P3: Food Planner

A Software Analysis and -Implementation

28/01-2015

Mathias Vestergaard Rasmussen *<mvra13@student.aau.dk>*
Christoffer Carlé Christensen *<ccch13@student.aau.dk>*
Kasper Østergaard Helsted *<khelst13@student.aau.dk>*
Anders Lykke Matthiassen *<amatt13@student.aau.dk>*
Christian Stephansen *<csteph13@student.aau.dk>*
Gideon Blegmand *<gblegm13@student.aau.dk>*

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk/

AALBORG UNIVERSITY
DENMARK

Agenda

Intro

Problem statement & Information Gathering

Quality Assurance

Unit Testing

Usability Testing

Reflection

Code Examples

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Intro

AALBORG UNIVERSITY
DENMARK

Content

- Causes
- Problem Statement
- XAML
- Code Exammples
- Unit Testing
- Usability Test
- Reflection

Problem statement & Information Gathering

AALBORG UNIVERSITY
DENMARK

Problem statement

- How can a software system assist the consumer in planning and managing a meal plan to minimize food waste and simultaneously be flexible enough to support their planning, shopping and cooking habits?

Information Gathering

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Quality Assurance

AALBORG UNIVERSITY
DENMARK

Quality assurance
Fulfilling the semester description

- Quality assurance (in terms of program testing and results and usability evaluation and results), including arguments for coverage and validity (Semester description)
  - Program testing = Unit testing
  - Usability evaluation = IDA (report section 11)

Unit Testing

AALBORG UNIVERSITY
DENMARK

Unit testing
What is unit testing?

- Initial stage of program testing
- Isolate smallest testable code
- Confirm if the unit behaves correctly

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Unit testing

Possible candidates for unit testing

- Classes e.g. User, Ingredient or Recipe
  - Testing their properties or methods
- List of candidates

Models	**ViewModels**
Graylist	InventoryViewModel*
Ingredient	MealPlanViewModel
InventoryIngredient*	RecipeSearchViewModel
InventoryListCombinedByQuantity	RecipeViewModel
LastMeal	SettingsViewModel
PublicQuerys	ShoppingListViewModel
Recipe	
RecipeIngredient	
SearchResults	
ShoppingClass	
ShoppingListIngredient	
User	

Unit testing
Code from InventoryIngredientTests.cs


```
1  [TestMethod]
2  public void PurchaseDate_AutoSetConstructor_SetToNow() {
3  //arrange
4  Ingredient testIngredient = new Ingredient();
5  DateTime expectedPurchaseDate = DateTime.Now;
6
7  //act - The property is set automaticly in the constructor
8  InventoryIngredient testInventoryIngredient = new
       InventoryIngredient(testIngredient, 750);
9  //assert
10 Assert.AreEqual(expectedPurchaseDate,
       testInventoryIngredient.PurchaseDate);
11 }
```

Usability Testing

AALBORG UNIVERSITY
DENMARK

IDA

- 3 persons
- Video camera
- Screen recorder
- Follow-up questions

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

IDA outcome

- Critical problems
  - Scheduled meal page, update button
  - Deleting meal
  - Automatical or manual update
- Serious problems
  - Planning meal the wrong way
  - Top bar in recipe screen
  - Changing the number of days to shop for
- Cosmetic problems
  - Dividing of setting screen
  - Adding ingredient to lists
  - Too specific rating

Usability notes

- Waterfall vs Iterative

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Reflection

AALBORG UNIVERSITY
DENMARK

missing features

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Optimization

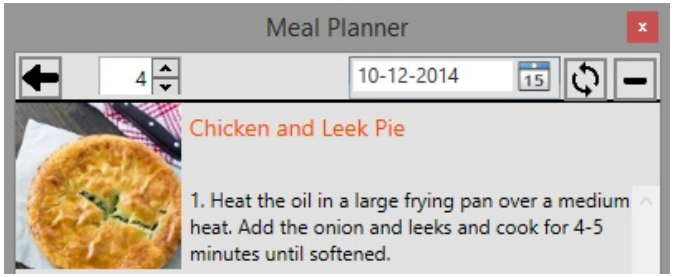
Minimize communication between server and client

- Search algorithm
  - .ToList()
- Cashing
  - Save meals locally
  - Local list of inventory
  - Better allowance of off-line usage

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Evaluation of design

- Scheduled meal page, update button



- Deleting meal

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Requirements

Problem based requirements:

- To much food being wasted
- Miscalculations happen in groups
- Shopping is time consuming
- Diets are difficult
- Consumers living alone have the most food waste
- Consumers living alone have a higher cost per meal

The two main problems:

- Planning and managing meals
- Minimizing food waste

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Requirements

Problem based requirements:

- To much food being wasted
- Miscalculations happen in groups
- Shopping is time consuming
- Diets are difficult
- Consumers living alone have the most food waste
- Consumers living alone have a higher cost per meal

The two main problems:

- Planning and managing meals
- Minimizing food waste

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Code Examples

Sorting

- Percentage Full Match
- Percentage Partial Match
- Rating
- Previous Ingredients
- Recipe Title

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

PublicQuerys


```
17          #region Fields
18          //A query that if multiple of the same ingredient is found in the users inventory, combines the sum into a single item, and combine the quantities onto one.
19          public IQueryable<inventoryListGroupedByQuantity> inventoryIQueryable = from ii in App.db.InventoryIngredients
20                                                                                  join i in App.db.Ingredients on ii.IngredientID equals i.ID
21                                                                                  where ii.UserID == App.CurrentUser.ID
22                                                                                  group ii by ii.IngredientID into iig
23                                                                                  select new inventoryListGroupedByQuantity()
24                                                                                  {
25                                                                                      IngredientID = iig.FirstOrDefault().IngredientID,
26                                                                                      Unit = iig.FirstOrDefault().Ingredient.Unit,
27                                                                                      Quantity = iig.Sum(i => i.Quantity),
28                                                                                      Ingredient = iig.FirstOrDefault().Ingredient,
29                                                                                      ExpirationDate = iig.FirstOrDefault().ExpirationDate,
30                                                                                      PurchaseDate = iig.FirstOrDefault().PurchaseDate,
31                                                                                      User = iig.FirstOrDefault().User,
32                                                                                      UserID = iig.FirstOrDefault().UserID
33                                                                                  };
34
35          //Same as last query execpt it's a list not a IQueryable.
36          public List<inventoryListGroupedByQuantity> inventoryList = (from ii in App.db.InventoryIngredients
37                                                                       join i in App.db.Ingredients on ii.IngredientID equals i.ID
38                                                                       where ii.UserID == App.CurrentUser.ID
39                                                                       group ii by ii.IngredientID into iig
40                                                                       select new inventoryListGroupedByQuantity()
41                                                                       {
42                                                                           IngredientID = iig.FirstOrDefault().IngredientID,
43                                                                           Unit = iig.FirstOrDefault().Ingredient.Unit,
44                                                                           Quantity = iig.Sum(i => i.Quantity),
45                                                                           Ingredient = iig.FirstOrDefault().Ingredient,
46                                                                           ExpirationDate = iig.FirstOrDefault().ExpirationDate,
47                                                                           PurchaseDate = iig.FirstOrDefault().PurchaseDate,
48                                                                           User = iig.FirstOrDefault().User,
49                                                                           UserID = iig.FirstOrDefault().UserID
50                                                                       }).ToList();
51
52          //A query that get all the ingredients from earlier meals, and group them, where the ingredientCount is the numbers of times the specfic ingredient has been used.
53          public List<LastMeal> ingredientsFromLastMeals = (from meals in App.db.Meals
54                                                            join ri in App.db.RecipeIngredients on meals.RecipeID equals ri.RecipeID
55                                                            join i in App.db.Ingredients on ri.IngredientID equals i.ID
56                                                            where meals.UserID == App.CurrentUser.ID && meals.Date <= DateTime.Now
57                                                            group i by i.ID into igrouped
58                                                            select new LastMeal()
59                                                            {
60                                                                ingredientID = igrouped.FirstOrDefault().ID,
61                                                                ingredientCount = igrouped.Count()
62                                                            }).ToList();
63
64          //A query that gets a list recipes where a blacklisted ingredient is used.
65          public List<int> blackList = (from bl in App.db.BlacklistIngredients
66                                        join ri in App.db.RecipeIngredients on bl.IngredientID equals ri.IngredientID
67                                        where bl.UserID == App.CurrentUser.ID
68                                        group ri by ri.RecipeID into ri
69                                        select ri.FirstOrDefault().RecipeID).ToList();
70
71          //get a list of graylisted ingredients.
72          public List<GrayList> grayList = (from gl in App.db.GraylistIngredients
73                                            join i in App.db.Ingredients on gl.IngredientID equals i.ID
74                                            where gl.UserID == App.CurrentUser.ID
75                                            group gl by gl.IngredientID into gl
76                                            select new GrayList()
77                                            {
78                                                ingredient = gl.FirstOrDefault().Ingredient,
79                                                rating = (gl.Sum(r => r.IngredientValue) / gl.Count())
80                                            }).ToList();
81          #endregion
```

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Search by recipes


```
83          #region Methods
            2 references
84          public IQueryable<IGrouping<int, Result>> search(List<int> recipeIDs)
85          {
86              //IQueryable that finds all the ingredients that belong to the found recipesm and how much of the found ingredient.
87              IQueryable<IGrouping<int, Result>> recipeIngredients = from ri in App.db.RecipeIngredients
88                                                                     join i in App.db.Ingredients on ri.IngredientID equals i.ID
89                                                                     join r in App.db.Recipes on ri.RecipeID equals r.ID
90                                                                     where recipeIDs.Contains(ri.RecipeID)
91                                                                     group new Result()
92                                                                     {
93                                                                         recipe = ri.Recipe,
94                                                                         ingredient = ri.Ingredient,
95                                                                         quantity = ri.Quantity
96                                                                     } by ri.RecipeID;
97
98              return recipeIngredients;
99          }
```

Sorting of results


```
2 references
public ObservableCollection<SearchResults> addValuesToSearch(IQueryable<IGrouping<int, Result>> userInput, List<string> searchKeywords = null)
{
    //Create ObservableCollection which allows for updates in the view.
    ObservableCollection<SearchResults> result = new ObservableCollection<SearchResults>();

    //Traverse through all found ingredients and add them to the recipe
    foreach (IGrouping<int, Result> ri in userInput)
    {
        Recipe recipe = ri.FirstOrDefault().recipe;

        //initialize the searchResult that is to be shown
        SearchResults searchResult = new SearchResults(recipe);

        //Make sure that searchKeywords is set, if null assume that it's a non-search that need to have added value.
        if (searchKeywords != null)
        {
            //check that the keyword is found in the recipe, it's case sensitive
            if (searchKeywords.Any(s => recipe.Title.ToLower().Contains(s.ToLower())))
            {
                searchResult.keyWordMatch++;
            }
        }

        //Traverse through all results which is ingredients
        foreach (Result res in ri)
        {
            //Add the ingredient to the recipe
            searchResult.addIngredient(res.ingredient);

            //Check if the ingredient is found in the users' inventory
            if (inventoryList.Where(il => il.IngredientID == res.ingredient.ID).Count() != 0)
            {
                //if found check if it's a partial match or full match
                //full match means that all of the ingredient is found in the inventory
                if (inventoryList.Where(il => il.IngredientID == res.ingredient.ID).First().Quantity >= res.quantity)
                {
                    searchResult.fullMatch++;
                }
                else
                {
                    //if only partial match, and the partial match percentage to the partialMatch
                    searchResult.partialMatch += inventoryList.Where(il => il.IngredientID == res.ingredient.ID).First().Quantity / res.quantity;
                }
```

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk

Sorting of results


```
                    }

                    //Make sure that searchKeywords is set, if null assume that it's a non-search that need to have added value.
                    if (searchKeywords != null)
                    {
                        //check that the keyword is found in the ingredient, it's case sensitive
                        if (searchKeywords.Any(s => res.ingredient.Name.ToLower().Contains(s.ToLower())))
                        {
                            searchResult.keyWordMatch++;
                        }
                    }

                    //Check if the ingredient is used in previous meals.
                    if (ingredientsFromLastMeals.Where(iflm => iflm.ingredientID == res.ingredient.ID).Count() != 0)
                    {
                        //if found add the amount of times the ingredient has been used to the prevIngredients
                        searchResult.prevIngredients += ingredientsFromLastMeals.Where(iflm => iflm.ingredientID == res.ingredient.ID).Single().ingredientCount;
                    }

                    //Check if the ingredient is found in the gray list, if it is add the value to setRating
                    if (grayList.Where(gl => res.ingredient.ID == gl.ingredient.ID).Count() != 0)
                    {
                        searchResult.setRating = grayList.Where(gl => res.ingredient.ID == gl.ingredient.ID).SingleOrDefault().rating;
                    }
                    else
                    {
                        //50 is the default value of nonrated items
                        searchResult.setRating = 50;
                    }
                }

                //add the SearchResults to the ObservableCollection of SearchResults
                result.Add(searchResult);
            }

            //return the ObservableCollection ordered.
            return new ObservableCollection<SearchResults>(result.OrderByDescending(res => res.percentageFullMatch).ThenByDescending(res => res.percentagePartialMatch).ThenByDescending(res =
        }
        #endregion
    }
}

```

Department of Computer Science
Selma Lagerlöfs Vej 300
DK-9220 Aalborg Ø
http://cs.aau.dk