

Design

Table of Content

1. Overall algorithm
 2. Databases and Tables
 3. Class, UML Diagram and UML Relationships
 4. User Interface rough draft
 5. Data Structures
 6. Flow Chart
 7. Test Plan
 8. Amendments
 - a. Databases and Tables
 - b. Class, UML Diagram and UML Relationships
 - c. User Interface final draft
 - d. Data Structures
-

Overall algorithm

- “Welcome” Home class is open, there are “Input dishes” button, “Daily input” button, “Prediction” button, “View Data” button.
- Option in Jmenu for program instruction such as FAQ and Help, and a home option for user to return home

if (“Input dishes” is pressed)

{

4 radio buttons will appear: Dishes, Desserts, Salads and Fruits, Drink

1 return button for the user to return to the previous frame;

if (“Dishes” is checked) {

Appear 4 text fields: Name, main ingredient, sauce, other ingredients;

An inserted button for user to insert data;

1 return button for the user to return to the previous frame;

} **if** (“Desserts” is checked) {

Appear 2 text fields: Name, sauce;

An inserted button for user to insert data;

1 return button for the user to return to the previous frame;

} **if** (“Salads and Fruits” is checked) {

Appear 3 text fields: Name, ingredient 1, ingredient 2;

An inserted button for user to insert data;

1 return button for the user to return to the previous frame;

} **if** (“Drinks” is checked) {

Appear 1 text fields: Name;

An inserted button for user to insert data;

1 return button for the user to return to the previous frame;

}

}

if (“Daily input” is pressed) {

Appear scroll panel with dishes, 2 text fields: amount of food left and amount of food made;

Option for user to input amount of food left and amount of food made of dishes;

Feature illogical input detection;

1 return button for the user to return to the previous frame;

}

if (“Prediction” is pressed) {

Appear 3 buttons: Predict food, Predict ingredient, Desired outcome;

1 return button for the user to return to the previous frame;

if (“predict food” is pressed) {

Display the dish that has the highest ratio of eaten, and the frequency of of the ratio;

```

        Display the dish that has the highest high ratio frequency;
        1 return button for the user to return to the previous frame;
    }
    if ("Prediction ingredient" is pressed) {
        Display the ingredients that has highest ratio of eaten, and the frequency
        of the ratio;
        Display the ingredients that has the highest high ratio frequency;
        1 return button for the user to return to the previous frame;
    }
    if ("Desired outcome" is pressed) {
        Appear 1 text field for user to enter the desired outcome/ ratio.;
        Display all dishes that have ratio equal or greater than desired ratio;
        1 return button for the user to return to the previous frame;
    }
}
if ("View data" is pressed) {
    Display 4 Jtable for Dishes , Desserts, Salads and Fruits, Drinks;
    Each table shows the data that is stored in the database of each category.
    Option for user to change, delete, manipulate the data in the table;
    1 return button for the user to return to the previous frame;
}
if ("Help" is pressed) {
    Option to choose a document of the programme or a video tutorial, or images
    procedures;
    1 return button for the user to return to the previous frame;
}
if ("FAQ" is pressed)
{
    Appear scroll panel of common questions about the programme along with
    solution/ answer for the question when the user chooses it;
    1 return button for the user to return to the previous frame;
}
if ("Home" is pressed) {
    The program returns to home screen;
}

```

Database and Tables

Unnormalized Database

Name (varchar)	Type (varchar)	Ingredients (varchar)	Sauces or spices (varchar)	Ratio (int)
	Entree			
	Desert	none		
	Salad			
	Drink	none	none	
	fruits		none	

Normalized Database

“Type: Entree” table for main ingredient

Entree’s name (varchar)	Ingredient (varchar)

“Type: Entree” table for Sauces or spices

Entree’s name (varchar)	Sauce or spice (varchar)

“Type: Entree” table for ratio

Entree’s name (varchar)	Ratio (int)

“Type: Dessert” table for sauces or spices

Dessert’s name (varchar)	Sauces or spices (varchar)

“Type: Dessert” table for ratio

Dessert’s name (varchar)	Ratio (int)

“Type: Salad” table for Ingredient

Salad's name (varchar)	Ingredient (varchar)

“Type: Salad” table for sauce or spice

Salad's name (varchar)	Sauce or spice (varchar)

“Type: Salad” table for ratio

Salad's name (varchar)	Ratio (int)

“Type: Drink” table for ratio

Drink's name (varchar)	Ratio (int)

“Type: Fruit” table for ingredient

Fruit's name (varchar)	Ingredient (varchar)

“Type: Fruit” table for ratio

Fruit's name (varchar)	Ratio (int)

Ingredient Table

Ingredient (varchar)	Ratio (int)

Spices and Sauce table

Spices and Sauce (varchar)	Ratio (int)

Classes & UML diagrams

Class Table

Class name	Description
Welcome	Welcome screen provides the user about the information of the program.
Home	Home frame where the user can click buttons to access different frames/functions of the program.
Error	Warn the user that he or she has entered illogical data.
SystemError	Warn the user that the system has run into problems within the internal calculation of the system.
Help	Provide the user with different type of helps/tutorials about the program
Documentation	Provide the user with documentation of the program, the frame also includes pictures of procedures/functionality of different buttons and frames.
Video	Provide users with a video tutorial of the program.
FAQ	Provide the user with frequently asked questions, questions about common error, and answers to all the questions.
NewDish	The user can input dishes alongside with what type of dishes it is, and ingredients/sauces inside the dishes.
Input	The user can select the dishes that are imputed in the system. Then, the user can input the amount of food made and the amount of food taken on that day.
Data	The user can view the data that they have inputted.
CalculateRatio	Computational class to calculate the ratio between amount of food made and the amount of food taken when the user input the data.

CheckRatio	The user can insert the ratio that he or she desired, and the system will display all the dishes that have met the ratio.
RatioMeet	Computational class to scan all the dishes inside the database to see which dishes have the ratio that is equal or higher than the user's desired ratio.
Prediction	The user can choose the dishes that he or she wants. Then the program will display the predicted ratio
CalculatePrediction	Computational class of calculating the probability of a dish will be successful based on the ingredients that are in the dish.
Rank	Provide users with the overall view of the top items of each categorization.
TableSorter	Computation class to sort out the order of food, the order is based on the highest ratio to the lowest in each categorize.
IngrdientRank	Provide users with the rank of the ingredients based on their ratio from high to low
EntreeRank	Provide users with the rank of the dishes/entrees based on their ratio from high to low
SaladRank	Provide users with the rank of the Salads based on their ratio from high to low
FruitRank	Provide users with the rank of the fruits based on their ratio from high to low
DrinkRank	Provide users with the rank of the drinks based on their ratio from high to low
SauceRank	Provide users with the rank of the sauces based on their ratio from high to low
InstallDatabase	Insert SQL Database once
JavaDatabase	Allows for SQL Databases accesses

UML DIAGRAM

Welcome
- GUI components
+ actionPerformed(ActionEvent e): void + Welcome():

Home
- GUI components
+ actionPerformed(ActionEvent e): void + home():

Error
- GUI components
+ actionPerformed(ActionEvent e): void + Error():

SystemError
- GUI components
+ actionPerformed(ActionEvent e): void + SystemError():

Help
- GUI components
+ actionPerformed(ActionEvent e): void + Help():

Documentation
- GUI components

- + actionPerformed(ActionEvent e): void
- + Documentation():

Video

- GUI components
- + actionPerformed(ActionEvent e): void
- + New Video();

FAQ

- GUI components
- + actionPerformed(ActionEvent e): void
- + FAQ():

NewDish

- GUI components
- Entree: boolean
- Desert: boolean
- Salad: boolean
- Drink: boolean
- Fruit: boolean
- + Entree: String[] {Name, ingredient, sauce, ratio}
- + Desert: String[] {Name, sauce}
- + Salad: String[] {Name, ingredient, sauce, ratio}
- + Drink: String[] {Name, ratio}
- + Fruit: String[] {Name, ingredient}
- + actionPerformed(ActionEvent e): void
- + NewDish():

Input

- GUI components
- FoodMade: float
- FoodLeft: float
- + actionPerformed(ActionEvent e): void

+ Input():

Data

- GUI components
- JTable

- + actionPerformed(ActionEvent e): void
- + New Data();

CalculateRatio

- foodMade: float
- foodLeft: float
- fatio: float

- + Ratio(float FoodMade, float FoodLeft):
- + setFoodMade(float FoodMade): void
- + setFoodLeft(float FoodLeft): void
- + getFoodMade(): float
- + getFoodLeft(): float

CheckRatio

- GUI components
- + desiredRatio: float

- + actionPerformed(ActionEvent e): void
- + CheckRatio():

RatioMeet

- desiredRatio: float
- checkedRatio: float
- ratio: float
- + meetRatio: String[] {Name}
- + notMeetRatio: String[] {Name}

- + RatioMeet(float desiredRatio):
- + setRatioMeet(string[] meetRatio): void
- + setRatioNotMeet(string[] notMeetRatiot): void
- + getRatioMeet();
- + getRatioNotMeet();

Prediction
<ul style="list-style-type: none"> - GUI components
<ul style="list-style-type: none"> + actionPerformed(ActionEvent e): void + Prediction():

CalculatePrediction
<ul style="list-style-type: none"> - probability: float - check: boolean - standardDeviation: float - dishName: string - dishRatioSum: string[] {Name, ingredientRatio, sauceRatio }
<ul style="list-style-type: none"> - standardDeviation(string[] dishRationSum) + probability(string dishName, standardDeviation)

Rank
<ul style="list-style-type: none"> - GUI components - IngredientRank: string[] - EntreeRank: string[] - SaladRank: string[] - FruitRank: string [] - DrinkRank: string[] - SauceRank: string[]
<ul style="list-style-type: none"> + actionPerformed(ActionEvent e): void + Rank():

TableSorter
<ul style="list-style-type: none"> - targetSort: string - entree: boolean - ingredient: boolean - dish: boolean - salad: boolean - fruit: boolean - drink: boolean - nameRatio: string[] {Name, ratio} - allRatioC: ArrayList<>

<ul style="list-style-type: none"> - allRatioA: string[] {nameRatio,..} - rank: string[]
<ul style="list-style-type: none"> - ArrayListToArray(ArrayList<> arrayList); + RankSort(string targetSort); + setRankSort(string[] rank): void + getRankSort()

IngredientRank
<ul style="list-style-type: none"> - IngredientRatio: string[] {Ingredient, ratio} - allRatioC: ArrayList<> - allRatioA: string[] {nameRatio,..} - IngrdientRank: string[]
<ul style="list-style-type: none"> - ArrayListToArray(ArrayList<> arrayList); + IngredientRank(): + setIngredientRank(string[] IngrdientRank): void + getIngredientRank()

EntreeRank
<ul style="list-style-type: none"> - EntreeRatio: string[] {Entree, ratio} - allRatioC: ArrayList<> - allRatioA: string[] {nameRatio,..} - EntreeRank: string[]
<ul style="list-style-type: none"> - ArrayListToArray(ArrayList<> arrayList); + EntreeRank(): + setEntreeRank(string[] EntreeRank): void + getEntreeRank()

SaladRank
<ul style="list-style-type: none"> - SaladRatio: string[] {Salad, ratio} - allRatioC: ArrayList<> - allRatioA: string[] {nameRatio,..} - SaladRank: string[]
<ul style="list-style-type: none"> - ArrayListToArray(ArrayList<> arrayList); + SaladRank(): + setSaladRank(string[] SaladRank): void + getSaladRank()

FruitRank

- FruitRatio: string[] {Fruit, ratio}
 - allRatioC: ArrayList<>
 - allRatioA: string[] {nameRatio,..}
 - FruitRank: string[]
-
- ArrayListToArray(ArrayList<> arrayList);
 - + FruitRank():
 - + setFruitRank(string[] FruitRank): void
 - + getFruitRank()

DrinkRank

- DrinkRatio: string[] {Drink, ratio}
 - allRatioC: ArrayList<>
 - allRatioA: string[] {nameRatio,..}
 - DrinkRank: string[]
-
- ArrayListToArray(ArrayList<> arrayList);
 - + DrinkRank():
 - + setDrinkRank(string[] DrinkRank): void
 - + getDrinkRank()

SauceRank

- SauceRatio: string[] {Sauce, ratio}
 - allRatioC: ArrayList<>
 - allRatioA: string[] {nameRatio,..}
 - SauceRank: string[]
-
- ArrayListToArray(ArrayList<> arrayList);
 - + SauceRank():
 - + setSauceRank(string[] SauceRank): void
 - + getSauceRank()

InstallDatabase

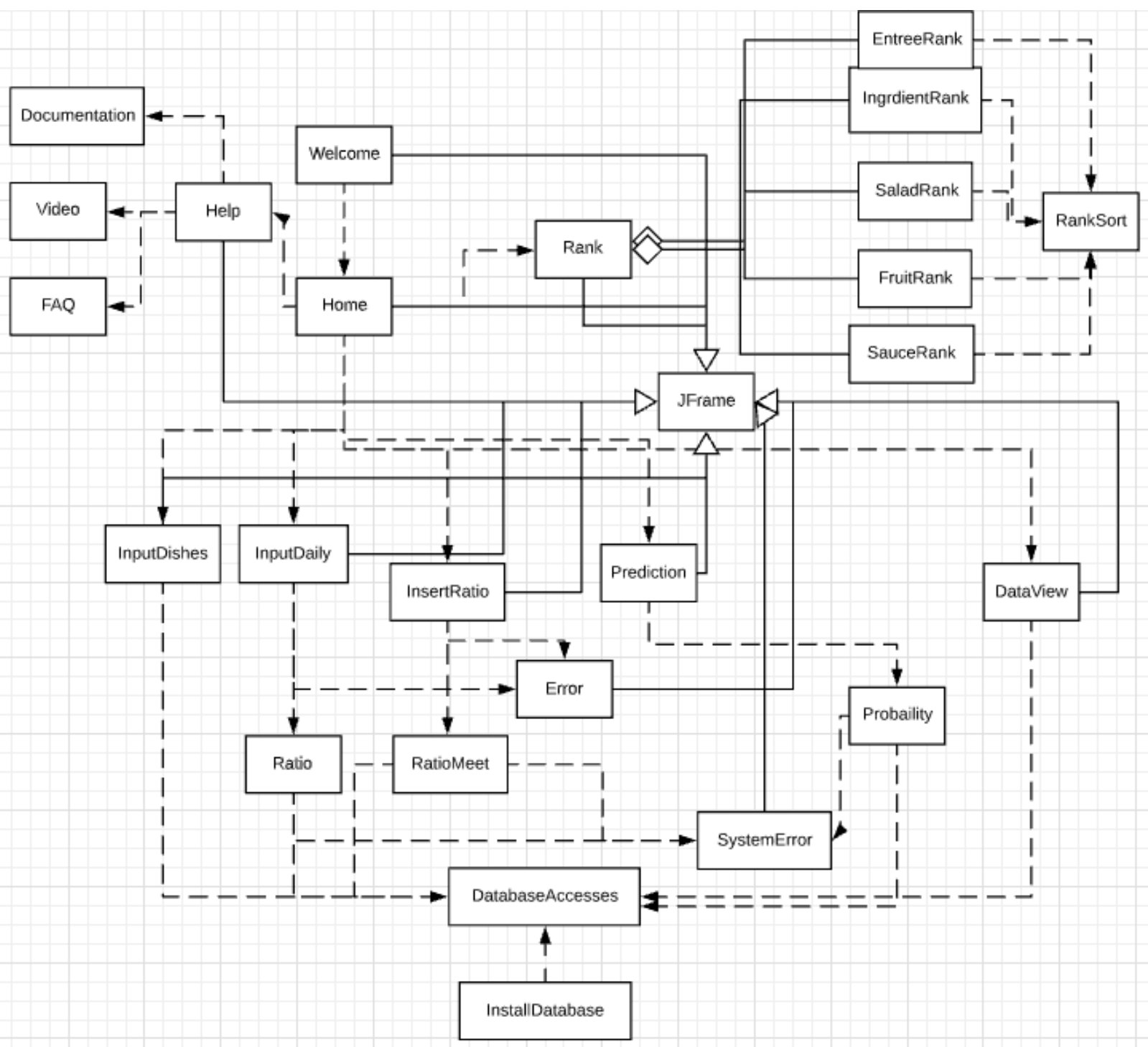
- dbName: String
 - newTable: String
-
- + JavaDatabase():

JavaDatabase

- dbName: String
- dbConnect: Connection
- Data: ArrayList

- + setDbName(String dbName): void
- + setDbConn(): void
- + setData(ArrayList data): void
- + setDBName(): string
- + getDbConn(): void
- + getData(): ArrayList
- + JavaDatabase(String DbName)
- + JavaDatabase():

CLASSES RELATIONSHIP DIAGRAM



GUI rough draft

Welcome frame

Help			
JLabel - Name of the program			
<div>Home</div>			
	<div>Close</div>		

Home frame

Help			
<div>New Dishes</div>	JLabel - Name of the program		<div>Rank</div>
<div>Input</div>			<div>Data</div>
	<div>Ratio</div>	<div>Prediction</div>	
	<div>Back</div>		<div>Close</div>

Data frame

Help			
JTable - Display all Data input and calculation stored in Database			
	<div>Home</div>	<div>Close</div>	

Input dishes frame

Help			
<div>Dish Type</div>			
<div>Dish Name</div>	<div>Ingredient</div>	<div>Sauce/spice</div>	
	<div>Input</div>		
	<div>Home</div>	<div>Close</div>	

Input daily frame

Help			
<div>Dish Type</div> <div>Dish Name</div> <div>Amount made</div> <div>Amount left</div> <div>Input</div>			
	Home		Close

Rank frame

Help			
<div>Ingrdient Rank</div> <div>Fruit Rank</div> <div>Entree Rank</div> <div>Salad Rank</div> <div>Sauce Rank</div>			
	Home		Close

Input ratio frame

Help			
<div>Dish Type</div> <div>Ratio</div> <div>Check</div> <div>JTable - listing all food meet the ratio</div>			
	Home		Close

Prediction frame

Help			
<div>Dish Type</div> <div>Dish Name</div> <div>Ingredient</div> <div>Sauce/spice</div> <div>Ratio</div> <div>Predict</div> <div>Probility of passing</div>			
	Home		Close

[illegible]

The image shows a Java Swing window with a title bar that says "Help". The window has a large central area containing a JLabel with the text "JLabel - Display the error in user input". At the bottom of the window, there is a horizontal bar with five buttons: "Home", "Back", and "Close" are labeled, while the two buttons on the far left and right are empty.

The image shows a Java Swing window titled "Help". The window has a title bar with the text "Help". The main content area is a large text field containing the text "JLabel - Display the error that has occurred in the system". At the bottom of the window, there is a footer bar with three buttons: "Home", "Back", and "Close".

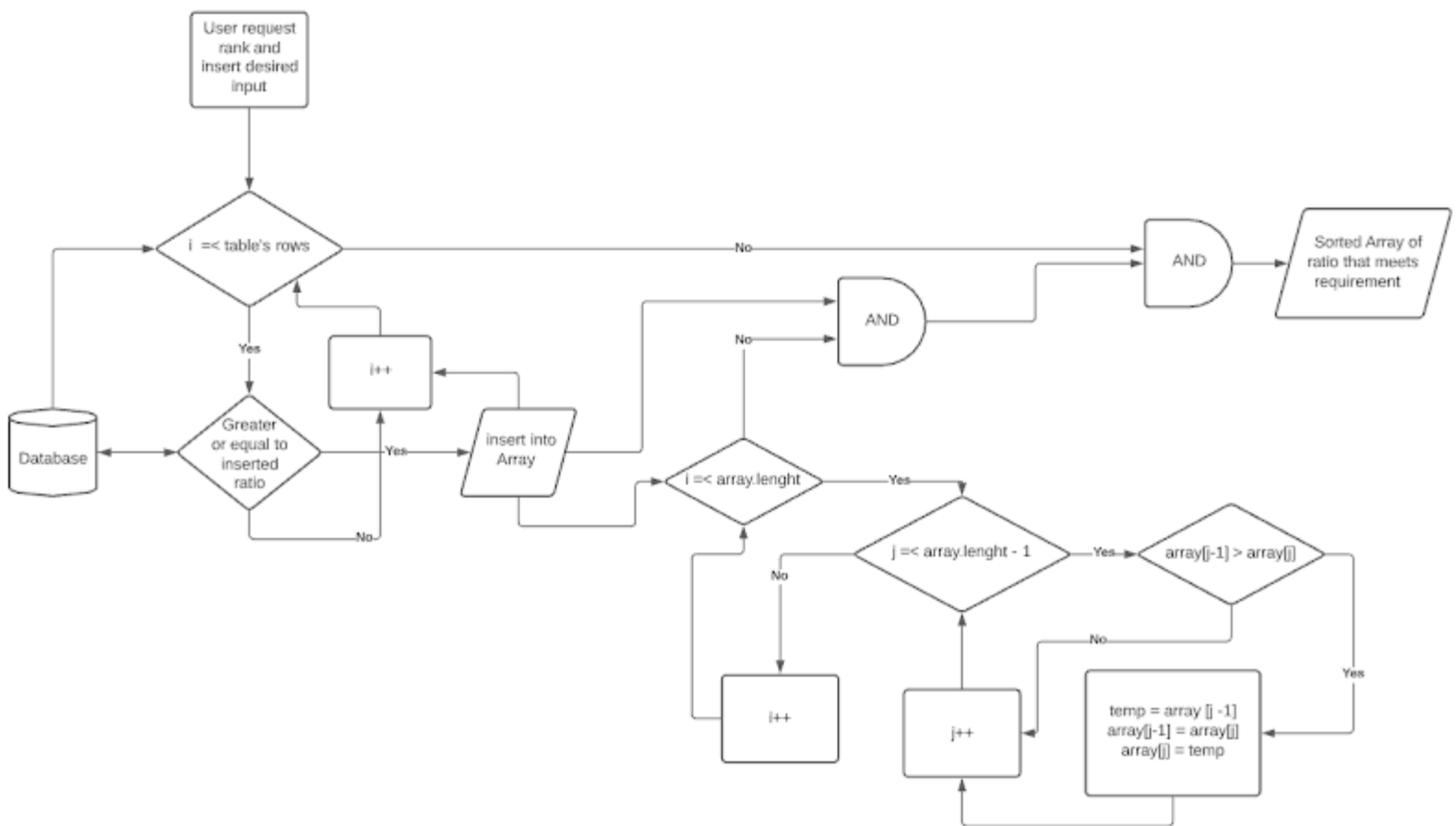
Flow charts & data structures

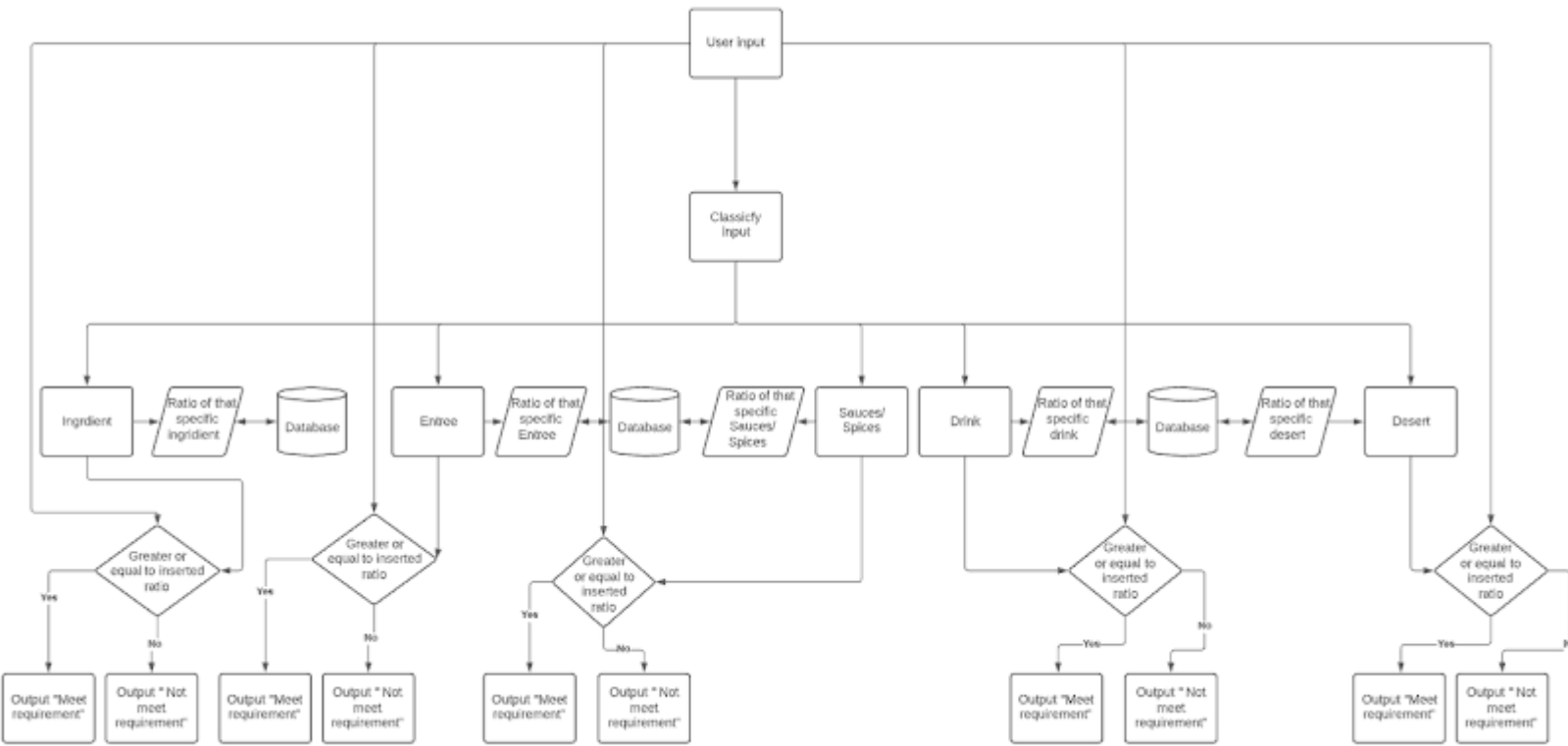
Data Structure

Data Structure	Description
ArrayList<ArrayList <float>>	Store numbers from calculation and store ratio.
ArrayList<ArrayList <String>>	Store data and manipulate in the database.
Object [] []	Use to display data in JTable
String []	Use to hold table headers

Flow Chart

Data ranking system (chart 1) and Data detection and checking system (chart 2)





Test plan

Features	Type of testing how the testing
A main or welcome window that has buttons for the user to maneuver or switch to different windows.	User acceptance testing The user will test to see if she or he can maneuver from different frames utilizing buttons. The user will give feedback wherever the buttons are comprehensive or easily accessible.
Help window is provided to display a basic tutorial and procedure of the program for the user.	User acceptance testing The user will view the procedure and give feedback whenever it is comprehensive.
Data entry that allows the user to input the type of dish they are serving and the type of ingredients as well as sauces or spices in the dish.	Black box testing User will enter the input such as entree, into the program along with all the components: sauce, spices, ingredients...
Window for daily input by the user to enter.	Black box testing User will select the name of the dish they have input and the user input how many how many were made as well as what was left over there for that type of dish.
Users can see what type of dish there is and what is the ingredient in it.	Black box testing User will access the data view table through the home frame. User will view the data through a table in the view data frame.
Feature to change the ingredients in a dish that has been inputted.	Black box testing User will change the name of the dish, the ingredient, the sauce, the desert, the water, the ratio in the data table in view data frame.
Feature option to fix the amount of food made or left over for the type of dish the user has entered on a specific date.	Black box testing User will change the amount of food made or left over for the type of dish the user has entered on a specific date.

Feature to store all data in the database and accessible by the user	White box testing Developer will put test data, and see if it is stored correctly and accessible.
Calculates the ratio of the amount of food made and the amount of food left over and stores it in the database.	White box testing Test data will be inputted, along with that the developer will calculate by hand and see if the result matches.
Determine if the food is preferable or not by inputting the user's desirable ratio of the amount of food made and the amount of food left over.	White box testing Test data will be inputted, along with that the developer will determine the output beforehand to see if the result will match. .
Displays preferable food when the user presses a button asking for favorite food.	User acceptance testing User will have a general idea of what the outcome is and will see if the program output is expected by the user.
Provides the user with what ingredients that have been preferred by scanning each dish with the ingredients and their ratio outcome.	White box testing Developer will scan the data beforehand and determine the output, then judge the program's output based on the expected output.
Feature an error window if the user entered illogical data	White box testing Illogical data will be purposely entered into the program to check if the program is robust.
Provide probability of new dishes will meet the desired ratio based on the elements inside the dishes.	White box testing The developer will feed the program with training data. The developer will also have an expected result within a range of acceptance value. The data output by the program will be judged base how far it is to the expected result

Amendments

Database and Tables

Unnormalized Database

DishID (int)	Name (varchar)	Type (varchar)	Ingredients (varchar)	Sauces or spices (varchar)	Ratio (int)
		Entree			
		Desert			
		Salad			
		Drink			
		fruits			

Normalized Database

DishName

DishID	Dish_Name

DishType

DishID	DishType

Ingredient

DishID	Ingredient

Sauce

DishID	Sauce
--------	-------

DishRatio

DishID	Ratio

Classes & UML diagrams

Classes Remove

Class	Reason
RatioMeet	This class is unnecessary because it has a similar function to checkRatio class.
Documentation	Making a class for Documentation is unnecessary and the function of providing documentation can be in help class
FAQ	Making a class for FAQ is unnecessary and the function of providing FAQ can be in help class
Video	Video class is unnecessary after speaking with the client
IngredientRank	This class is unnecessary because the ranking ability can be put under one class.
EntreeRank	This class is unnecessary because the ranking ability can be put under one class.
SaladRank	This class is unnecessary because the ranking ability can be put under one class.
FruitRank	This class is unnecessary because the ranking ability can be put under one class.
DrinkRank	This class is unnecessary because the ranking ability can be put under one class.
SauceRank	This class is unnecessary because the ranking ability can be put under one class.

New Classes

Class Name	Description
DishName	Accessing and retrieving data from DishName table from database.
DishType	Accessing and retrieving data from DishType table from database.
Ingredient	Accessing and retrieving data from Ingredient table from database and holding method to combine ingredients for display.
Sauce	Accessing and retrieving data from DishName table from database.
Ratio	Accessing and retrieving data from Ratio table from database and holding method to convert double to String converter.
ComebineTable	A computational class used to combine many different tables into one with the same dish ID, return tables combined data.
TypeTable	A computational class used to identify specific types of data in the DishType table and extract only that data with a specific type requirement.
TableSorter	A computational class that uses sorting class to sort data and manipulate data for displaying in JTable.
UpdateFrame	A GUI class provides the user the ability to update data in the database.
WarningClose	A GUI class that warns the user before closing the program.
LineOfRegression	A computation class that holds methods to calculate and construct lines of regression.
CheckRatioTable	A computational class that uses CheckRatio class to retrieve specific data and manipulate data for displaying in JTable.
IngredientDisplay	A GUI class provides users with ingredient names in the database.

RankedTable	A computational class that uses ranking class to rank data and manipulate data for displaying in JTable.
SelectTableRatio	A computational class that uses selectRatio class to extract specific required data and manipulate data for displaying in JTable.
DataTable	A computational class that uses DishTypetable to extract specific type of data requested and manipulate data for displaying in JTable.

UML Diagram

Update UML Diagram

NewDish
<ul style="list-style-type: none"> - GUI components - inputMany: JCheckBox - ingredientCount: int - dishTypeOption: JComboBox - ingredientNumberBox: JComboBox - dishNameTextField: JTextField - ingredientTextField1: JTextField - ingredientTextField2: JTextField - ingredientTextField3: JTextField - ingredientTextField4: JTextField - ingredientTextField5: JTextField - ingredientTextField6: JTextField - ingredientTextField7: JTextField - ingredientTextField8: JTextField - ingredientTextField9: JTextField
<ul style="list-style-type: none"> - actionPerformed(ActionEvent e): void - input() - resetIngredient + NewDish():

Input
<ul style="list-style-type: none"> - GUI components - foodMade: double

<ul style="list-style-type: none"> - foodLeft: double - Ratio: double - dishID: int - entreeTable: JTable - fruitTable: JTable - saladTable: JTable - desertTable: JTable - drinkTable: JTable - dataTable: Object [] []
<ul style="list-style-type: none"> - actionPerformed(ActionEvent e): void + Input(): - addInputTextField - resetsPanels - resetTextField

Data
<ul style="list-style-type: none"> - GUI components - tableOption: JTable - Datatable: Object[] [] - mainTable: JTable - entreeTable: JTable - fruitTable: JTable - saladTable: JTable - desertTable: JTable - drinkTable: JTable - sauceTable: JTable - ingredientTable: JTable
<ul style="list-style-type: none"> - actionPerformed(ActionEvent e): void + Data(): - reset()

CalculateRatio
<ul style="list-style-type: none"> - foodMade: double - foodLeft: double - ratio: double
<ul style="list-style-type: none"> + CalculateRatio(double foodMade, double foodLeft): + setFoodMade(double FoodMade): void + setFoodLeft(double FoodLeft): void + setCalculatedRatio(): void + getCalculatedRatio(): double

- + getFoodMade(): double
- + getFoodLeft(): double

CheckRatio

- GUI component
 - checkRatio: double
 - userTableSelection: String
 - displayTable: JTable
-
- actionPerformed(ActionEvent e): void
 - + CheckRatio():
 - identifyColumn(): String []

CalculatePrediction

- Ingredient: String
 - Ingredient1: String
 - Ingredient2: String
 - Ingredient3: String
 - Ingredient4: String
 - Ingredient5: String
 - Ingredient6: String
 - Ingredient7: String
 - Ingredient8: String
 - Ingredient9: String
 - predictedRatio: double
 - ingredientCount: int
 - ingredientList: ArrayList<String>
-
- + CalculatePrediction():
 - + getters and setters
 - + getIngredientCount(): int
 - + setIngredientCount(int IngredientCount): void
 - + setIngredientList(): void
 - + getIngredientList: ArrayList<String>
 - + setPredictedRatio(): void
 - + getPredictedRatio(): double

TableSorter

- unsortedData: ArrayList<ArrayList<String>>
- sortedData: ArrayList<ArrayList<String>>
- ratioColumn: int

- columnCount: int
+ getColumnCount: int + setColumnCount(int columnCount): void + getRatioColumn: int + setRatioColumn(int ratioColumn): void + getUnsortedData(): ArrayList<ArrayList<String>> + setUnsortedData(ArrayList<ArrayList<String>> unsortedData): void + getSortedData(): ArrayList<ArrayList<String>> + setSortedData();

JavaDataBase
- dbName: String - dbConnect: Connection - Data: ArrayList
+ setDbName(String dbName): void + setDbConn(): void + setData(ArrayList data): void + setDBName(): string + getDbConn(): void + getData(): ArrayList + JavaDatabase(String DbName) + JavaDatabase(): + to2dArray(): Object[] [] + getLatestID(): int + generateID(): int + insertData(String tableName, int IDNumber, String columnName, String insert): void + insertData(String tableName, int IDNumber, String columnName, double insert): void + update(String tableName, int IDNumber, String columnName, String change): void + update(String tableName, int IDNumber, String columnName, int change): void + delete(int IDNumber): void

New class UML Diagram

DishName
- dishName: String - dishID: int - data: ArrayList<ArrayList<String>>
+ DishName(): + DishName(int dishID, String dishName) + getters and setters + setDishNameTable(): void + getDishNametable(): ArrayList<ArrayList<String>>

DishType

- dishType: String
- dishID: int
- data: ArrayList<ArrayList<String>>

- + DishName():
- + DishName(int dishID, String dishType)
- + getters and setters
- + setDishTypeTable(): void
- + getDishTypeTable(): ArrayList<ArrayList<String>>

Sauce

- sauce: String
- dishID: int
- data: ArrayList<ArrayList<String>>

- + Sauce():
- + Sauce(int dishID, String sauce)
- + getters and setters
- + setDishSauceTable(): void
- + getDishSauceTable(): ArrayList<ArrayList<String>>

Ingredient

- ingredient: String
- dishID: int
- data: ArrayList<ArrayList<String>>
- combineData: ArrayList<ArrayList<String>>
- allIngredientName: ArrayList<ArrayList<String>>

- + Ingredient():
- + Ingredient(int dishID, String ingredient)
- + getters and setters
- + setIngredientTable(): void
- + getIngredientTable(): ArrayList<ArrayList<String>>
- + CombineIngredient(ArrayList<ArrayList<String>> data): ArrayList<ArrayList<String>>
- + getAllIngredientName(): ArrayList<ArrayList<String>>

Ratio

<ul style="list-style-type: none"> - ratio: double - dishID: double - checkRatio: double - data: ArrayList<ArrayList<Double>> - dataTable: ArrayList<ArrayList<String>> - dataChecked: ArrayList<ArrayList<Double>>
<ul style="list-style-type: none"> + Ratio(): + getters and setters + serData(): void + getData(): ArrayList<ArrayList<Double>> + intToStringArray (ArrayList<ArrayList<Double>> data): ArrayList<ArrayList<String>> + setDataChecked(): void

CombineTable
<ul style="list-style-type: none"> - dataCombine: ArrayList<ArrayList<String>> - dataTable1: ArrayList<ArrayList<String>> - dataTable2: ArrayList<ArrayList<String>> - dataTable3: ArrayList<ArrayList<String>> - dataTable4: ArrayList<ArrayList<String>> - dataTable5: ArrayList<ArrayList<String>> - tableCount: int
<ul style="list-style-type: none"> + Combinetable(): + getters and setters + setDataCombine(): void + getTableData(): ArrayList<ArrayList<String>>

TypeTable
<ul style="list-style-type: none"> - dishType: String - data : ArrayList<ArrayList<String>> - dishTypedata: ArrayList<ArrayList<String>>
<ul style="list-style-type: none"> + TypeTable (String dishType): void + setTypeTable(): void + getTypeTable(): ArrayList<ArrayList<String>>

TableSorter
<ul style="list-style-type: none"> - unsortedData: ArrayList<ArrayList<String>>

<ul style="list-style-type: none"> - sortedData: ArrayList<ArrayList<String>> - ratioColumn: int - columnCount: int
<ul style="list-style-type: none"> + TableSorter(): + getters and setters + setSortedData(ArrayList<ArrayList<String>> unsortedData): void + getUnsortedData(): ArrayList<ArrayList<String>> + setSortedData(): void + getSortedData(): ArrayList<ArrayList<String>>

UpdateFrame
<ul style="list-style-type: none"> - GUI components - typeUpdate: boolean - nameUpdate: boolean - ingredientUpdate: boolean - sauceUpdate: boolean - typeSelect: String - typeName: String[] - typeOption: JComboBox
<ul style="list-style-type: none"> + UpdateFrame(): - actionPerformed((ActionEvent e): void - removeOptions(): void - reset(): void

WarningClose
<ul style="list-style-type: none"> - GUI components
<ul style="list-style-type: none"> - actionPerformed((ActionEvent e): void + WarningClose():

CheckRatioTable
<ul style="list-style-type: none"> - data: ArrayList<ArrayList<String>> - dataTable: Object[] [] - targetTable: String - checkRatio: double
<ul style="list-style-type: none"> + CheckRatioTable(): + getters and setters

- + setData(): void
- + to2dArray(ArrayList<ArrayList<String>> data): Object [] []
- + getDataTable(): Object [] []

IngredientDisplay

- GUI components
- actionPerformed(ActionEvent e): void
- + IngredientDisplay():

RankedTable

- data: ArrayList<ArrayList<String>>
- dataTable: Object [] []
- targetTable: String
- + RankedTable():
- + setters and getters
- + setData(): void
- + getRankedTable: ArrayList<ArrayList<String>>
- + to2dArray(ArrayList<ArrayList<String>> data): Object [] []

SelectTableRatio

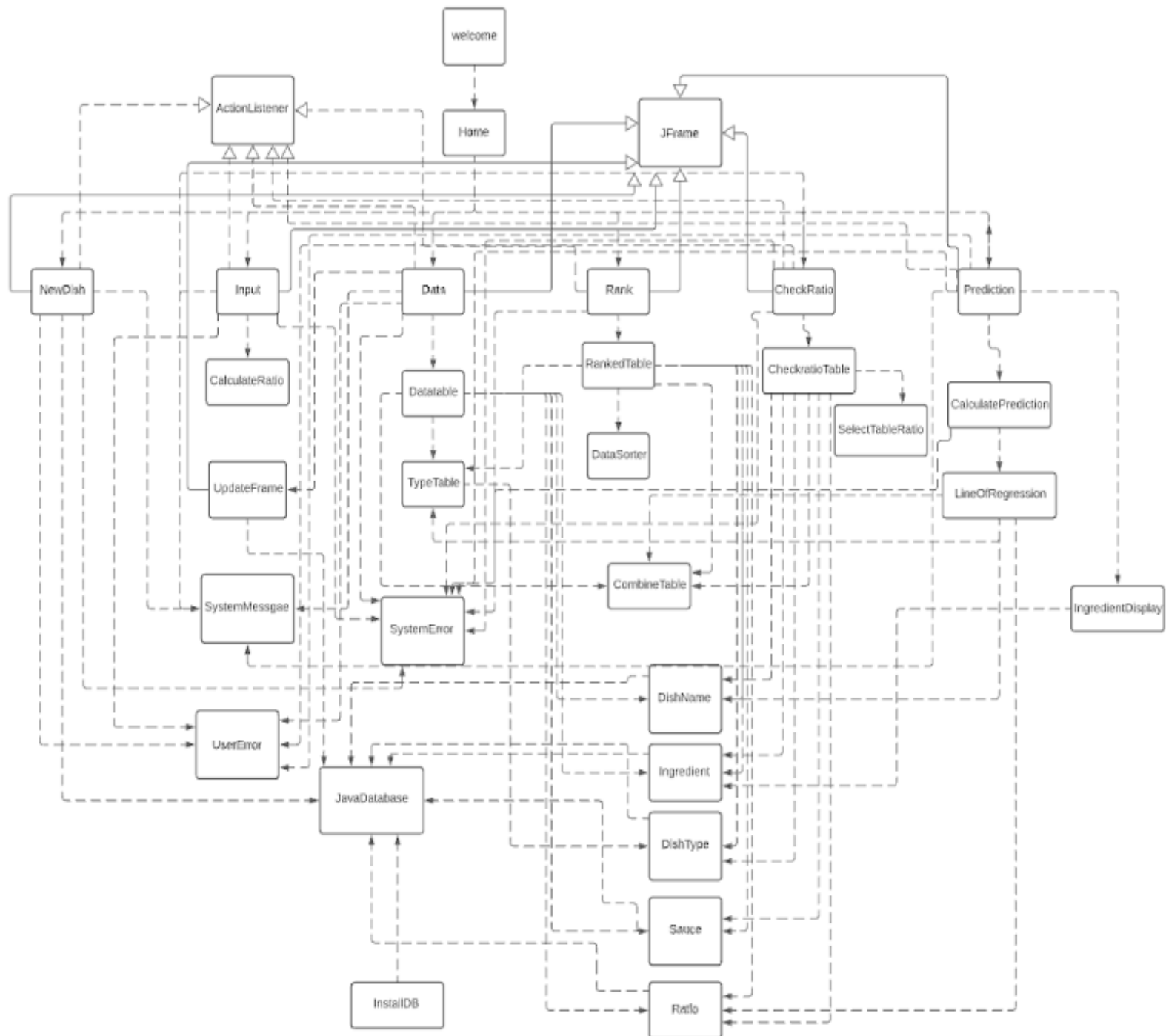
- unSelectedData: ArrayList<ArrayList<String>>
- SelectedData: ArrayList<ArrayList<String>>
- ratioColumn: int
- columnCount: int
- checkRatio: double
- + SelectTableRatio():
- + getters and setters
- + setSelectedData(): void
- + setUnsortedData (ArrayList<ArrayList<String>> unsortedData): void
- + getSelectedData(): ArrayList<ArrayList<String>>

DataTable

<ul style="list-style-type: none"> - targetTable: String - data: ArrayList<ArrayList<String>> - dataTable: Object [] []
<ul style="list-style-type: none"> + DataTable (String targetTable): void + setDataTable(): void + getDataTable(): Object [] [] + to2dArray(ArrayList<ArrayList<String>> data): Object [] []

LineOfRegression
<ul style="list-style-type: none"> - ySet: Double[] - xSet: Double[] - ingredientList: ArrayList<String> - dataIngredient: ArrayList<ArrayList<String>> - dataDishName: ArrayList<ArrayList<String>> - dataRatioInt: ArrayList<ArrayList<Double>> - ingredientName: String - ingredientAvgSum: double - dishAvgSum: double - mSlope: double - bIntercept: double - sumationXY: double - sumationX: double - sumationY: double - sumationXsquared: double - sumationYsquared: double
<ul style="list-style-type: none"> + LineOfRegression(): + setLineOfRegression(): void + setDishAvgSum(): void + setYset(): void + setXset(): void + setIngredientAvg(): void + CalculateAvg(Double[] dataset): double + setSumationXY(Double[] xSet, Double[] ySet): void + setSumationX(Double[] xSet): void + setSumationY(Double[] ySet): void + setSumationXsquared(Double[] xSet): void + setSumationYsquared(Double[] ySet): void + roundDouble(double value, int places): double + getMSlope(): double + getBIntercept():double + getDishAvgSum(): double

Class Relationship



GUI Final Draft

Home Frame

Data Frame

Help

New Dish

Input

Data

Picture

Rank

Ratio

Prediction

Close

Back

Help

Main Table

Update

Delete

Reload Table

JTable - displaying dishes in database

Close

NewDish Frame

Input Frame

Help

Entree

Input many

Dish ID

Ingredient

1

sauce and spices

input entree

Close

Help

Entree Table

Dish ID

Food Made


Food Left

input





JTable - displaying dishes in database

Close


Rank Frame

Help		
Main Table 	JTable - displaying sorted data in database	
		Close

Ratio Frame

Help		
Main Table 	JTable - displaying dishes in database	
Favorite 		
Half 		
Hate 		
Advance 		
		Close

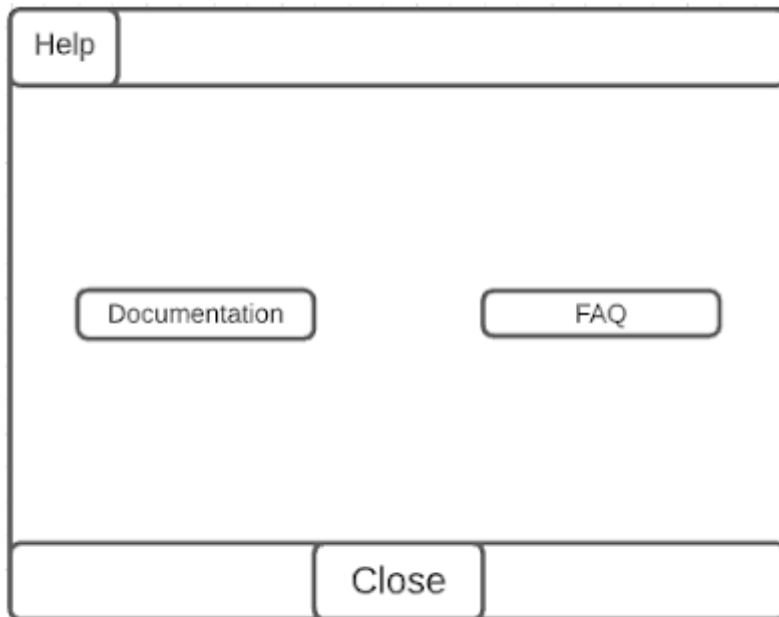
Prediction Frame

Help		
Ingredient 1  <input type="text"/>	Ingredient <input type="button" value="Display"/>	<input type="button" value="Predict"/>
		Close

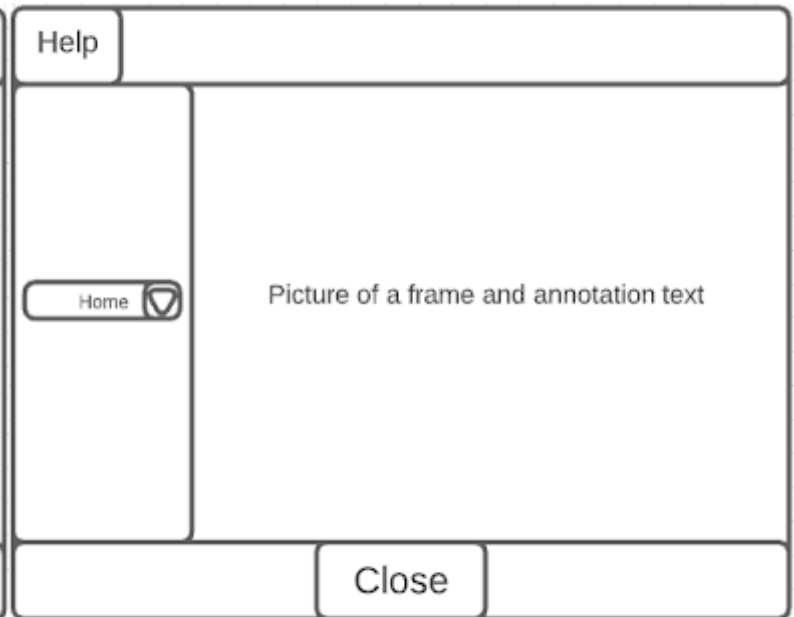
WarningClose Frame

Are you sure you want to quit ?	
<input type="button" value="Yes"/>	<input type="button" value="No"/>

Help Frame



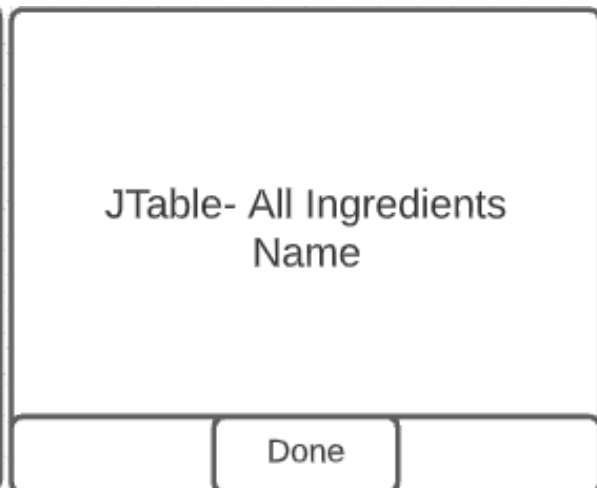
Help(Documentation) Frame



SystemMessage Frame



IngredientDisplay Frame



Data Structure

Data Structure	Description
ArrayList<ArrayList<String>>	Used to hold data from database, and used to hold data to be manipulated
ArrayList<ArrayList<Double>>	Used to hold data that is a number from database, and used to hold number data to be manipulated
Object []	Used to hold data to be displayed on JTable
String []	Used to hold the table header of JTable
Int []	Used to hold integers for calculation and sorting.
Double []	Used to hold double for calculator and sorting.
ArrayList<String>	Used to hold data and used as Object to be put inside ArrayList
ArrayList<Double>	Used to hold number data and used as Object to be put inside ArrayList