Bradley Culligan, E5

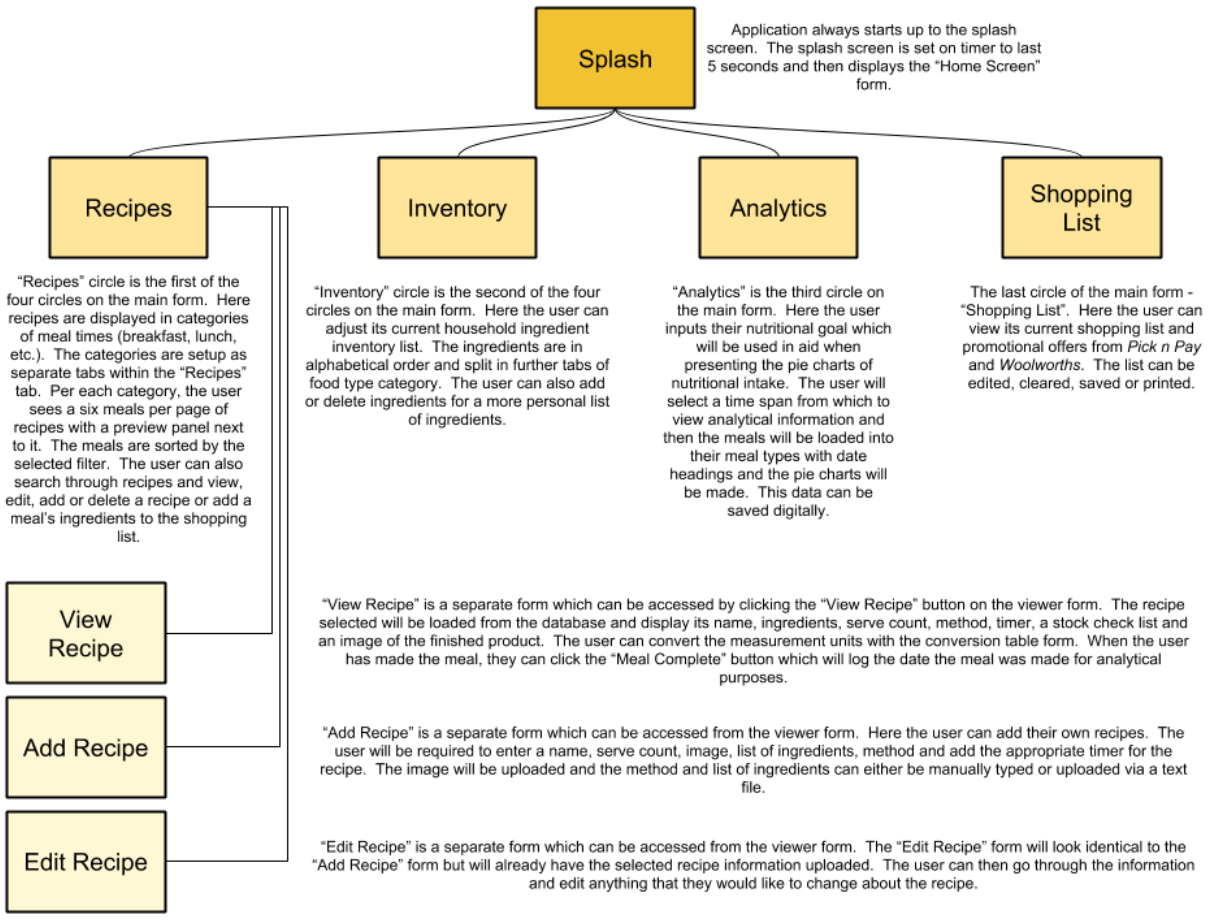
IT PAT, Phase 1

Slice ‘n Dice

2018

Slice 'n Dice

Basic Design:



Application always starts up to the splash screen. The splash screen is set on timer to last 5 seconds and then displays the "Home Screen" form.

"Recipes" circle is the first of the four circles on the main form. Here recipes are displayed in categories of meal times (breakfast, lunch, etc.). The categories are setup as separate tabs within the "Recipes" tab. Per each category, the user sees a six meals per page of recipes with a preview panel next to it. The meals are sorted by the selected filter. The user can also search through recipes and view, edit, add or delete a recipe or add a meal's ingredients to the shopping list.

"Inventory" circle is the second of the four circles on the main form. Here the user can adjust its current household ingredient inventory list. The ingredients are in alphabetical order and split in further tabs of food type category. The user can also add or delete ingredients for a more personal list of ingredients.

"Analytics" is the third circle on the main form. Here the user inputs their nutritional goal which will be used in aid when presenting the pie charts of nutritional intake. The user will select a time span from which to view analytical information and then the meals will be loaded into their meal types with date headings and the pie charts will be made. This data can be saved digitally.

The last circle of the main form - "Shopping List". Here the user can view its current shopping list and promotional offers from *Pick n Pay* and *Woolworths*. The list can be edited, cleared, saved or printed.

"View Recipe" is a separate form which can be accessed by clicking the "View Recipe" button on the viewer form. The recipe selected will be loaded from the database and display its name, ingredients, serve count, method, timer, a stock check list and an image of the finished product. The user can convert the measurement units with the conversion table form. When the user has made the meal, they can click the "Meal Complete" button which will log the date the meal was made for analytical purposes.

"Add Recipe" is a separate form which can be accessed from the viewer form. Here the user can add their own recipes. The user will be required to enter a name, serve count, image, list of ingredients, method and add the appropriate timer for the recipe. The image will be uploaded and the method and list of ingredients can either be manually typed or uploaded via a text file.

"Edit Recipe" is a separate form which can be accessed from the viewer form. The "Edit Recipe" form will look identical to the "Add Recipe" form but will already have the selected recipe information uploaded. The user can then go through the information and edit anything that they would like to change about the recipe.

Database Design:

tblRecipe				
FIELDS	**DATA TYPE**	**SIZE**	**EXAMPLE**	**VALIDATION**
RecID	AutoNumber	Long integer	2	>0; numeric only
RecName	Short Text	100	Baked Cheesecake	Required; do not allow zero length
RecImage	Short Text	255	Cheesecake.jpg	Application adds file extension type
RecType	Short Text	9	Dessert	Required; application drop menu
RecServe	Number	Byte	8	Defaults to 0; application forces numeric value
RecMethod	Long Text	64 000	Preheat oven to 180°. Crush biscuits in a food processor and mix with melted butter. Line the base of 22cm cake tin with baking paper and place biscuit mix in the tin. Press down the biscuit mix with a spoon and bake for 10 minutes. Mix the cream cheese, eggs, sugar, flour, 250g creme fraiche plus two extra tablespoons of creme fraiche and vanilla extract in a bowl. Pour into the baked base and	Do not allow zero length

			bake for 15 minutes at 180°, then turn the oven down to 140° and bake for 50 minutes. Turn off oven and allow cheesecake to cool in oven. Refrigerate the cake for 1½ hours before you take it out the tin. Top it with the extra creme fraiche and strawberries and dust with icing sugar.	
RecCookTime	Number	Integer	140	Default 0; application forces numeric value
RecTimer	Short Text	3	10	Default 0; do not allow zero length; application forces numeric value

tblIngredient				
FIELDS	**DATA TYPE**	**SIZE**	**EXAMPLE**	**VALIDATION**
IngID	AutoNumber	Long Integer	12	>0; numeric only
IngName	Short Text	190	Egg	Required; do not allow zero length
IngAvailable	Yes/No	Yes/No	Yes	Default No
IngType	Short Text	20	Dairy	Application forces a choice from drop menu

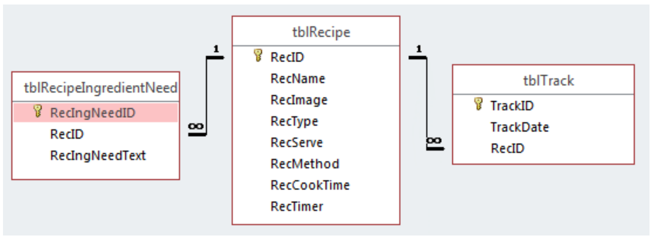
tblRecipeIngredientNeed				
FIELDS	**DATA TYPE**	**SIZE**	**EXAMPLE**	**VALIDATION**

RecIngNeedID	AutoNumber	Long Integer	6	>0; numeric only
RecID	Number	Long Integer	1	Connected to tblRecipe
RecIngNeedText	Short Text	200	7 apples	Application

tblTrack				
FIELDS	**DATA TYPE**	**SIZE**	**EXAMPLE**	**VALIDATION**
TrackID	AutoNumber	Long Integer	4	>0; numeric only
TrackDate	Date/Time	Short Date	2018/08/16	Default current date; format yyyy/mm/dd
RecID	Number	Long Integer	94	Connected to tblRecipe

tblDietaryGoal				
FIELDS	**DATA TYPE**	**SIZE**	**EXAMPLE**	**VALIDATION**
DGoalID	AutoNumber	Long Integer	4	>0; numeric only
DGoalPercent	Number	Byte	7	Default 0; application forces numeric only
DGoalType	Short Text	20	Fruit and Vegetables	Uneditable by the user

Relationship:



Data Dictionary:

DATA STRUCTURE	NAME	DATA STORED	COMMENTS
Arrays	CtrlState	Integer	DFCS_BUTTONCHECK, DFCS_BUTTONCHECK or DFCS_CHECKED which determines the checkbox style to be drawn in the dbGrids
	arrKeys	String	6 values which hold the RecIDs of the recipes that can be seen in the viewer
	arrRecID	String	Contains the RecIDs of the recipes within the “to” and “from” dates
	arrIngredients	String	Contains all ingredient text of all the ingredients that are needed for the appropriate RecIDs in arrRecID
	arrTypeCount	Integer	Keeps count of the quantities of the various food types based on their quantity found in arrIngredients
	arrColours	TColor	Colours to be used for the pie chart
	arrLabels	String	Labels to be used for the pie chart
	arrGoal	String	Contains all goal dietary values that the user can edit
Constants	Rect	TRect	Data about the rectangular parameters
	DataCol	Integer	Data column
	Column	TColumn	Appropriate column
	State	TGridDrawState	The current draw state of the dbGrid
	iSS	Integer	Remains 0 - value for seconds in the timer which is used in formatting the timer
Database	tblRecipe	Recipes	All information pertaining to recipes except for their ingredient text
	tblRecipeIngredientNeed	Ingredient text for recipes	Ingredient text needed for recipes
	tblIngredient	Inventory stock	List of all ingredients shown in the

			Inventory form of the application
	tblTrack	Meal tracking	Dates of past meals for analytical data
	tblDietaryGoal	Dietary goal values	Keeps track of the user's dietary goal values
Object	objDependencies	TDependencies	Contains (as string): - SelectedID - MealType - AddOREdit - FromDate - ToDate These values are used to carry data across to other forms
Text files	tFile (Shopping List.txt)	Shopping list data	Text file stored in application folder which holds the data of the current shopping list. Items are added to this text file when the shopping list in the application is updated and is used to load the application shopping list when the user closes the application and reopens it at a later stage
	tShopList (Shopping List yyyymmdd.txt)	Shopping list data	User's own save of the shopping list for them to digitally store on their local hard drive or to upload to their online storage. yyyymmdd is replaced with the current date
	tFile (sFile)	List of ingredient text needed for a recipe	User selects a text file with the ingredient data they want to upload and then the data is uploaded to the listbox
	tAnalytics (Analytics yyyymmdd.txt)	File made by application to be saved for user based on analytical data	User's save (in text file format) of the data shown in the analytical columns. yyyymmdd is replaced by current date
Variables	sDate	String	Current date in format yyyymmdd
	sItem	String	Item to be added to shopping list
	sQuantity	String	Quantity of item to be added to shopping list
	sLine	String	Line of text from application shopping list to be written to text file shopping list or vice versa

	iLineCounter	Integer	Used to get last two lines of text in the shopping list which gets added to the text file shopping list
	FOriginalOptions	TDBGridOptions	Keeps the dbGrid's original options so options can quickly be changed back to original options after options have changed for editing function
	iImageCount	Integer	Used to count through the six images shown in the catalog of the viewer
	iPage	Integer	Keeps track of the page number of the catalog part of the viewer
	iRecordNumber	Integer	Number of records within a recipe type category which is used to limit loading recipes further than this number as this would cause an error
	sID	String	The RecID of the currently loading recipe to add to arrKeys
	sRecName	String	The recipe name of the selected recipe which is used in confirming with the user about deleting the recipe
	sSearchedID	String	Sets the SelectedID in objDependencies based on what item was searched
	sSearchText	String	The text that the search bar is using to search in the dbGrid below it
	sImageField	String	The name of the file which pertains to the necessary image to be loaded
	sName	String	Recipe name to be loaded
	sServe	String	Serve count of recipe to be loaded
	sTime	String	Total time of cooking to be displayed for the loaded recipe
	dynLogoIMG	TImage	Dynamic variable used to load the logo image in place of the timer if no timer is required for a recipe upon viewing

	bTimerActive	Boolean	Keeps track of whether the timer is active or not
	iTime	Integer	Total timer value which is used to split into hour and minute values
	bTimerNeeded	Boolean	Tracks whether a timer is needed or not to know if a timer or logo image must be displayed
	iSelect	Integer	Used in confirming with the user whether they want to “go back”
	iHH	Integer	Hour value entered by the user in resetting the timer
	iMM	Integer	Minute value entered by the user in resetting the timer
	iHH	Integer	Hour value based on splitting the iTime variable for the timer that is displayed to the user
	iMM	Integer	Minute value based on splitting the iTime variable for the timer that is displayed to the user
	sIng	String	Used to load the list of ingredients when viewing a recipe
	iTimerSS	Integer	Used in updating the timer value shown to the user
	iTimerMM	Integer	Used in updating the timer value shown to the user
	iTimerHH	Integer	Used in updating the timer value shown to the user
	bTimerDone	Boolean	Determines whether the timer is complete or still working its way down the set time
	sIng	String	Used to load the list of ingredients when using the “Add Meal to Shopping List” pop up form
	PieCharts	TPieCharts	Used to adjust the slices of the achieved pie chart
	q	Integer	Counter used to add ingredients to arrIngredients
	w	Integer	Counter to go through arrIngredients and add appropriate

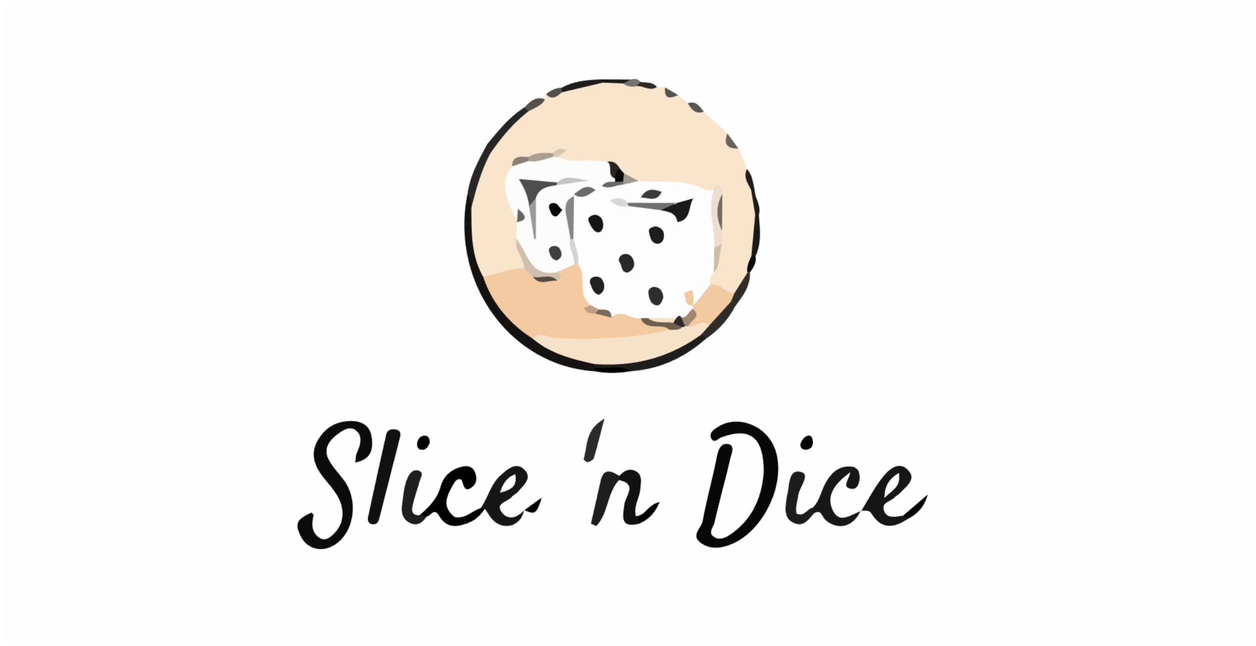
			values to arrTypeCount
	r	Integer	Counter to go through arrTypeCount and initialise the array
	iCount	Integer	Counter used when adding data to the new series of the pie chart (achieved)
	dynLogoIMG	TImage	Dynamic variable to create the image of the logo in the timer’s place for adding or editing a recipe
	bImgChange	Boolean	Tracks whether the image for a recipe was changed or not so the application knows what statements to execute from there on
	sIng	String	Contains the ingredient text of a recipe which the user is adding to the ingredient listbox
	sFile	String	Name of text file from which the ingredients/method will be loaded into the application
	iIngCount	Integer	Counter when going through the ingredient listbox to add data to the database
	iWidth	Integer	Width of current item in ingredient listbox
	iMaxWidth	Integer	Width of longest item in ingredient listbox
	iItemCount	Integer	Counter when going through the ingredient listbox
	jIngCount	Integer	Counter when going through the ingredient listbox
	Bmp	TBitmap	Bitmap variable to assign the loaded image to from which the image will be saved
	bSettingAllowed	Boolean	Allows or restricts ability of setting the “to” and “from” dates in the objDependencies to be used in the achieved pie chart
	k	Integer	Counter used in updating database with dietary goal values entered by user

	k	Integer	Counter used in updating array with dietary goal values from database
	iTotal	Integer	Total value of dietary goal. This value must equal 100 to validate that the user has entered the values in percentage form

Input Process Output:

Splash

Slice_n_Dice_Splash_u - SnDSplash



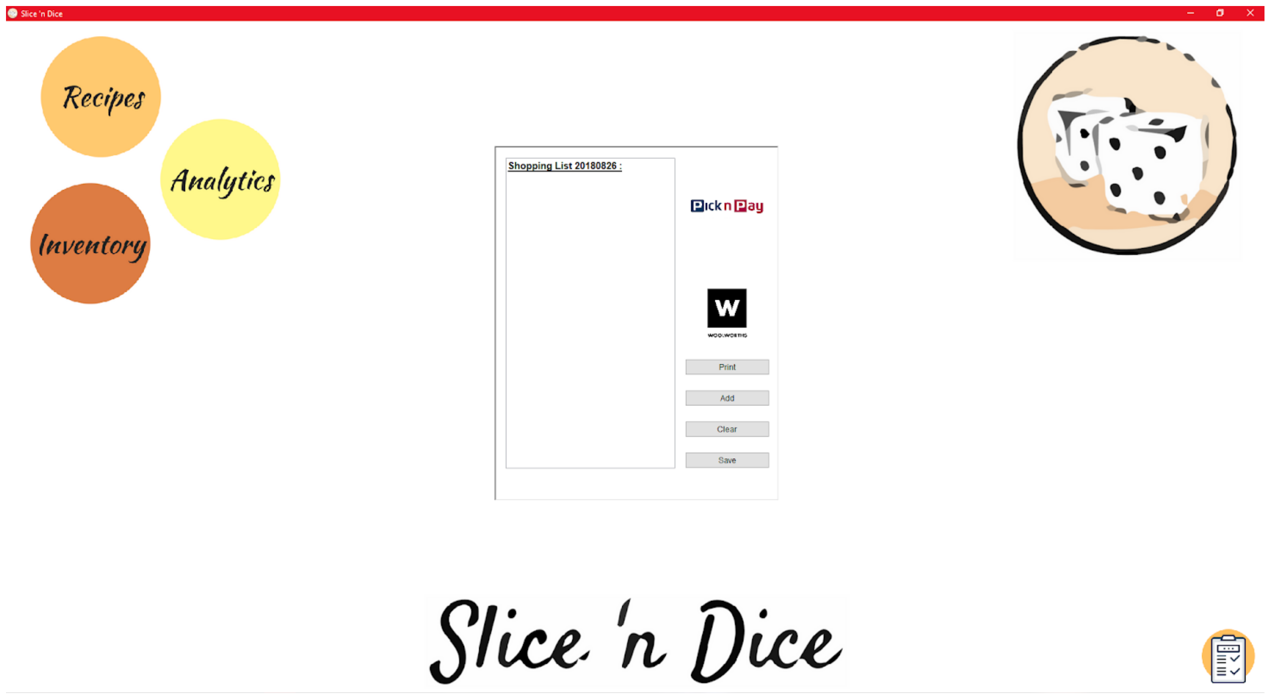
INPUT	PROCESS	OUTPUT
User starts application	- Timer component is active while splash image is displayed - Timer runs down to zero	Splash screen closes and home screen is displayed for user to take further action

CODE	EXPLANATION

procedure TSnDSplash.splashTimerTimer(Sender: TObject);
begin
  splashTimer.Enabled := false; // when timer is complete, close form to show home screen
  close;
end;
	When timer component is complete, close form and show home screen

Home Screen

Slice_n_Dice_Home_u - SnDHome



INPUT	PROCESS	OUTPUT
Mouse hover over "Recipes" circle	Background image is loaded with new image found in application folder	Top right corner image changes to match the theme of recipes

CODE	EXPLANATION

procedure TSnDHome.imgRecipesMouseEnter(Sender: TObject);
begin
imgBackground.Picture.LoadFromFile('SnDHomeTwo.png');
end;
	Load recipe image which is backing image

INPUT	PROCESS	OUTPUT
Mouse hover over “Inventory” circle	Background image is loaded with new image found in application folder	Top right corner image changes to match the theme of inventory ingredients

CODE	EXPLANATION

procedure TSnDHome.imgInventoryMouseEnter(Sender: TObject);
begin
imgBackground.Picture.LoadFromFile('SnDHomeThree.png');
end;
	Load inventory image which is backing image

INPUT	PROCESS	OUTPUT
Mouse hover over “Analytics” circle	Background image is loaded with new image found in application folder	Top right corner image changes to match the theme of analytics

CODE	EXPLANATION

procedure TSnDHome.imgAnalyticsMouseEnter(Sender: TObject);
begin
imgBackground.Picture.LoadFromFile('SnDHomeFour.png');
end;
	Load analytics image which is backing image

INPUT	PROCESS	OUTPUT
Mouse hover over circle with shopping list icon	Receives text hint	Text hint is displayed over icon

INPUT	PROCESS	OUTPUT
User clicks shopping list icon	Application determines whether the shopping list is currently visible	If shopping list is visible - hide shopping list If shopping list is invisible - show shopping list

CODE	EXPLANATION

procedure
TSnDHome.imgShoppingListClick(Sender:
TObject);
begin
  if pnlShopList.Visible = false then
  begin
    pnlShopList.Visible := true;
  end
  else if pnlShopList.Visible = true then
  begin
    pnlShopList.Visible := false;
  end;
end;
	If shopping list is visible - hide shopping list
	If shopping list is invisible - show shopping list

INPUT	PROCESS	OUTPUT
User clicks *Pick n Pay* logo image	Application starts default internet browser and loads URL with set URL	Default internet browser is opened on the *Pick n Pay* promotions page

CODE	EXPLANATION

procedure TSnDHome.imgPicknPayClick(Sender:
TObject);
begin
  ShellExecute(0, 'open',
'https://www.pnp.co.za/welcome?rf=y&_ga=2.258
583689.2005233561.1525418824-1984058615.15
25418822', nil, nil, SW_ShowNormal);
end;
	ShellAPI is required for the ShellExecute function which allows *Slice 'n Dice* to open applications on the user's computer

INPUT	PROCESS	OUTPUT
User clicks *Woolworths* logo image	Application starts default internet browser and loads URL with set URL	Default internet browser is opened on the *Woolworths* promotions page

CODE	EXPLANATION

procedure TSnDHome.imgWoolworthsClick(Sender: TObject);
begin
  ShellExecute(0, 'open', 'http://www.woolworths.co.za/store/cat/_/N-1z13sk5Z1ha6kkl', nil, nil,
    SW_ShowNormal);
end;
	ShellAPI is required for the ShellExecute function which allows *Slice 'n Dice* to open applications on the user's computer

INPUT	PROCESS	OUTPUT
User clicks "Print" button	Print dialog is opened for user to execute print from	If executed, the shopping list will be printed to the printer; else the print will be cancelled and the dialog will close

CODE	EXPLANATION

procedure TSnDHome.btnPrintSLClick(Sender: TObject);
begin
  if dlgPrint.Execute then
  begin
    Application.ProcessMessages;
    redShopList.Print('Shopping List');
  end;
end;
	If the print dialog is executed, call the print function of the richedit containing the text to be printed

INPUT	PROCESS	OUTPUT
User clicks "Add" button	Receives data from user pertaining to the ingredient name and quantity to be added to the shopping list	Shopping list in application is updated with newly added value and the text file shopping list also receives the updated value to keep the text file shopping list up to date with the most recent version of the shopping list

CODE	EXPLANATION

procedure TSnDHome.btnAddSLClick(Sender:
TObject);
var
  sItem, sQuantity, sLine: string;
  tFile: TextFile;
  iLineCounter: integer;
 	
begin
  sItem := InputBox('Add to shopping list',
    'What ingredient would you like to add to your
shopping list?', '');
  // gets item from user
  sQuantity := InputBox('Add to shopping list',
    'How much of the ingredient do you need to
buy?', '');
  // gets quantity of item from user
  redShopList.Lines.Add('');
  redShopList.Lines.Add(sItem + #9 + sQuantity);
  // adds item and quantity to shopping list
 	
  AssignFile(tFile, 'Shopping List.txt'); // assign
shopping list textfile to variable
  if fileexists('Shopping List.txt') = true then
  begin
    Append(tFile); // if shopping list textfile is
present, go to end of textfile
  end
  else
    Rewrite(tFile); // create new text file for
shopping list
  for iLineCounter := redShopList.Lines.Count
downto redShopList.Lines.Count -
    1 do // last line has most recently added item
and line before that contains space for ease of
readibility
  begin
    sLine := redShopList.Lines[iLineCounter];
    Writeln(tFile, sLine); // writes these lines to the
shopping list textfile so shopping list data is saved
  end;
  CloseFile(tFile);
end;
	Receive item and quantity to add to shopping list. Adjust both the application and the text file shopping list with newly added value

INPUT	PROCESS	OUTPUT
User clicks “Clear” button	Clears the shopping list richedit and text file contents and calls the Shop List Header procedure to add the heading	Clear richedit shopping list with dated heading on top

CODE	EXPLANATION

procedure TSnDHome.btnClearSLClick(Sender: TObject);
var
  tFile: TextFile;
 	
begin
  redShopList.Clear;
  ShopListHeader; // rewrites the shopping list heading to the richedit which is now empty
 	
  AssignFile(tFile, 'Shopping List.txt');
  Rewrite(tFile); // Clears text file
  CloseFile(tFile);
end;
	Clears richedit and text file. Calls ShopListHeader procedure

procedure TSnDHome.ShopListHeader;
var
  sDate: string;
 	
begin
  sDate := FormatDateTime('yyyymmdd', Now); // saves current date to variable sDate in format 'yyyymmdd'
  redShopList.Lines.Add('Shopping List ' + sDate + ' :'); // heading for shopping list which includes current date
  redShopList.SelStart := 0;
  redShopList.SelLength := 24; // selects heading text
  redShopList.SelAttributes.Style := redShopList.SelAttributes.Style + [fsBold]
    + [fsUnderline]; // makes heading text bold and underlined
end;
	Gets current date; adds date to “Shopping List” header text; sets header text to be bold and underlined on the richedit

INPUT	PROCESS	OUTPUT
User clicks “Save” button	Save dialog is opened for user to execute. Text in richedit is added to text file to be saved	User has a saved text file in their local storage

CODE	EXPLANATION

procedure TSnDHome.btnSaveSLClick(Sender: TObject);
var
  tShopList: TextFile;
  sDate: string;
 	
begin
  dlgSaveTxtFileShopList.Filter := 'Text file|*.txt';
  // filters save dialog to textfile formats
  dlgSaveTxtFileShopList.DefaultExt := 'txt'; // automatically sets file to be saved's extension as .txt
 	
  sDate := FormatDateTime('yyyymmdd', Now); // gets current date, formatted
  dlgSaveTxtFileShopList.FileName := 'Shopping List ' + sDate; // automatically sets textfiles name to be 'Shopping List' with the date afterwards
 	
  if dlgSaveTxtFileShopList.Execute then
  begin
    try
      redShopList.PlainText := true; // sets shopping list to plaintext to be saved
      AssignFile(tShopList, dlgSaveTxtFileShopList.FileName);
      Rewrite(tShopList); // textfile made
      Writeln(tShopList, redShopList.Text); // shopping list contents added to textfile
      CloseFile(tShopList);
      redShopList.PlainText := false; // set shopping list richedit back to formatted
      ShowMessage('File has successfully been saved');
    except
      ShowMessage('File save was cancelled'); // tell user that shopping list save has been cancelled
    end
  end;
 	
end;
	Limits save dialog to .txt files only and automatically sets the file name to be “Shopping List yyyymmdd.txt” with yyyymmdd replaced by the current date. File is either saved by user or dialog is closed

INPUT	PROCESS	OUTPUT
User clicks “Recipes” circle	Builds the appropriate form	Appropriate form is displayed

CODE	EXPLANATION

procedure TSnDHome.imgRecipesClick(Sender: TObject);
begin
  ViewRecipe;
end;
____
procedure TSnDHome.ViewRecipe;
begin
  SnDRecipes := TSnDRecipes.Create(self); // Create the form
  try
    SnDRecipes.ShowModal; // Show the form
  finally
    SnDRecipes.Free; // Free the form from memory
  end;
end;
	Calls ViewRecipe procedure which creates the appropriate form in memory

INPUT	PROCESS	OUTPUT
User clicks “Inventory” circle	Builds the appropriate form	Appropriate form is displayed

CODE	EXPLANATION

procedure TSnDHome.imgInventoryClick(Sender: TObject);
begin
  ViewInventory;
End;
____
procedure TSnDHome.ViewInventory;
begin
  SnDInventory := TSnDInventory.Create(self);
  try
    SnDInventory.ShowModal;
  finally
    SnDInventory.Free;
  end;
	Calls ViewInventory procedure which creates the appropriate form in memory

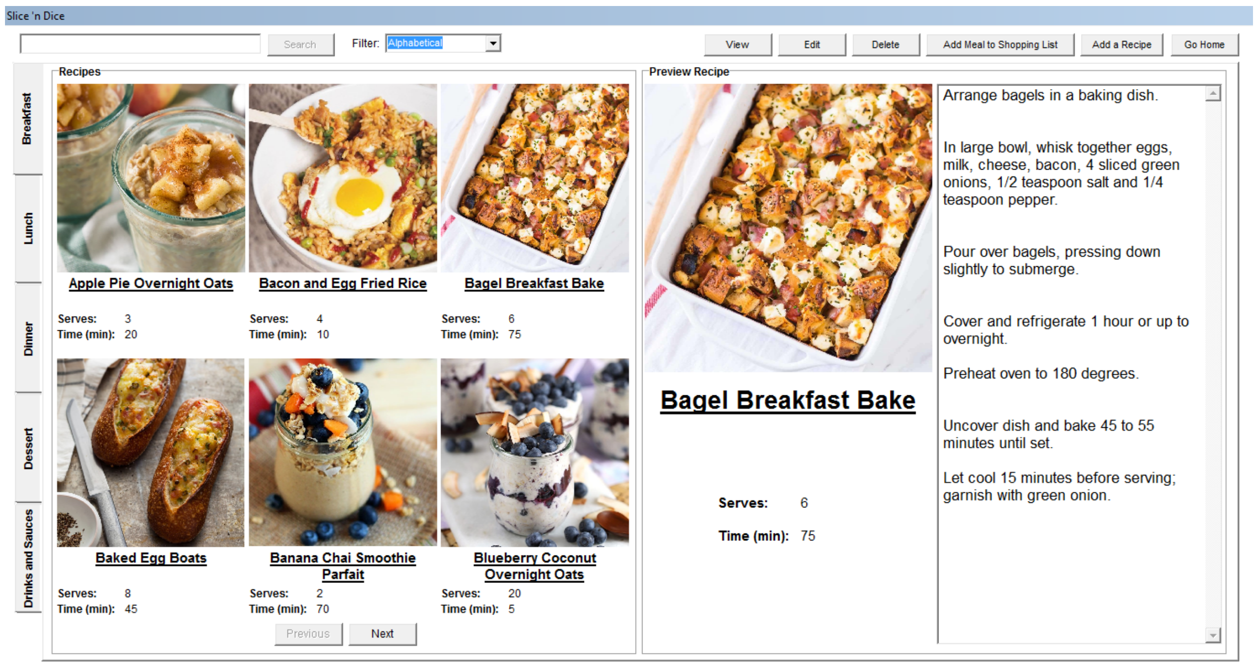
INPUT	PROCESS	OUTPUT
User clicks “Analytics” circle	Builds the appropriate form	Appropriate form is displayed

CODE	EXPLANATION

procedure TSnDHome.imgAnalyticsClick(Sender: TObject);
begin
  ViewAnalytics;
End;
--- 	
procedure TSnDHome.ViewAnalytics;
begin
  SnDAnalytics := TSnDAnalytics.Create(self);
  try
    SnDAnalytics.ShowModal;
  finally
    SnDAnalytics.Free;
  end;
end;
	Calls ViewAnalytics procedure which creates the appropriate form in memory

Recipe Viewer

Slice_n_Dice_u - SnDRecipes



Apple Pie Overnight Oats
Serves: 3
Time (min): 20

Bacon and Egg Fried Rice
Serves: 4
Time (min): 10

Bagel Breakfast Bake
Serves: 6
Time (min): 75

Baked Egg Boats
Serves: 8
Time (min): 45

Banana Chai Smoothie Parfait
Serves: 2
Time (min): 70

Blueberry Coconut Overnight Oats
Serves: 20
Time (min): 5

Bagel Breakfast Bake

Serves: 6

Time (min): 75

Arrange bagels in a baking dish.

In large bowl, whisk together eggs, milk, cheese, bacon, 4 sliced green onions, 1/2 teaspoon salt and 1/4 teaspoon pepper.

Pour over bagels, pressing down slightly to submerge.

Cover and refrigerate 1 hour or up to overnight.

Preheat oven to 180 degrees.

Uncover dish and bake 45 to 55 minutes until set.

Let cool 15 minutes before serving; garnish with green onion.

INPUT	PROCESS	OUTPUT
User selects tab of meal type	Database filter updates based on tab index	All recipes displayed are of the new filter type. Buttons become disabled because no recipe is selected and *Slice 'n Dice* logo is displayed in the preview panel

CODE	EXPLANATION

procedure
TSnDRecipes.tcRecipesChange(Sender:
TObject);
begin
  iPage := 0; // meal type change, therefore go to start of records
  btnPrevious.Enabled := false;
  btnNext.Enabled := true;
 	
  if tcRecipes.TabIndex = 0 then
  begin
 	
SnDHome.objDependencies.SetMealType('Breakfast'); // save meal type in dependencies unit for if user selects "Add" button to auto select meal type
    with dmSlicenDice do
    begin
      tblRecipe.Filter := 'RecType = ''Breakfast''';
      iRecordNumber := tblRecipe.RecordCount;
    end;
  end
  else if tcRecipes.TabIndex = 1 then
  begin
 	
SnDHome.objDependencies.SetMealType('Lunch');
    with dmSlicenDice do
    begin
      tblRecipe.Filter := 'RecType = ''Lunch''';
      iRecordNumber := tblRecipe.RecordCount;
    end;
  end
  else if tcRecipes.TabIndex = 2 then
  begin
 	
SnDHome.objDependencies.SetMealType('Dinner');
    with dmSlicenDice do
    begin
      tblRecipe.Filter := 'RecType = ''Dinner''';
      iRecordNumber := tblRecipe.RecordCount;
    end;
  end
	tcRecipesChange:
	The database will reset its filter to a new recipe type filter, therefore must load from the first item and the page index must be reset to 0. This means the "Next" button must be enabled and the "Previous" button disabled.
	Based on the tab index selected by the user:
	Set the meal type value in the objDependencies. This value is used to autofill the drop menu when a user adds a recipe.
	New database filter is set and the iRecordNumber index is set to the record count of the new filtered table. This is used to limit the application loading recipes beyond the count of recipes which would cause an error.
	When the new filter has been set, the application calls the NewRecipeSet procedure

```
  else if tcRecipes.TabIndex = 3 then
  begin

SnDHome.objDependencies.SetMealType('Desse
rt');
    with dmSlicenDice do
    begin
      tblRecipe.Filter := 'RecType = ''Dessert''';
      iRecordNumber := tblRecipe.RecordCount;
    end;
  end
  else if tcRecipes.TabIndex = 4 then
  begin

SnDHome.objDependencies.SetMealType('Drinks'
);
    with dmSlicenDice do
    begin
      tblRecipe.Filter := 'RecType = ''Drinks''';
      iRecordNumber := tblRecipe.RecordCount;
    end;
  end;

  NewRecipeSet;
end;
```

```
procedure TSnDRecipes.NewRecipeSet;
begin
  iImageCount := 0; // initialise image counter for
the six images
  NextRecipe(img1, lblName1, lblServesOut1,
lblTimeOut1);
  // load appropriate components with data
  arrKeys[1] := sID; // update array with RecIDs
  NextRecipe(img2, lblName2, lblServesOut2,
lblTimeOut2);
  arrKeys[2] := sID;
  NextRecipe(img3, lblName3, lblServesOut3,
lblTimeOut3);
  arrKeys[3] := sID;
  NextRecipe(img4, lblName4, lblServesOut4,
lblTimeOut4);
  arrKeys[4] := sID;
  NextRecipe(img5, lblName5, lblServesOut5,
lblTimeOut5);
  arrKeys[5] := sID;
  NextRecipe(img6, lblName6, lblServesOut6,
lblTimeOut6);
  arrKeys[6] := sID;
  HideDuplicateRecipes(img2, img3, img4, img5,
img6, lblName2, lblServes2,
    lblServesOut2, lblTime2, lblTimeOut2,
lblName3, lblServes3, lblServesOut3,
    lblTime3, lblTimeOut3, lblName4, lblServes4,
lblServesOut4, lblTime4,
```

NewRecipeSet:
This procedure loads each of the six recipes in the viewer with their appropriate information. This information is loaded with the NextRecipe procedure. The iImageCount counter is used in the NextRecipe procedure and must therefore be initialized to zero when calling the NewRecipeSet procedure.
Once all items have been loaded with the NextRecipe Procedure, the HideDuplicateRecipes procedure is called to hide any recipes in the catalog viewer which may be duplicates.
Finally the appropriate buttons are disabled because no recipe has been selected and the preview panel is replaced with the *Slice ‘n Dice* logo for an appealing look

```
   lblTimeOut4, lblName5, lblServes5,
lblServesOut5, lblTime5, lblTimeOut5,
   lblName6, lblServes6, lblServesOut6, lblTime6,
lblTimeOut6);
  // check for duplicates and hide if needs be

  btnView.Enabled := false; // nothing is selected
therefore disable components affecting selected
meals
  btnEdit.Enabled := false;
  btnAddShoppingList.Enabled := false;
  btnDelete.Enabled := false;
  grpbxPreview.Visible := false;
end;
```

```
procedure TSnDRecipes.NextRecipe(pImg:
TImage; pName, pServe, pTime: TLabel);
var
  sImageField, sName, sServe, sTime: string;

begin
  with dmSlicenDice do
  begin
    inc(iImageCount); // goes through the six meals
in viewer
    tblRecipe.RecNo := iImageCount + (iPage * 6);
// sets record to load based on image and page
count
    if tblRecipe.RecNo <= iRecordNumber then // if
record is still within number of records in database
    begin
      sImageField := tblRecipe['RecImage']; // load
image
      pImg.Picture.LoadFromFile

(ExpandFileName(ExtractFileDir(Application.ExeN
ame))
        + '/Recipe Images/' + sImageField);

      sName := tblRecipe['RecName']; // load name
      sID := tblRecipe['RecID'];
      pName.Caption := sName;

      sServe := tblRecipe['RecServe']; // load serve
      pServe.Caption := sServe;

      sTime := tblRecipe['RecCookTime']; // load
time
      pTime.Caption := sTime;
    end;
  end;
End;
```

NextRecipe:
Each of the six individual recipes in the catalog viewer are loaded with data using this procedure.
The iImageCount counter is used to determine which meal is being loaded with data and must therefore be incremented on each call. The appropriate record number to receive data is then determined by taking the iImageCount counter and adding it to the iPage counter which is multiplied by six (the six recipes viewable in the catalog). If the application is on page 1, iPage is 0 and will therefore result in nothing being added to the iImageCount variable.
The record number to be loaded is first checked against iRecordNumber which prevents the application loading data further than there are records available.
Appropriate variables are filled by the database and used to load the contents into the viewer. The image component receives the file name from the database and then loads the appropriate image from the folder of images contained with the application

---

```
procedure
TSnDRecipes.HideDuplicateRecipes(pImg,
```

HideDuplicateRecipes:
The array with all recipe IDs is checked value x

```
pImg2, pImg3, pImg4,
  pImg5: TImage; pName, pServeOut, pServe,
pTimeOut, pTime, pName2,
  pServeOut2, pServe2, pTimeOut2, pTime2,
pName3, pServeOut3, pServe3,
  pTimeOut3, pTime3, pName4, pServeOut4,
pServe4, pTimeOut4, pTime4, pName5,
  pServeOut5, pServe5, pTimeOut5, pTime5:
TLabel);
// compares current RecID to next meal's RecID in
array containing RecIDs.  If duplicate - hide all
components for meals after the first one else show
all components because not a duplicate
begin
  if arrKeys[2] = arrKeys[1] then
  begin
    pImg.Visible := false;
    pName.Visible := false;
    pServeOut.Visible := false;
    pServe.Visible := false;
    pTimeOut.Visible := false;
    pTime.Visible := false;
  end
  else
  begin
    pImg.Visible := true;
    pName.Visible := true;
    pServeOut.Visible := true;
    pServe.Visible := true;
    pTimeOut.Visible := true;
    pTime.Visible := true;
  end;

  if arrKeys[3] = arrKeys[2] then
  begin
    pImg2.Visible := false;
    pName2.Visible := false;
    pServeOut2.Visible := false;
    pServe2.Visible := false;
    pTimeOut2.Visible := false;
    pTime2.Visible := false;
  end
  else
  begin
    pImg2.Visible := true;
    pName2.Visible := true;
    pServeOut2.Visible := true;
    pServe2.Visible := true;
    pTimeOut2.Visible := true;
    pTime2.Visible := true;
  end;

  if arrKeys[4] = arrKeys[3] then
  begin
    pImg3.Visible := false;
```
	against x+1 for a duplicate value. If these values are equal, the appropriate components on the viewer catalog are made invisible else they are made visible

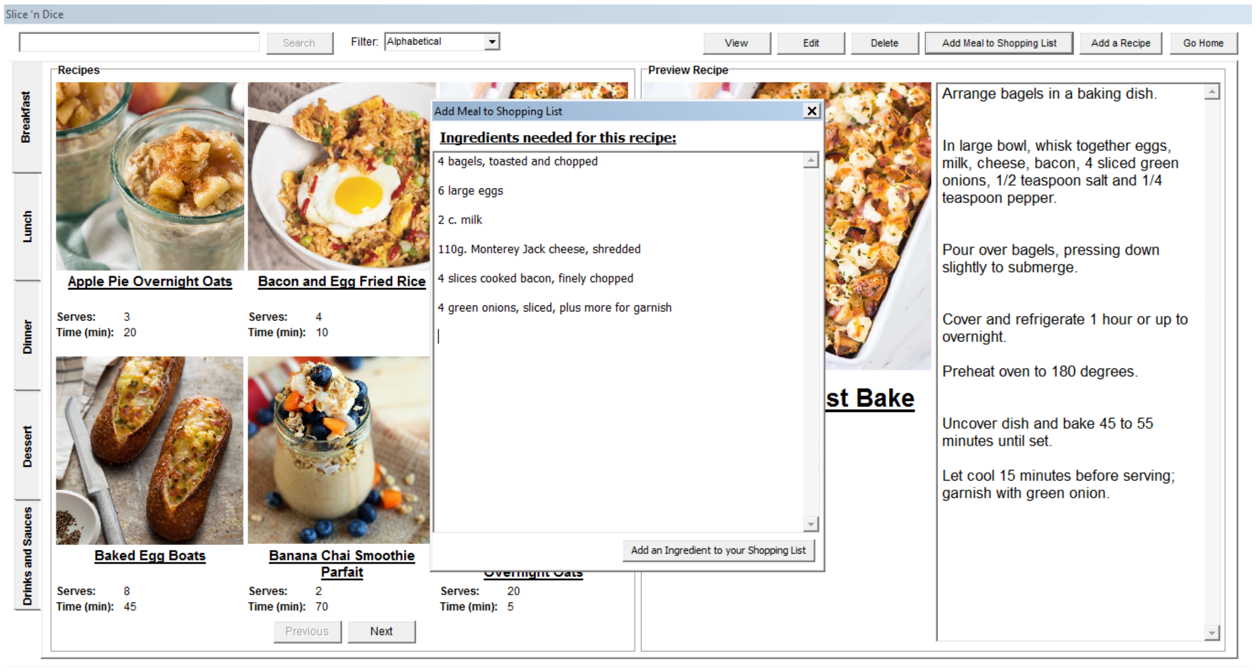
```
    pName3.Visible := false;
    pServeOut3.Visible := false;
    pServe3.Visible := false;
    pTimeOut3.Visible := false;
    pTime3.Visible := false;
  end
  else
  begin
    pImg3.Visible := true;
    pName3.Visible := true;
    pServeOut3.Visible := true;
    pServe3.Visible := true;
    pTimeOut3.Visible := true;
    pTime3.Visible := true;
  end;

  if arrKeys[5] = arrKeys[4] then
  begin
    pImg4.Visible := false;
    pName4.Visible := false;
    pServeOut4.Visible := false;
    pServe4.Visible := false;
    pTimeOut4.Visible := false;
    pTime4.Visible := false;
  end
  else
  begin
    pImg4.Visible := true;
    pName4.Visible := true;
    pServeOut4.Visible := true;
    pServe4.Visible := true;
    pTimeOut4.Visible := true;
    pTime4.Visible := true;
  end;

  if arrKeys[6] = arrKeys[5] then
  begin
    pImg5.Visible := false;
    pName5.Visible := false;
    pServeOut5.Visible := false;
    pServe5.Visible := false;
    pTimeOut5.Visible := false;
    pTime5.Visible := false;
  end
  else
  begin
    pImg5.Visible := true;
    pName5.Visible := true;
    pServeOut5.Visible := true;
    pServe5.Visible := true;
    pTimeOut5.Visible := true;
    pTime5.Visible := true;
  end;
end;
```

INPUT	PROCESS	OUTPUT
“Next” or “Previous” button is clicked by user	Increment the page counter and check whether or not the “Next” or “Previous” button must be disabled and enable or disabled respectively. Also reload the catalog viewer with the new recipe data	New set of six recipes are loaded into the catalog viewer and the “Next” or “Previous” button is either disabled or enabled

INPUT	PROCESS	OUTPUT
User types into search bar	Database is searched based on the text in the search bar	dbGrid is displayed underneath the search bar showing results related to the text in the search bar
“Search” button is clicked	Determine the RecID of the searched recipe	Selected recipe is opened in the view form

INPUT	PROCESS	OUTPUT
Filter drop menu is clicked	User selects new filter and filter is applied to database	New recipe set is loaded in the catalog viewer based on the new filter

INPUT	PROCESS	OUTPUT
“Add Meal to Shopping List” button is clicked	“Add Meal to Shopping List” form is created and the ingredients which connect to the appropriate selected recipe are loaded	“Add Meal to Shopping List” form is shown to user as output



Serves: 3
Time (min): 20

Serves: 4
Time (min): 10

Serves: 8
Time (min): 45

Serves: 2
Time (min): 70

Serves: 20
Time (min): 5

Ingredients needed for this recipe:

4 bagels, toasted and chopped

6 large eggs

2 c. milk

110g. Monterey Jack cheese, shredded

4 slices cooked bacon, finely chopped

4 green onions, sliced, plus more for garnish

Arrange bagels in a baking dish.

In large bowl, whisk together eggs, milk, cheese, bacon, 4 sliced green onions, 1/2 teaspoon salt and 1/4 teaspoon pepper.

Pour over bagels, pressing down slightly to submerge.

Cover and refrigerate 1 hour or up to overnight.

Preheat oven to 180 degrees.

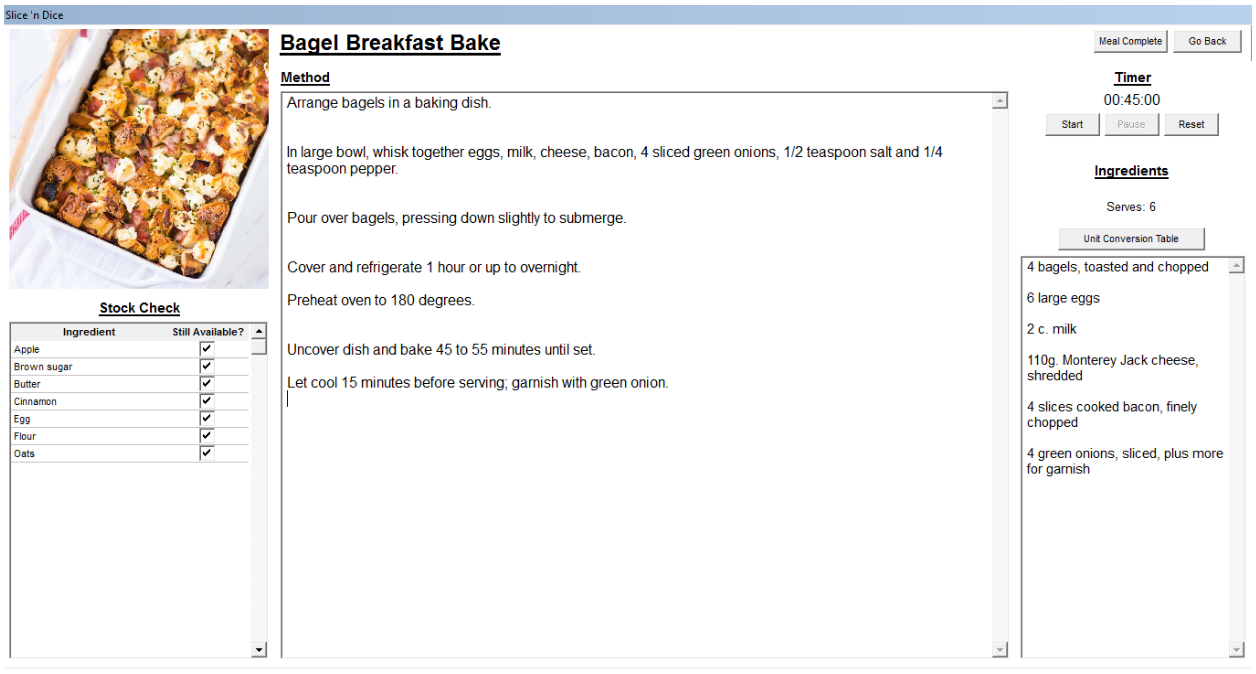
Uncover dish and bake 45 to 55 minutes until set.

Let cool 15 minutes before serving; garnish with green onion.

INPUT	PROCESS	OUTPUT
"Add Ingredient to Shopping List" button is clicked	Calls ingredient adding to shopping list procedure from home screen	Shopping list is updated

Viewing a Recipe

Slice_n_Dice_View_u - SelectedRecipe



Bagel Breakfast Bake

Stock Check

Ingredient	Still Available?
Apple	✓
Brown sugar	✓
Butter	✓
Cinnamon	✓
Egg	✓
Flour	✓
Oats	✓

Method

Arrange bagels in a baking dish.

In large bowl, whisk together eggs, milk, cheese, bacon, 4 sliced green onions, 1/2 teaspoon salt and 1/4 teaspoon pepper.

Pour over bagels, pressing down slightly to submerge.

Cover and refrigerate 1 hour or up to overnight.

Preheat oven to 180 degrees.

Uncover dish and bake 45 to 55 minutes until set.

Let cool 15 minutes before serving; garnish with green onion.

Timer

00:45:00

Ingredients

Serves: 6

4 bagels, toasted and chopped

6 large eggs

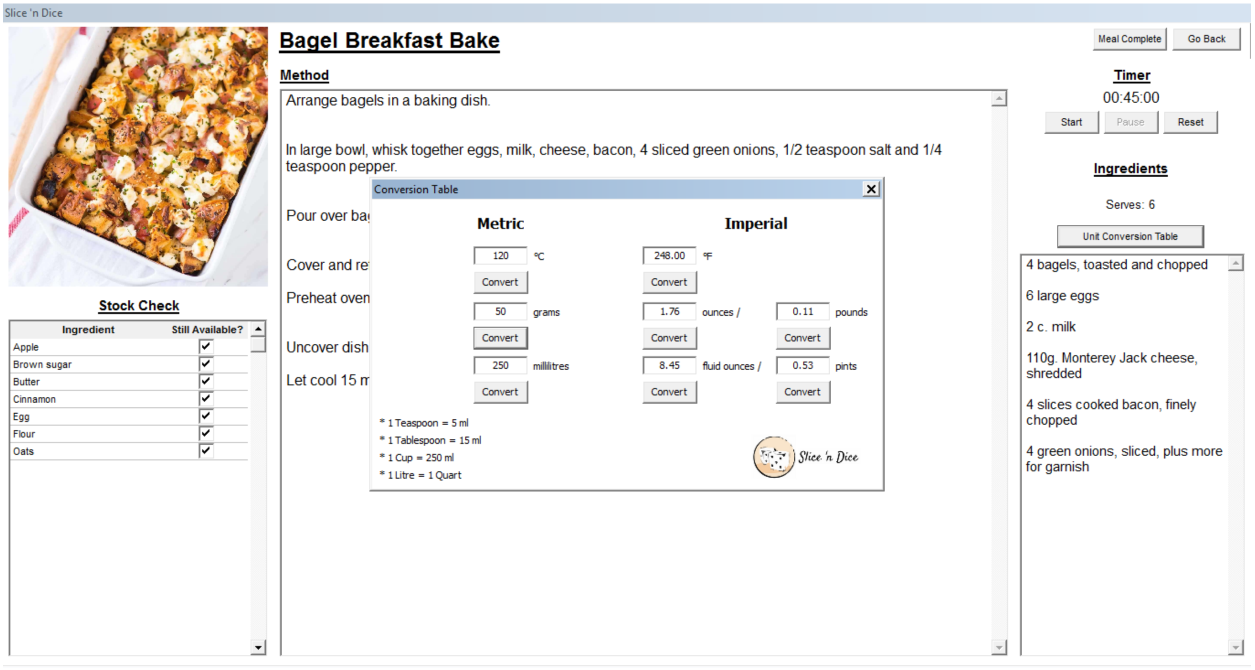
2 c. milk

110g. Monterey Jack cheese, shredded

4 slices cooked bacon, finely chopped

4 green onions, sliced, plus more for garnish

INPUT	PROCESS	OUTPUT
User clicks “Start” button	Timer component is enabled	Timer displayed to user is updated based on timer component
User clicks “Stop” button	Timer component is disabled	Timer displayed to user is frozen based on timer component
User clicks “Reset” button	Timer component is updated with new timer interval	Timer displayed to user is updated based on timer component
User clicks “Unit Conversion Table” button	“Unit Conversion Table” form is created	“Unit Conversion Table” is shown to user as pop up
User clicks “Meal Complete” button	Current date and meal name is logged	“View” form is closed and data is logged in database


Slice 'n Dice

Bagel Breakfast Bake

Method

Arrange bagels in a baking dish.

In large bowl, whisk together eggs, milk, cheese, bacon, 4 sliced green onions, 1/2 teaspoon salt and 1/4 teaspoon pepper.

Pour over ba

Cover and re

Preheat oven

Uncover dish

Let cool 15 m

Timer

00:45:00

Ingredients

Serves: 6

4 bagels, toasted and chopped

6 large eggs

2 c. milk

110g. Monterey Jack cheese, shredded

4 slices cooked bacon, finely chopped

4 green onions, sliced, plus more for garnish

Stock Check

Ingredient	Still Available?
Apple	✓
Brown sugar	✓
Butter	✓
Cinnamon	✓
Egg	✓
Flour	✓
Oats	✓

Conversion Table

Metric		Imperial			
120	°C	248.00	°F		
50	grams	1.76	ounces /	0.11	pounds
250	millilitres	8.45	fluid ounces /	0.53	pints

* 1 Teaspoon = 5 ml

* 1 Tablespoon = 15 ml

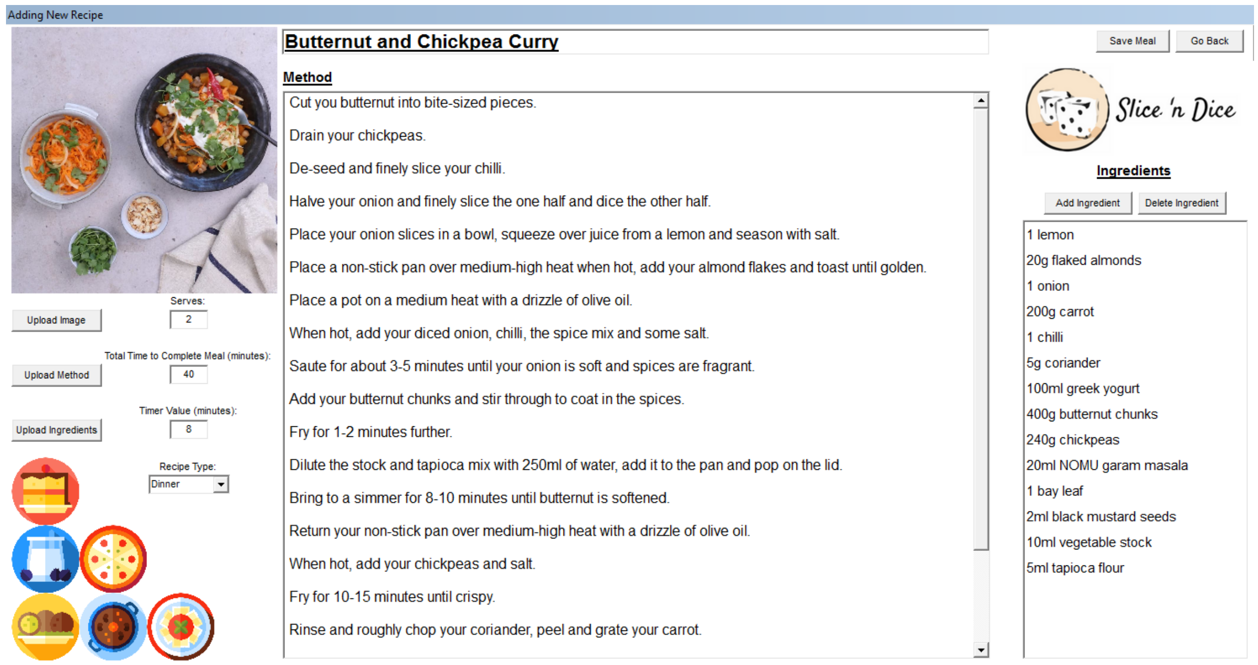
* 1 Cup = 250 ml

* 1 Litre = 1 Quart

INPUT	PROCESS	OUTPUT
"Convert" button is clicked	Math is done to calculate new value in imperial or metric form depending on which was entered by user	Imperial or metric unit is updated based on user's entered data to convert

Adding or Editing a Recipe

Slice_n_Dice_Add_OR_Edit_u - AddOREditRecipe



Butternut and Chickpea Curry

Method

Cut you butternut into bite-sized pieces.

Drain your chickpeas.

De-seed and finely slice your chilli.

Halve your onion and finely slice the one half and dice the other half.

Place your onion slices in a bowl, squeeze over juice from a lemon and season with salt.

Place a non-stick pan over medium-high heat when hot, add your almond flakes and toast until golden.

Place a pot on a medium heat with a drizzle of olive oil.

When hot, add your diced onion, chilli, the spice mix and some salt.

Saute for about 3-5 minutes until your onion is soft and spices are fragrant.

Add your butternut chunks and stir through to coat in the spices.

Fry for 1-2 minutes further.

Dilute the stock and tapioca mix with 250ml of water, add it to the pan and pop on the lid.

Bring to a simmer for 8-10 minutes until butternut is softened.

Return your non-stick pan over medium-high heat with a drizzle of olive oil.

When hot, add your chickpeas and salt.

Fry for 10-15 minutes until crispy.

Rinse and roughly chop your coriander, peel and grate your carrot.

Serves: 2

Total Time to Complete Meal (minutes): 40

Timer Value (minutes): 8

Recipe Type: Dinner

Ingredients

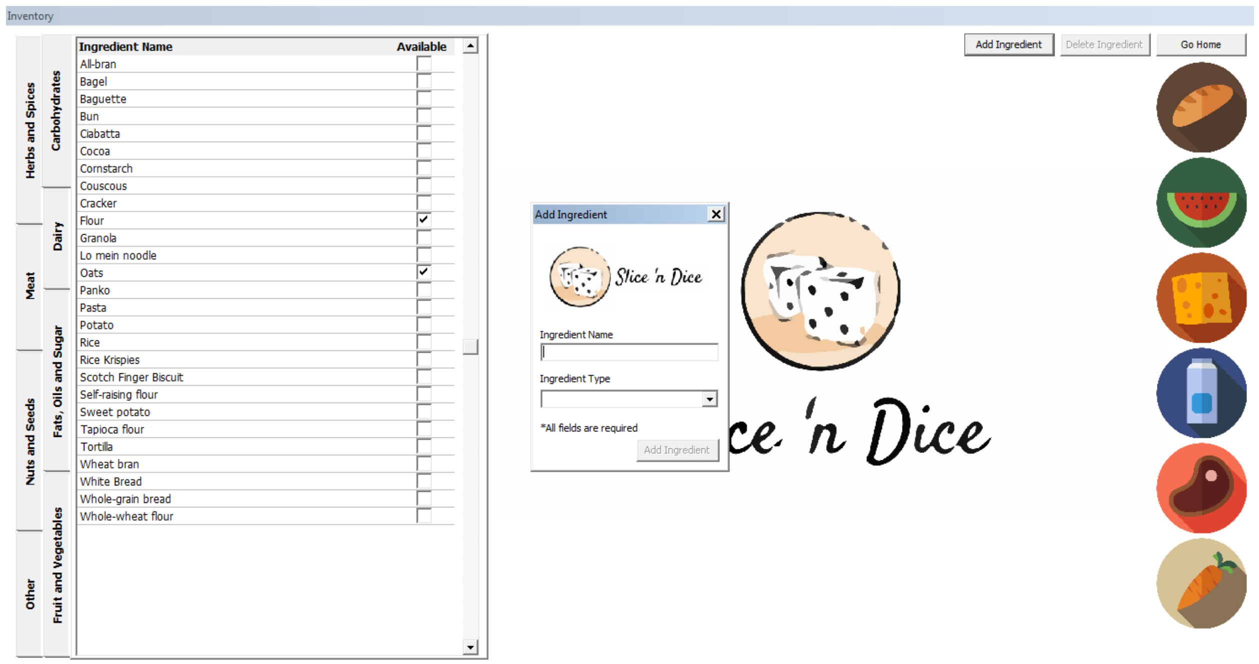
1 lemon
20g flaked almonds
1 onion
200g carrot
1 chilli
5g coriander
100ml greek yogurt
400g butternut chunks
240g chickpeas
20ml NOMU garam masala
1 bay leaf
2ml black mustard seeds
10ml vegetable stock
5ml tapioca flour

The same form is used for adding or editing a recipe. The application knows whether the user is adding or editing a recipe from the objDependencies. If the user is editing a recipe, the add form has the fields filled in with the appropriate data from the database.

INPUT	PROCESS	OUTPUT
User clicks on any of the "Upload" buttons	Upload dialogs are created and displayed for user to select appropriate files	Appropriate file is uploaded to form application
"Add Ingredient" button is clicked	Dialog's displayed for user to enter new ingredient text	New ingredient text is added to the end of the ingredient listbox
"Delete Ingredient" button is clicked	Listbox finds which item is selected to be deleted, then the item(s) is/are deleted	Ingredient listbox is updated with appropriate data
"Save Meal" button is clicked	Data in this application form is written to the database to either update, insert or delete values that are in the database	User is shown dialog to tell them that the meal has been saved and the form closes
"Go Back" button is clicked	User confirmation about going back and that doing so discards all written data	Form is either closed and data is discarded or form remains open

Inventory

Slice_n_Dice_Inventory_u - SnDInventory

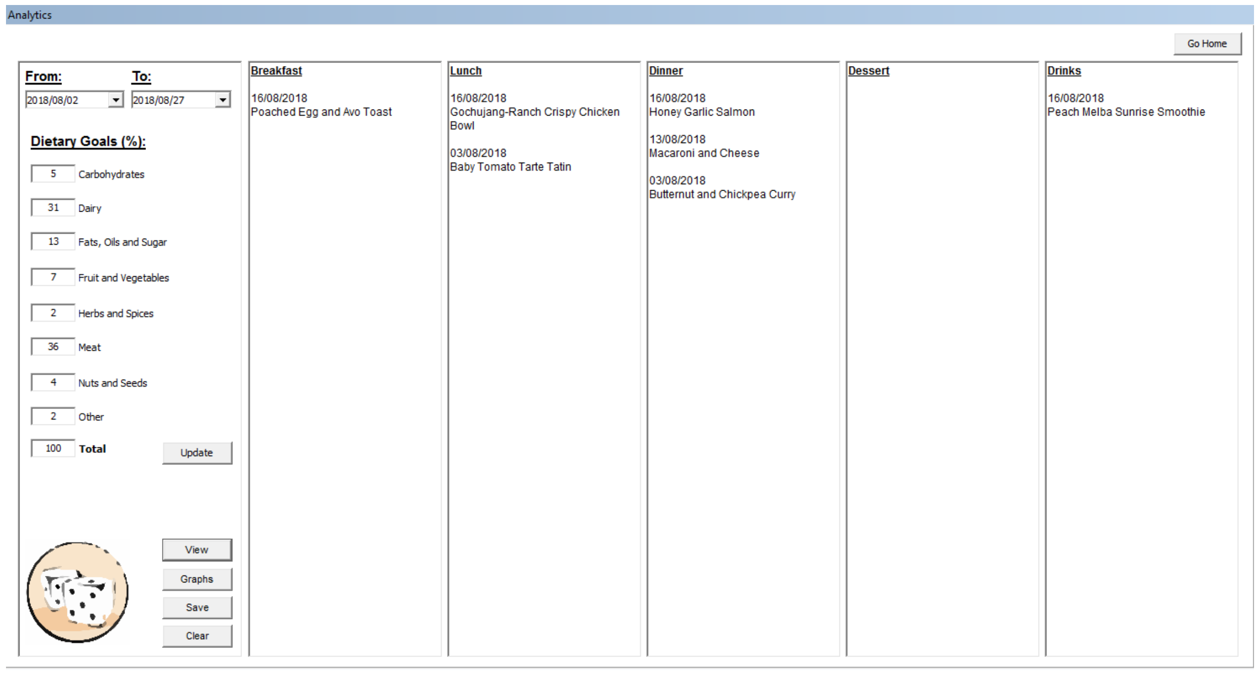


Ingredient Name	Available
All-bran	☐
Bagel	☐
Baguette	☐
Bun	☐
Ciabatta	☐
Cocoa	☐
Cornstarch	☐
Couscous	☐
Cracker	☐
Flour	✓
Granola	☐
Lo mein noodle	☐
Oats	✓
Panko	☐
Pasta	☐
Potato	☐
Rice	☐
Rice Krispies	☐
Scotch Finger Biscuit	☐
Self-raising flour	☐
Sweet potato	☐
Tapioca flour	☐
Tortilla	☐
Wheat bran	☐
White Bread	☐
Whole-grain bread	☐
Whole-wheat flour	☐

INPUT	PROCESS	OUTPUT
User changes tab on tab control	Database is re-filtered based on tab index	Ingredient data displayed is updated based on filter
User selects the “Available” checkbox	Check if ingredient is set as available or unavailable	Change value in application and database to opposite of what it was
“Add Ingredient” button is clicked	Create “Add Ingredient” form	“Add Ingredient” form is displayed
Both fields have data entered and user clicks the “Add Ingredient” button	Database is updated with new ingredient	Application is refreshed to display newly added ingredient; fields are cleared for user to enter another new ingredient
“Delete Ingredient” button clicked	Check what ingredient is selected and find it in the database to delete it	Delete the selected ingredient and update the values shown in the application

Analytics

Slice_n_Dice_Analytics_u - SnDAnalytics



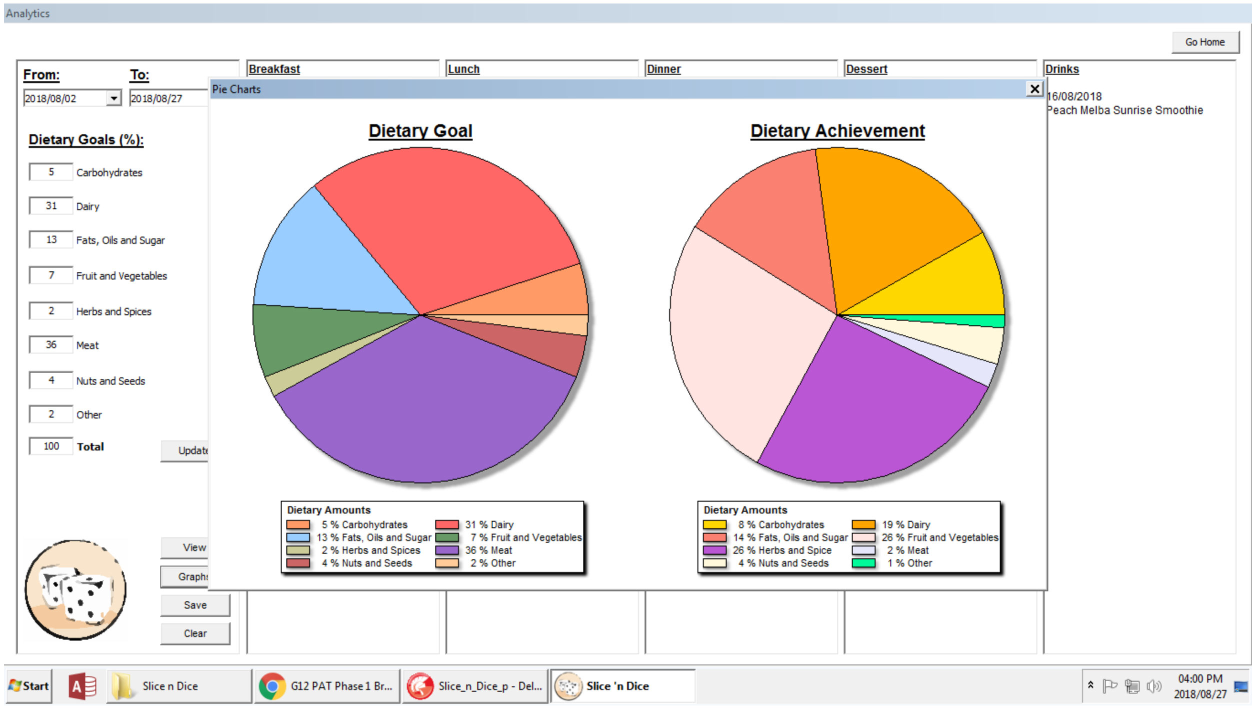
From: 2018/08/02 To: 2018/08/27

Dietary Goals (%):

5	Carbohydrates
31	Dairy
13	Fats, Oils and Sugar
7	Fruit and Vegetables
2	Herbs and Spices
36	Meat
4	Nuts and Seeds
2	Other
100	Total

Breakfast	Lunch	Dinner	Dessert	Drinks
16/08/2018 Poached Egg and Avo Toast	16/08/2018 Gochujang-Ranch Crispy Chicken Bowl	16/08/2018 Honey Garlic Salmon		16/08/2018 Peach Melba Sunrise Smoothie
	03/08/2018 Baby Tomato Tarte Tatin	13/08/2018 Macaroni and Cheese		
		03/08/2018 Butternut and Chickpea Curry		

INPUT	PROCESS	OUTPUT
User select “to” and “from” date	Checks if valid, logical dates are selected	Functions are enabled or disabled based on if logical dates were selected by the user
User changes values in the “Dietary Goal” section	Adds values in the input fields	“Total” field is updated to show current “Dietary Goal” count
“Update” button is clicked	Dietary goal values are updated in database and array	Pop up confirms with user about the updated values
“View” button is clicked	Goes through database checking for values which lie in the “to” and “from” dates selected by the user	Columns on the right are updated with recipe names based on “to” and “from” dates
“Save” button is clicked	Loads the data in columns into a text file to be saved	Confirms with user about text file save



From: 2018/08/02 To: 2018/08/27

16/08/2018
Peach Melba Sunrise Smoothie

Dietary Goals (%):

5	Carbohydrates
31	Dairy
13	Fats, Oils and Sugar
7	Fruit and Vegetables
2	Herbs and Spices
36	Meat
4	Nuts and Seeds
2	Other
100	**Total**

Dietary Goal

Dietary Amounts
5 % Carbohydrates, 31 % Dairy
13 % Fats, Oils and Sugar, 7 % Fruit and Vegetables
2 % Herbs and Spices, 36 % Meat
4 % Nuts and Seeds, 2 % Other

Dietary Achievement

Dietary Amounts
8 % Carbohydrates, 19 % Dairy
14 % Fats, Oils and Sugar, 26 % Fruit and Vegetables
26 % Herbs and Spice, 2 % Meat
4 % Nuts and Seeds, 1 % Other

INPUT	PROCESS	OUTPUT
User views graphs by clicking “Graphs” button	Goes through database to search for appropriate values and updates array of ingredient type which counts the amount of an ingredient type used in creating the pie chart “Dietary Achievement”. “Dietary Goal” pie chart gets values directly from a table in the database	Pie charts are displayed with appropriate values