Intro

Content

Intro

Problem statement &
Information Gathering

Quality Assurance

Unit Testing

Usability Testing

Reflection

Code Examples

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

Problem statement & Information Gathering

▶ 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?

▶

Quality Assurance

Food Planner

DS301E14

▶ 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

9

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

Food Planner

DS301E14

- ► Classes e.g. User, Ingredient or Recipe
  - ► Testing their properties or methods
- ► List of candidates
  - ► MODELS
  - ► Graylist
  - ► Ingredient
  - ► InventoryIngredient*
  - ► InventoryListCombinedByQuantity
  - ► LastMeal
  - ► PublicQuerys
  - ► Recipe
  - ► RecipeIngredient
  - ► SearchResults
  - ► ShoppingClass
  - ► ShoppingListIngredient
  - ► User
  - ► VIEWMODELS
  - ► InventoryViewModel*
  - ► MealPlanViewModel
  - ► RecipeSearchViewModel

Food Planner

DS301E14

Intro

Problem statement &
Information Gathering

Quality Assurance

Unit Testing

Usability Testing

Reflection

Code Examples

11

```
[ TestMethod ]
public void PurchaseDate_AutoSetConstructor_SetToN
// arrange
Ingredient testIngredient = new Ingredient ( ) ;
DateTime expectedPurchaseDate = DateTime . Now ;

// act - The property is set automaticly in the co
InventoryIngredient testInventoryIngredient = new
// assert
Assert . AreEqual ( expectedPurchaseDate , testInvento
}
```

Usability Testing

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

- ► 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

► Waterfall vs Iterative

Reflection

▶

►

►

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

Code Examples

Sorting

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

Food Planner

DS301E14

Intro

Problem statement &
Information Gathering

Quality Assurance

Unit Testing

Usability Testing

Reflection

Code Examples

```csharp
#region Fields
//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 into one.
public IQueryable<inventoryListGroupedByQuantity> inventoryIQueryable = from ii in App.db.InventoryIngredients
                    join i in App.db.Ingredients on ii.IngredientID equals i.ID
                    where ii.UserID == App.CurrentUser.ID
                    group ii by ii.IngredientID into iig
                    select new inventoryListGroupedByQuantity()
                    {
                        IngredientID = iig.FirstOrDefault().IngredientID,
                        Unit = iig.FirstOrDefault().Ingredient.Unit,
                        Quantity = iig.Sum(i => i.Quantity),
                        Ingredient = iig.FirstOrDefault().Ingredient,
                        ExpirationDate = iig.FirstOrDefault().ExpirationDate,
                        PurchaseDate = iig.FirstOrDefault().PurchaseDate,
                        User = iig.FirstOrDefault().User,
                        UserID = iig.FirstOrDefault().UserID
                    };

//Same as last query except it's a list not a IQueryable.
public List<inventoryListGroupedByQuantity> inventoryList = (from ii in App.db.InventoryIngredients
                    join i in App.db.Ingredients on ii.IngredientID equals i.ID
                    where ii.UserID == App.CurrentUser.ID
                    group ii by ii.IngredientID into iig
                    select new inventoryListGroupedByQuantity()
                    {
                        IngredientID = iig.FirstOrDefault().IngredientID,
                        Unit = iig.FirstOrDefault().Ingredient.Unit,
                        Quantity = iig.Sum(i => i.Quantity),
                        Ingredient = iig.FirstOrDefault().Ingredient,
                        ExpirationDate = iig.FirstOrDefault().ExpirationDate,
                        PurchaseDate = iig.FirstOrDefault().PurchaseDate,
                        User = iig.FirstOrDefault().User,
                        UserID = iig.FirstOrDefault().UserID
                    }).ToList();

//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.
public List<LastMeal> ingredientsFromLastMeals = (from meals in App.db.Meals
                    join ri in App.db.RecipeIngredients on meals.RecipeID equals ri.RecipeID
                    join i in App.db.Ingredients on ri.IngredientID equals i.ID
                    where meals.UserID == App.CurrentUser.ID && meals.Date <= DateTime.Now
                    group i by i.ID into igrouped
                    select new LastMeal()
                    {
                        IngredientID = igrouped.FirstOrDefault().ID,
                        IngredientCount = igrouped.Count()
                    }).ToList();

//A query that gets a list recipes where a blacklisted ingredient is used.
public List<int> blackList = (from bl in App.db.BlacklistIngredients
                    join ri in App.db.RecipeIngredients on bl.IngredientID equals ri.IngredientID
                    where bl.UserID == App.CurrentUser.ID
                    group ri by ri.RecipeID into ri
                    select ri.RecipeID into ri).RecipeID).ToList();

//get a list of graylisted ingredients.
public List<GrayList> grayList = (from gl in App.db.GraylistIngredients
                    join i in App.db.Ingredients on gl.IngredientID equals i.ID
                    where gl.UserID == App.CurrentUser.ID
                    group gl by gl.IngredientID into gl
                    select new GrayList()
                    {
                        Ingredient = gl.FirstOrDefault().Ingredient,
                        rating = (gl.Sum(r => r.IngredientValue) / gl.Count())
                    }).ToList();
#endregion
```

Food Planner

DS301E14

Intro

Problem statement &
Information Gathering

Quality Assurance

Unit Testing

Usability Testing

Reflection

Code Examples 24

```csharp
#region Methods
1reference
public IQueryable<IGrouping<int, Result>> search(List<int> recipeIDs)
{
    //IQueryable that finds all the ingredients that belong to the found recipese and how much of the found ingredient.
    IQueryable<IGrouping<int, Result>> recipeIngredients = from ri in App.db.RecipeIngredients
                                                           join i in App.db.Ingredients on ri.IngredientID equals i.ID
                                                           join r in App.db.Recipes on ri.RecipeID equals r.ID
                                                           where recipeIDs.Contains(ri.RecipeID)
                                                           group new Result()
                                                           {
                                                               recipe = ri.Recipe,
                                                               ingredient = ri.Ingredient,
                                                               quantity = ri.Quantity
                                                           } by ri.RecipeID;

    return recipeIngredients;
}
```

Food Planner

DS301E14

```csharp
101  public ObservableCollection<SearchResults> addValuesToSearch(IQueryable<IGrouping<int, Result>> userInput, List<string> searchKeywords = null)
102  {
103      //Create ObservableCollection which allows for updates in the view.
104      ObservableCollection<SearchResults> result = new ObservableCollection<SearchResults>();
105
106      //Traverse through all found ingredients and add them to the recipe
107      foreach (IGrouping<int, Result> ri in userInput)
108      {
109          Recipe recipe = ri.FirstOrDefault().recipe;
110
111          //initialize the searchResult that is to be shown
112          SearchResults searchResult = new SearchResults(recipe);
113
114          //Make sure that searchKeywords is set, if null assume that it's a non-search that need to have added value.
115          if (searchKeywords != null)
116          {
117              //check that the keyword is found in the recipe, it's case sensitive
118              if (searchKeywords.Any(s => recipe.Title.ToLower().Contains(s.ToLower())))
119              {
120                  searchResult.keyWordMatch++;
121              }
122          }
123
124          //Traverse through all results which is ingredients
125          foreach (Result res in ri)
126          {
127              //Add the ingredient to the recipe
128              searchResult.addIngredient(res.Ingredient);
129
130              //Check if the ingredient is found in the users' inventory
131              if (inventoryList.Where(il => il.IngredientID == res.ingredient.ID).Count() != 0)
132              {
133                  //If found check if it's a partial match or full match
134                  //full match means that all of the ingredient is found in the inventory
135                  if (inventoryList.Where(il => il.IngredientID == res.ingredient.ID).First().Quantity >= res.quantity)
136                  {
137                      searchResult.fullMatch++;
138                  }
139                  else
140                  {
141                      //if only partial match, and the partial match percentage to the partialMatch
142                      searchResult.partialMatch += inventoryList.Where(il => il.IngredientID == res.ingredient.ID).First().Quantity / res.quantity;
143                  }
```

Food Planner

DS301E14

Intro

Problem statement &
Information Gathering

Quality Assurance

Unit Testing

Usability Testing

Reflection

Code Examples 26