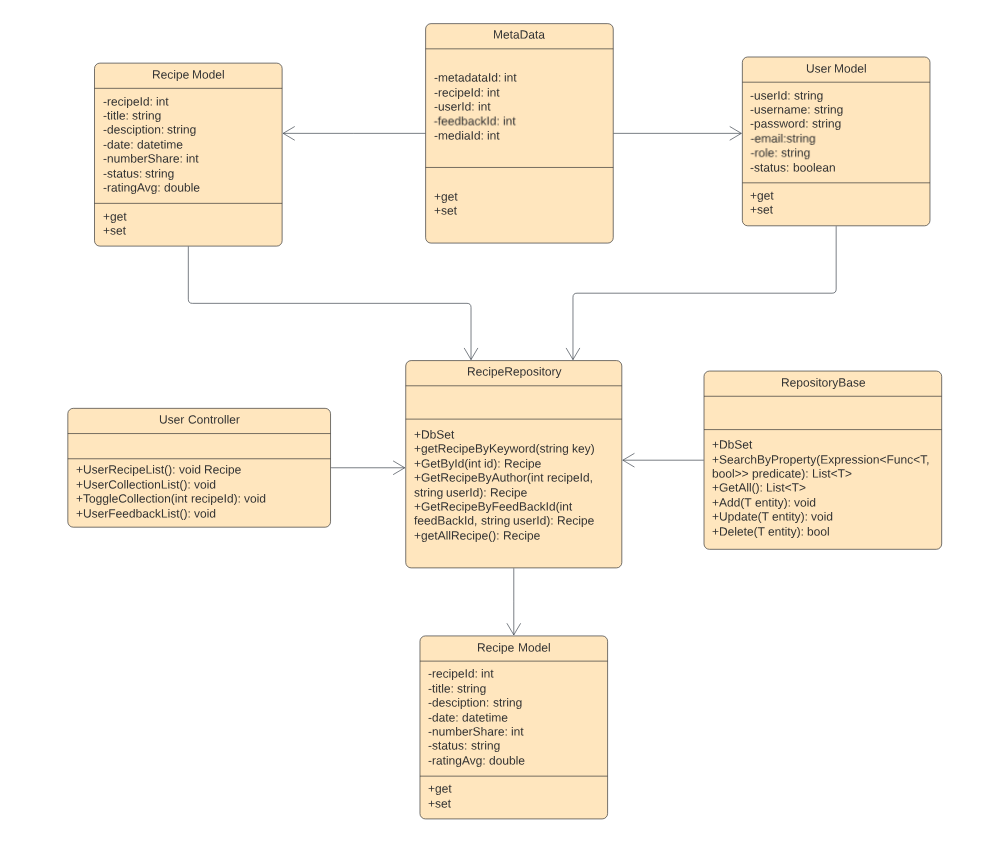
FROM Recipe r

INNER JOIN Collection c ON r.RecipeId = c.RecipeId

WHERE c.UserId = @userId

## **9. User Recipe Attribute List Page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **UserController Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet UserRecipeList() | Function to display all recipes contributed by the user |
| 02 | doGet UserCollectionList() | Function to display all recipes that the user has saved |
| 03 | doGet ToggleCollection(int recipeId) | Function to unsave recipes that the user has saved |
| 04 | doGet UserFeedbackList() | Function to display all the feedbacks that the user has sent |

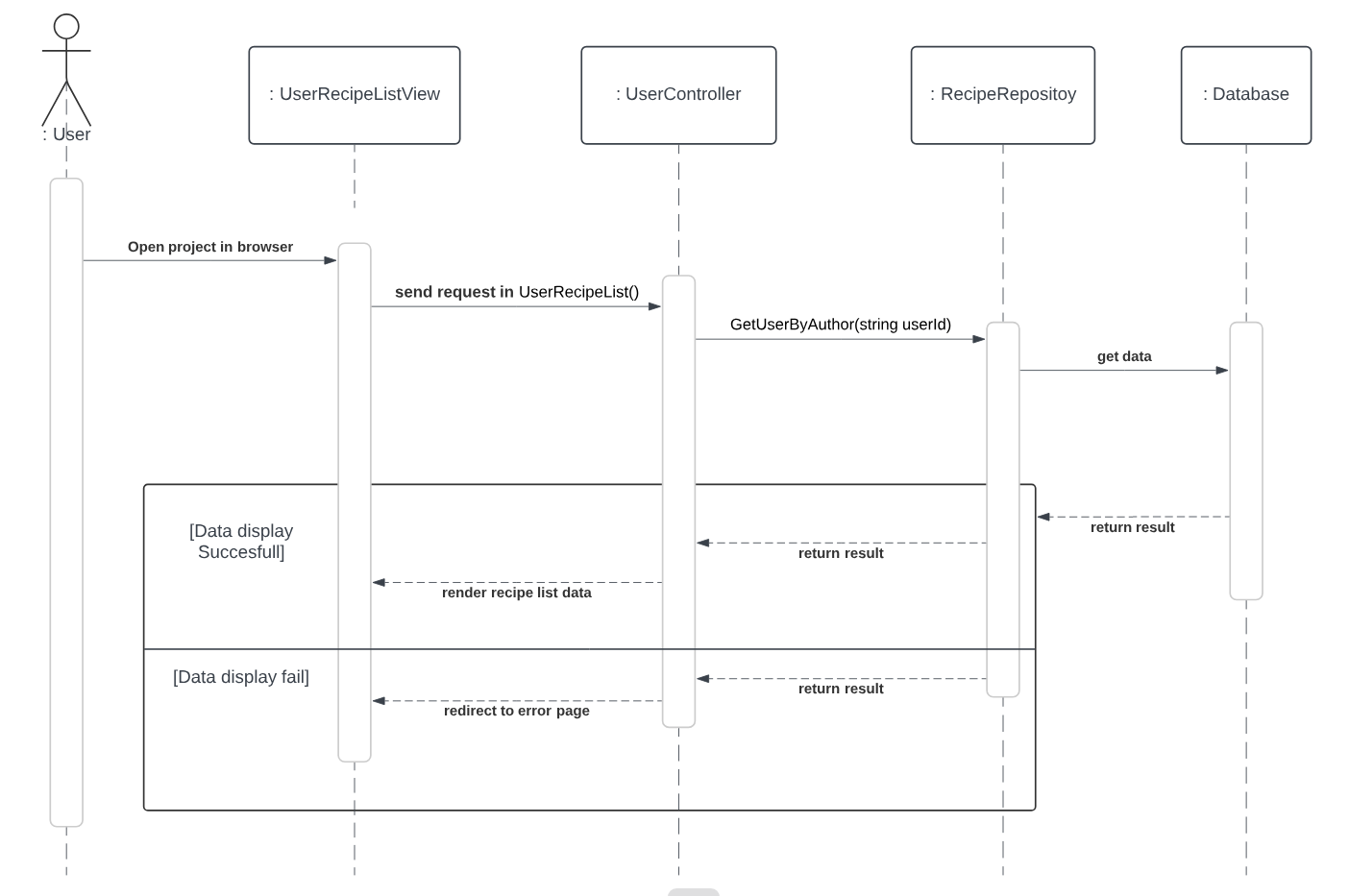
#### **RecipeRepository Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | getAllRecipe() | Function used to query data, get recipe list from database |
| 02 | GetById(int id) | The function is used to query data, get a recipe through recipeId from the database |
| 03 | getRecipeByKeyword(string keyword) | The function is used to query data, retrieve recipes through keywords from the database |
| 04 | IncreaseNumberShare(int recipeId) | The function used to change the data to increase the number of shares |
| 05 | GetUserByAuthor(string userId) | Function used to query data, get recipe list with condition userId from database |

#### **RepositoryBase Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SearchByProperty(Expression<Func<T, bool>> predicate) | Function used to query data, get model list from database |
| 02 | GetAll() | Function used to query data, get all in model from database |
| 03 | Add(T entity) | Function used to add data to the database |
| 04 | Update(T entity) | Function used to edit data into the database |
| 05 | Delete(T entity) | Function used to delete data into the database |

### **c. Sequence Diagram(s)**

**

### **d. Database Queries**

SELECT \*

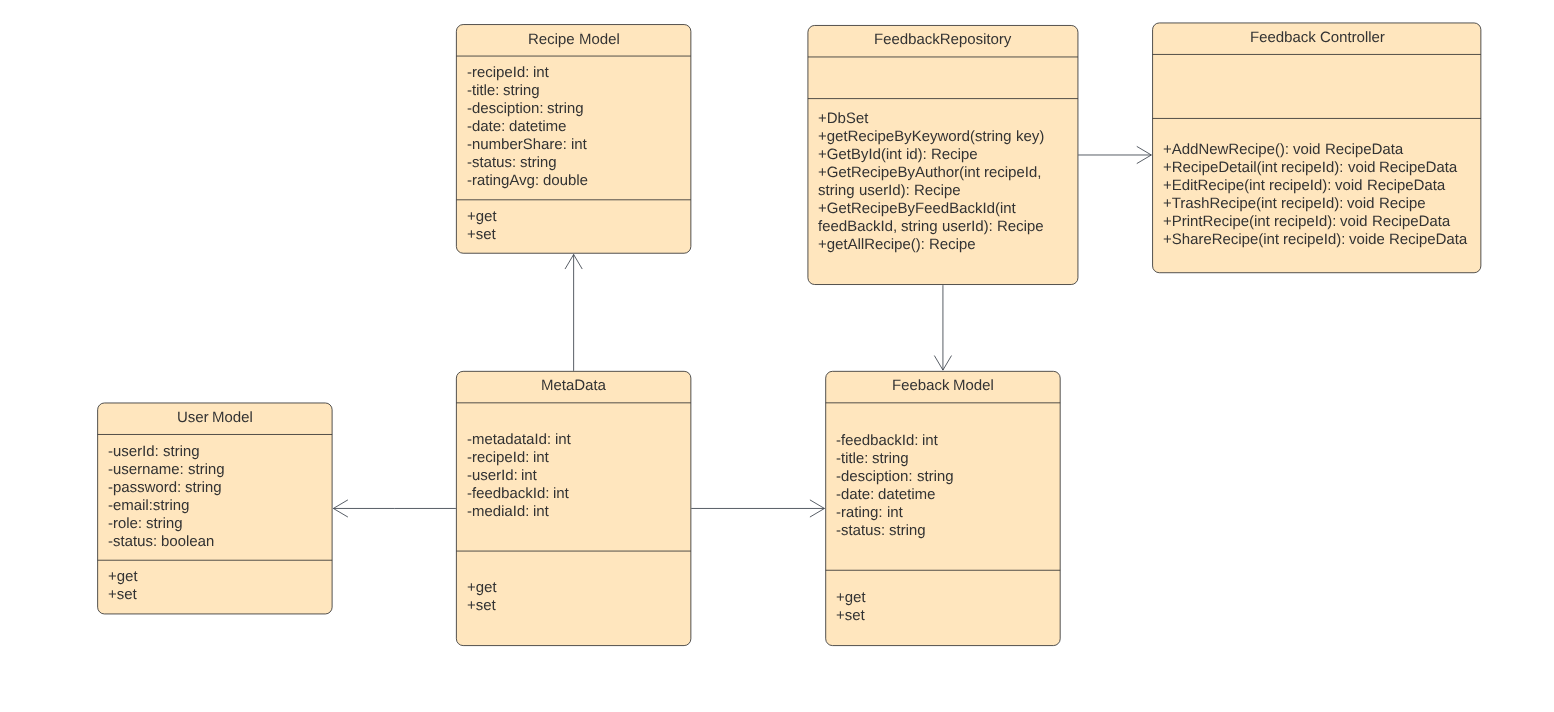
FROM Recipe r

INNER JOIN Metadata m ON r.RecipeId = m.RecipeId

WHERE m.UserId = 'UserId'

## **10. User Feedback List Page**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **FeedbackController Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | doGet UserFeedbackList() | Function to display all feedbacks of a user |
| 02 | doGet RecipeFeedbackList() | Function to display all feedbacks of a recipe |
| 03 | AddFeedback(FeedbackViewModel feedbackViewModel) | Function used to add a new feedback of a recipe from a user |

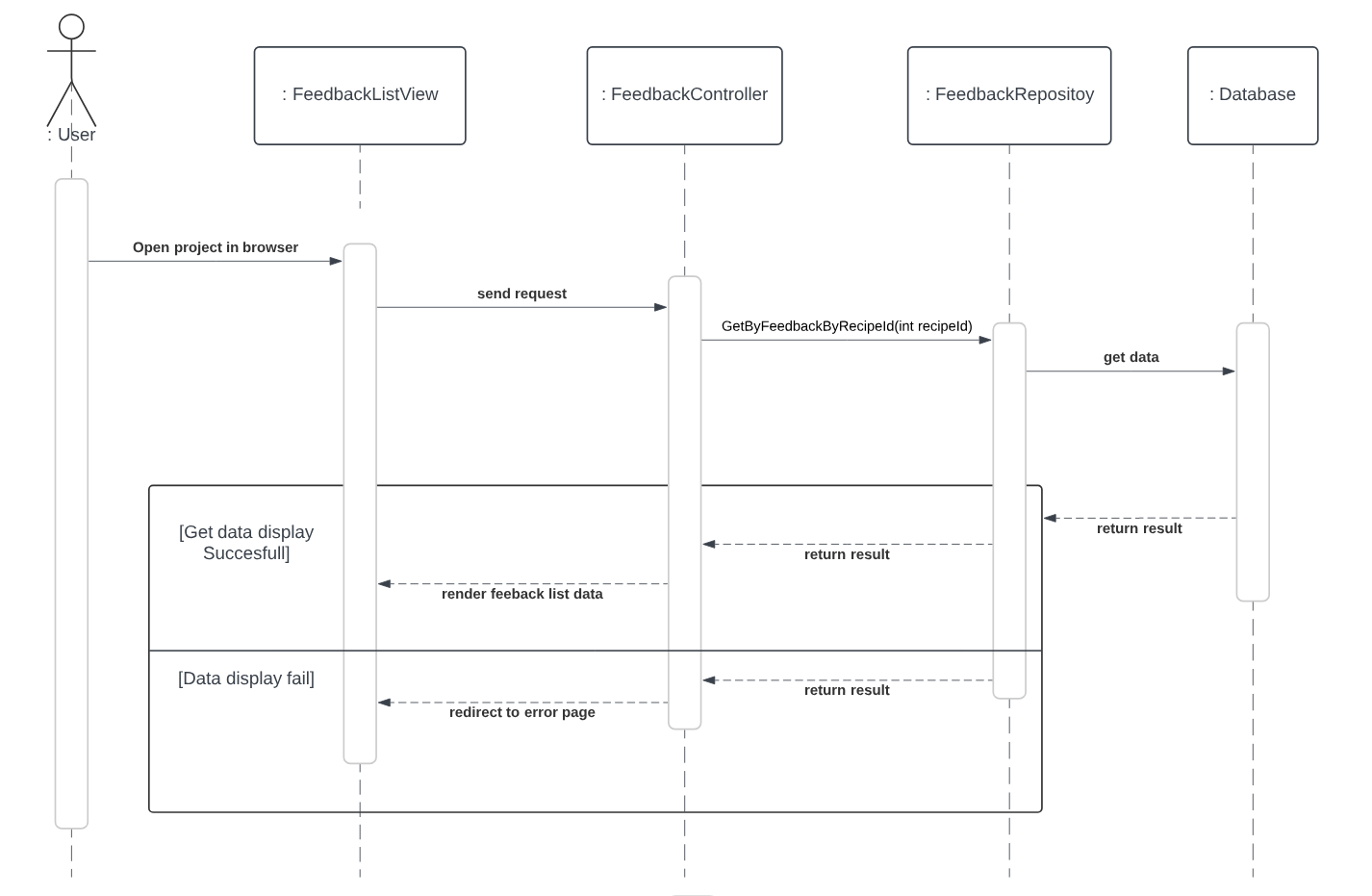
#### **FeedbackRepository Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | valueAvgRateRecipe(int recipeId) | The function is used to calculate the rating formula of the average value of a recipe and store it in the database |
| 02 | GetByFeedbackByUser(string userId) | The function is used to query the data, get feedbacks through the user id from the database |
| 03 | GetByFeedbackByRecipeId(int recipeId) | Function used to query data, get all feedback through recipe id from the database |

#### **RepositoryBase Class**

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SearchByProperty(Expression<Func<T, bool>> predicate) | Function used to query data, get model list from database |
| 02 | GetAll() | Function used to query data, get all in model from database |
| 03 | Add(T entity) | Function used to add data to the database |
| 04 | Update(T entity) | Function used to edit data into the database |
| 05 | Delete(T entity) | Function used to delete data into the database |

### **c. Sequence Diagram**

**

### **d. Database Queries**

SELECT \*

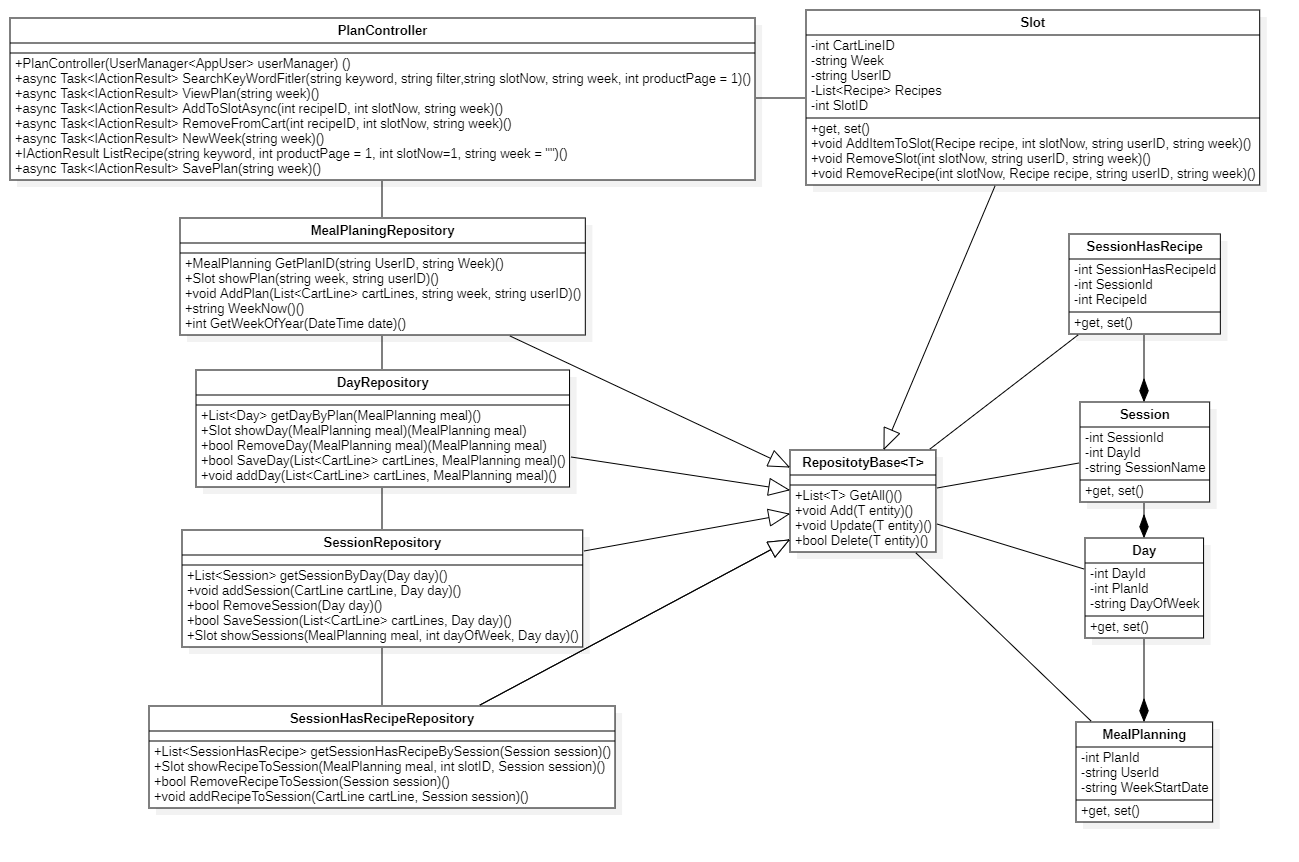
FROM Feedback f

INNER JOIN Metadata m ON f.FeedbackId = m.FeedbackId

WHERE m.UserId = 'UserId'

## **11. Show Meal Planning**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **PlanController Class**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | ViewPlan(string week) | Show view plan |

#### **MealPlanningRepository Class**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | showPlan() | Get plan from database, check day of plan, show slot |

#### **DayRepository Class**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | showDay() | Get day from database, check session of plan, add slot |

#### **SessionRepository Class**

***Class Methods***

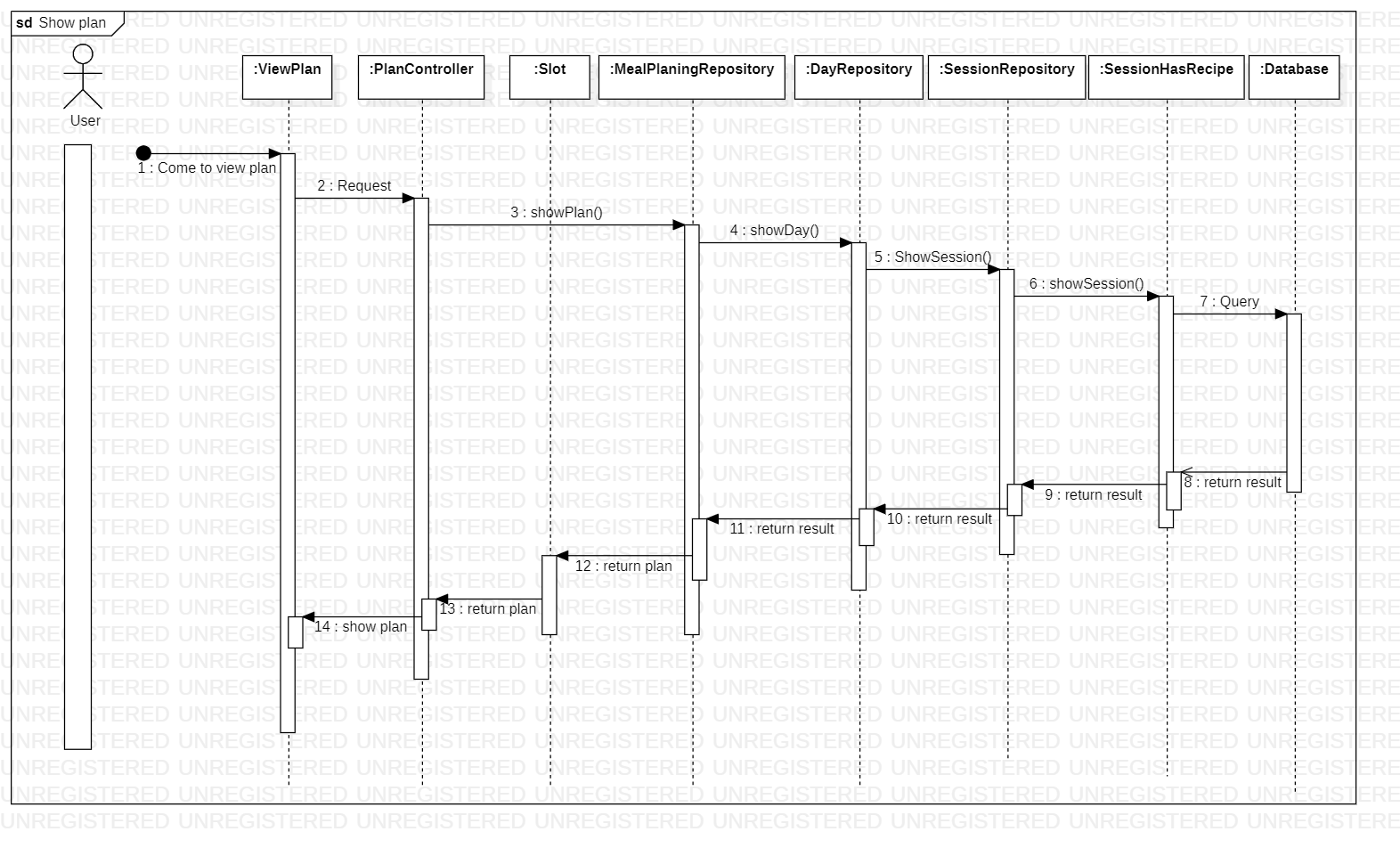
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | showSessions() | Get session from database, check recipe of plan, add slot |

#### **SessionHasRecipeRepositoryClass**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | showRecipeToSession() | Get recipe from database, add slot |

### **c. Sequence Diagram(s)**

**

### **d. Database Queries**

USE [Recipe\_Organizer]

GO

INSERT INTO [dbo].[MealPlanning]

([user\_id]

,[week\_start\_date])

VALUES

(<user\_id, nvarchar(450),>

,<week\_start\_date, varchar(10),>)

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Day](

[day\_id] [int] IDENTITY(1,1) NOT NULL,

[plan\_id] [int] NOT NULL,

[day\_of\_week] [varchar](10) NOT NULL,

CONSTRAINT [PK\_Day] PRIMARY KEY CLUSTERED

(

[day\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Day] WITH CHECK ADD CONSTRAINT [FK\_Day\_MealPlanning] FOREIGN KEY([plan\_id])

REFERENCES [dbo].[MealPlanning] ([plan\_id])

GO

ALTER TABLE [dbo].[Day] CHECK CONSTRAINT [FK\_Day\_MealPlanning]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Session](

[session\_id] [int] IDENTITY(1,1) NOT NULL,

[day\_id] [int] NOT NULL,

[session\_name] [varchar](10) NOT NULL,

CONSTRAINT [PK\_Session] PRIMARY KEY CLUSTERED

(

[session\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Session] WITH CHECK ADD CONSTRAINT [FK\_Session\_Day] FOREIGN KEY([day\_id])

REFERENCES [dbo].[Day] ([day\_id])

GO

ALTER TABLE [dbo].[Session] CHECK CONSTRAINT [FK\_Session\_Day]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Recipe\_has\_Category](

[category\_id] [int] NOT NULL,

[recipe\_id] [int] NOT NULL,

[id] [int] IDENTITY(1,1) NOT NULL,

CONSTRAINT [PK\_Recipe\_has\_Category] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Recipe\_has\_Category] WITH CHECK ADD CONSTRAINT [FK\_Recipe\_has\_Category\_Category1] FOREIGN KEY([category\_id])

REFERENCES [dbo].[Category] ([category\_id])

GO

ALTER TABLE [dbo].[Recipe\_has\_Category] CHECK CONSTRAINT [FK\_Recipe\_has\_Category\_Category1]

GO

ALTER TABLE [dbo].[Recipe\_has\_Category] WITH CHECK ADD CONSTRAINT [FK\_Recipe\_has\_Category\_Recipe] FOREIGN KEY([recipe\_id])

REFERENCES [dbo].[Recipe] ([recipe\_id])

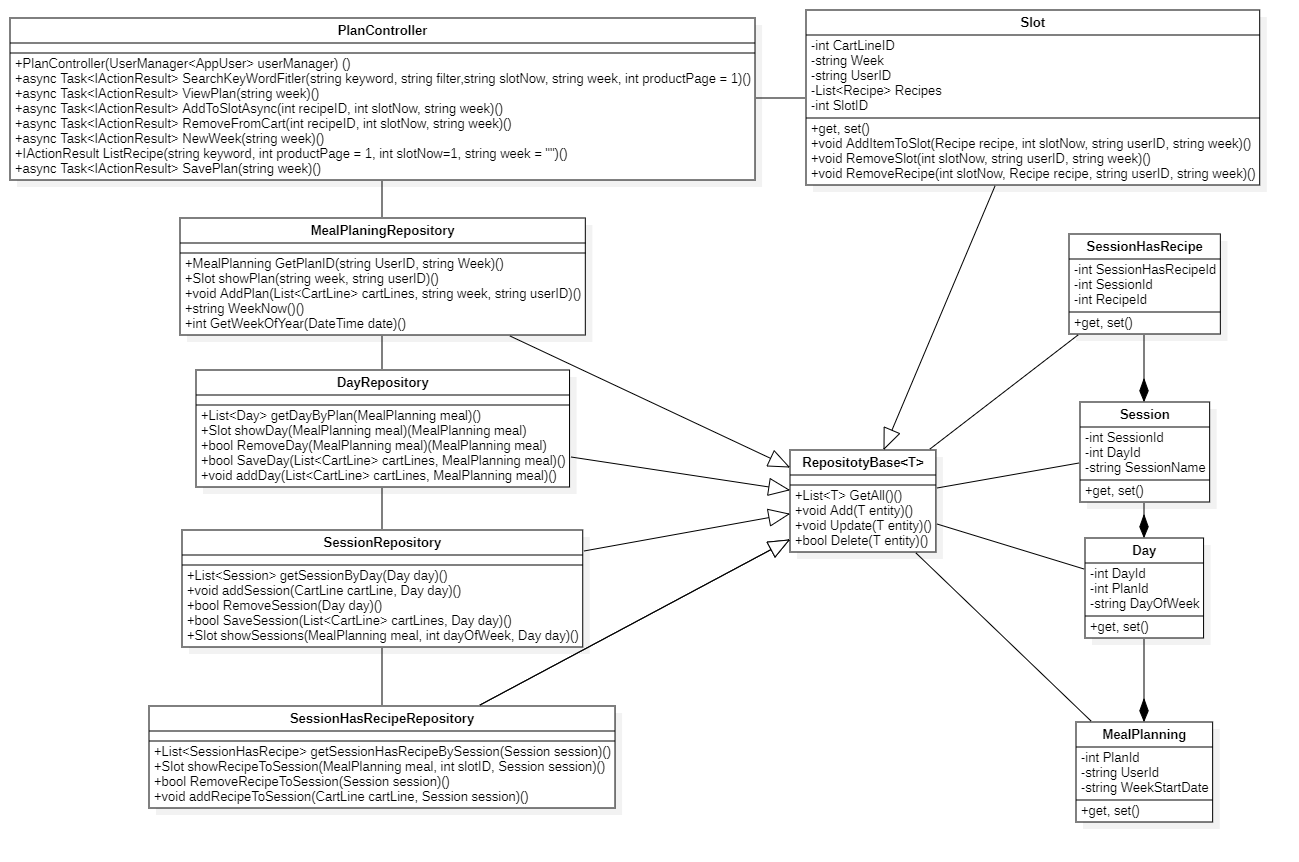
GO

ALTER TABLE [dbo].[Recipe\_has\_Category] CHECK CONSTRAINT [FK\_Recipe\_has\_Category\_Recipe]

GO

## **12. Add Recipe To Plan**

### **a. Class Diagram**



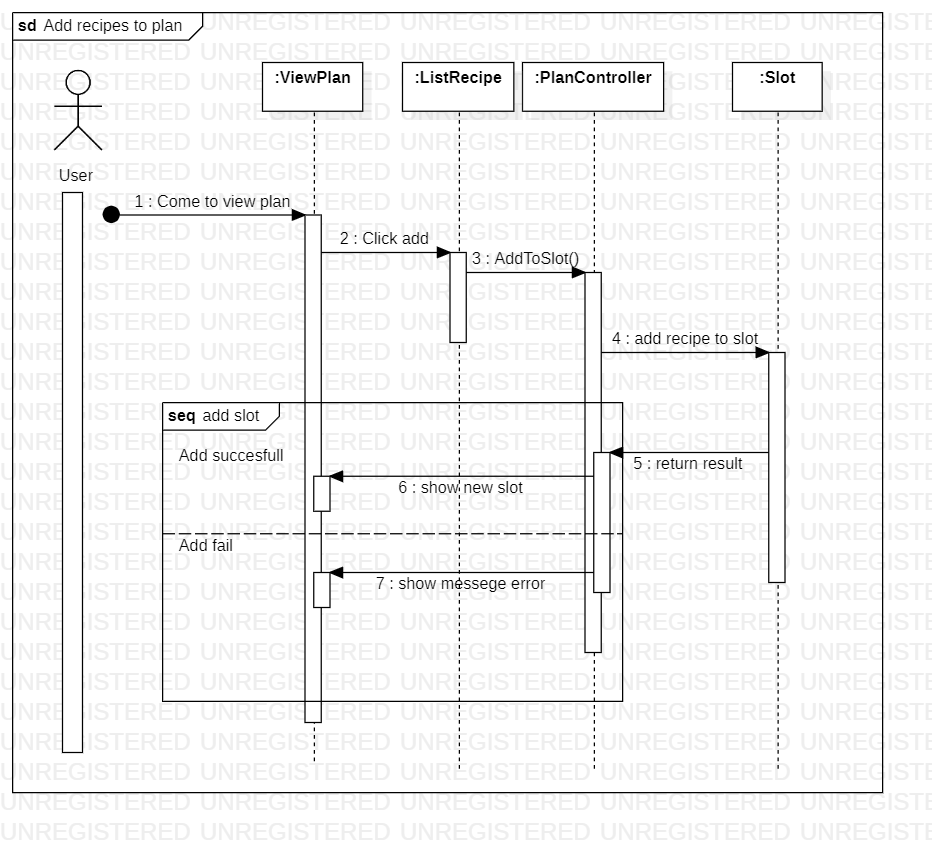
### **b. Class Specifications**

#### **PlanController Class**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | AddToSlot() | Check recipe in slot, add recipe to slot |

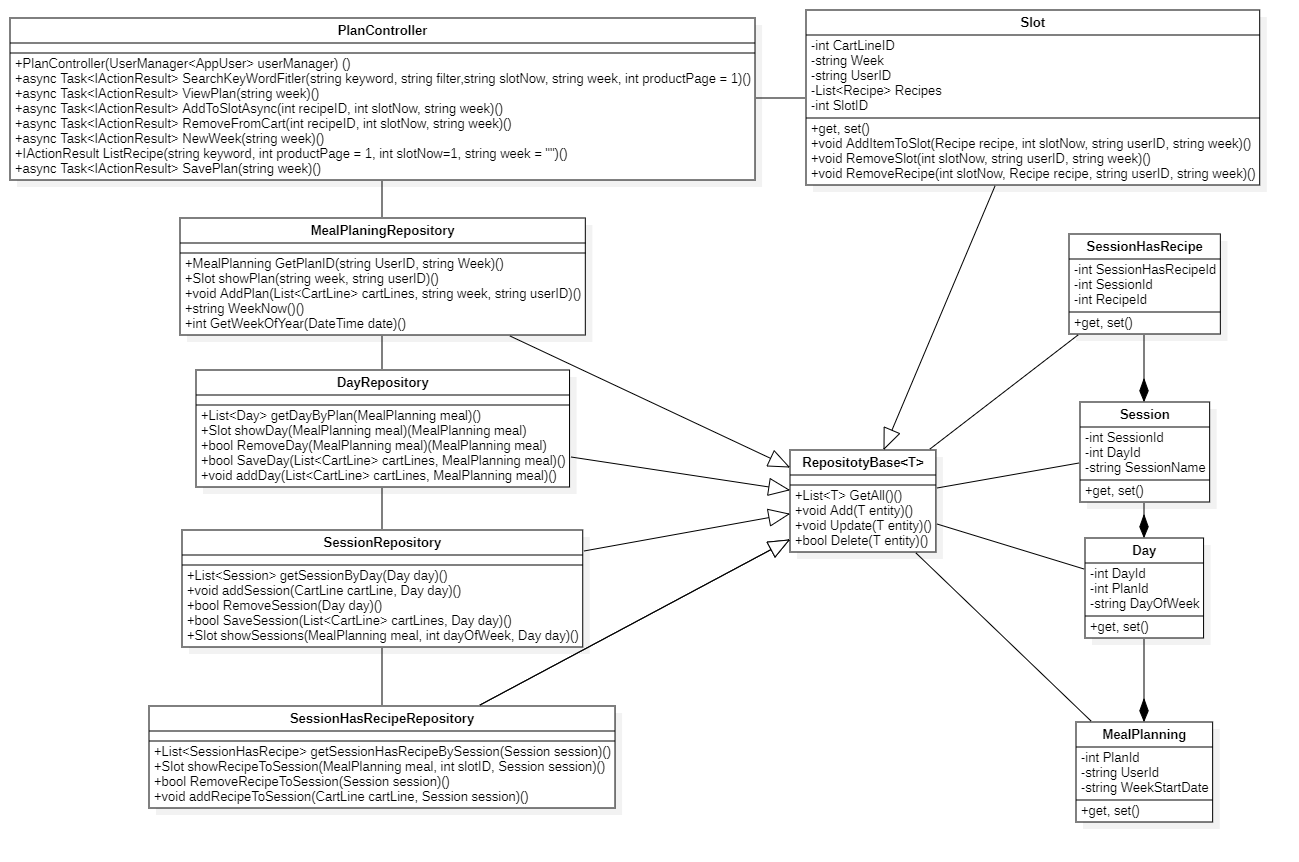
### **c. Sequence Diagram(s)**

**

### 

## **13. Remove Recipe From Plan**

### **a. Class Diagram**



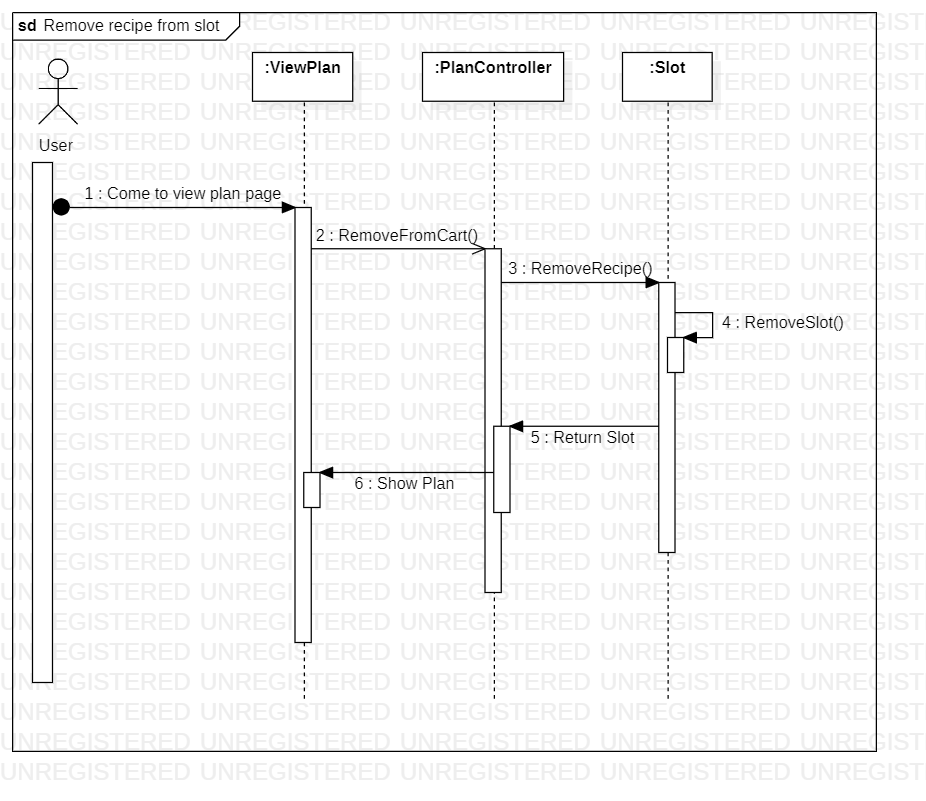
### **b. Class Specifications**

#### **PlanController Class**

***Class Methods***

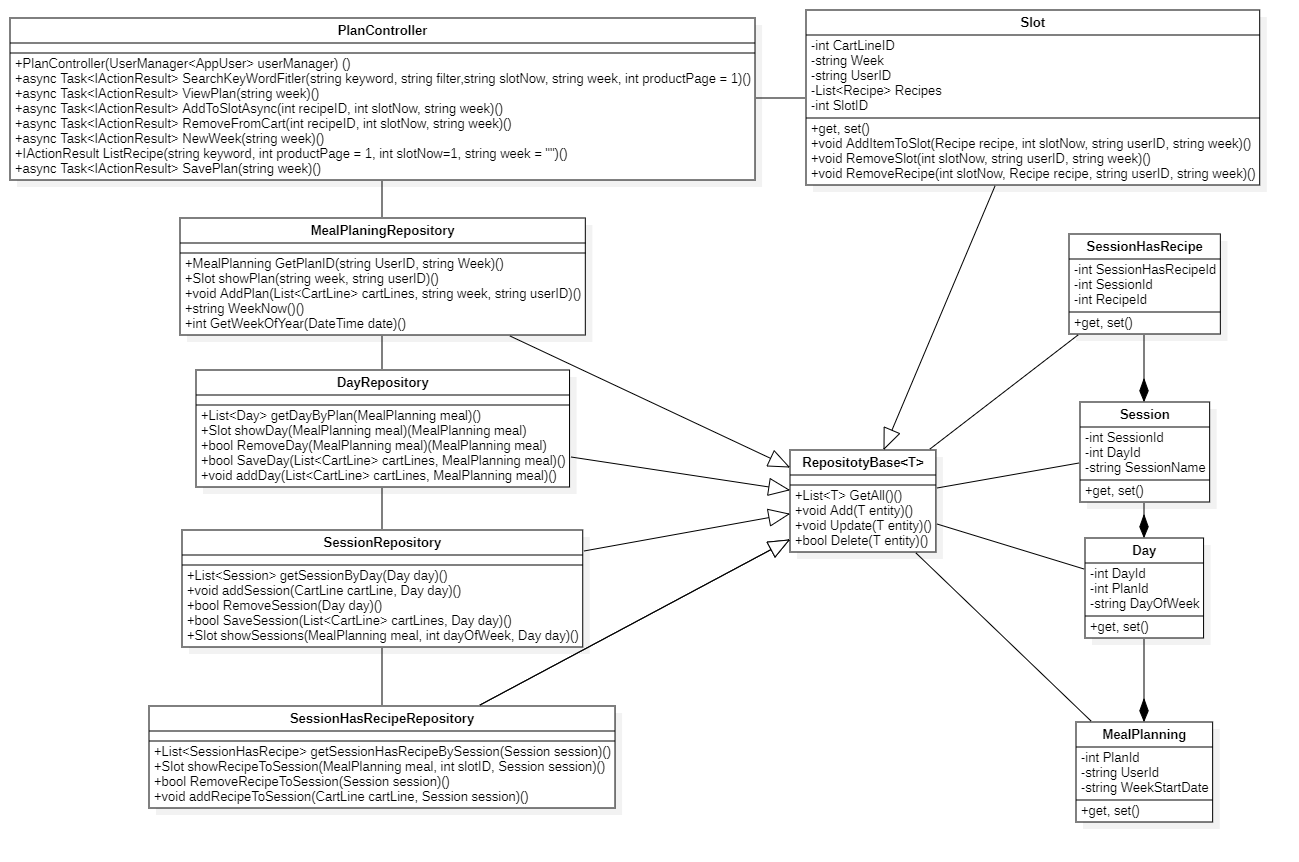
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | RemoveFromCart() | Check recipe in slot, remove recipe from slot |

### **c. Sequence Diagram(s)**

**

## **14. Save plan**

### **a. Class Diagram**

**

### **b. Class Specifications**

#### **PlanController Class**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SavePlan() | Get slot to save plan |

#### **MealPlanningRepository Class**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | AddPlan() | Check plan from slot, create plan in database |

#### **DayRepository Class**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SaveDay() | Check day from slot, create day in database |

#### **SessionRepository Class**

***Class Methods***

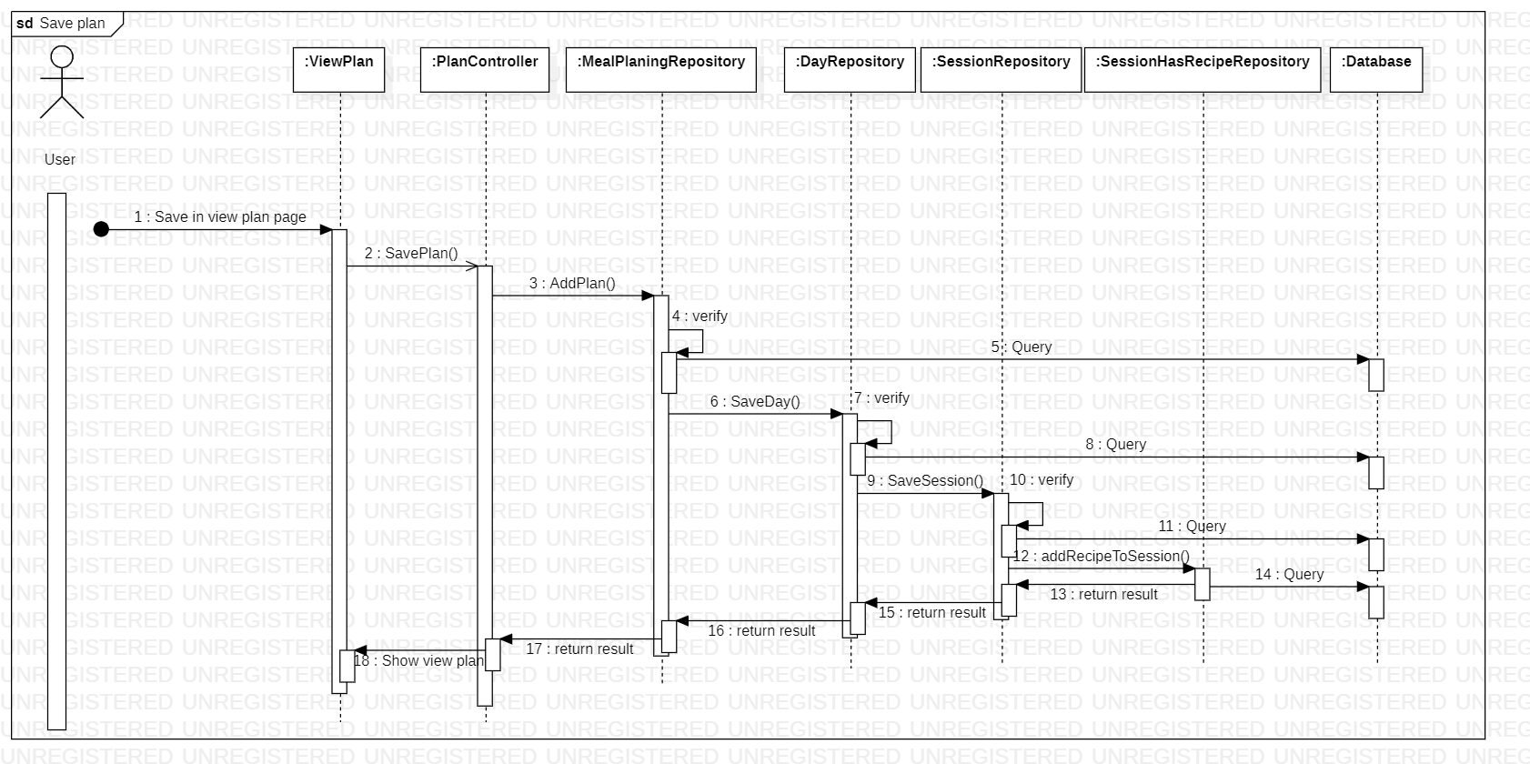
| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | SaveSession() | Check session from slot, create session in database |

#### **SessionHasRecipeRepositoryClass**

***Class Methods***

| **No** | **Method** | **Description** |
| --- | --- | --- |
| 01 | addRecipeToSession() | Check recipefrom slot, add or remove SessionHasRecipeRepository in database |

### **c. Sequence Diagram(s)**

**

### **d. Database Queries**

USE [Recipe\_Organizer]

GO

INSERT INTO [dbo].[MealPlanning]

([user\_id]

,[week\_start\_date])

VALUES

(<user\_id, nvarchar(450),>

,<week\_start\_date, varchar(10),>)

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Day](

[day\_id] [int] IDENTITY(1,1) NOT NULL,

[plan\_id] [int] NOT NULL,

[day\_of\_week] [varchar](10) NOT NULL,

CONSTRAINT [PK\_Day] PRIMARY KEY CLUSTERED

(

[day\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Day] WITH CHECK ADD CONSTRAINT [FK\_Day\_MealPlanning] FOREIGN KEY([plan\_id])

REFERENCES [dbo].[MealPlanning] ([plan\_id])

GO

ALTER TABLE [dbo].[Day] CHECK CONSTRAINT [FK\_Day\_MealPlanning]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Session](

[session\_id] [int] IDENTITY(1,1) NOT NULL,

[day\_id] [int] NOT NULL,

[session\_name] [varchar](10) NOT NULL,

CONSTRAINT [PK\_Session] PRIMARY KEY CLUSTERED

(

[session\_id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Session] WITH CHECK ADD CONSTRAINT [FK\_Session\_Day] FOREIGN KEY([day\_id])

REFERENCES [dbo].[Day] ([day\_id])

GO

ALTER TABLE [dbo].[Session] CHECK CONSTRAINT [FK\_Session\_Day]

GO

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Recipe\_has\_Category](

[category\_id] [int] NOT NULL,

[recipe\_id] [int] NOT NULL,

[id] [int] IDENTITY(1,1) NOT NULL,

CONSTRAINT [PK\_Recipe\_has\_Category] PRIMARY KEY CLUSTERED

(

[id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Recipe\_has\_Category] WITH CHECK ADD CONSTRAINT [FK\_Recipe\_has\_Category\_Category1] FOREIGN KEY([category\_id])

REFERENCES [dbo].[Category] ([category\_id])

GO

ALTER TABLE [dbo].[Recipe\_has\_Category] CHECK CONSTRAINT [FK\_Recipe\_has\_Category\_Category1]

GO

ALTER TABLE [dbo].[Recipe\_has\_Category] WITH CHECK ADD CONSTRAINT [FK\_Recipe\_has\_Category\_Recipe] FOREIGN KEY([recipe\_id])

REFERENCES [dbo].[Recipe] ([recipe\_id])

GO

ALTER TABLE [dbo].[Recipe\_has\_Category] CHECK CONSTRAINT [FK\_Recipe\_has\_Category\_Recipe]

GO

### 