The Kitchen Assistant

Software Requirements Specification / Design Document

Institute for Women in Technology
© 1999 Texas A&M University
## Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision</th>
<th>Description</th>
<th>Author</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/19/99</td>
<td>1.0</td>
<td>Initial Version</td>
<td>JF, JD, TLK, JG, JM</td>
</tr>
</tbody>
</table>
# Table of Contents

1. **INTRODUCTION** .................................................................................................................................\1
   1.1. PURPOSE OF THE PRODUCT REQUIREMENTS DOCUMENT (PRD) .................................................\1
   1.2. PRODUCT OVERVIEW .......................................................................................................................\1
   1.3. REFERENCES .......................................................................................................................................\1

2. **SOFTWARE REQUIREMENTS** ...........................................................................................................\2
   2.1. INTERFACE ...........................................................................................................................................\2
       2.1.1. Voice Recognition .......................................................................................................................\2
       2.1.2. Touch Screen ...............................................................................................................................\2
       2.1.3. Scanner ........................................................................................................................................\2
       2.1.4. Monitor, Speakers – Audio and Visual Output ...............................................................................\2
       2.1.5. Kitchen Device Interface ...........................................................................................................\2
   2.2. APPLICATION .......................................................................................................................................\3
       2.2.1. Voice Recognition .......................................................................................................................\3
       2.2.2. Database Interaction ....................................................................................................................\3
       2.2.3. Liaison .........................................................................................................................................\3
   2.3. DATABASE ............................................................................................................................................\3
       2.3.1. History .........................................................................................................................................\3
       2.3.2. Data Types ...................................................................................................................................\3
       2.3.3. Application Interface ..................................................................................................................\3
       2.3.4. Make and Platform ......................................................................................................................\3
       2.3.5. Multimedia .................................................................................................................................\4

3. **SYSTEM MODELS** ..............................................................................................................................\5
   3.1. OBJECT MODEL ..................................................................................................................................\5
   3.2 DYNAMIC MODEL ................................................................................................................................\8
   3.3 INTERFACE MODEL .............................................................................................................................\11
1. **Introduction**

1.1. **Purpose of the Product Requirements Document (PRD)**

This document regarding the Kitchen Assistant is an organizational document which will help our team to organize our program design, visualize our programming structure and refine our team responsibilities. This document will help in the clarification of the individual component of our program module and their relations. This document will also help the team to organize their own respective responsibilities. This document outlines capabilities needed by the user, and describes a method for providing these capabilities.

1.2. **Product Overview**

Our goal is to create a kitchen application that will help cooks of all skill levels to learn and improve their cooking skills. We plan to the public with an application that will provide the public with different features that will not only aid the user in food preparation, but also all around the kitchen. The Kitchen Assistant will help users with such things as creating a grocery shopping list, planning a nutritional diet, scheduling meals or menus in advance, and many other features explained in more detail in this document. The Kitchen Assistant will create a meal plan custom made for each user, based on the user’s preferences. It will allow them to create their favorite meal list and schedule them whenever they want to make them. We hope that our product will open the door for the kitchen of the future and allow users to interact with the whole kitchen.

1.3. **References**

TBA.
2. Software Requirements

The product may be divided into three (3) parts for the sake of classification; Interface is the user interface, Application is what calculates calorie counts and the meal planning schedules, etc., Database maintains the information needed to assist the user, such as the kitchen dictionary, user history, and multimedia clips. Each part requires separate software packages and capabilities.

2.1. Interface

2.1.1. Voice Recognition

The primary interface will be a vocal interpreter. Therefore the product interface will require a speech analysis program, commercially available if possible. Currently under consideration is IBM’s ViaVoice due to its lower cost and portability to operating systems other than Windows 95, specifically Linux.

2.1.1.1. Versatility

Since the product should have an intuitive interface, it should be able to interpret a wide variety of commands. The underlying interpretation software therefore will be keyed to a wide range of “buzzwords” that will identify the requested action.

2.1.2. Touch Screen

A more standard touch interface will also be available in the manner of a touch screen or keyboard. Industry standard hardware and software drivers should suffice for this purpose.

2.1.3. Scanner

In order to read and analyze the receipts, the product must be equipped with a scanning device. The standard software drivers are also available for scanners such as the ones desired.

2.1.4. Monitor, Speakers – Audio and Visual Output

The output shall be both visual and auditory, and therefore the product must have a monitor and speakers, as well as a sound device. Once again, industry standard hardware and software are sufficient for this purpose.

2.1.5. Kitchen Device Interface

For the case of the Smart Kitchen implementation, the product must interface with other kitchen devices (i.e. the toaster, the range, etc.). Such devices would have a new form of interface with the Kitchen Helper, and as such a new set of instructions between the appliances and the product will be developed.
2.2. Application

2.2.1. Voice Recognition
The application will include analyzing the request and performing the appropriate calculations needed. Therefore the application must be able to calculate a range of useful information including, but not limited to: calories per serving, cooking time for multiple recipes, estimated cost based on previous purchases, and so on.

2.2.2. Database Interaction
The application will also make and necessary calls to the database of the product. It should therefore be able to build intelligent queries using the Query object of the database. It should also pass newly acquired information, such as receipt prices or user preference settings, to the database to be stored for future use.

2.2.2.1. Query Results
In the interest of making the system more user-friendly, should the request return a large set of options or potential results, the application should be able to pare down the set by eliminating any unneeded or unwanted results.

2.2.3. Liaison
The application should be an intelligent go between for the Database and Interface parts of the product.

2.3. Database

2.3.1. History
Since the product should adapt to the user’s preferences, both manually set and observed, the database should maintain a user history, a log of user actions and choices so that the user’s past actions can be used to more effectively perform services for the user.

2.3.2. Data Types
In order to perform its main function as a cooking assistant, the database should maintain a variety of useful information tables, such as food, dictionary, and recipes to be used by the application and be passed along to the user.

2.3.3. Application Interface
The interface between the User Interface and the database is the application, therefore the database should include a Query object that the application can use to build SQL statements to the database by setting parameters and constraints on the query.

2.3.4. Make and Platform
Since the desired product would not be implemented on a Microsoft system, the product cannot use a Microsoft database. The database should be implemented using a portable database system, most likely and ORACLE database, since the system is free, well known and easily ported to Linux.
2.3.5. Multimedia

In order to maintain the application is memory-less, the database will maintain the multimedia files to be passed to the application when the application determines them to be necessary.
3. System Models

3.1. Object Model

```java
class Query:

attributes:
RecipeID rid;

methods:
void CookMeal()
void SetCookingLevel()
vector<RecipeID> GetRecipes()
Recipe GetRecipe(RecipeID rid)
void SortRecipes()
void MergeRecipes()
void PutIntoShoppingList()

class IO:

attributes:

methods:
void PrintRecipe()
void PrintReceipt()
void ScanRecipe()
void ScanReceipt()
void PrintShoppingList()

class Scheduler:

attributes:
bool isOnMicrowave;
bool isOnOven;
bool isOnCoffeeMaker;
bool isOnRiceCooker;
bool isOnToaster;

methods:
void TurnOn()
void TurnOff()
void ShowStatus()
void GetInputFromPhoneCall()

class Recipe:

attributes:
double prepareTime;
double cookingTime;
double calories;
vector<Utensil> utensils;
int cookinglevel;
AudioVideo av;

methods:
double GetPrepareTime()
double GetCookingTime()
double GetCaleries()
vector<Utensil> GetUtensils()
int GetCookingLevel
AudioVideo GetAudioVideo()

class ShoppingList:

attributes:
vector<Food> foodlist;

methods:
void Planning():
vector<Food> GetList();
double EstimateCost();
Date GetLastShoppingDate(Food fd);
Place GetLastShoppingPlace(Food fd)
double GetLastShoppingPrice(Food fd):

class User:

attributes:
int UserID;
int cookinglevel;

methods:
void GetUserInfo();
void GetHistory();
void GetPreferences();
void AddToHistory();
```

Object Query: Works as an interface class between user and the system.
- CookingMeal(): Record that a meal has been cooked. Update the inventory based on foodstuffs consumed.
- SetCookingLevel(): A proper detail level of descriptions is chosen according to user's skill, e.g. novice, master chief, etc.
- GetRecipes(): Return recipes that meet the query requirement(s).
- GetRecipe(): Return a specific recipe.
- SortRecipe(): Sort recipes by definable criteria.
- MergeRecipe(): Merge ingredients and instructions.
- PutIntoShoppingList(): Add items not available in inventory to shopping list.

Object Recipe
- GetPrepareTime(): Return preparing time for this recipe.
- GetCookingTime(): Return cooking time.
- GetCalories(): Return calories.
- GetUtensils(): Return utensils needed; warn user if some utensils is not on the inventory.
- GetCookingLevel(): Return cooking skill required for this recipe.
- GetAudioVideo(): Return audio or video attached to this recipe.

Object I/O: Input, output class.
- PrintRecipe(): Print the selected recipe to a printer.
- PrintReceipt(): Print the receipt.
- ScanRecipe(): Read in a recipe.
- ScanReceipt(): Read in a receipt.
- PrintShoppingList(): Print out shopping list.

Object Shopping List:
- Planning(): Arrange the next shopping: place, time, what to buy.
- GetList(): Display the list.
- EstimateCost(): Estimate the total cost for all the stuff on the list.
- GetLastShoppingDate(Food fd): Return last date one specific food was purchased.
- GetLastShoppingPlace(Food fd): Return the place.
- GetLastShoppingPrice(Food fd): Return the price.

---

```java
class Data:
attributes:
RecipeID rid;

methods:
void AddRecipe();
void DeleteRecipe();
Recipe GetRecipe(RecipeID rid);

class Inventory:
attributes:
vector<Food> foodlist;

methods:
void ModifyInventory();
```
Object Scheduler:
• TurnOn(): Turn on a specific device, e.g. oven.
• TurnOff(): Turn a specific device off.
• ShowStatus(): Display the status for all the devices.
• GetInputFromPhoneCall(): Read control input from phone line.

Object User
• GetUserInfo(): Get the current user’s personal info (e.g. name).
• GetHistory(): Get the user’s cooking history, a list of past recipes.
• GetPreferences(): Get the user’s preferences for the system.
• AddToHistory(): Add a recipe to the list of past recipe history.

Object Data
• AddRecipe(): Add a recipe to the database.
• DeleteRecipe(): Delete a recipe from the database.
• GetRecipe(RecipeID rid): Retrieve a recipe from the database.

Object Inventory
• ModifyInventory(): Modify (add/delete items) the current inventory.