Group 01

Foodaholic
Software Architecture Document

Version 1.0

Revision History

Date	Version	Description	Author
22/Jun/22	1.0	Initial draft	Lê Trọng Anh Tú Nguyễn Thiện Nhân Phan Tuấn Khải Lê Đăng Khoa

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system

1.2 Scope

This document is an architectural overview of the Foodaholic website. This website is hosted online which supplies recipes, trending food, etc.

This document describes the current architectural structure of our website and is intended to be read by future developers who will join the project, and create and interface their own website of this project.

1.3 Definitions, Acronyms and Abbreviations

There are some acronyms and abbreviations we use in this document:

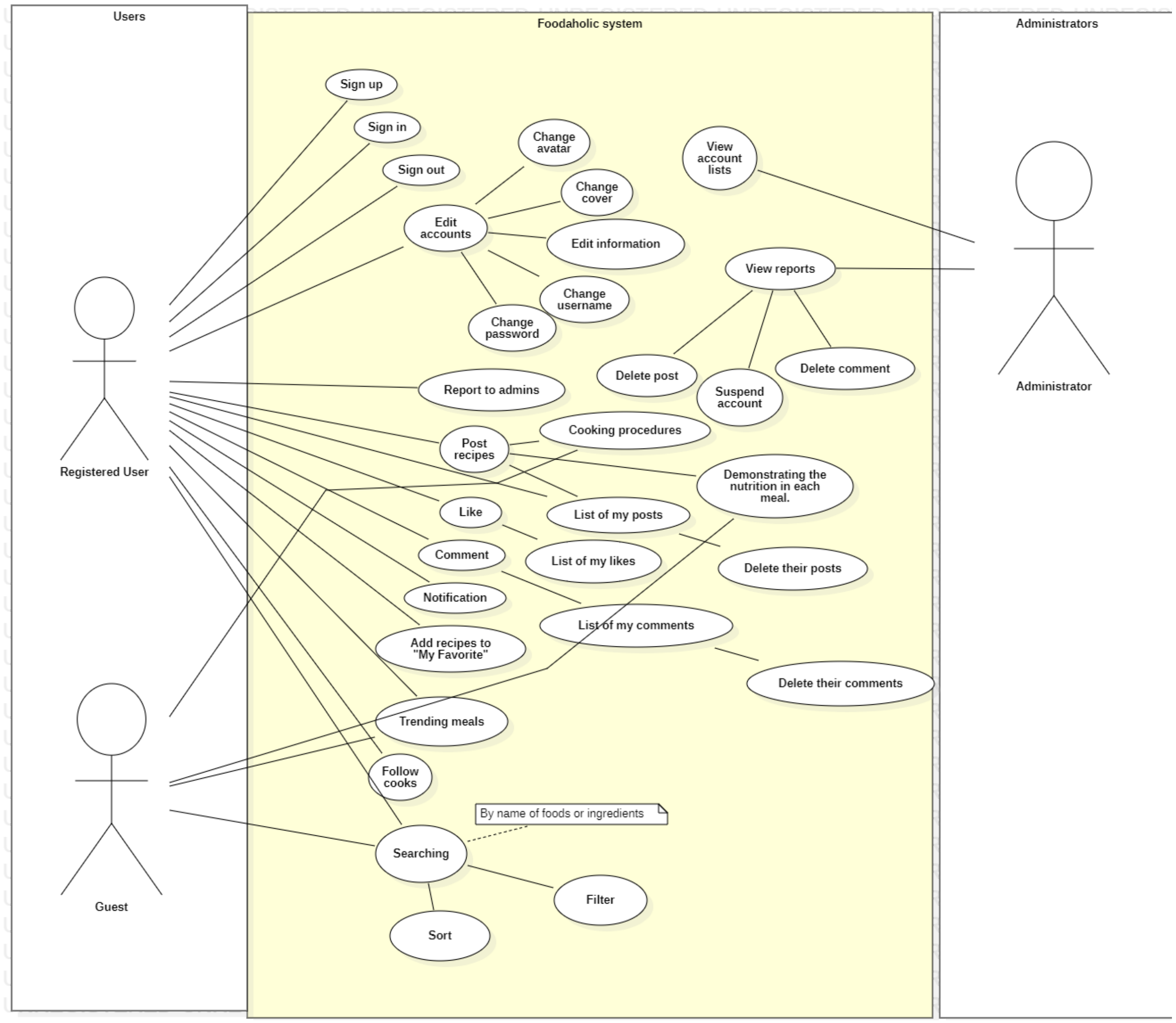
- GUI: Graphical User Interface - interact with user by interface
- SQL: Structured Query Language - a programming language to query database
- HTTP: Hypertext Transfer Protocol - a communicating method between web browsers and web servers
- API: Application Programming Interface - a software intermediary that allows two applications to talk to each other

2. Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impacts on the architecture and also captures the special constraints that may apply.

- **Safety:** make sure there is no malicious code affecting the user's devices.
- **Security:** authenticate the user with username and password. For Internet access, we need the following requirements:
  - *Confidential:* transferred documents cannot be viewed by anybody else but the sender and the final recipient.
  - *Integrity:* guarantee integrity of the transferred files.
- **Privacy:** ensure user's privacy, must not get their information leaked without any permissions.
- **Persistence:** data persistence will be addressed using a relational database.
- **Performance:** the web should reply to the clients' commands as soon as possible.
- **Availability**: the web must be available all time (24/7) except when there are maintenance activities.
- **Capacity:** server must have large storage to save data such as storing all posts, accounts, etc.
- **Server-Client:** both server and client must connect to the Internet to be accessible.

3. Use-Case Model

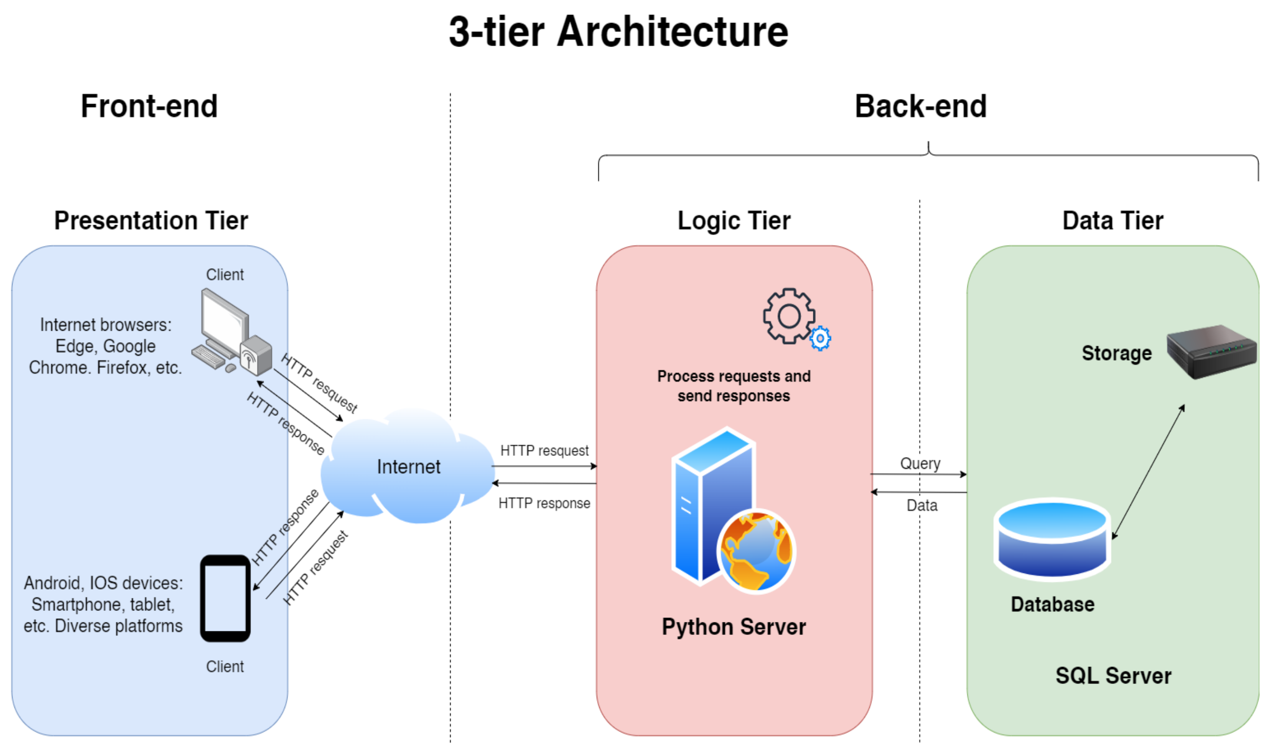


4. Logical View

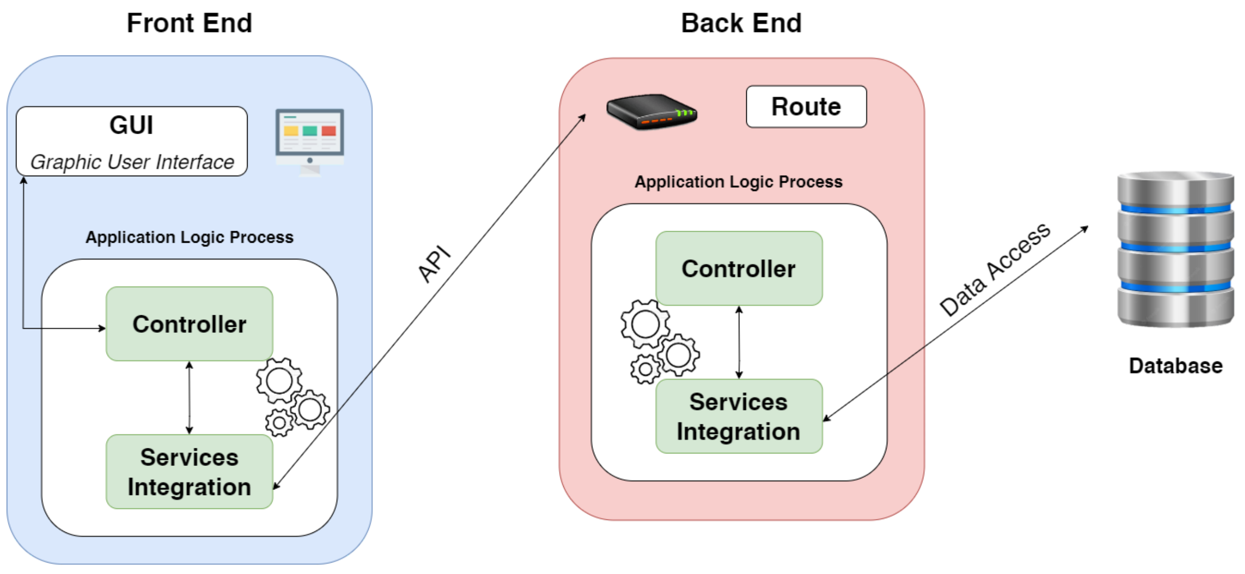
The logical view of the architecture describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers.

It also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

The web application is divided into layers based on 3-tier architecture.



And this is the particular model.

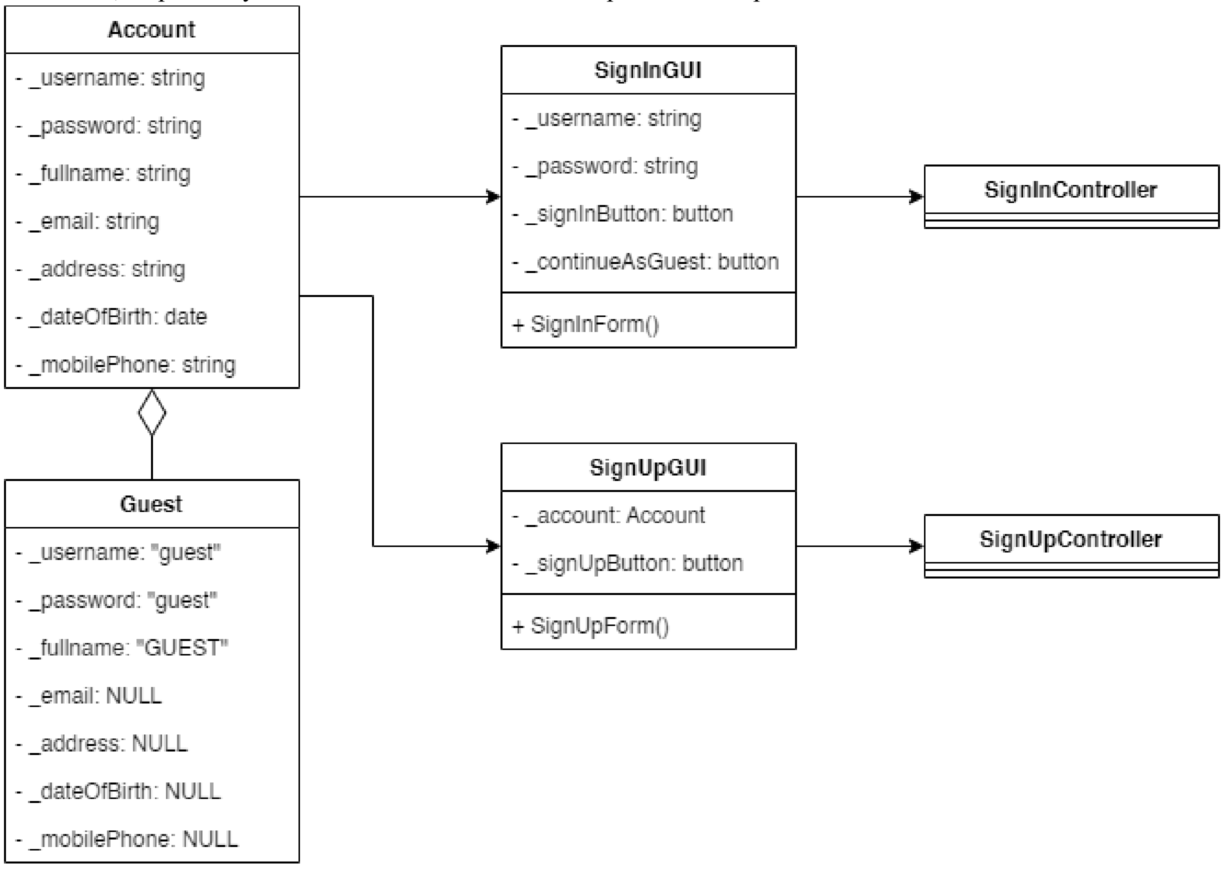


4.1 Component: Authentication GUI

In **Presentation tier**, the user will see the authentication forms. There are two forms: Sign In and Sign Up. Initially, the user is directed into “Sign In” page. They are required to enter the username and password in text bars, then click “Sign In” button to submit. On the other hand, if they do not have an account, the user will be then moved to sign up page aftering clicking "Don't have account?" button. After signing up successfully, they will be redirected back to "Sign In" form to login.

If the user clicks "Continue as guest" button in "Sign In" page, then they will access the website as a GUEST account.

Aftering logging in the account successfully (their account or GUEST account), they will be directed to the homepage. All the "Sign In" data the user entering in the SignInGUI and SignUpGUI will be sent to their suitable controllers, respectively. The controllers will handle and process the input data.

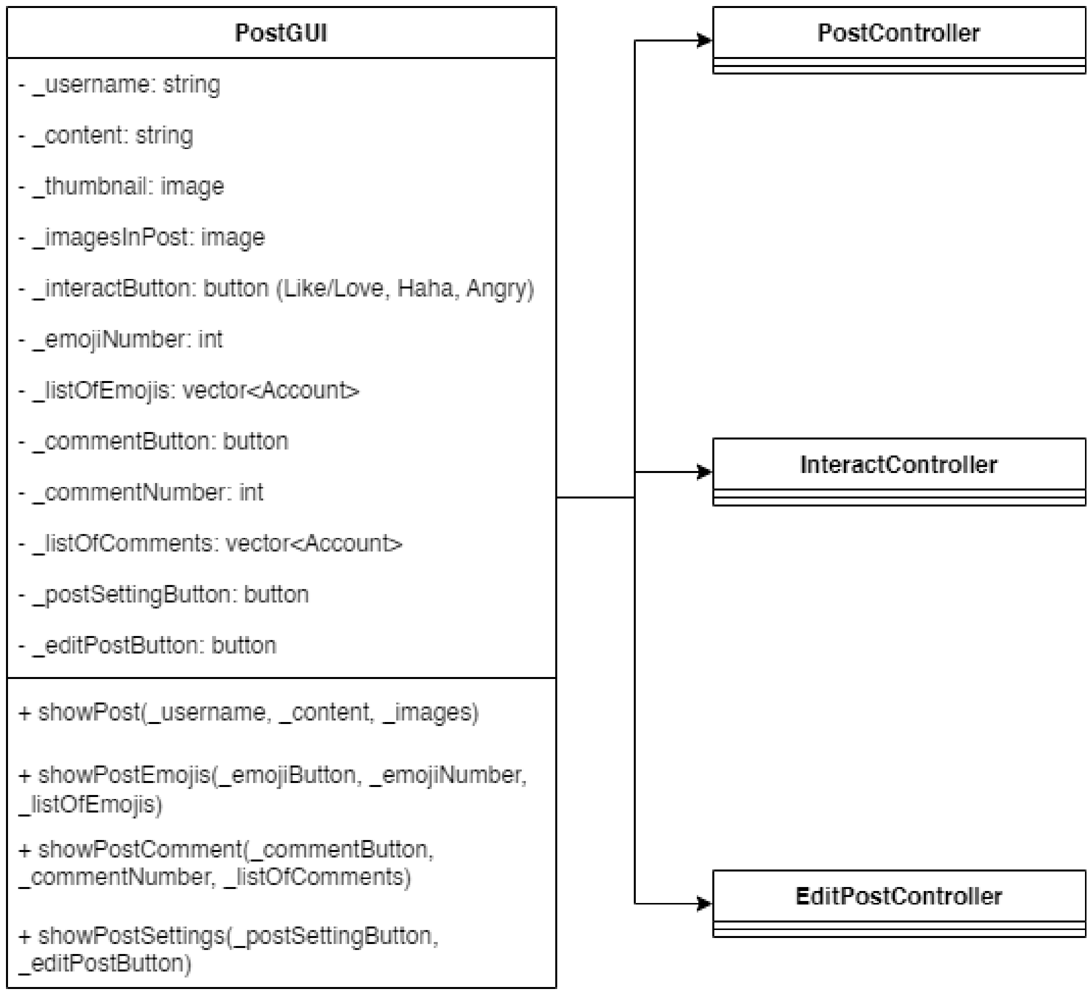


4.2 Component: Post GUI

In **Presentation tier**, the user is allowed to see a post. The post GUI will contain its content, images, and the posting account. Furthermore, there are buttons for users to interact with the post through emojis (Like/Love, Haha, Angry) and to open the textbox for commenting on the post. All of the interactions and comments would be put below the post content.

If the user chooses to post a new recipe, they will be shown the form to enter the recipe. Then the input data is sent to controller to handle the posting process.

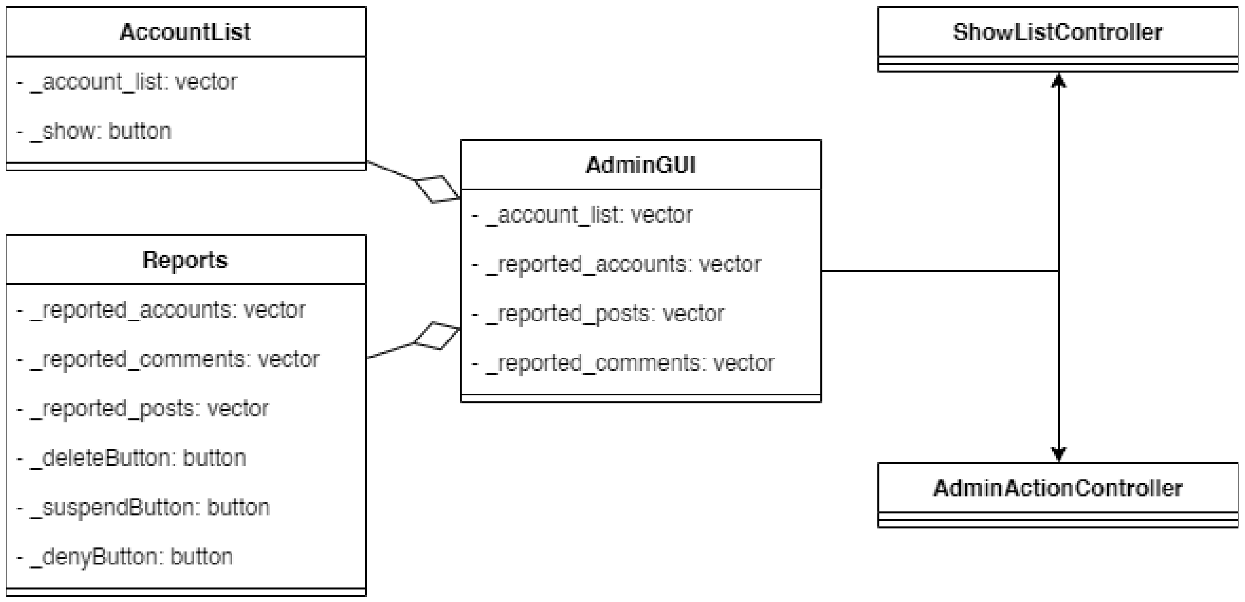
There will be three controllers which are PostController, InteractController, EditPostController to handle and process the data of the post.


PostGUI
- _username: string
- _content: string
- _thumbnail: image
- _imagesInPost: image
- _interactButton: button (Like/Love, Haha, Angry)
- _emojiNumber: int
- _listOfEmojis: vector<Account>
- _commentButton: button
- _commentNumber: int
- _listOfComments: vector<Account>
- _postSettingButton: button
- _editPostButton: button
+ showPost(_username, _content, _images)
+ showPostEmojis(_emojiButton, _emojiNumber, _listOfEmojis)
+ showPostComment(_commentButton, _commentNumber, _listOfComments)
+ showPostSettings(_postSettingButton, _editPostButton)
PostController
InteractController
EditPostController

4.3 Component: Admin GUI

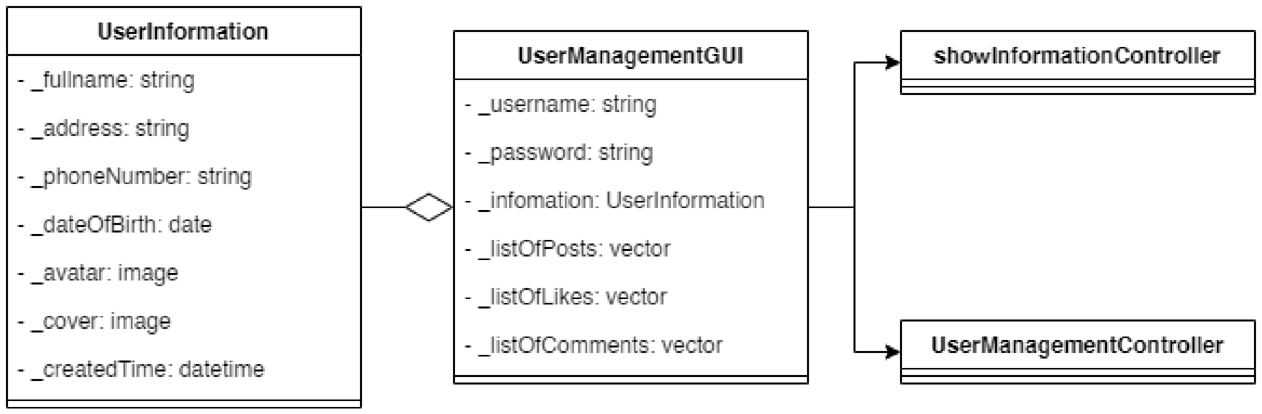
In **Presentation tier**, an administrator can view list of accounts and reports. The data will be processed by ShowListController to present a list of accounts, lists of reported accounts, posts, and comments on the admin screen.

When the admin choose to "Delete" a post or comment and "Suspend" an account, the AdminActionController will handle the task to delete or suspend it and the former list on the screen will be shortened because something was deleted. If the admin chooses "Deny" button, the list on the screen will still be shortened but the post/comment/account will not be deleted or banned.



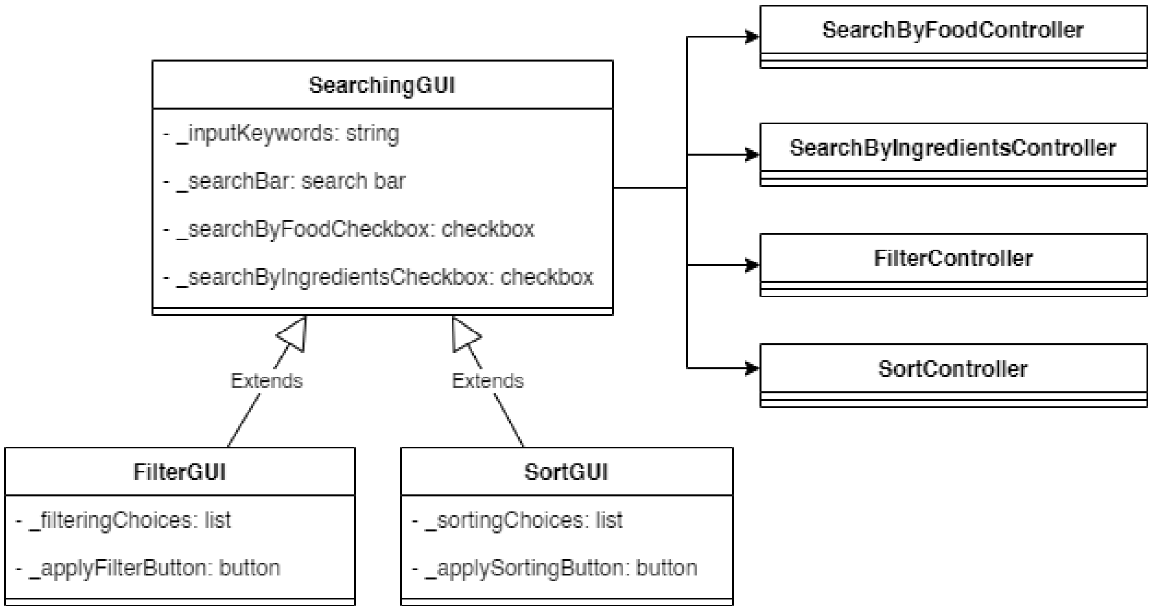
4.4 Component: Users Management GUI

In **Presentation tier**, the system shows all information of the user. They can view their username, password (hidden), detailed information, avatar, cover, and view their likes/posts/comments. The presentation of information in user's account page will be handled by showInformationController. If any part is edited, the information will be processed by UserManagementController and that on the screen will be soon updated after refreshing current page.



4.5 Component: Searching Result GUI

In **Presentation tier**, there is a search bar for the user to enter the keywords. Beside the search bar, there will be two checkboxes: searchByFoodCheckbox and searchByIngredientsCheckbox. Only on checkbox can be selected at the same time. There will be a filter for input requirements below before searching for the recipes. After the client receives the data from the server, in their page, there will be the list of results which can be applied to be filtered or sorted (by clicking "Apply Filter" or "Sorting" button). All of the presentation and process will be handled by the SearchingController.


SearchingGUI
- _inputKeywords: string
- _searchBar: search bar
- _searchByFoodCheckbox: checkbox
- _searchByIngredientsCheckbox: checkbox
SearchByFoodController
SearchByIngredientsController
FilterController
SortController
Extends
Extends
FilterGUI
- _filteringChoices: list
- _applyFilterButton: button
SortGUI
- _sortingChoices: list
- _applySortingButton: button

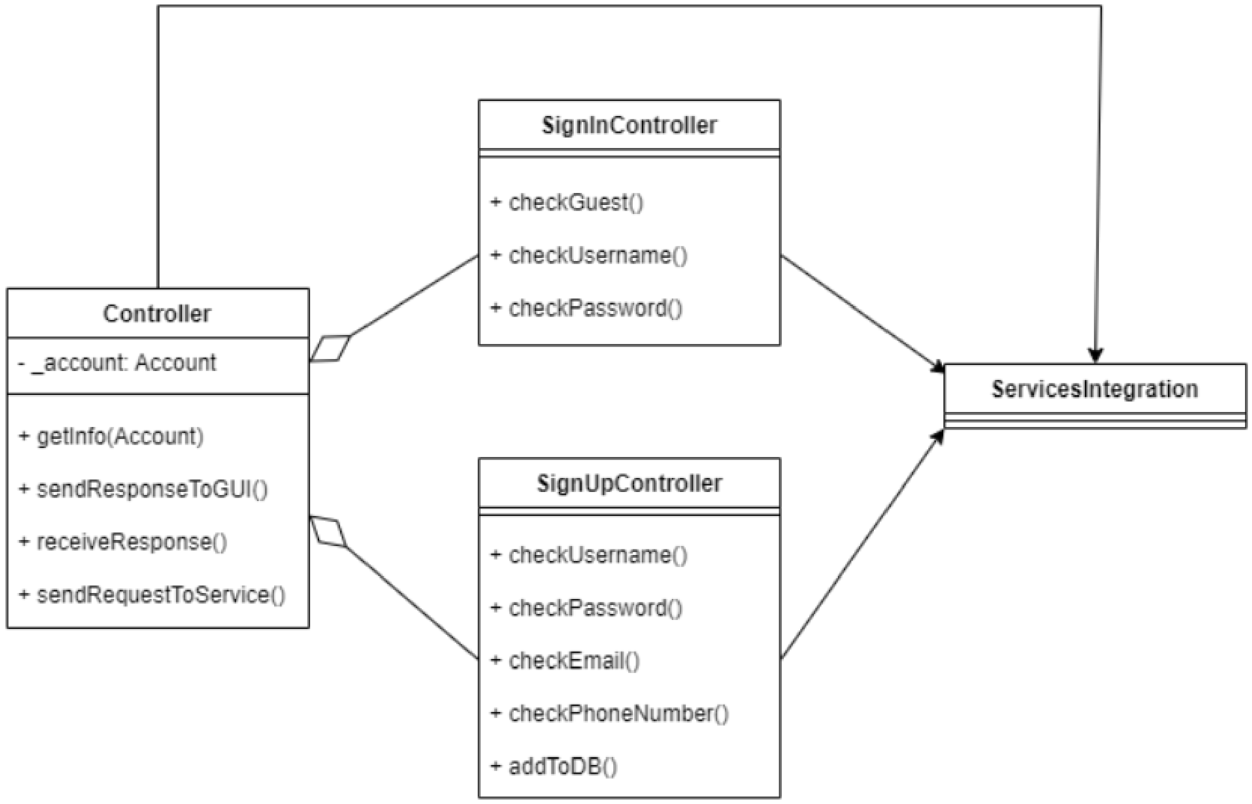
4.6 Component: Controller Authentication

For Authentication Controller, we build a father class called Controller. This class will receive requests from the GUI and get username and password entered by the user through function getInfo(). This class also receives responses from deeper processing components (or more detailed - Services Integration) and sends these responses to GUI to display on the user interface.

SignInController class - child class of Controller, used for checking account and password that the user enters through calling the Service so that the Service can access the Database to get all the accounts with the same username and check the password. Services Integration will send the response to the Controller whether sign in is successful or not, then the Controller will send that response to the GUI.

- About Guest, username and password which sent to Controller is “guest”

SignUpController class - another child class of Controller, used for checking whether the information entered by the user is valid (does the account exist in the Database, does the password match the re-entered password, is this email/phone number registered or not) through a Service similar to the SignInController class. If the information is valid to sign up, add that account to the Database through function addToDB(). Finally the Controller sends a response to the GUI to verify that account is successfully registered.



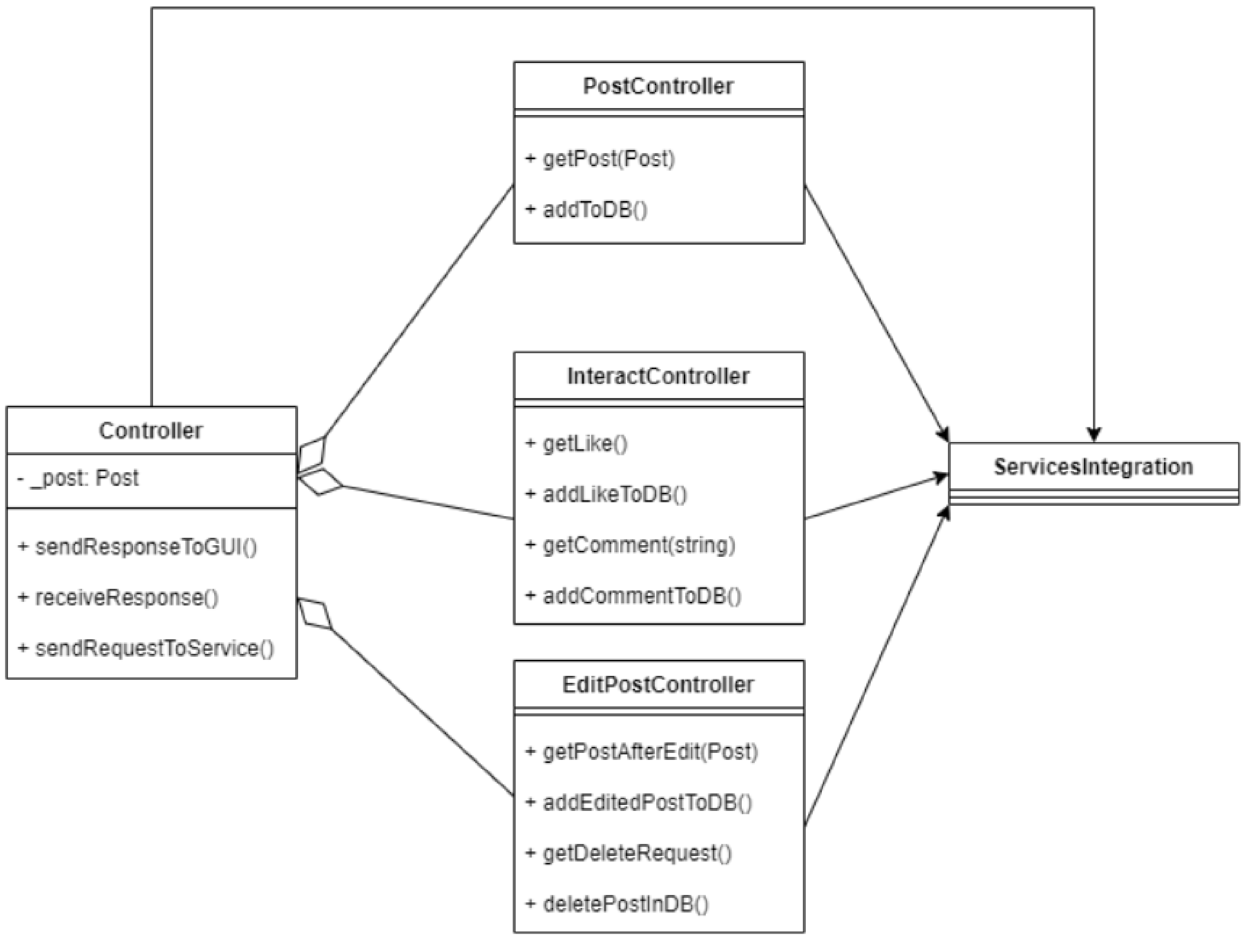
4.7 Component: Controller User Post

Controller (parent class) in this component will perform 3 tasks as 3 child classes: post a recipe (class PostController), interacting on posts (class InteractController) and editing/deleting posts (class EditPostController).

With the PostController class, we get the content of the post entered by the user (including images) into the Controller through the function getPost() with an input parameter of data type “Post”. Then pass it to the Services Integration to add to the Database by using the function addPostToDB(). After successfully added, the Services will send a response to the Controller to notice that the article has been updated in Database and posted successfully.

With the InteractController class we have 2 small tasks. Firstly, when someone likes a post, Services will call Database to find that post and increase the number of likes by 1. The second is to handle the comment on the post, we will put the comment content in the Database and update the comment in that post.

With the EditPostController class, we perform 2 tasks: edit post and delete post. When someone edits a post, the content of the post will be changed with the new one. When someone deletes the post, this post will be deleted in the Database.

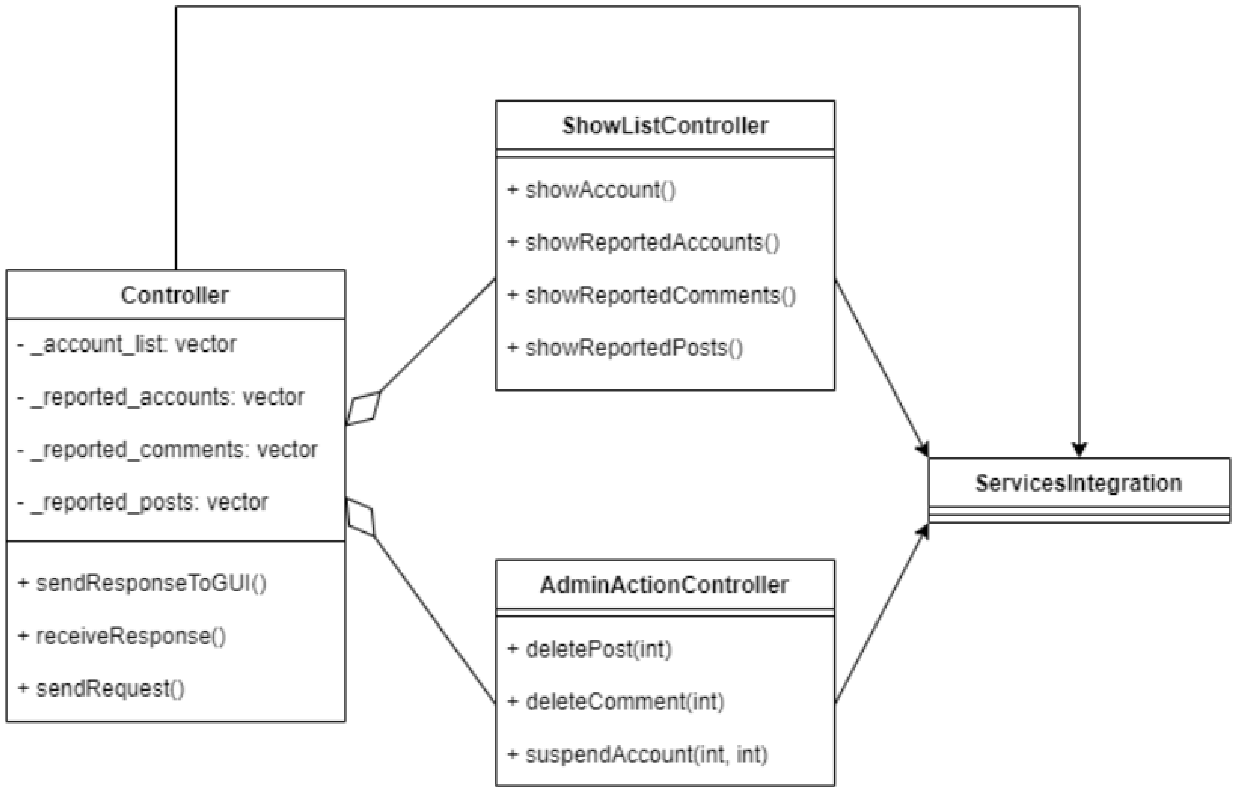


4.8 Component: Controller Admin Action

In the Presentation tier, we handle 2 main functions for the admin: display a list of accounts/reports and delete comments/posts or ban accounts within how many days.

In the ShowListController class, we call the function to display the list of accounts, display reports about accounts, comments or posts (see the image below for more information). The above functions will call Service Integration to access the Database to update data into the private variables in the Controller (parent class) as well as give the results (send response) to the GUI to display to the admin.

In the AdminActionController class, depending on the admin's choice at the GUI, the functions will be used or not. If the admin agrees with the report, these functions will call Service Integration to access the Database to delete the comment based on the id which admin passed in, delete the post based on the id which admin passed in, or update the account's suspend variable to some number (the first integer represents the account id and the second integer represents the number of days suspended)

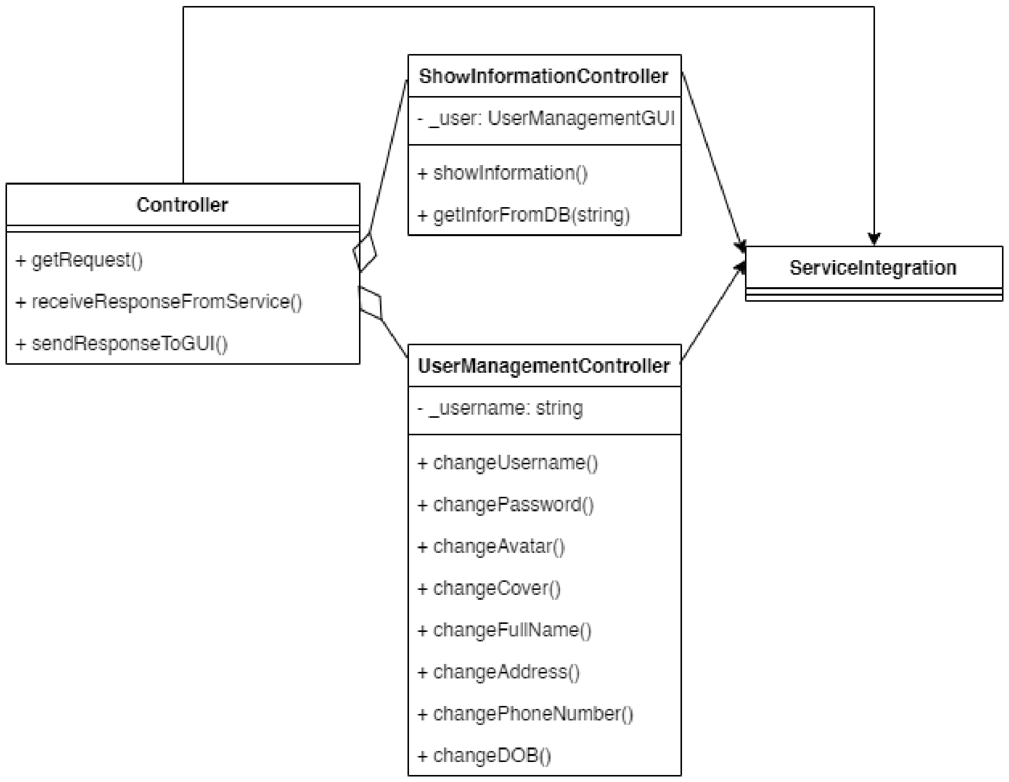


4.9 Component: User Management Controller

When the controller receives resquests from the GUI, it will analyze the command to know the user wants to display information (class ShowInformationController) or edit information (class UserManagementController)

In the ShowInformationController class, it will get the user information based on the username (because each account has a unique username) through the Service and give it to the private variable of this class. After finishing accessing the database to get the data, it will call the function to display that information to the user

In the UserManagementController class, the user will be able to change his/her information through the functions listed in the image below. These functions will ask the Service to access the Database to update database for that account.

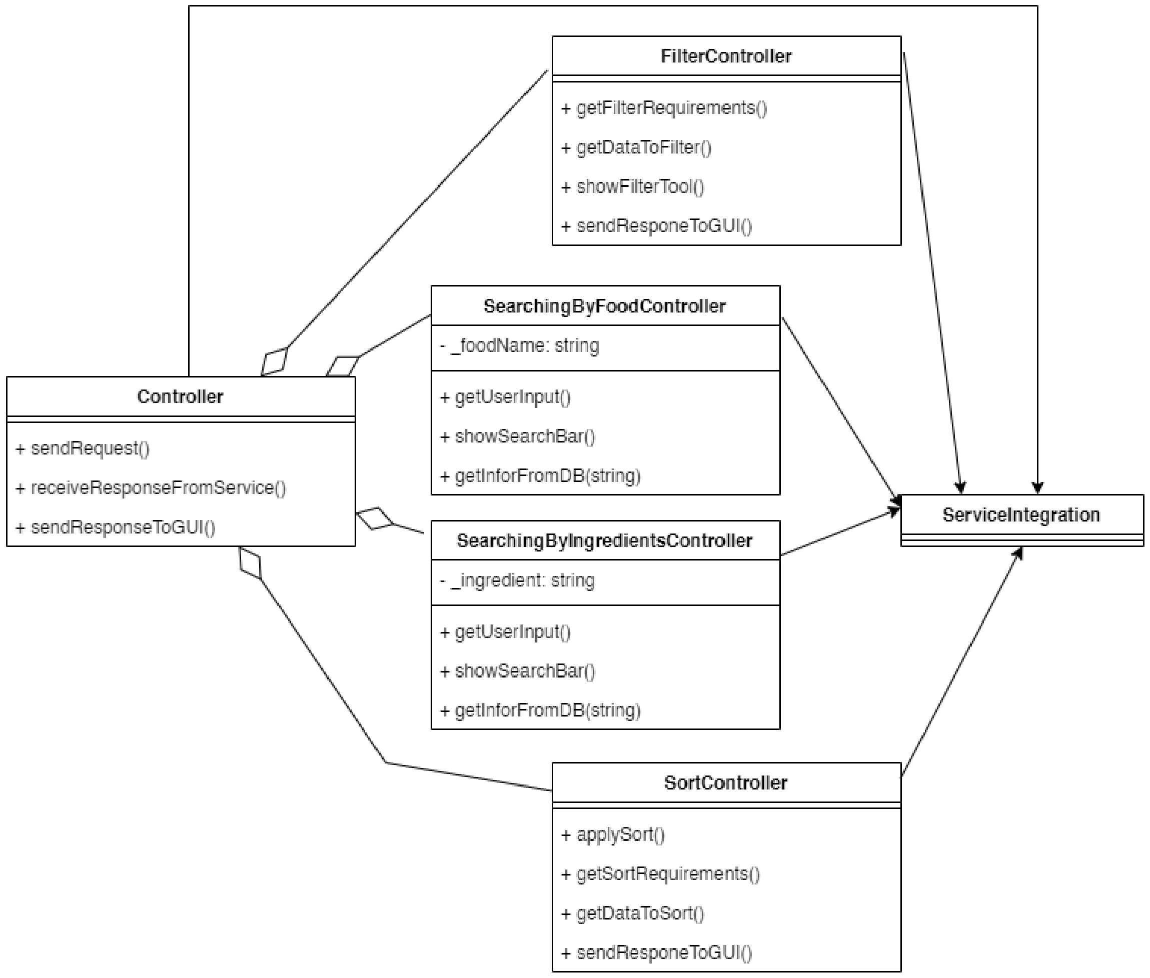


4.10 Component: Searching Controller

When the controller receives requests from the GUI, it will analyze to know the user wants to search by food (class SearchingByFoodController), to search by ingredients (class SearchingByIngredientsController), to filter (class FilterController), or to sort (SortController) the input data in the search bar and filter/sort tools.

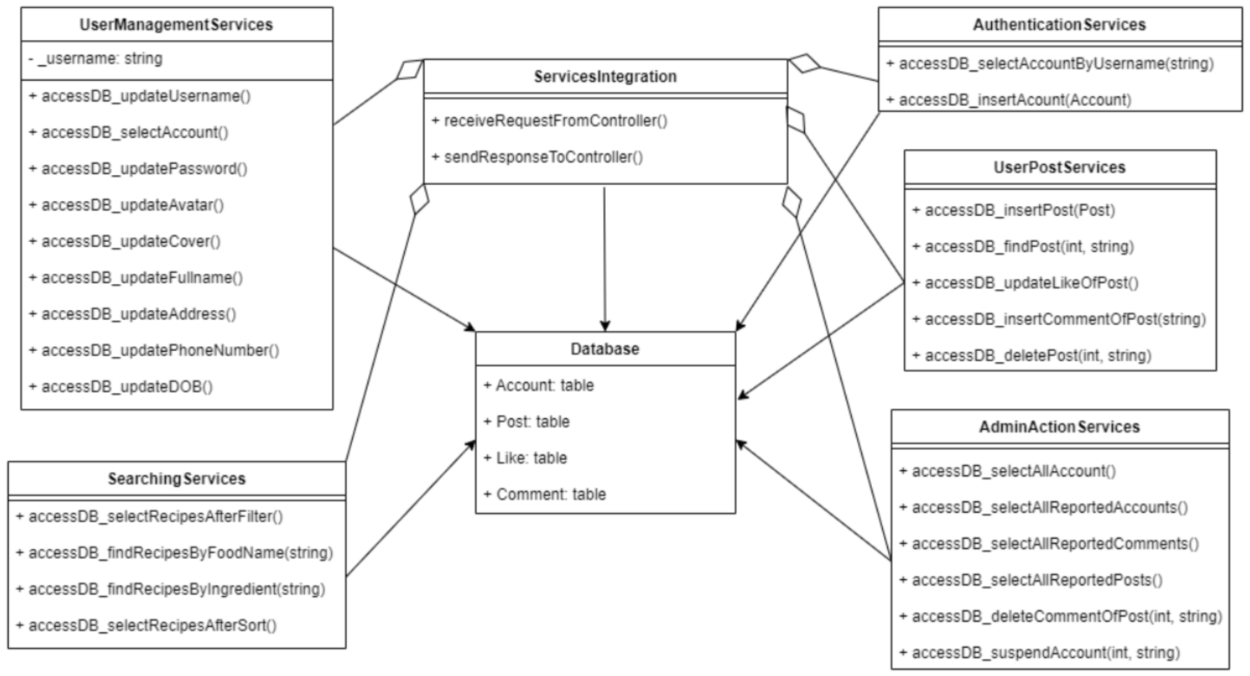
In the SearchingByFoodController class, it will get the food string in search bar and in the SearchingByIngredientsController class, it will get input string of ingredients (separated by commas) through the Service Integrations and send it to the private variable of this class. After finishing accessing the database to get the data, it will call the function to display the results to the user.

In the FilterController and SortController classes, it will access the input data in the search bar and chosen criteria in the filter/sort tools. Then, the data will be sent to Service Integrations to handle the process. After finishing accessing the database to get the data, the results will be updated and the newly-found results will be displayed to the user..



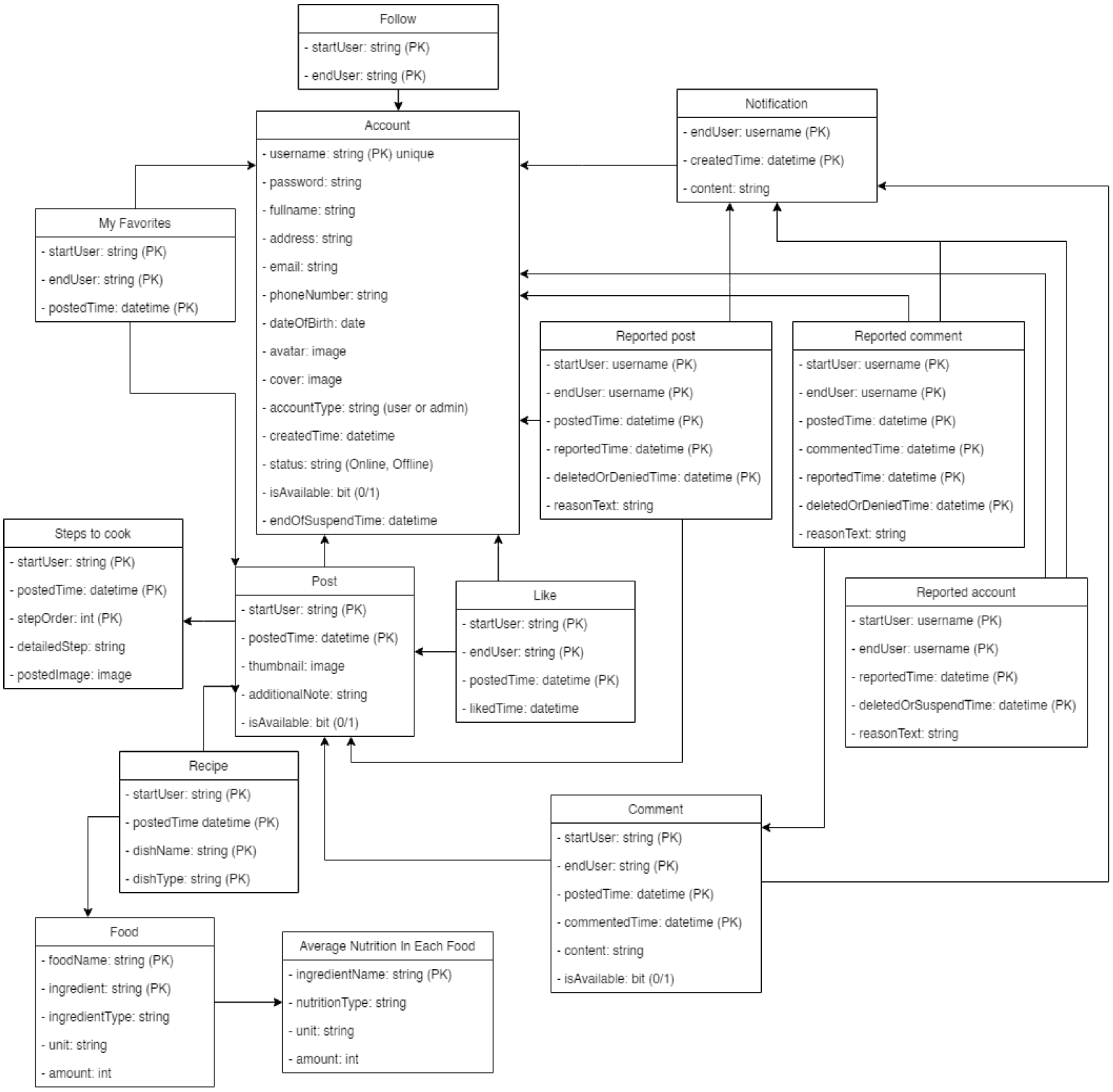
4.11 Component: Service Integration

The Service Integration is constructed to communicate between client and server side. From the client, the Integration will receive the commands from the Controllers, send them to server (by using GET and POST methods) to request the server to access the database. Then, the accessed data will be sent back to the client side to display on the GUIs.


UserManagementServices
- _username: string
+ accessDB_updateUsername()
+ accessDB_selectAccount()
+ accessDB_updatePassword()
+ accessDB_updateAvatar()
+ accessDB_updateCover()
+ accessDB_updateFullname()
+ accessDB_updateAddress()
+ accessDB_updatePhoneNumber()
+ accessDB_updateDOB()
ServicesIntegration
+ receiveRequestFromController()
+ sendResponseToController()
AuthenticationServices
+ accessDB_selectAccountByUsername(string)
+ accessDB_insertAcount(Account)
UserPostServices
+ accessDB_insertPost(Post)
+ accessDB_findPost(int, string)
+ accessDB_updateLikeOfPost()
+ accessDB_insertCommentOfPost(string)
+ accessDB_deletePost(int, string)
Database
+ Account: table
+ Post: table
+ Like: table
+ Comment: table
SearchingServices
+ accessDB_selectRecipesAfterFilter()
+ accessDB_findRecipesByFoodName(string)
+ accessDB_findRecipesByIngredient(string)
+ accessDB_selectRecipesAfterSort()
AdminActionServices
+ accessDB_selectAllAccount()
+ accessDB_selectAllReportedAccounts()
+ accessDB_selectAllReportedComments()
+ accessDB_selectAllReportedPosts()
+ accessDB_deleteCommentOfPost(int, string)
+ accessDB_suspendAccount(int, string)

4.12 Component: Database

The database of our website is sketched in the class diagram below. It shows how each table is constructed and how they connect to each other.



5. Deployment

None

6. Implementation View

None