Tasty Table Lite

1. Introduction
2. Proposal/Pitch
3. Data Dictionary
4. Sequence diagrams/User Manual
5. Login
6. Register
7. Home
8. Search
9. Add Recipes
10. About Us
11. User Stories
12. Context Diagram and Diagram Zero Login
13. Context Diagram
14. Diagram Zero
15. Future Plan
16. Source Code

Introduction

TASTY TABLE VERSION 0.0.0.9 INTRODUCTION

Welcome to my recipe application! Our recipe application is built using C#, a powerful programming language that enables us to create a user-friendly interface and seamless functionality. With this app, you can explore a vast collection of recipes, ranging from simple and quick meals to more complex dishes that require a bit more time and effort.

Whether you're an experienced chef or a novice cook, this application is designed to provide you with all the information you need to create delicious meals in your own kitchen. The application provides step-by-step instructions for each recipe, as well as a list of necessary ingredients and equipment, so you can prepare your meals with confidence. Additionally, you can save your favorite recipes for future reference or share them with friends and family.

The application is secure and stores your data securely with our state of the art hashing algorithms, so you can rest easy knowing that you can browse safely. We believe that cooking should be fun and accessible to everyone, and our recipe application reflects that philosophy. So why not give it a try and discover a whole new world of culinary possibilities? We hope you enjoy using our app and creating delicious meals for yourself and your loved ones.

Proposal/Pitch

Graphical user interface, application

Description automatically generated

A picture containing text

Description automatically generated

Website, timeline

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

Chart, box and whisker chart

Description automatically generated

Timeline

Description automatically generated

Table

Description automatically generated

Data Dictionary

Recipe Table

1.RecID – Recipe ID autogenerated Primary Key

2.RecName – Recipe Name

3.TempNum – Temperature integer value

4.TempChar – Temperature discriminator, Celsius or Fahrenheit

Ingredient Table

1. IngId – Ingredient ID autogenerated Primary Key

2. IngName – Ingredient name

3. Quantity – Amount of Unit

4. Unit – A general unit or measurement, cups, ounces, etc

Instructions Table

1. InstID – Instruction ID Primary Key

2. StepNum – Number of steps

3. Description – Description of steps

4. RecID – RecipeID Foreign Key

RecipeIngr (Bridging Table)

1. RecID

2. IngID

UserLogin

1. ID - ID

2. Username - Username

3. FName – First Name

4. LName – Last Name

5. Hashpass – Password but hashed

6. Salt – Unique discriminator

These formulate the basics of the database structure used for the program. Within the program itself, we have a lot of data access elements that help us to search and query the database inside to be used in the Tasty Table program itself.

We have several Objects we create and populate within the program,

Text

Description automatically generated

This is a Recipe Object we can use to create recipes inside the form, with an addition of the Ingredients and Instructions created from the Bridging Table. Values coincide with the Database

Text

Description automatically generated

Text

Description automatically generated

Self-explanatory Object classes. These also coincide with the Database. Also populated by the program itself and Forms.

Text

Description automatically generated with medium confidence  
Recipe Bridge Object for the bridging table.

Shape

Description automatically generated with medium confidence

This is a User Object, which is for login and Registration. Alongside this basic constructor, we also have methods inside this class as well.

Text

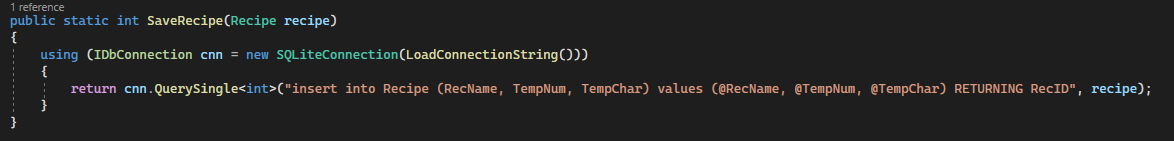
Description automatically generatedThis is a hashing algorithm using the SHA256 Cryptography Hash. This generates a unique 256-bit signature for text. It’s then Encoded into UTF8 and used to split the password and store data safely and securely. The for loop is what generates the Hash.

Text

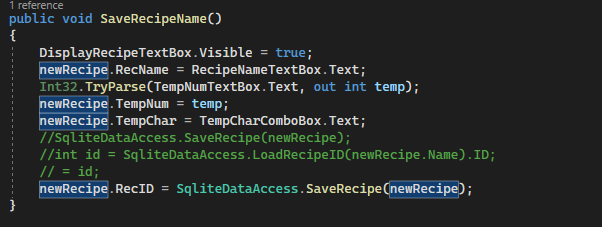
Description automatically generated

This is a program to create a Salt to generate noise for the password, so if it’s stored separately from the hash, the hash becomes useless without it.

Data Access



This Method, In conjunction with

Graphical user interface

Description automatically generated

This form and method save a recipe to the database, additionally, there are also similar methods for Ingredients and Instructions I will not add due to redundancy.

Graphical user interface, application

Description automatically generated

Login field, for existing users to log into the database.

A picture containing graphical user interface

Description automatically generated

Registration form, to write a new user to the database, and works in conjunction with Data Access Methods to write a new user, along with encryption.

Home Page

A picture containing chart

Description automatically generated

Blurb

Chart

Description automatically generated with medium confidence

Not implemented yet Recipe book

Chart, waterfall chart

Description automatically generated

General UI and feature outlook

Sequence Diagrams/User Manual

1. Login

Graphical user interface, application

Description automatically generated

First the login page pops up and you either enter your login info or Register.

Graphical user interface, application

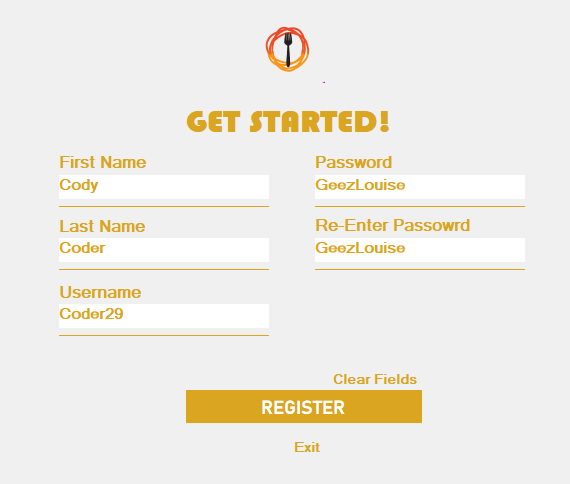
Description automatically generated

Graphical user interface, application

Description automatically generated

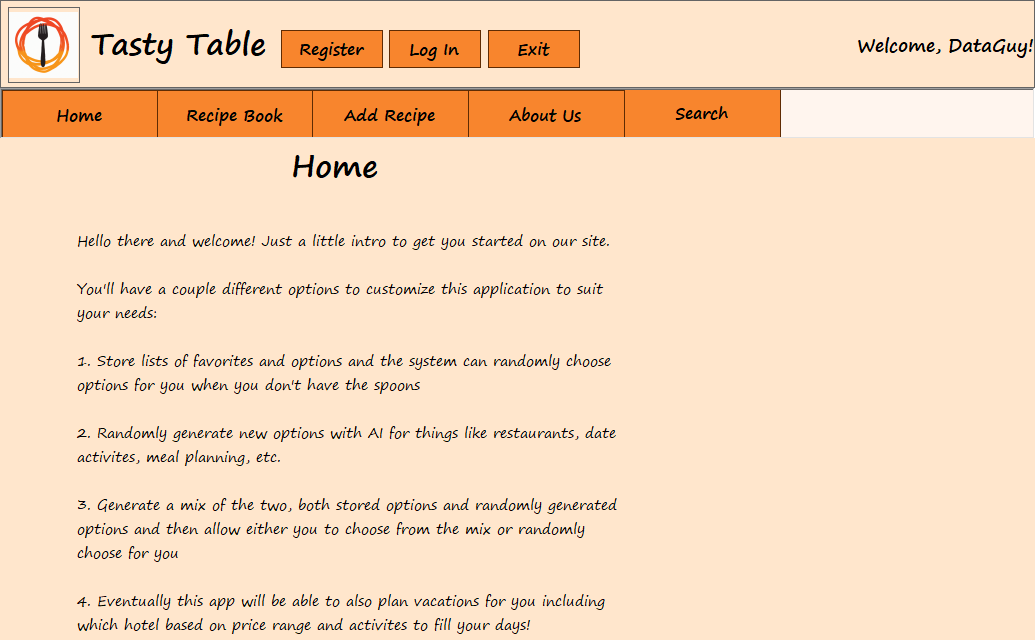
A little message box will pop up to let you know that you are good to go or that you need to try again.

1. Register



1. Home

The landing page will give you a little welcome message at the top of the page. There is also an introduction to the site which explains what it’s features (and future features) are.



1. Search

On the search page, users will be able to search through recipes by ingredient. It will pull up a list of all recipes containing that ingredient and display them in a scroll bar text box, formatted for readability. Be careful to spell correctly.

Type one ingredient in the search bar



Hit the search button and a list will display



Users can scroll through to read all the recipes.





1. Add Recipes

Click the add recipe button

Chart, waterfall chart

Description automatically generated

Once you hit the button, text boxes will pop up for Recipe name and temperature.



After you add the recipe name, you can add ingredients by pressing the prompt button. This action will also save the recipe and in the same action, immediately pull the recipe ID in order to match the ingredients to the appropriate recipe upon entry.



The Add ingredients button will Display the labels and the add ingredient plus sign button. Click on this button and input boxes will appear. The User will need to press this button again for each ingredient.

There are dropdown boxes for ease of input and less issues with input.

Chart, waterfall chart

Description automatically generated

The only issue is the user will have to save each ingredient with a button as they go, but this action will ensure that they are matched up to the appropriate recipe ID.

Graphical user interface

Description automatically generated

When finished with the ingredients, hit done with ingredients button to start the instructions input.

An input box will pop up asking for the number of steps in the recipe so that they can be entered one at a time.

Graphical user interface

Description automatically generated

Graphical user interface, application

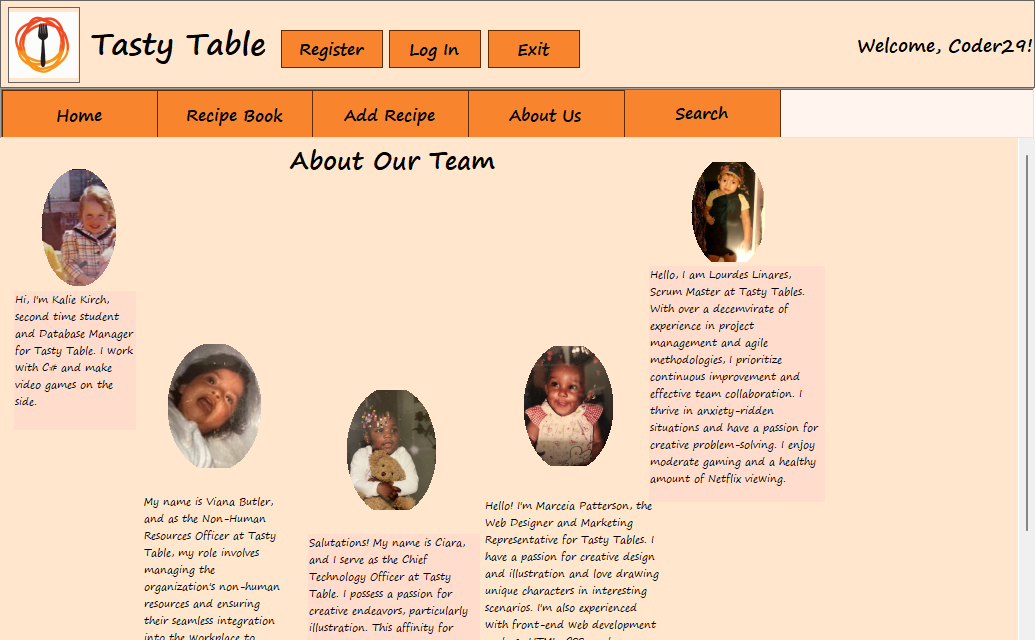
Description automatically generated

Now it’s in the database and can get pulled immediately!



1. About Us

This page displays our team and a blurb about each person.



Timeline

Description automatically generated

User Stories

As a busy professional, I want to be able to quickly search for recipes that I can make with the ingredients I already have in my pantry, so that I can avoid having to make an extra trip to the grocery store.

As a health-conscious individual, I want to be able to filter recipes by dietary restrictions, so that I can find recipes that fit my specific needs, such as vegan, gluten-free, or low-carb.

As a beginner cook, I want to be able to access easy-to-follow recipes with step-by-step instructions and clear pictures, so that I can build my confidence in the kitchen and try new things.

As a seasoned chef, I want to be able to find more advanced and challenging recipes that will push my skills and creativity in the kitchen.

As a social media influencer, I want to be able to easily share my favorite recipes with my followers, so that I can inspire them to try new things and build a community around cooking.

As a parent, I want to be able to find recipes that are kid-friendly and easy to prepare, so that I can involve my children in the cooking process and make mealtime a fun and educational experience.

As a food blogger, I want to be able to access a wide variety of recipes and ingredients, so that I can create unique and interesting content for my blog and social media channels.

As a meal planner, I want to be able to save recipes for later and create shopping lists based on the ingredients I need, so that I can streamline my meal planning process and save time and money.

As a traveler, I want to be able to find recipes from different parts of the world, so that I can experience new cuisines and cultures even when I'm not able to travel.

As a member of a cooking club, I want to be able to share and discuss recipes with other members, so that we can learn from each other and build a community around cooking.

As a college student with limited cooking experience, I want to discover recipes that are simple and beginner-friendly, so that I can learn how to cook for myself and save money on takeout.

Context Diagram and Diagram ZeroDiagram

Description automatically generated

Diagram Zero

Diagram

Description automatically generated

Future Plan

- Rating Feature: A feature where users are able to rate recipes and review based on their personal experience with the recipe in question.

- Saving Feature: Feature where users are able to save recipes to their personal lists so that may come back to it in the case that they might need to look at the steps again.

-List sorting: Users can name their lists and make multiple of them so that it may be easier for them to find. (i.e "Breakfast Recipes", "Lunch Recipes", or "Dinner Recipes".)

- Ingredient Substitutions Feature :

Users can access a list of possible substitutes for certain ingredients that may make any dietary restrictions easier to find meals with

- Category Feature :

Multiple categories are added onto the search query. These categories would include things like what type of cuisine it is, when it is usually eaten (Like breakfast), if it is recommended for children, the level of complexity, cost, etc etc.

- Community Blog Feature:

Feature where users are able to communicate and talk to each other regarding recipes and what not. This feature could be used to ask for suggestions from other users in case they want a human opinion.

- Recommended Recipe Feature:

This feature would either be AI generated based on the recipes that the user normally saves. It would take all of the categories that is in their saves and uses the most commonly found ones to create some suggestions for the user in the case they may not want to go through the trouble of asking the community blog.

Source Code

TastyTableForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinUI

{

public partial class TastyTableForm : Form

{

public TastyTableForm()

{

InitializeComponent();

}

private void TastyTableForm\_Load(object sender, EventArgs e)

{

// Hide All User Controls except the home page

// TODO Maybe make this the login page, either that or have an overlay login

// Background color is 255, 230, 204

recipeBookUCF.Hide();

addRecipeUCF.Hide();

aboutUCF.Hide();

searchUCF.Hide();

string user = LogInForm.ReturnUser();

label1.Text = "Welcome, " + user + "!";

}

private void HomeButton\_Click(object sender, EventArgs e)

{

// Hide All User Controls except the home page

recipeBookUCF.Hide();

addRecipeUCF.Hide();

aboutUCF.Hide();

searchUCF.Hide();

// Bring Home page back to front

homeUCF.Show();

homeUCF.BringToFront();

}

private void RecipeBookButton\_Click(object sender, EventArgs e)

{

// Hide All User Controls except the Recipe Book page

homeUCF.Hide();

addRecipeUCF.Hide();

aboutUCF.Hide();

searchUCF.Hide();

// Bring Recipe Book page back to front

recipeBookUCF.Show();

recipeBookUCF.BringToFront();

}

private void AddRecipeButton\_Click(object sender, EventArgs e)

{

// Hide All User Controls except the Add Recipe page

homeUCF.Hide();

recipeBookUCF.Hide();

aboutUCF.Hide();

searchUCF.Hide();

// Bring Add Recipe page back to front

addRecipeUCF.Show();

addRecipeUCF.BringToFront();

}

private void AboutButton\_Click(object sender, EventArgs e)

{

// Hide All User Controls except the About page

homeUCF.Hide();

recipeBookUCF.Hide();

addRecipeUCF.Hide();

searchUCF.Hide();

// Bring About page back to front

aboutUCF.Show();

aboutUCF.BringToFront();

}

private void SearchButton\_Click(object sender, EventArgs e)

{

// Hide All User Controls except the Search page

homeUCF.Hide();

recipeBookUCF.Hide();

addRecipeUCF.Hide();

aboutUCF.Hide();

// Bring Search page back to front

searchUCF.Show();

searchUCF.BringToFront();

}

private void registerButton\_Click(object sender, EventArgs e)

{

new RegistrationForm().Show();

}

private void loginButton\_Click(object sender, EventArgs e)

{

LogInForm login = new LogInForm();

login.Show();

}

private void ExitButton\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

RegistrationForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using TastyTableClassLibrary;

namespace WinUI

{

public partial class RegistrationForm : Form

{

public RegistrationForm()

{

InitializeComponent();

}

private void registerButton\_Click(object sender, EventArgs e)

{

if ((userNametxt.Text == "") || (firstNametxt.Text == "") || (lastNametxt.Text == "") || (passwordtxt.Text == "") || (confirmpasstxt.Text == ""))

{

MessageBox.Show("One or more of the fields are not filled in.");

}

else

{

// check if passwords match

// if passwords match create account

// if passwords do not match then show message box

if (passwordtxt.Text == confirmpasstxt.Text)

{

User user = new User();

user.Username = userNametxt.Text;

user.FName = firstNametxt.Text;

user.LName = lastNametxt.Text;

user.Salt = User.CreateSalt(10);

user.HashPass = User.EncryptPassword(passwordtxt.Text, user.Salt);

SqliteDataAccess.SaveUser(user);

MessageBox.Show("You have successfully made an account!");

this.Close();

}

else

{

MessageBox.Show("Your passwords do not match.");

}

}

}

private void exitLabel\_Click(object sender, EventArgs e)

{

this.Close();

}

}

}

LogInForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using TastyTableClassLibrary;

namespace WinUI

{

public partial class LogInForm : Form

{

public LogInForm()

{

InitializeComponent();

}

public static User user = new User();

private void loginButton\_Click(object sender, EventArgs e)

{

//checks if user exists via username

// if i also did password i couldn't verify it properly

//User user = new User();

//checks if username is null and if it is displays message box

// if it's filled it, it goes onto the next step of checking the password and etc.

if (usernametxt.Text == "")

{

MessageBox.Show("Please enter in a username in order to log in!");

}

else

{

string username = usernametxt.Text;

user = SqliteDataAccess.LoadUser(username);

// check for password/if user even exists

if (user == null)

{

MessageBox.Show("Invalid username or password! Try again or register for an account!");

}

else

{

string attemptedHashPass = User.EncryptPassword(txtPassword.Text, user.Salt);

if (attemptedHashPass == user.HashPass)

{

MessageBox.Show("You've successfully logged in, welcome!");

this.Close();

}

else

{

MessageBox.Show("Invalid username or password! Try again or register for an account!");

}

}

}

}

internal static string ReturnUser()

{

return user.Username;

}

//private void RegisterFormOpen(RegistrationForm form)

//{

// form.BringToFront();

//}

private void exitLabel\_Click(object sender, EventArgs e)

{

this.Close();

}

// private RegistrationForm form = new RegistrationForm();

private void RegisterLabel\_Click(object sender, EventArgs e)

{

//Want to pull up the registerform but couldn't figure it out. Moving on since that can be done other ways

}

}

}

SearchUC.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using TastyTableClassLibrary;

using System.Linq;

using System.Net.NetworkInformation;

namespace WinUI

{

public partial class SearchUC : UserControl

{

public SearchUC()

{

InitializeComponent();

}

private void SearchButton\_Click(object sender, EventArgs e)

{

// Search from database using text

// Clears Display

DisplayTextBox.Text = "";

// List to pull ingredients that match the search

List<Ingredient> ingSearch;

// List to store the ingredient IDs so we can search bridging table

List<int> ingIDs = new List<int>();

// List to store the recipe IDs we got from the bridging table

List<int> recBrid = new List<int>();

// the var to store the search term

string search = SearchTextBox.Text;

// pulls from database to fill list of ingredients that match the search

ingSearch = SqliteDataAccess.LoadIngredients(search);

//loops through the above list and adds the IDs to a list that we will use for searching

foreach (Ingredient item in ingSearch)

{

ingIDs.Add(item.IngID);

}

// Searches based on ID

foreach (int ingID in ingIDs)

{

recBrid.Add(SqliteDataAccess.LoadRecFromBridge(ingID));

}

// Makes a unique list of all recipes without duplicates

List<int> uniqueRecBrid = recBrid.Distinct().ToList();

// Pull the Recipe name, temp, and C or F based on uniqueRecBrid

foreach (int recID in uniqueRecBrid)

{

// Creates a list of object for each recipe pulled

Recipe recipe = SqliteDataAccess.LoadRecipeOnID(recID);

// Displays Recipe name for each recipe

DisplayTextBox.Text += " \r\n\r\n" + recipe.RecName + " \r\n" + string.Format("{0}\u00B0", recipe.TempNum) + recipe.TempChar + " \r\n";

// Goes to bridging table to get the ID of each ingredient that matches the recipe and stores it to list

List<int> ingsIDs = SqliteDataAccess.LoadIngFromBridge(recID);

// loops through ID list and pulls each ingredient as object to display

List<Ingredient> ings = new List<Ingredient>();

foreach(int ingID in ingsIDs)

{

ings.Add(SqliteDataAccess.LoadIngredientID(ingID));

}

// Displays ingredients

foreach (Ingredient ing in ings)

{

DisplayTextBox.Text += ing.Quantity + " " + ing.Unit + " " + ing.IngName + " \r\n";

}

List<Instruction> insts = SqliteDataAccess.LoadInstructions(recID);

DisplayTextBox.Text += " \r\n" + "Directions" + " \r\n\r\n";

foreach (Instruction inst in insts)

{

DisplayTextBox.Text += " \r\n" + inst.StepNum + " " + inst.Description + " \r\n";

}

}

}

}

}

RecipeBookUC.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinUI

{

public partial class RecipeBookUC : UserControl

{

public RecipeBookUC()

{

InitializeComponent();

}

}

}

HomeUC.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinUI

{

public partial class HomeUC : UserControl

{

public HomeUC()

{

InitializeComponent();

}

}

}

AddRecipeUC.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net.NetworkInformation;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using TastyTableClassLibrary;

using WinUI.UserControls;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using Button = System.Windows.Forms.Button;

using TextBox = System.Windows.Forms.TextBox;

using Microsoft.VisualBasic;

namespace WinUI

{

public partial class AddRecipeUC : UserControl

{

public static Recipe newRecipe = new Recipe();

public AddRecipeUC()

{

InitializeComponent();

QuantityLabel.Visible= false;

UnitLabel.Visible= false;

IngNameLabel.Visible = false;

AddIngButton.Visible = false;

TempCharComboBox.Visible = false;

TempCharLabel.Visible = false;

RecipeNameLabel.Visible = false;

RecipeNameTextBox.Visible = false;

TempNumLabel.Visible = false;

TempNumTextBox.Visible = false;

DisplayRecipeTextBox.Visible = false;

ReadyButton.Visible = false;

DonWIngButton.Visible = false;

}

int addIngrControl = 5;

private void AddIngButton\_Click(object sender, EventArgs e)

{

AddIngredientUC newIng = new AddIngredientUC();

this.Controls.Add(newIng);

newIng.Top = addIngrControl \* 40;

newIng.Left = 11;

addIngrControl += 1;

}

private void AddNewRecipeButton\_Click(object sender, EventArgs e)

{

//QuantityLabel.Visible = true;

//UnitLabel.Visible = true;

//IngNameLabel.Visible = true;

//AddIngButton.Visible = true;

TempCharComboBox.Visible = true;

TempCharLabel.Visible = true;

RecipeNameLabel.Visible = true;

RecipeNameTextBox.Visible = true;

TempNumLabel.Visible = true;

TempNumTextBox.Visible = true;

ReadyButton.Visible = true;

}

public void SaveRecipeName()

{

DisplayRecipeTextBox.Visible = true;

newRecipe.RecName = RecipeNameTextBox.Text;

Int32.TryParse(TempNumTextBox.Text, out int temp);

newRecipe.TempNum = temp;

newRecipe.TempChar = TempCharComboBox.Text;

//SqliteDataAccess.SaveRecipe(newRecipe);

//int id = SqliteDataAccess.LoadRecipeID(newRecipe.Name).ID;

// = id;

newRecipe.RecID = SqliteDataAccess.SaveRecipe(newRecipe);

}

private void ReadyButton\_Click(object sender, EventArgs e)

{

QuantityLabel.Visible = true;

UnitLabel.Visible = true;

IngNameLabel.Visible = true;

AddIngButton.Visible = true;

DonWIngButton.Visible = true;

SaveRecipeName();

}

internal static int PassRecipeID()

{

return newRecipe.RecID;

}

private void DonWIngButton\_Click(object sender, EventArgs e)

{

AddIngButton.Visible = false;

string message = "How many steps are in this recipe?";

string title = "Recipe Instructions";

string numofInstrS = Interaction.InputBox(message, title);

Int32.TryParse(numofInstrS, out int numofInstrN);

for (int i = 1; i < numofInstrN + 1; i++)

{

Instruction inst = new Instruction();

inst.StepNum = i;

message = "Step " + i;

title = "Recipe Steps";

inst.Description = Interaction.InputBox(message, title);

inst.RecID = newRecipe.RecID;

SqliteDataAccess.SaveInstructions(inst);

}

}

}

}

AddIngredientUC.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using TastyTableClassLibrary;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WinUI.UserControls

{

public partial class AddIngredientUC : UserControl

{

public AddIngredientUC()

{

InitializeComponent();

}

public List<Ingredient> IngredientsForRecipe = new List<Ingredient>();

//public RecipeBridge recipeBridge = new RecipeBridge();

public int recipeID;

public int ingredientID;

public void SelectedIngr()

{

Ingredient ing = new Ingredient();

string name = textBox1.Text;

double quantity = double.Parse(QuantityComboBox.Text);

string unit = UnitComboBox.Text;

RecipeBridge recipeBridge = new RecipeBridge();

ing.IngName = name;

ing.Quantity = quantity;

ing.Unit = unit;

//SqliteDataAccess.SaveIngredients(ing);

ingredientID = SqliteDataAccess.SaveIngrReturnID(ing);

MessageBox.Show("Saved!"); //Comment out later

IngredientsForRecipe.Add(ing);

recipeID = AddRecipeUC.PassRecipeID();

//ingredientID = SqliteDataAccess.LoadIngredientID(ing.Name).ID;

//MessageBox.Show(recipeID.ToString() + ingredientID.ToString());

recipeBridge.RecID = recipeID;

recipeBridge.IngID = ingredientID;

SqliteDataAccess.SavetoBridge(recipeBridge);

// Fix recipeID retrieval

}

private void SaveButton\_Click(object sender, EventArgs e)

{

SelectedIngr();

}

}

}

AboutUC.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinUI

{

public partial class AboutUC : UserControl

{

public AboutUC()

{

InitializeComponent();

}

}

}

User.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

namespace TastyTableClassLibrary

{

public class User

{

public User() { }

public int ID { get; set; }

public string Username { get; set; }

public string FName { get; set; }

public string LName { get; set; }

public string HashPass { get; set; }

public string Salt { get; set; }

public static string EncryptPassword(string password, string salt)

{

// No need to reverse the hash, we just store the salt, use the same salt when hashing attempted password

// and then compare the two already hashed passwords

using (SHA256 sha256 = SHA256.Create())

{

byte[] hashBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password + salt));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < hashBytes.Length; i++)

{

builder.Append(hashBytes[i].ToString("x2"));

}

return builder.ToString();

}

}

public static String CreateSalt(int size)

{

var rng = new System.Security.Cryptography.RNGCryptoServiceProvider();

var buff = new byte[size];

rng.GetBytes(buff);

return Convert.ToBase64String(buff);

}

}

}

SqliteDataAccess.cs

using Dapper;

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SQLite;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Xml.Linq;

namespace TastyTableClassLibrary

{

public class SqliteDataAccess

{

public static User LoadUser(string Username)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var parameters = new { Username = Username };

var output = cnn.QuerySingle<User>("SELECT \* from UserLogin WHERE Username = @Username", parameters);

return output;

}

}

public static void SaveUser(User user)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

cnn.Execute("insert into UserLogin (Username, FName, LName, HashPass, Salt) values (@Username, @FName, @LName, @HashPass, @Salt)", user);

}

}

public static List<Ingredient> LoadIngredients(string IngName)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var parameters = new {IngName = IngName };

//return cnn.Query<Ingredient>("SELECT \* from Ingredient WHERE IngName LIKE \"%@IngName%\"", parameters).ToList();

return cnn.Query<Ingredient>("SELECT \* from Ingredient WHERE IngName LIKE '%' || @IngName || '%'", parameters).ToList();

}

}

public static int SaveIngrReturnID(Ingredient ingr)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

return cnn.QuerySingle<int>("insert into Ingredient (IngName, Quantity, Unit) values (@IngName, @Quantity, @Unit) RETURNING IngID", ingr);

}

}

public static void SaveInstructions(Instruction inst)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

cnn.Execute("insert into Instruction (StepNum, Description, RecID) values (@StepNum, @Description, @RecID)", inst);

}

}

public static List<Instruction> LoadInstructions(int RecID)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var parameters = new { RecID = RecID };

return cnn.Query<Instruction>("SELECT \* from Instruction WHERE RecID = @RecID", parameters).ToList();

}

}

//public static int SaveInstructions(Instruction inst)

//{

// using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

// {

// return cnn.QuerySingle<int>("insert into Instruction (StepNum, Description, RecID) values (@StepNum, @Description, @RecID) RETURNING ID", inst);

// }

//}

public static Ingredient LoadIngredientID(int IngID)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var parameters = new { IngID = IngID };

var output = cnn.QuerySingle<Ingredient>("SELECT \* from Ingredient WHERE IngID = @IngID", parameters);

return output;

}

}

public static List<Recipe> LoadRecipe()

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var output = cnn.Query<Recipe>("SELECT \* from Recipe", new DynamicParameters());

return output.ToList();

}

}

public static Recipe LoadRecipeOnID(int RecID)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var parameters = new { RecID = RecID };

return cnn.QuerySingle<Recipe>("SELECT \* from Recipe WHERE RecID = @RecID", parameters);

}

}

public static int SaveRecipe(Recipe recipe)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

return cnn.QuerySingle<int>("insert into Recipe (RecName, TempNum, TempChar) values (@RecName, @TempNum, @TempChar) RETURNING RecID", recipe);

}

}

public static void SavetoBridge(RecipeBridge recipeBridge)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

cnn.Execute("insert into RecipeIngr (RecID, IngID) values (@RecID, @IngID)", recipeBridge);

}

}

public static int LoadRecFromBridge(int IngID)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var parameters = new { IngID = IngID };

return cnn.QuerySingle<int>("SELECT DISTINCT RecID from RecipeIngr WHERE IngID = @IngID", parameters);

}

}

public static List<int> LoadIngFromBridge(int RecID)

{

using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

{

var parameters = new { RecID = RecID };

return cnn.Query<int>("SELECT IngID from RecipeIngr WHERE RecID = @RecID", parameters).ToList();

}

}

//public static List<string> PullRecipe(string RecName)

//{

// using (IDbConnection cnn = new SQLiteConnection(LoadConnectionString()))

// {

// var parameters = new { RecName = RecName };

// return cnn.Query<string>("SELECT Recipe.RecName, Recipe.TempNum, Recipe.TempChar, Instruction.StepNum, Instruction.Description, Ingredient.IngName, Ingredient.Quantity," +

// " Ingredient.Unit FROM Recipe JOIN Instruction ON Recipe.RecID = Instruction.RecID JOIN RecipeIngr ON Recipe.RecID = RecipeIngr.RecID JOIN Ingredient " +

// "ON RecipeIngr.IngID = Ingredient.IngID WHERE Recipe.RecName = @RecName", parameters).ToList();

// }

//}

private static string LoadConnectionString(string id = "Default")

{

return ConfigurationManager.ConnectionStrings[id].ConnectionString;

}

}

}

RecipeBridge.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TastyTableClassLibrary

{

public class RecipeBridge

{

public RecipeBridge() { }

public int RecID { get; set; }

public int IngID { get; set; }

}

}

Recipe.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TastyTableClassLibrary

{

public class Recipe

{

public Recipe() { }

public int RecID { get; set; }

public string RecName { get; set; }

public int TempNum { get; set; }

public string TempChar { get; set; }

// We'll get these VV from the bridging table's reference and sort them out that way

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

public List<Instruction> Instructions { get; set;}

}

}

Instruction.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TastyTableClassLibrary

{

public class Instruction

{

public Instruction() { }

public int InstID { get; set; } //An autoincremented unique identifier

public int StepNum { get; set; } //Each step is numbered like 1, 2, 3

public string Description { get; set; }

public int RecID { get; set; }

}

}

Ingredient.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TastyTableClassLibrary

{

public class Ingredient

{

public Ingredient() { }

public int IngID { get; set; }

public string IngName { get; set; }

public double Quantity { get; set; }

public string Unit { get; set; }

}

}

App.config

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<connectionStrings>

<add name="Default" connectionString="Data Source=.\TastyTableDB.db;Version=3;" providerName="System.Data.SqlClient"/>

</connectionStrings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

</configuration>

packages.config

<?xml version="1.0" encoding="utf-8"?>

<packages>

<package id="Stub.System.Data.SQLite.Core.NetFramework" version="1.0.117.0" targetFramework="net472" />

<package id="System.Data.SQLite.Core" version="1.0.117.0" targetFramework="net472" />

</packages>

Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinUI

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LogInForm());

Application.Run(new TastyTableForm());

}

}

}