  
  
  
CS 353

Database Systems  
Project Final Report  
  
Project Name: Daily Food

Group 1

Pınar Bayata – 21401687

Özgür Can Erdoğan – 21300586

Kübra Nur Güzel – 21400946

Arif Can Terzioğlu- 21302061

Table of Contents

[1. Overview 3](#_Toc501712129)

[2.Revised E/R Diagram 4](#_Toc501712130)

[3. Relation Schemas 6](#_Toc501712131)

[3.1 Account 6](#_Toc501712132)

[3.2 Messages 6](#_Toc501712133)

[3.3 Member 6](#_Toc501712134)

[3.4 GoldMember 7](#_Toc501712135)

[3.5 Admin 7](#_Toc501712136)

[3.6 Category 8](#_Toc501712137)

[3.7 SubCategory 8](#_Toc501712138)

[3.8 Ingredient 8](#_Toc501712139)

[3.9 Comment 9](#_Toc501712140)

[3.10 Recipe 9](#_Toc501712141)

[3.11 Share 10](#_Toc501712142)

[3.12 Create 10](#_Toc501712143)

[3.13 Has 10](#_Toc501712144)

[3.14 Rate 11](#_Toc501712145)

[3.15 Follow 11](#_Toc501712146)

[3.16 Favorite 12](#_Toc501712147)

[4. Implementation Details 12](#_Toc501712148)

[5. Advanced Database Features and Reports 15](#_Toc501712149)

[ Showing food per category 15](#_Toc501712150)

[ Gender distribution 15](#_Toc501712151)

[ Montly added recipe 16](#_Toc501712152)

[ Montly word 16](#_Toc501712153)

[ Subcategories 17](#_Toc501712154)

[ Total Recipes Added Last Month 17](#_Toc501712155)

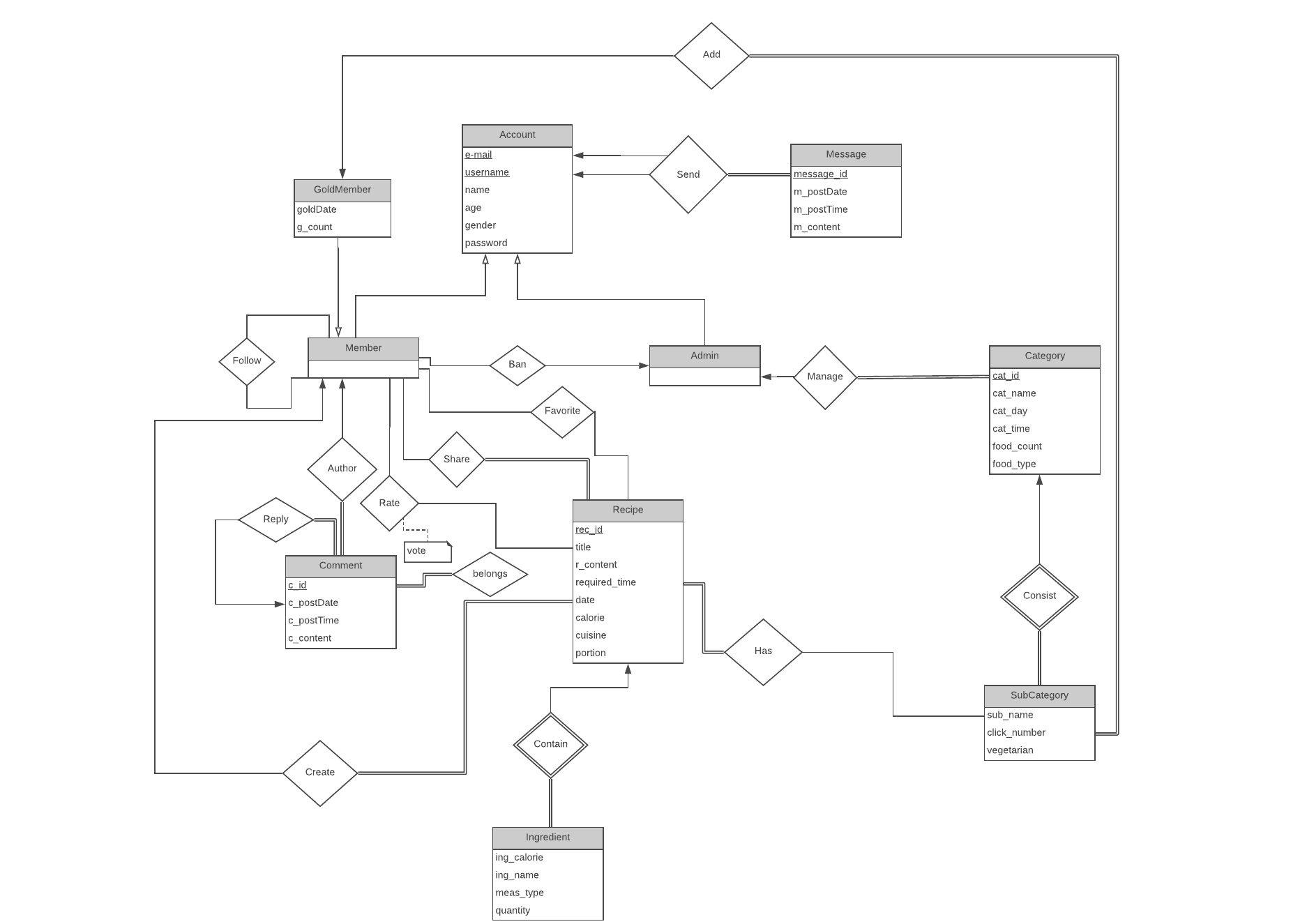
[6. User’s Manual 18](#_Toc501712156)

# Overview

As a Hypertext Dictionary we decided to make a food recipe sharing website called “Daily Food”. In our website, users can be members, share and read recipes which are separated into categories, search for recipes, follow other users, rate and comment on recipes. Users can search for recipes by their categories using the search bar.

# 2.Revised E/R Diagram

* We removed the relation Edit between Member and Recipe.
* We removed sub\_id from SubCategory entity.
* We make the relation ‘Reply’ between comments a total participation and one-to-many.
* We make the relation ‘Create’ which is between Recipe and Member, a one-to-many



# 3. Relation Schemas

## 3.1 Account

**Relational Model:**

Account(username, email, name, age, gender, password)

**Functional Dependencies:**

username -> email ,name, age ,gender, password

**Candidate Keys:**

{(username)}

**Foreign Keys:**

NONE

**Normal Form:**

BCNF

## 3.2 Messages

**Relational Model:**

Message(messageID, postDate, content, rec\_username, send\_username)

**Functional Dependencies:**

messageID -> postDate, content, rec\_username, send\_username

**Candidate Keys:**

{(messageID)}

**Foreign Keys:**

FOREIGN KEY (`rec\_username`) REFERENCES `Account` (`username`)

FOREIGN KEY (`send\_username`) REFERENCES `Account` (`username`)

**Normal Form:**

BCNF

## 3.3 Member

**Relational Model:**

Member(username,recipeCount)

**Functional Dependencies:**

username -> recipeCount

**Candidate Keys:**

{(username)}

**Foreign Keys:**

FOREIGN KEY (`username`) REFERENCES `Account` (`username`)

**Normal Form:**

BCNF

## 3.4 GoldMember

**Relational Model:**

GoldMember(username,goldDate)

**Functional Dependencies:**

username -> goldDate

**Candidate Keys:**

{(username)}

**Foreign Keys:**

FOREIGN KEY (`username`) REFERENCES `Member` (`username`)

**Normal Form:**

BCNF

## 3.5 Admin

**Relational Model:**

Admin(username)

**Functional Dependencies:**

NONE

**Candidate Keys:**

{(username)}

**Foreign Keys:**

FOREIGN KEY (`username`) REFERENCES `Account` (`username`)

**Normal Form:**

BCNF

## 3.6 Category

**Relational Model:**

Category(cat\_id,cat\_name, food\_count, adusername)

**Functional Dependencies:**

cat\_id -> cat\_name, food\_count, adusername

**Candidate Keys:**

{(cat\_id)}

**Foreign Keys:**

FOREIGN KEY (`adusername`) REFERENCES `Account` (`username`)

**Normal Form:**

BCNF

## 3.7 SubCategory

**Relational Model:**

SubCategory(catID,subName, vegetarian, musername)

**Functional Dependencies:**

cat\_id, subName -> vegetarian, musername

**Candidate Keys:**

{(cat\_id, subName)}

**Foreign Keys:**

FOREIGN KEY (`catID`) REFERENCES `Category` (`cat\_id`)

FOREIGN KEY (`musername`) REFERENCES `Member` (`username`)

**Normal Form:**

BCNF

## 3.8 Ingredient

**Relational Model:**

Ingredient(recID,ingName, ingCalorie, measType, quantity)

**Functional Dependencies:**

recID, ingName -> ingCalorie, measType, quantity

**Candidate Keys:**

{( recID, ingName)}

**Foreign Keys:**

FOREIGN KEY (`recID`) REFERENCES `Recipe` (`recipeID`)

**Normal Form:**

BCNF

## 3.9 Comment

**Relational Model:**

Comment(comID,comContent, reply\_username, musername, rec\_id)

**Functional Dependencies:**

comID -> comContent, reply\_username, musername, rec\_id

**Candidate Keys:**

{( comID)}

**Foreign Keys:**

FOREIGN KEY (`musername`) REFERENCES `Member` (`username`)

FOREIGN KEY (`rec\_id`) REFERENCES `Recipe` (`recipeID`)

**Normal Form:**

BCNF

## 3.10 Recipe

**Relational Model:**

Recipe(recipeID,title, content, requiredTime, date, calorie, cuisine, portion, username, subname)

**Functional Dependencies:**

recipeID -> title, content, requiredTime, date, calorie, cuisine, portion, username, subname

**Candidate Keys:**

{( recipeID)}

**Foreign Keys:**

NONE

**Normal Form:**

BCNF

## 3.11 Share

**Relational Model:**

Share(musername, recID)

**Functional Dependencies:**

NONE

**Candidate Keys:**

{( musername, recID)}

**Foreign Keys:**

FOREIGN KEY (`musername`) REFERENCES `Member` (`username`)

FOREIGN KEY (`recID`) REFERENCES `Recipe` (`recipeID`)

**Normal Form:**

BCNF

## 3.12 Create

**Relational Model:**

Create(username, recID)

**Functional Dependencies:**

NONE

**Candidate Keys:**

{( username, recID)}

**Foreign Keys:**

FOREIGN KEY (`recID`) REFERENCES `Recipe` (`recipeID`)

FOREIGN KEY (`username`) REFERENCES `Member` (`username`)

**Normal Form:**

BCNF

## 3.13 Has

**Relational Model:**

Has(rec-id, subname)

**Functional Dependencies:**

NONE

**Candidate Keys:**

{(rec-id, subname)}

**Foreign Keys:**

FOREIGN KEY (`rec-id`) REFERENCES `Recipe` (`recipeID`)

FOREIGN KEY (`subname`) REFERENCES `SubCategory` (`subName`)

**Normal Form:**

BCNF

## 3.14 Rate

**Relational Model:**

Rate(rec-id, vote, musername)

**Functional Dependencies:**

rec-id -> vote, musername

**Candidate Keys:**

{(rec-id)}

**Foreign Keys:**

FOREIGN KEY (`rec-id`) REFERENCES `Recipe` (`recipeID`)

FOREIGN KEY (`musername`) REFERENCES `Member` (`username`)

**Normal Form:**

BCNF

## 3.15 Follow

**Relational Model:**

Follow(username, fol\_username)

**Functional Dependencies:**

NONE

**Candidate Keys:**

{(username, fol\_username)}

**Foreign Keys:**

FOREIGN KEY (`username`) REFERENCES `Account` (`username`)

FOREIGN KEY (`fol\_username`) REFERENCES `SubCategory` (`username`)

**Normal Form:**

BCNF

## 3.16 Favorite

**Relational Model:**

Favorite(musername, recID)

**Functional Dependencies:**

NONE

**Candidate Keys:**

{(musername, recID)}

**Foreign Keys:**

FOREIGN KEY (`musername`) REFERENCES `Member` (`username`)

FOREIGN KEY (`recID`) REFERENCES `Recipe` (`recipeID`)

**Normal Form:**

BCNF

# 4. Implementation Details

**About the GUI:**

Most of the people from our group took GRA210 class named Web Design and our knowledge from that course helped us a lot while writing the html and css codes line by line. For some pages, we used a wrapper to stabilize the objects and for some we did not because it was needless. Since our priority was functionality and database, we tried our best to make a user-friendly interface which we think we made. And some for other pages we used templates to make design easier. We give a lot of time designing some pages because we could not use a compiler for html and css which lead to the waste of time due to the unknown errors. However, we worked well and fortunately, we made a user interface which we think will satisfy the expectations from the project.

**About the ASP.NET and MySQL:**

We downloaded Visual Studio from dreamspark Bilkent via our accounts. We used MySQL Workbench as our program. First, we entered our tables, attributes, keys to the database one by one and then we started coding first the GUI then the database. We connected our database in our MySqlDatabase like in the following:

The libraries we used are:

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

We first created a connection object as it can be seen below:

  private MySqlConnection connection;

Then we used Initialize() function to connect the database and this method is called from the constructor from the class MySqlDatabase.aspx.

  public MysqlDatabase()

        {

            i1 = new string[128];

            i2 = new string[128];

            recipeList = new string[3, 2];

            favList = new string[3, 2];

            genderDist = new string[2, 2];

            Initialize();

        }

        //Initialize values

        private void Initialize()

        {

            server = "[db353.cuhyc2prztfl.eu-west-1.rds.amazonaws.com](http://db353.cuhyc2prztfl.eu-west-1.rds.amazonaws.com/)";

            database = "db353";

            uid = "db353";

            password = "01480148";

            string connectionString;

            connectionString = "SERVER=" + server + ";" + "DATABASE=" +

            database + ";" + "UID=" + uid + ";" + "PASSWORD=" + password + ";";

            connection = new MySqlConnection(connectionString);

        }

We created SQL queries in button clicks like in the following and after defining the query, we connected to the database by the OpenConnection() method and after executing the query, we called the closeConnection() method:

 //open connection to database

        private bool OpenConnection()

        {

            try

            {

                connection.Open();

                return true;

            }

            catch (MySqlException ex)

            {

                //When handling errors, you can your application's response based

                //on the error number.

                //The two most common error numbers when connecting are as follows:

                //0: Cannot connect to server.

                //1045: Invalid user name and/or password.

                switch (ex.Number)

                {

                    case 0:

                        Console.WriteLine("Cannot connect to server.  Contact administrator");

                        break;

                    case 1045:

                        Console.WriteLine("Invalid username/password, please try again");

                        break;

                }

                return false;

            }

        }

//Close connection

        private bool CloseConnection()

        {

            try

            {

                connection.Close();

                return true;

            }

            catch (MySqlException ex)

            {

                Console.WriteLine(ex.Message);

                return false;

            }

        }

An example of a query:

 public void getResultKeyOnly(string key)

        {

            string query = "select \* from resultWord where title like '%" + key + "%';";

            OpenConnection();

            MySqlCommand cmd = new MySqlCommand(query, connection);

            MySqlDataReader dataReader = cmd.ExecuteReader();

            int i = 0;

            while (dataReader.Read())

            {

                i1[i] = dataReader["title"].ToString();

                i1[i + 1] = dataReader["date"].ToString();

                i1[i + 2] = dataReader["username"].ToString();

                i1[i + 3] = dataReader["cat\_name"].ToString();

                i1[i + 4] = dataReader["subname"].ToString();

                i = i + 5;

            }

            dataReader.Close();

            this.CloseConnection();

        }

Then we defined the behaviours of the buttons when clicked in aspx.cs classes. Detailed code is in the Code section. Because our E/R model is designed well, we did not confront serious problems. One problem that we had can be the use of GitHub tool in our Visual Studio. We wanted to integrate GitHub so that we can make changes and see the changes that have been made but other project members easily. But we couldn’t be able to do it and gave up after trying it for a few days.

# 5. Advanced Database Features and Reports

## Showing food per category

CREATE

ALGORITHM = UNDEFINED

DEFINER = `db353`@`%`

SQL SECURITY DEFINER

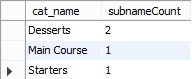
VIEW `db353`.`totalRecipesAddedLastMonth` AS

SELECT

SUM(`monthly\_added\_recipe`.`recipeCount`) AS `totalRecipe`

FROM

`db353`.`monthly\_added\_recipe`



## Gender distribution

CREATE

ALGORITHM = UNDEFINED

DEFINER = `db353`@`%`

SQL SECURITY DEFINER

VIEW `db353`.`genderDist` AS

SELECT

`db353`.`Account`.`gender` AS `gender`,

COUNT(DISTINCT `db353`.`Account`.`gender`) AS `genderCount`

FROM

`db353`.`Account`

GROUP BY `db353`.`Account`.`gender`



## Montly added recipe

CREATE

ALGORITHM = UNDEFINED

DEFINER = `db353`@`%`

SQL SECURITY DEFINER

VIEW `db353`.`monthly\_added\_recipe` AS

SELECT

`db353`.`Recipe`.`username` AS `username`,

COUNT(DISTINCT `db353`.`Recipe`.`recipeID`) AS `recipeCount`

FROM

`db353`.`Recipe`

WHERE

((TO\_DAYS(CURDATE()) - TO\_DAYS(`db353`.`Recipe`.`date`)) <= 30)

GROUP BY `db353`.`Recipe`.`username`

ORDER BY `recipeCount` DESC



## Montly Recipe List

CREATE

ALGORITHM = UNDEFINED

DEFINER = `db353`@`%`

SQL SECURITY DEFINER

VIEW `db353`.`resultWord` AS

SELECT

`db353`.`Category`.`cat\_name` AS `cat\_name`,

`db353`.`Recipe`.`title` AS `title`,

`db353`.`Recipe`.`date` AS `date`,

`db353`.`Recipe`.`username` AS `username`,

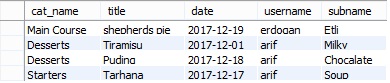
`db353`.`Recipe`.`subname` AS `subname`

FROM

((`db353`.`Recipe`

JOIN `db353`.`SubCategory` ON ((`db353`.`Recipe`.`subname` = `db353`.`SubCategory`.`subName`)))

JOIN `db353`.`Category` ON ((`db353`.`SubCategory`.`catID` = `db353`.`Category`.`cat\_id`)))



## Subcategories

CREATE

ALGORITHM = UNDEFINED

DEFINER = `db353`@`%`

SQL SECURITY DEFINER

VIEW `db353`.`subCategories` AS

SELECT

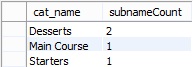
`resultWord`.`cat\_name` AS `cat\_name`,

COUNT(`resultWord`.`subname`) AS `subnameCount`

FROM

`db353`.`resultWord`

GROUP BY `resultWord`.`cat\_name`



## Total Recipes Added Last Month

CREATE

ALGORITHM = UNDEFINED

DEFINER = `db353`@`%`

SQL SECURITY DEFINER

VIEW `db353`.`totalRecipesAddedLastMonth` AS

SELECT

SUM(`monthly\_added\_recipe`.`recipeCount`) AS `totalRecipe`

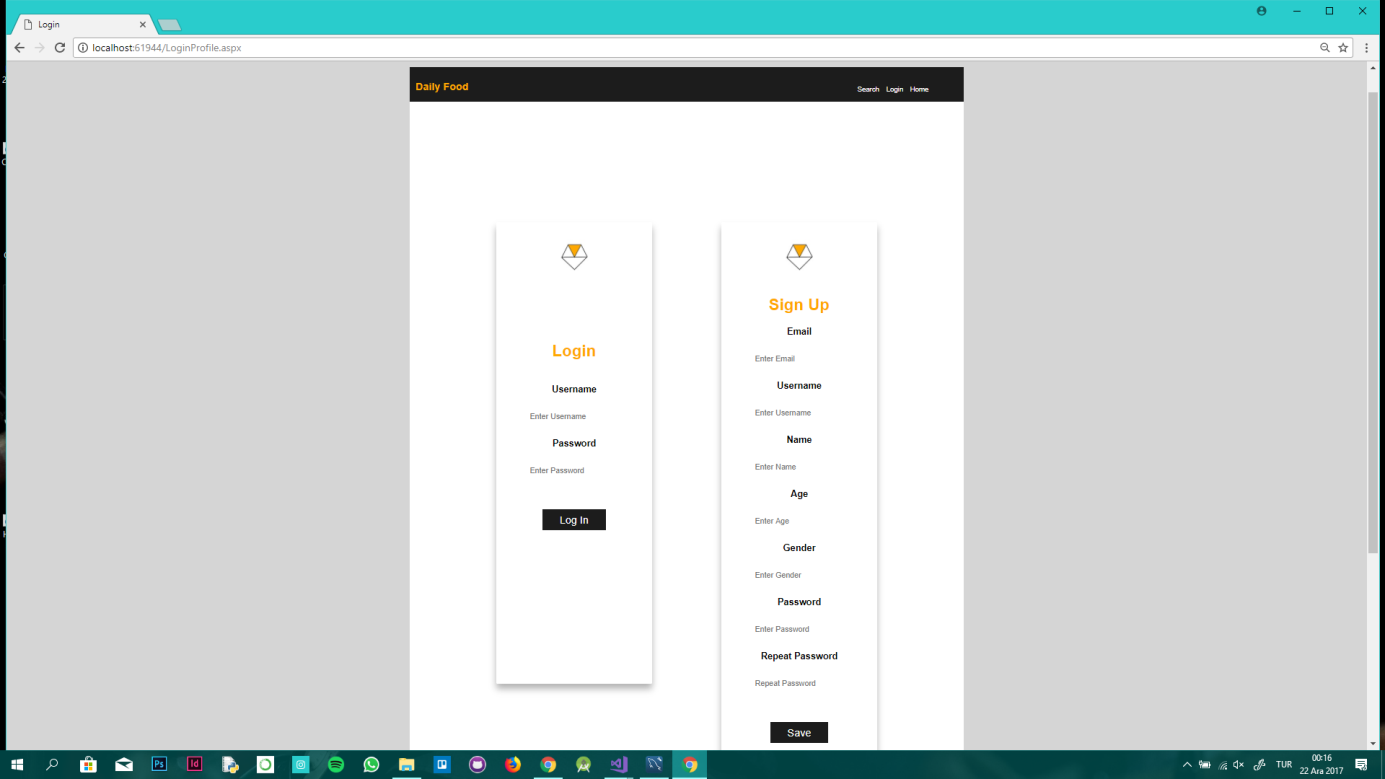
FROM

`db353`.`monthly\_added\_recipe`



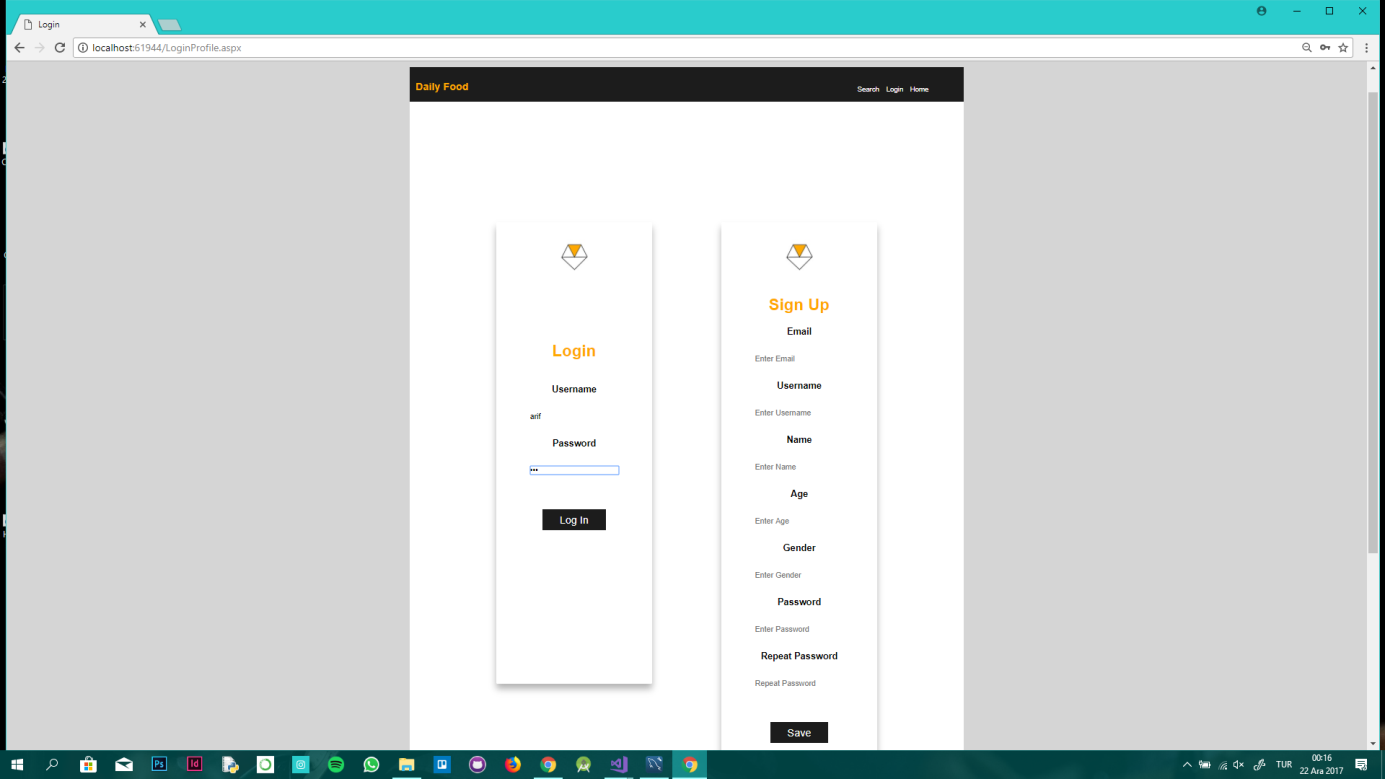
# 6. User’s Manual

1. Login & Signup Page

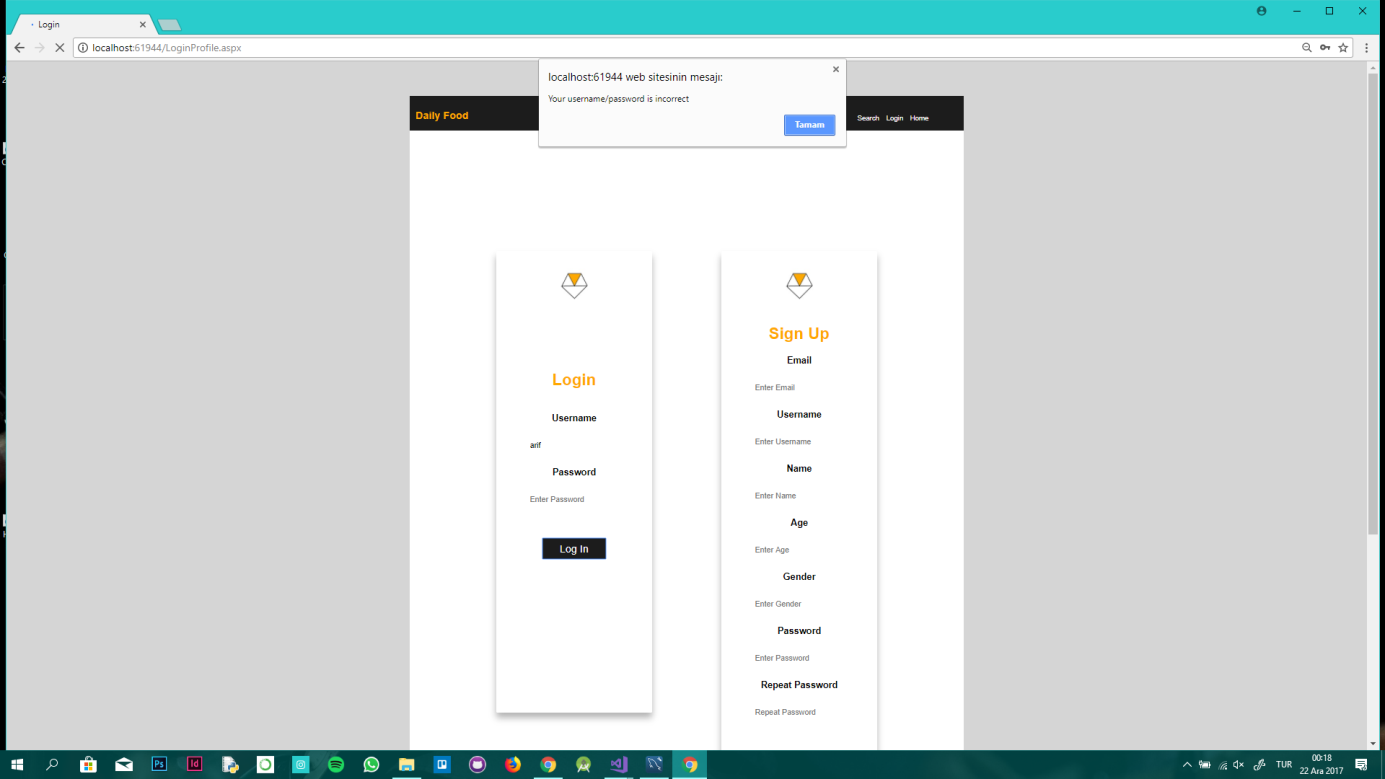


1. Sample Login & Signup Page

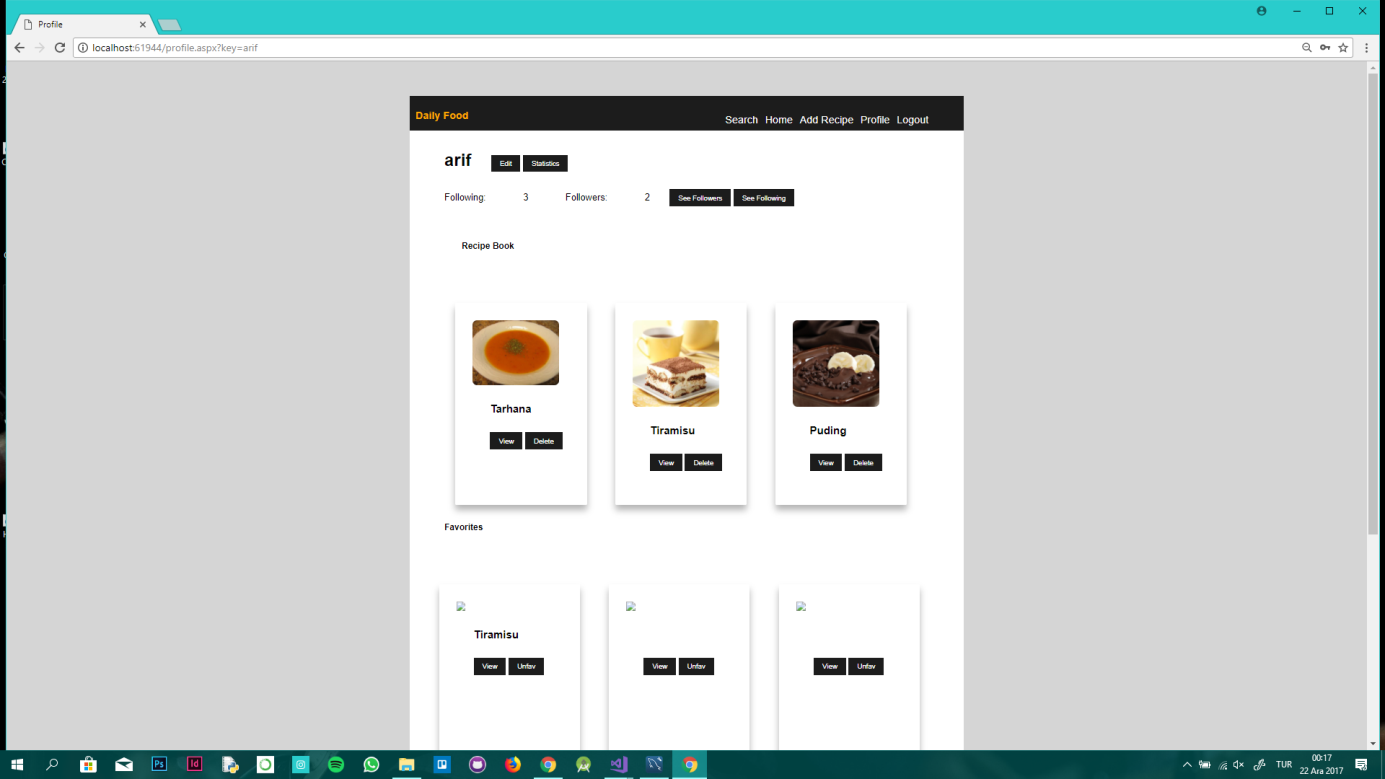
Username = arif Password= mrb



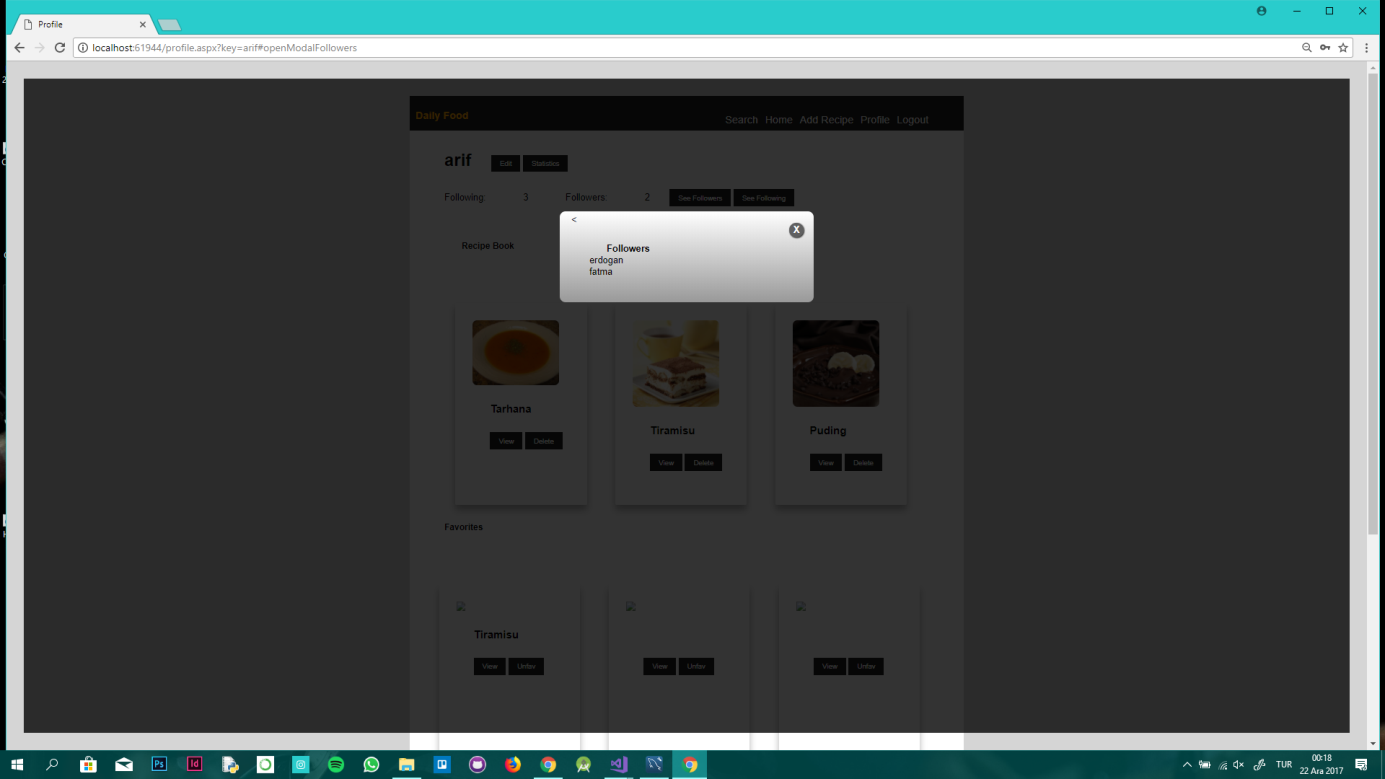
1. If the username or password is wrong, system gives an error message.



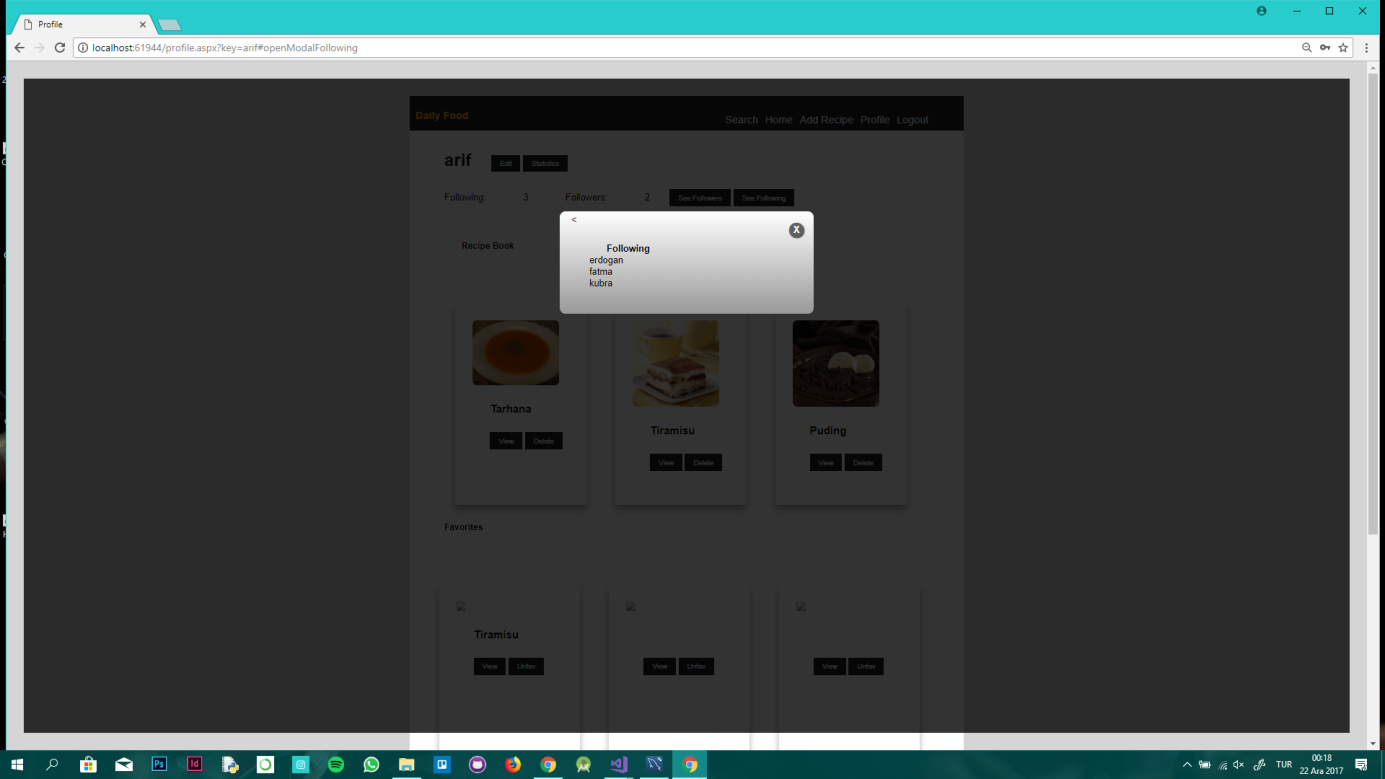
1. User’s Profile



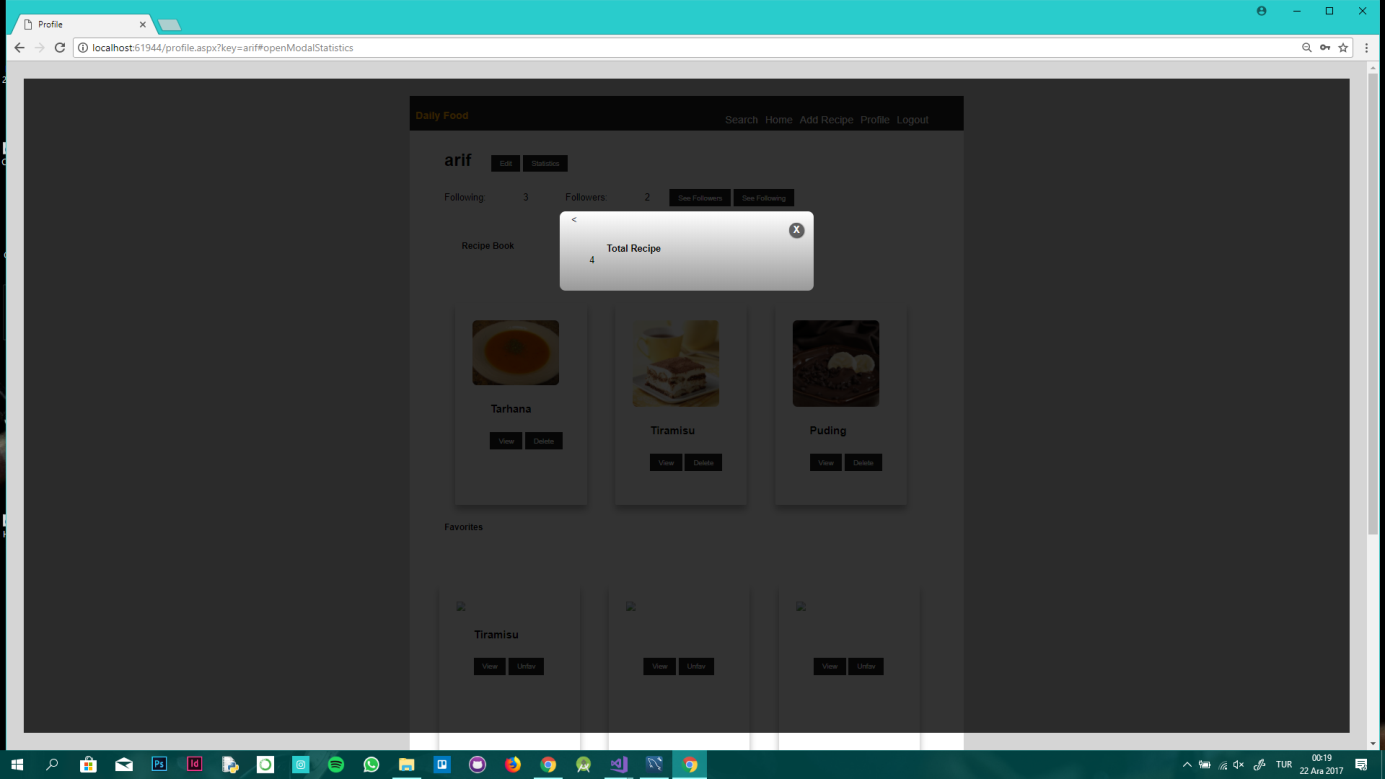
1. Users can see their Followers by clicking on the ‘See Followers” button. As it can be seen below.



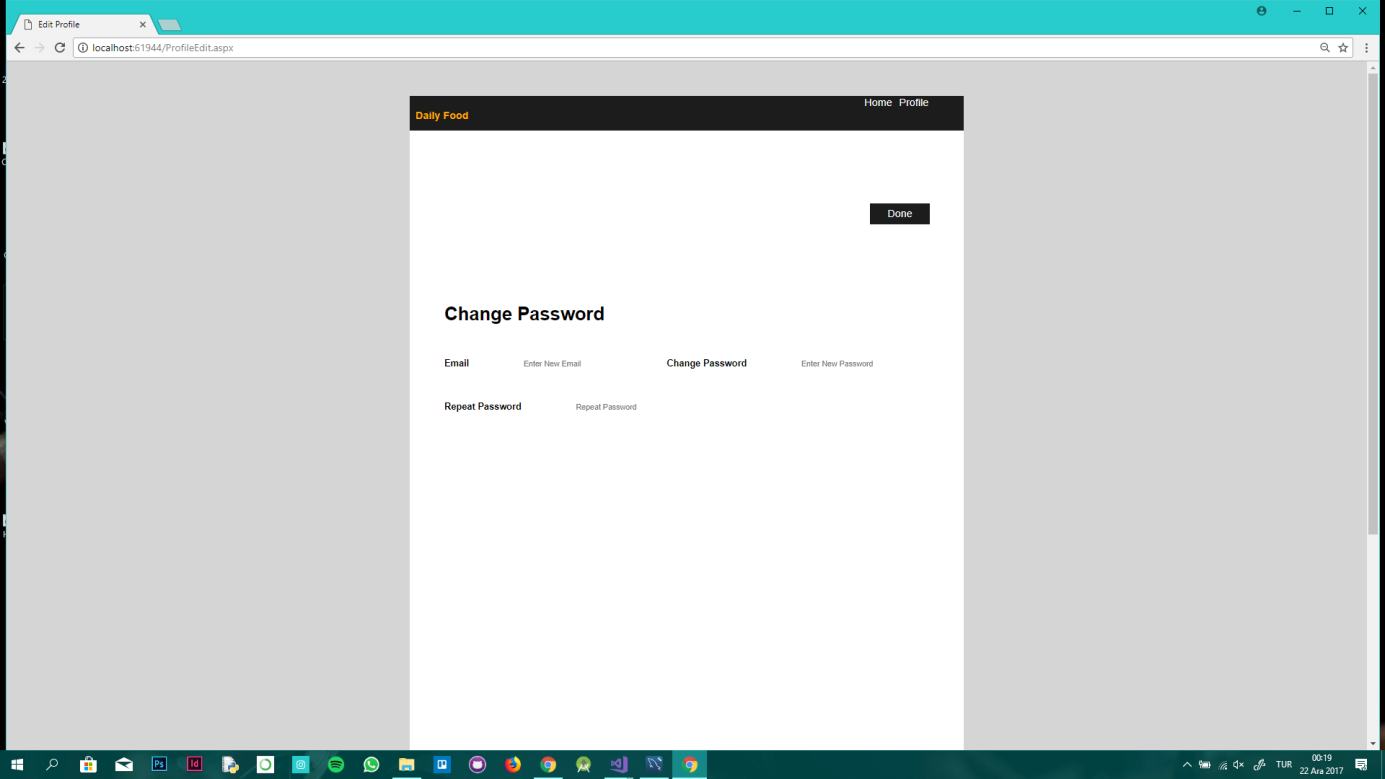
1. Users can see also the people whom a user follows by clicking on “See Following” button.



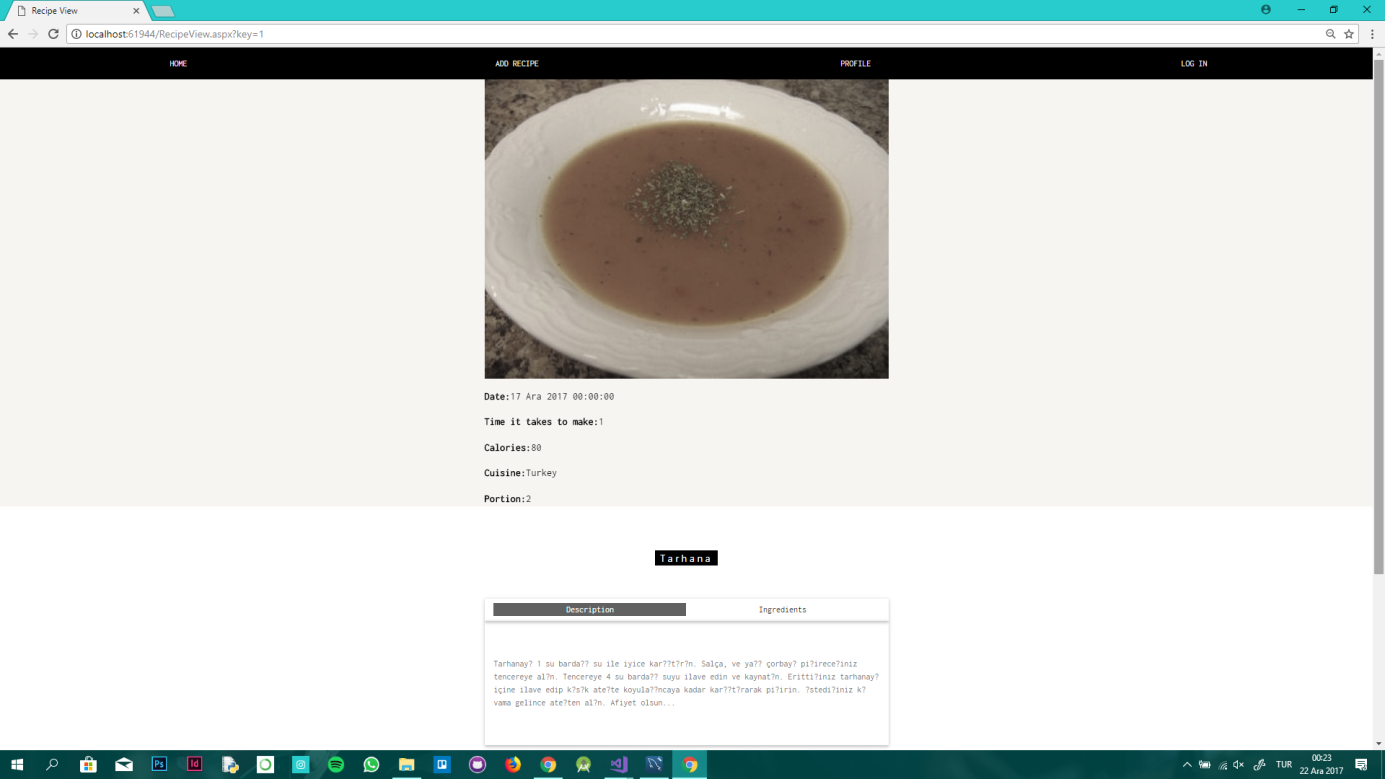
1. Also, users can see their total number of recipes by clicking on the ‘Statistics’ button near username.



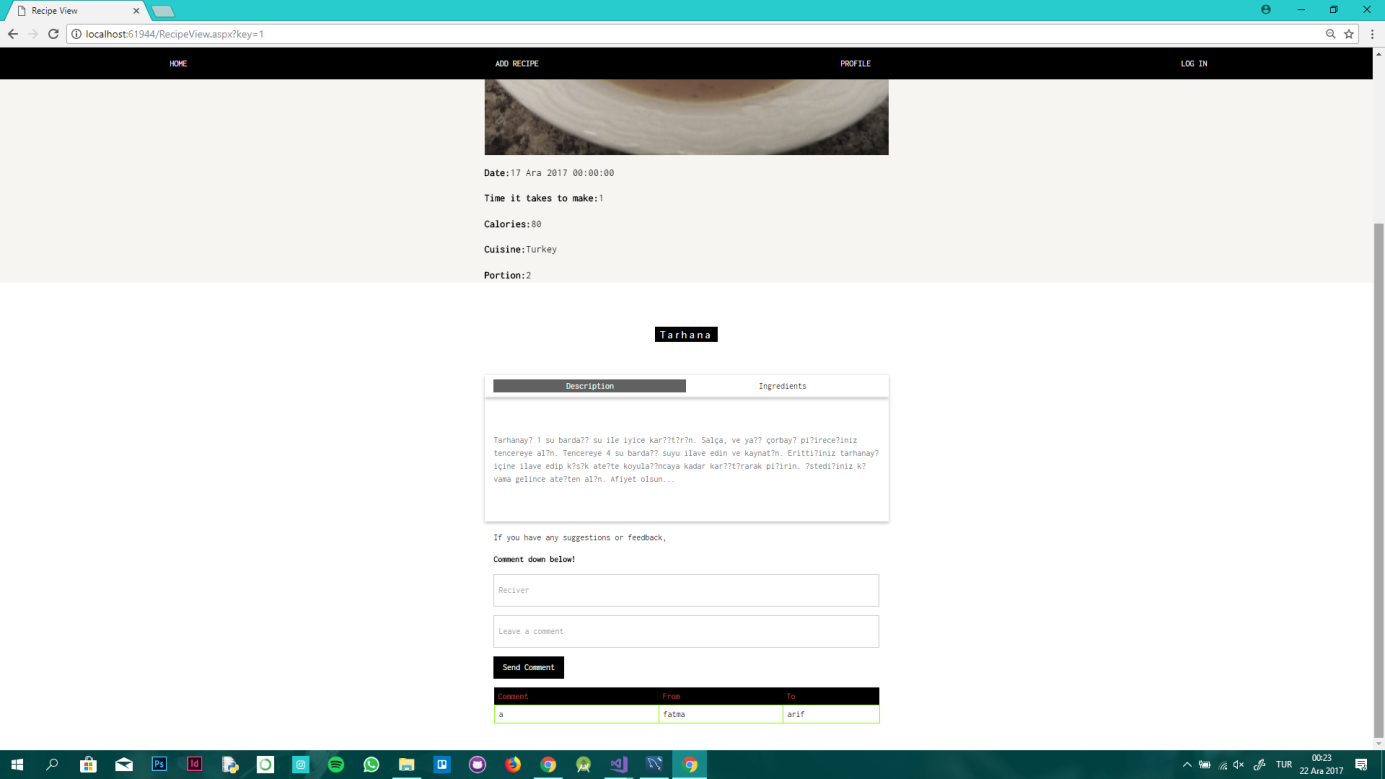
1. Users can change their password by clicking on the ‘Edit’ button near username in the Profile page. The image below shows the screen after clicking on the ‘Edit’ button. Changes will be saved after clicking on the ‘Done’ button.



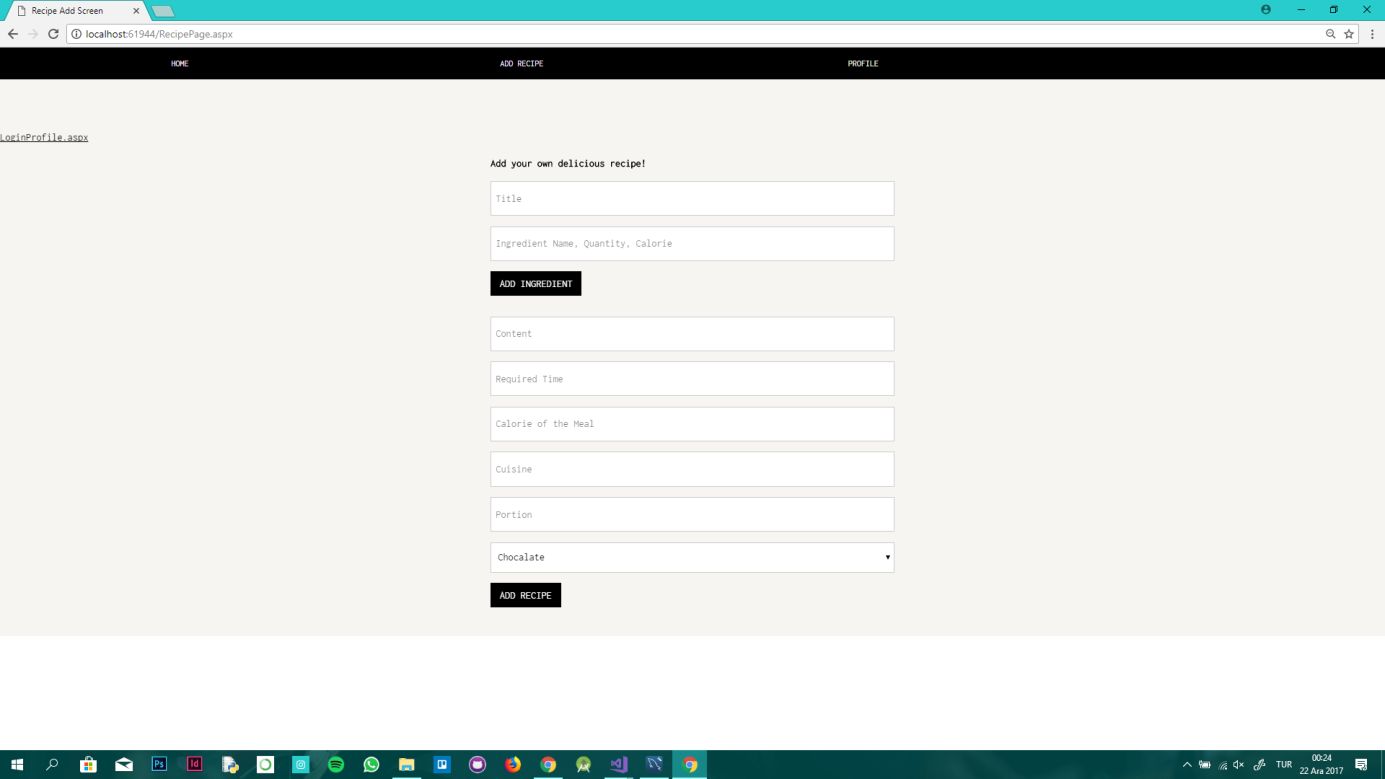
1. In the user’s profile page, users can view their recipes by clicking on the ‘View’ button on which recipe they want to view.



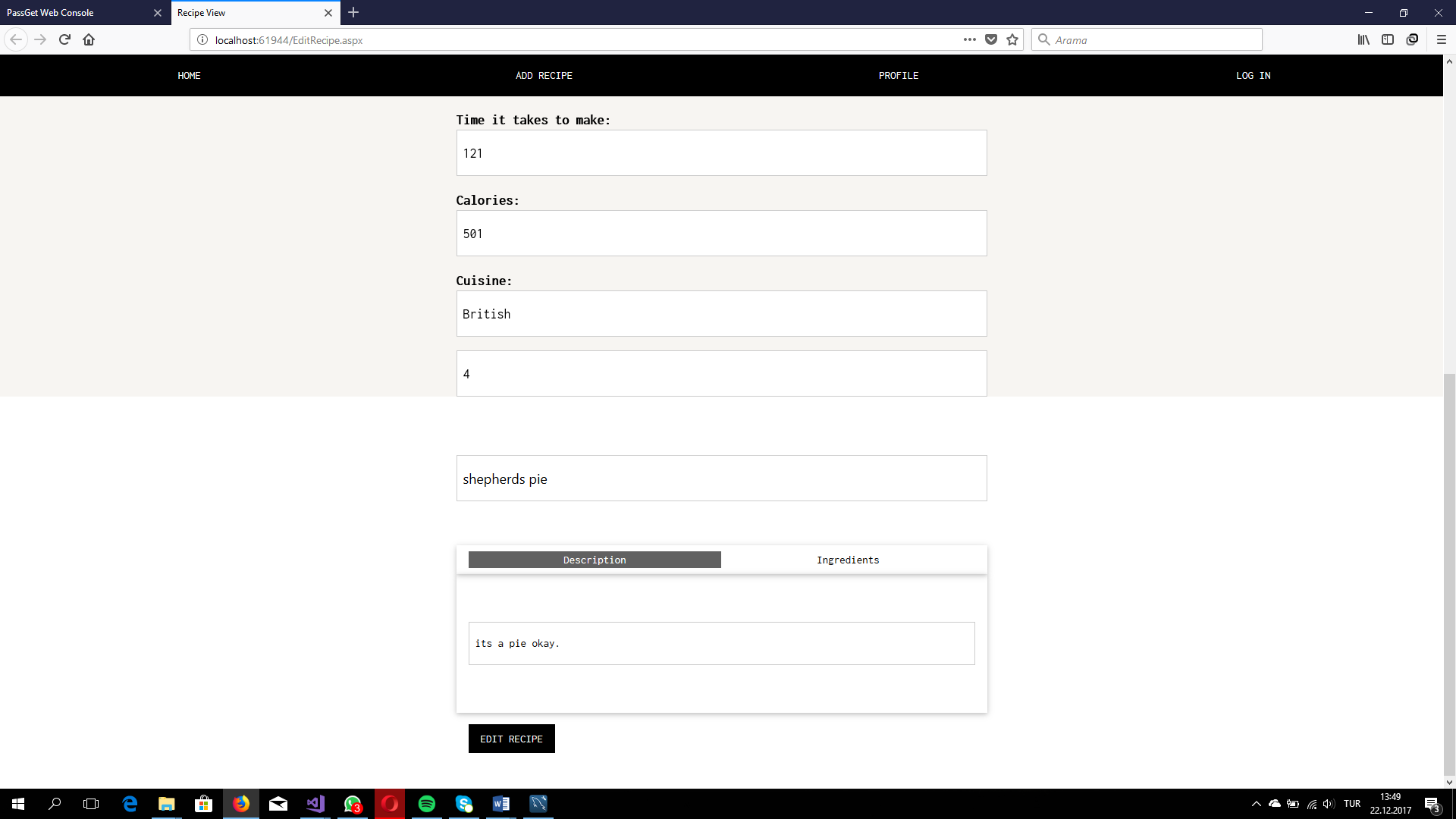
1. Comment part in the Recipe View page is as in the following.



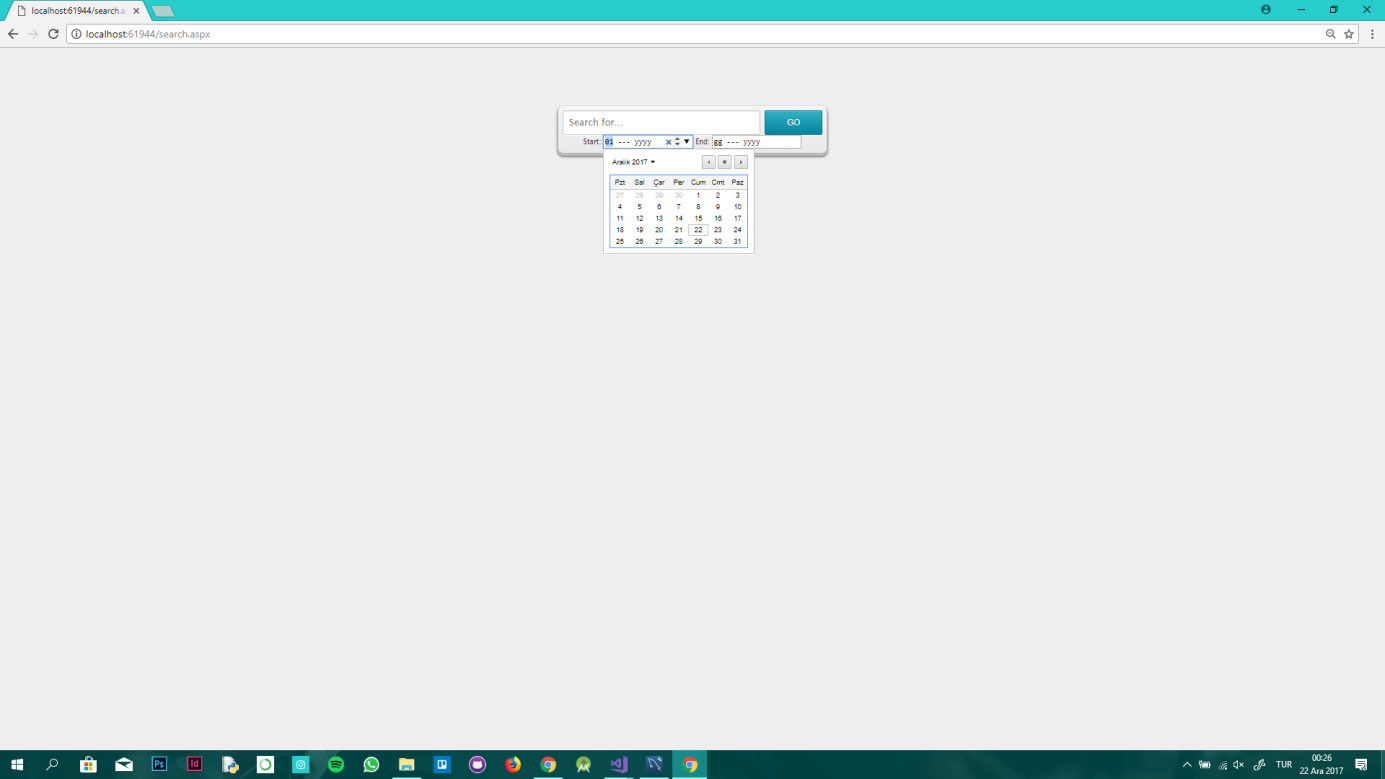
1. Users can add recipe by clicking on the ‘Add Recipe’ button on the menu bar in their profile page. After clicking on that button, Add Recipe screen shows up as below. Users must enter the title of the recipe and then must enter all the properties related with that recipe.



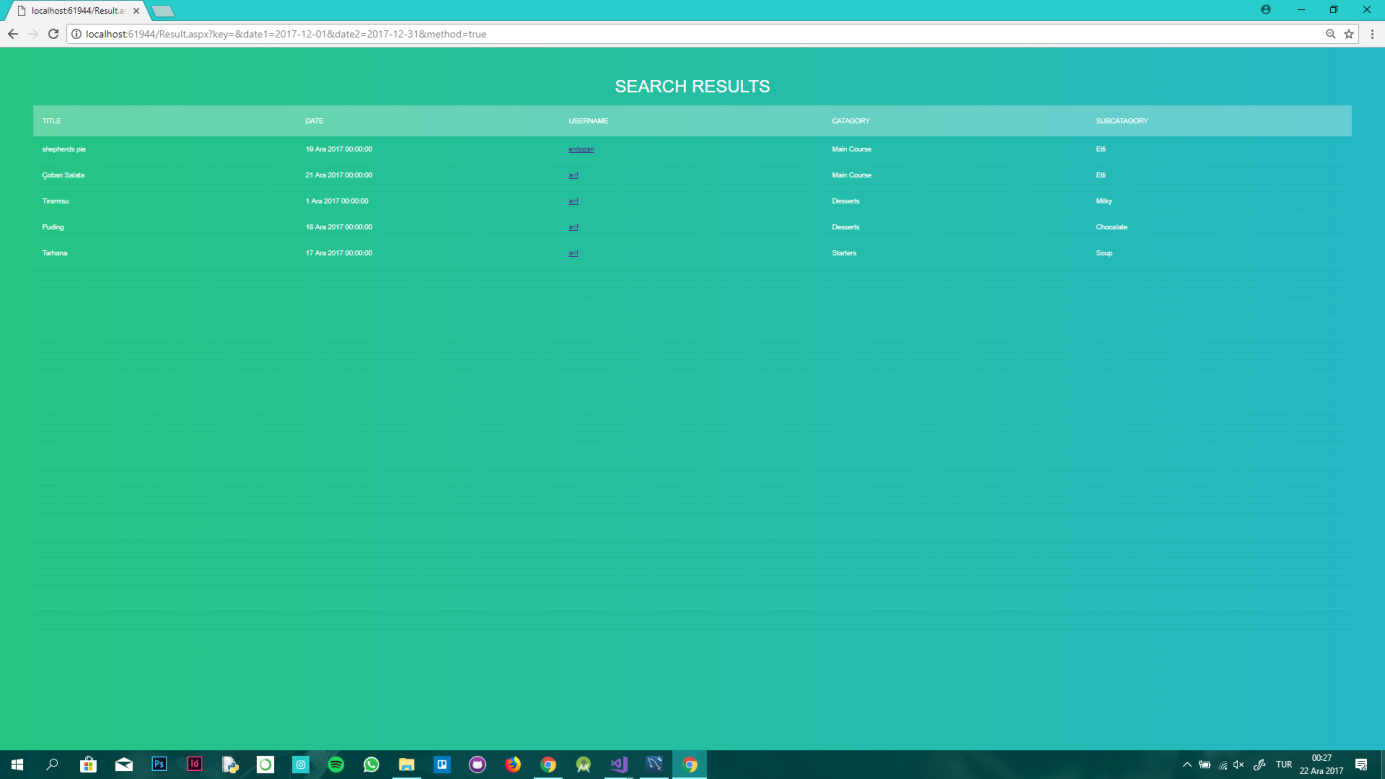
1. User can also edit their recipes after adding them.



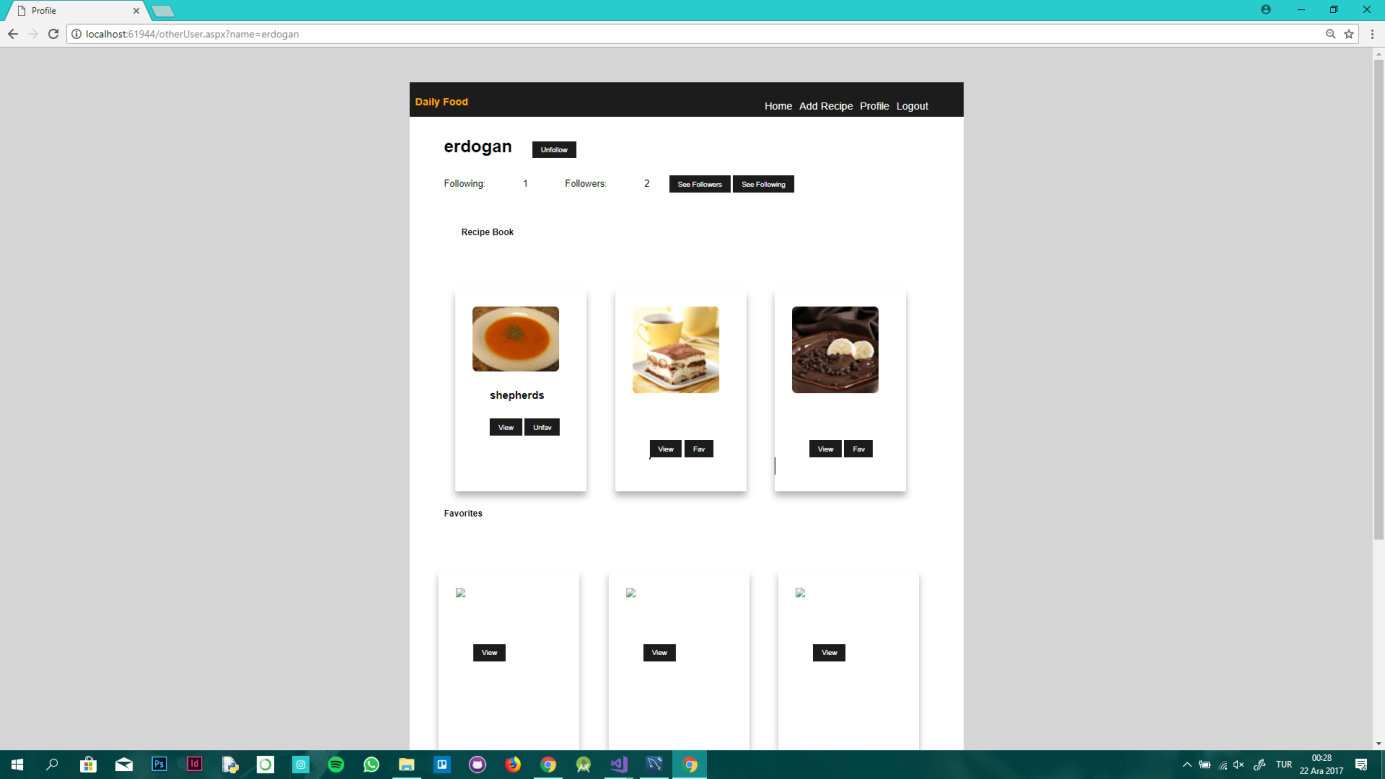
1. Users can also search for recipes by clicking on the’ Search’ button on the menu bar which is on top of the page. After clicking on that button, a page like in the following appears.



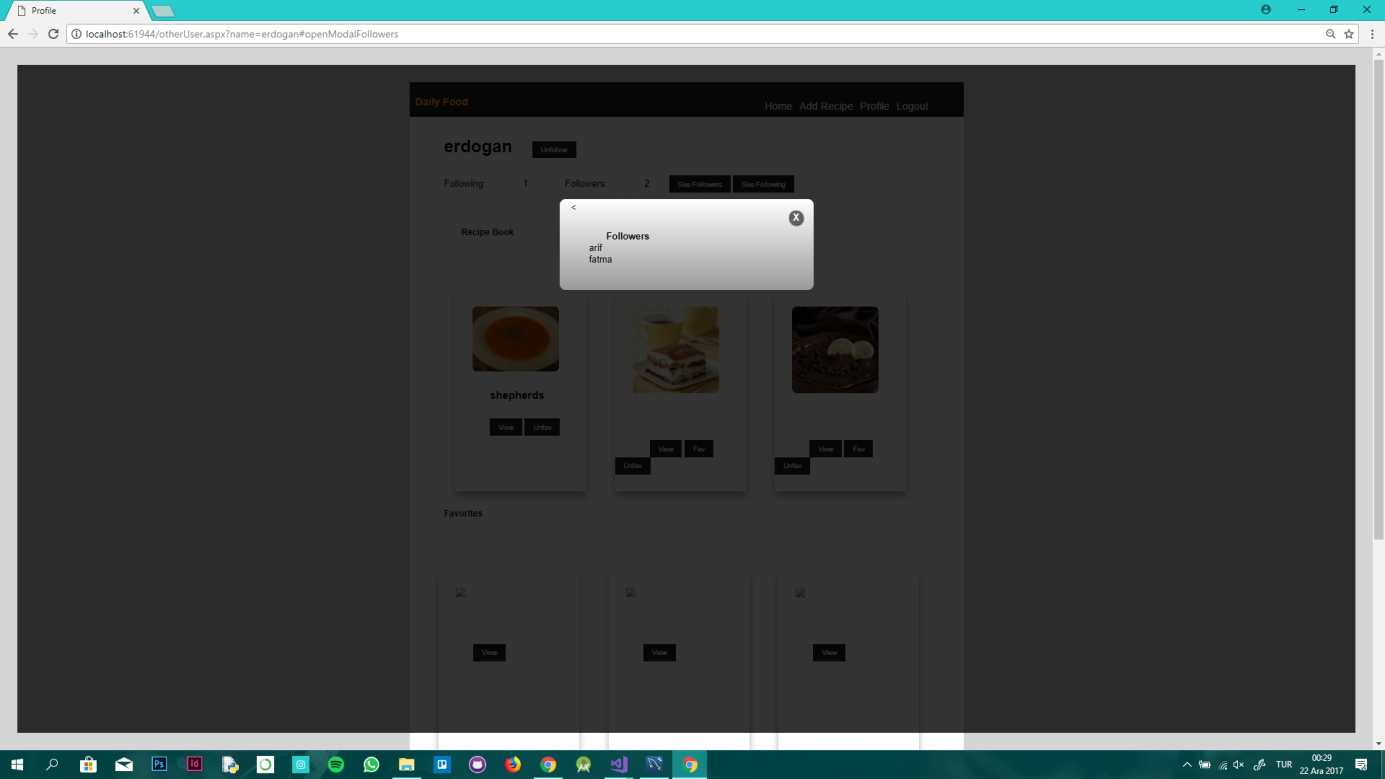
1. After entering the recipe name, if a recipe like that appears on the database, the system will show the records as in the below.

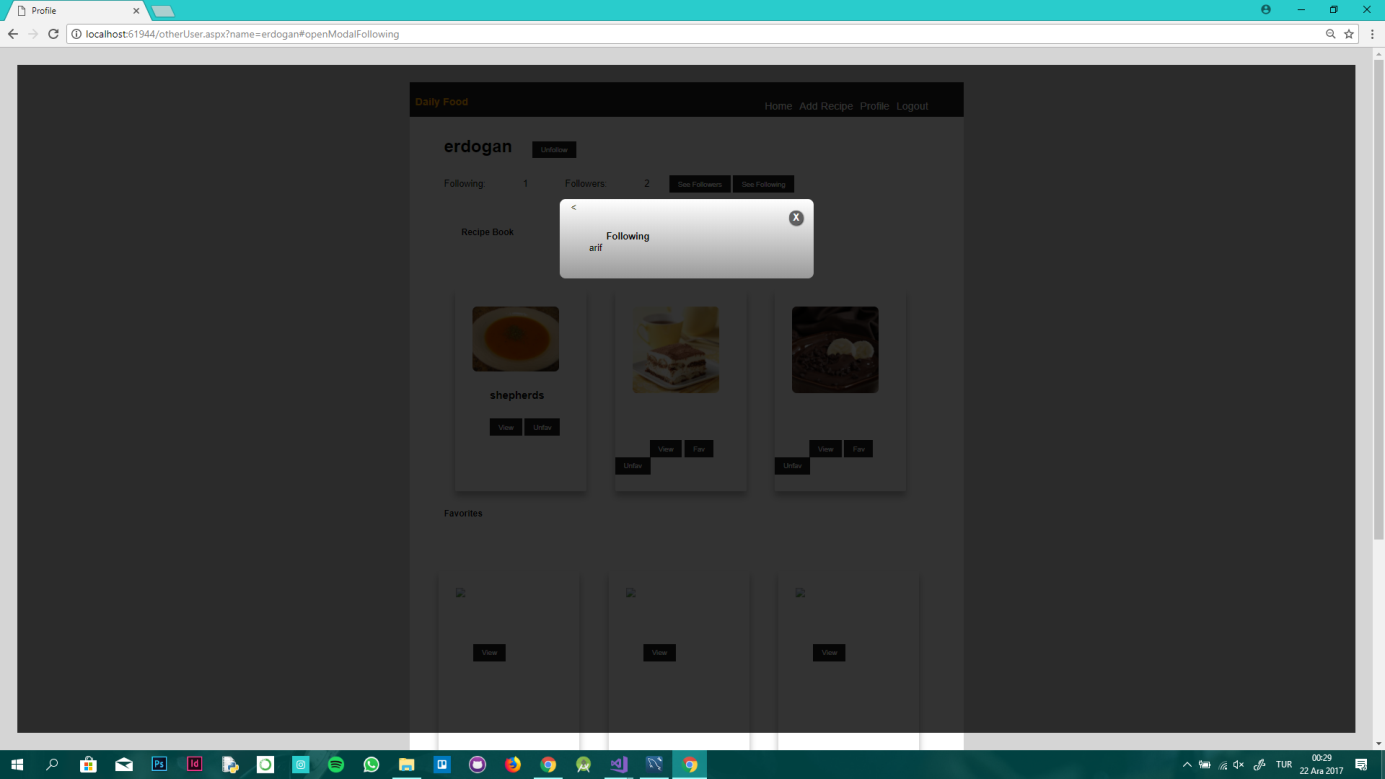


1. Users can also view other member’s profiles by clicking on an username in the username section.

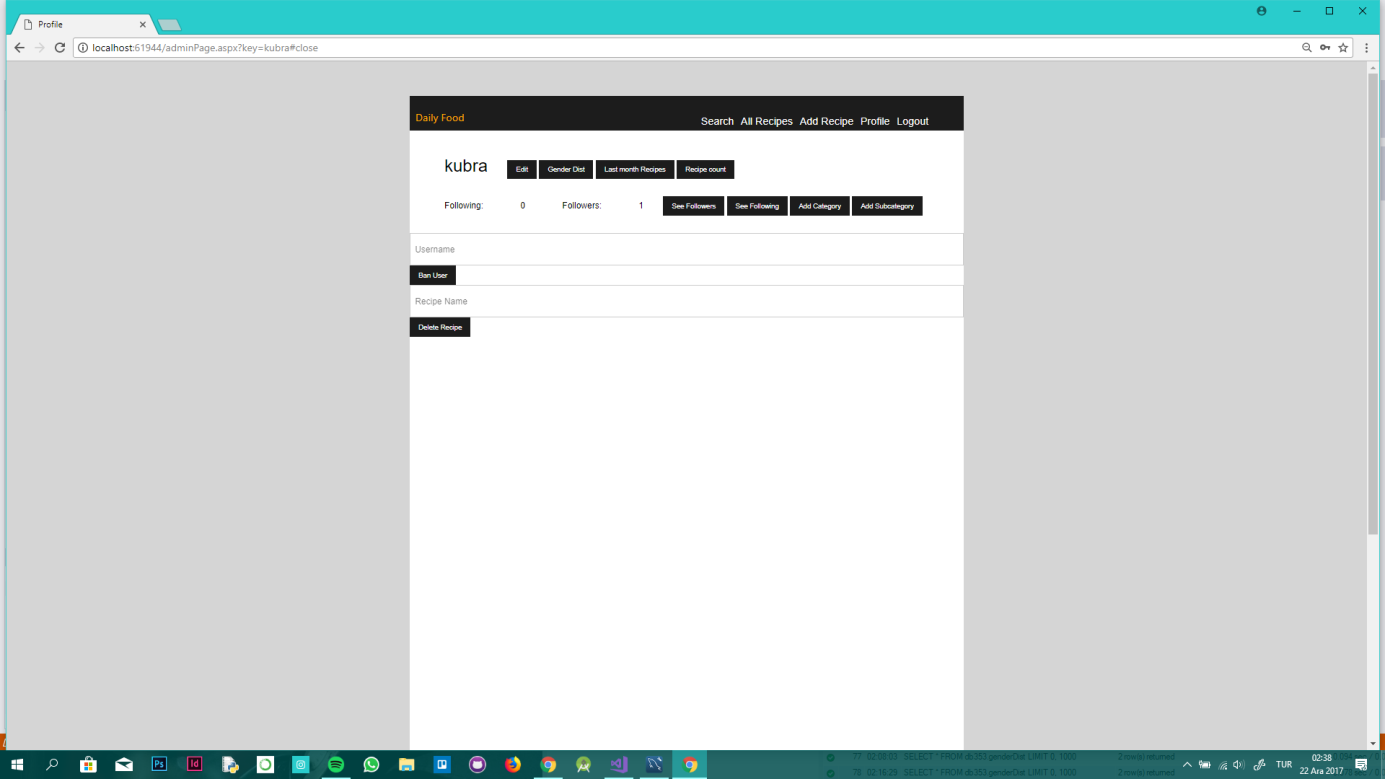


1. As users can view their followers and followings, they can see also other member’s following and followers.

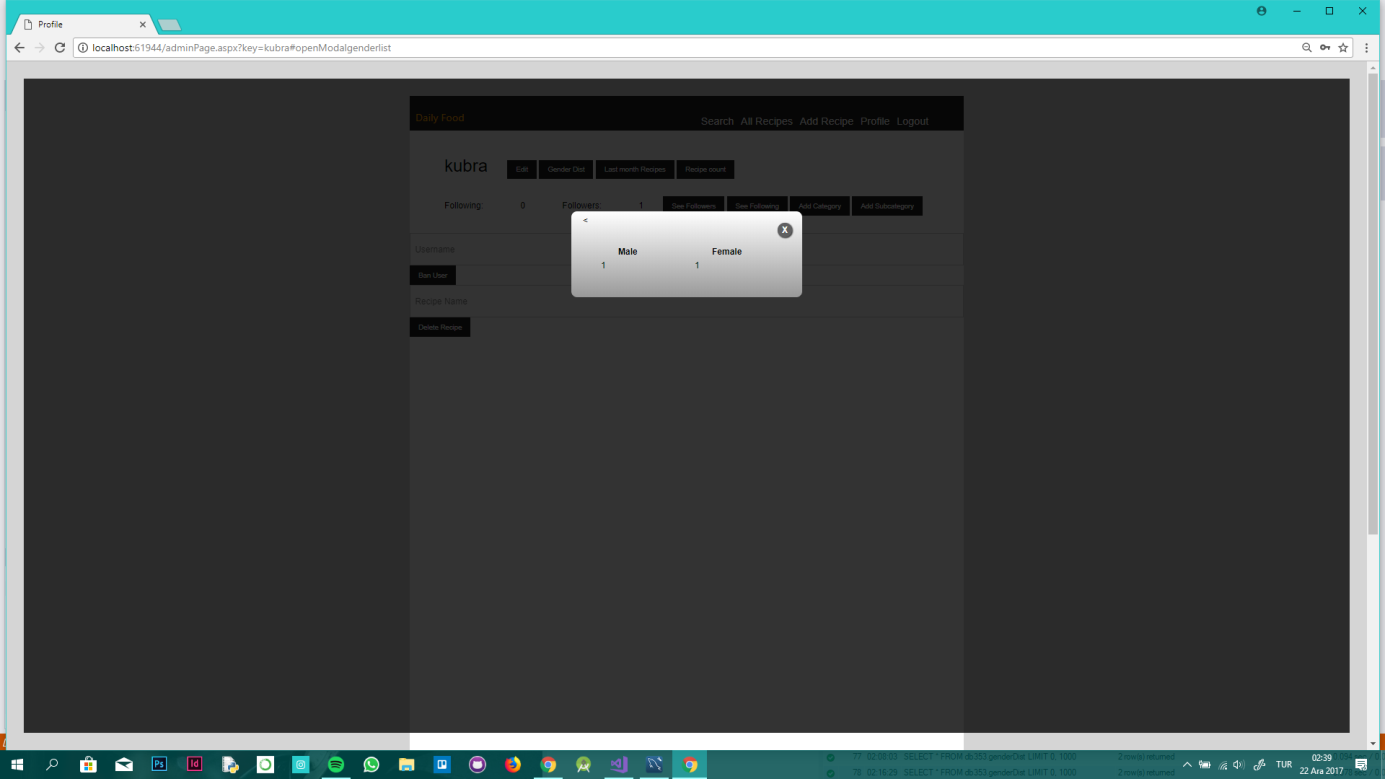




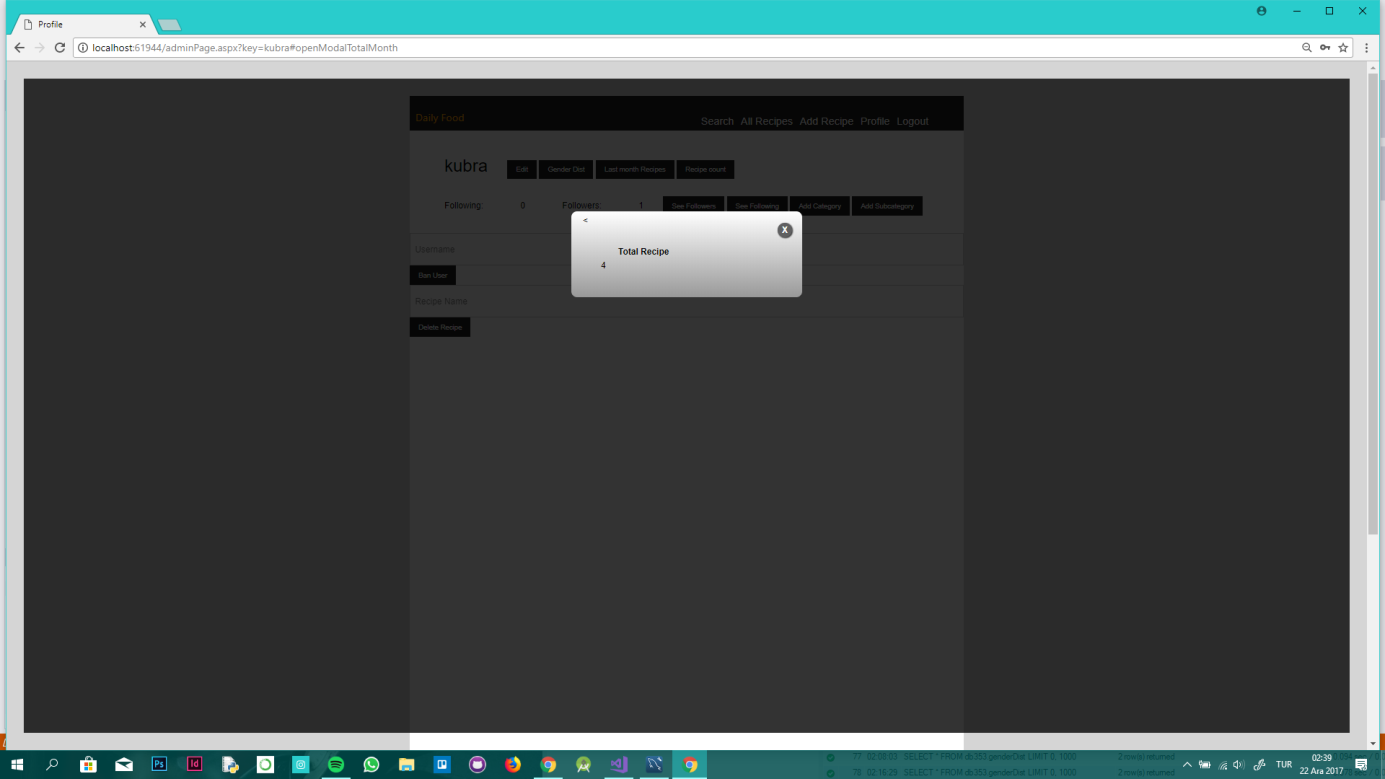
1. An admin’s profile page is different from the user’s since an admin can ban a user by typing the username, can add a category and subcategoryand so forth.



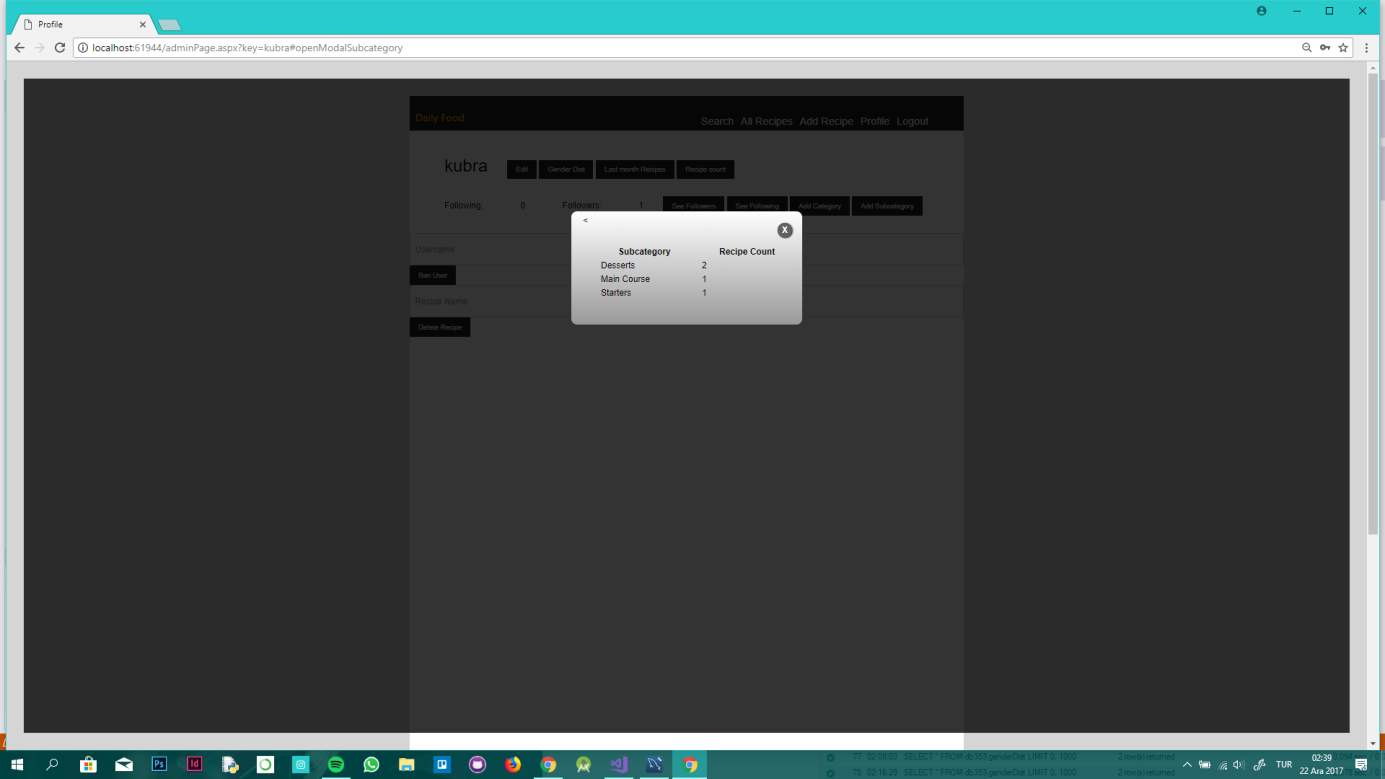
1. Apart from the users, an admin can view gender distribution of the users by clicking on Gender Dist button.



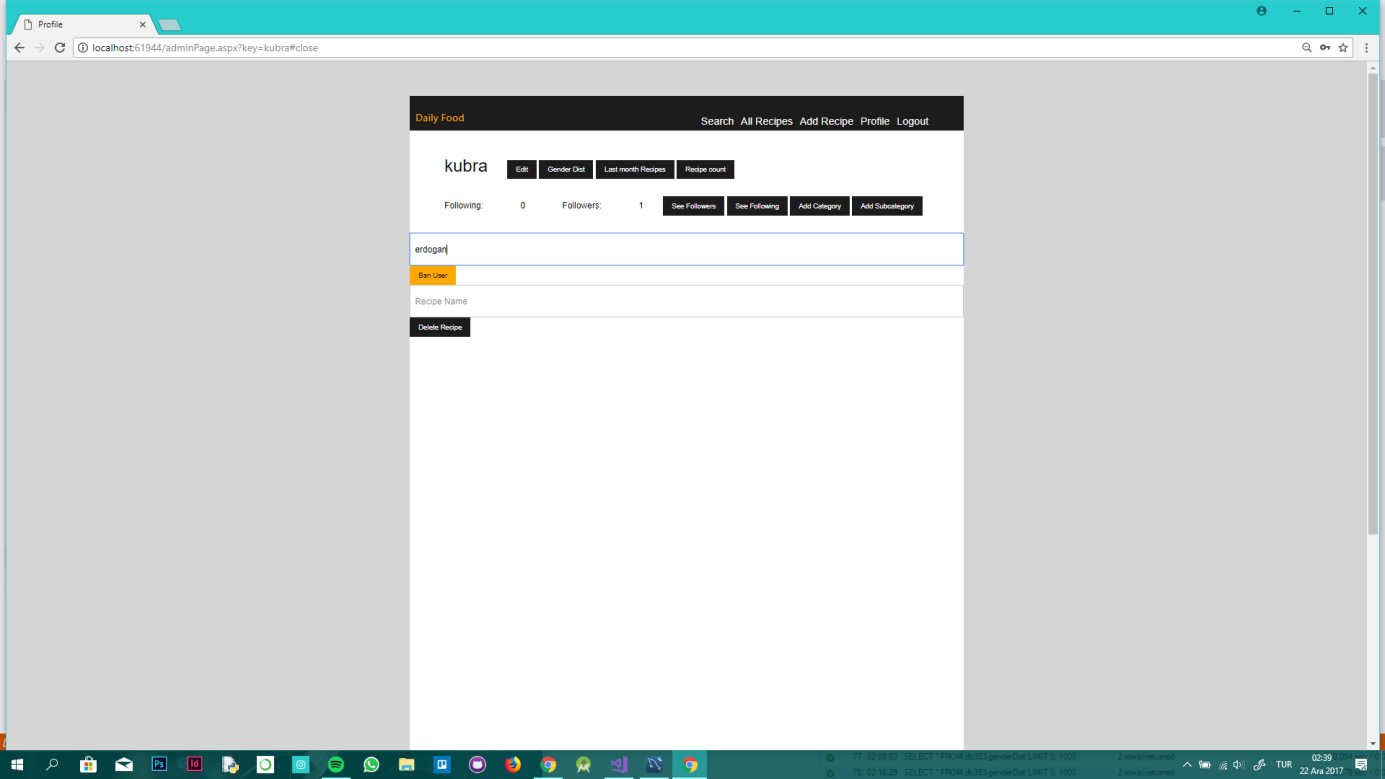
1. An admin can view total recipes of the page by clicking on the ‘Recipe Count’ button.



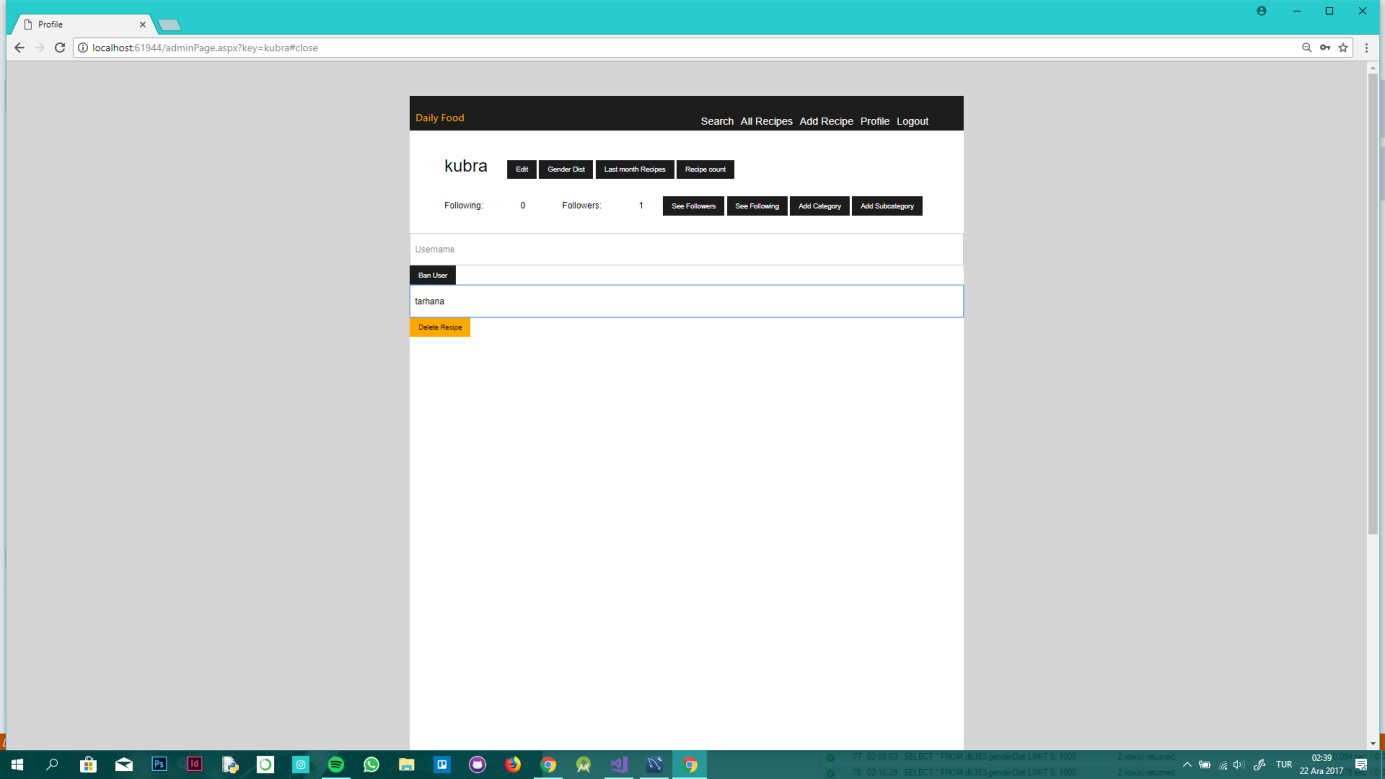
1. An admin can see the recipes from last month by clicking on the ‘Last Month Recipe’ button.



1. As mentioned, an admin can ban a user by typing his/her username.

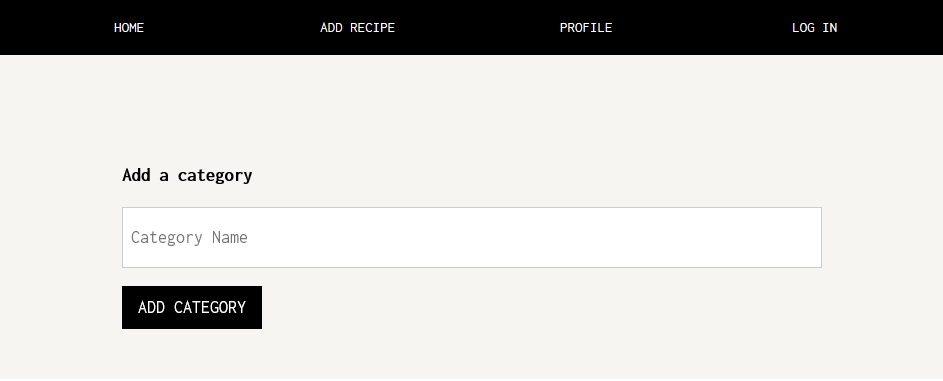


1. An admin can also delete a recipe by typing the name of the recipe.



1. An admin can add a category and a subcategory by using his/her profile.

The Add Category page:



The Add Subcategory Page:

