Iteration 3: RAD Document and First Implementation

Introduction

Purpose of The System

There are unlimited meals to cook. Every country has its own traditional and regional cuisine. We are working for you to cook your own traditional and regional meals and improve your imagination and taste. Improve your skills with Masterfridge.

Scope of The System

MasterFridge is the application that shows the recipes, according to users' foods in their own fridges. It can show food types according to the breakfast, lunch and dinner. Additionally, when the meal name is entered, the recipe can be seen. We have categorized recipes as breakfast, lunch and dinner to keep the program simple.

Objectives and Success Criteria of the Project

The success of the application depends upon meeting the following core set of objectives:

- The algorithm that provides to match recipes with foods in fridge.
- Making a design to make a choice from different variations of three meals during working time.
- To design an interface to enter new recipes by programmer.

Definitions, Acronyms, and Abbreviations

Important terms and concepts are listed here.

Sign Up

- User registers a system. In the meantime, two step verification is used.

Log In

- User logs in to the application.

Guest

- User can only searches and displays recipes.

Main Page

- Shows Log In and Sign Up. Just for displaying recipes, it includes Guest.

References

- http://docwiki.embarcadero.com/RADStudio/Rio/en/Mobile_Tutorial:_Creating_an_Application_for_Mobile_Platforms_(iOS_and_Android)
- http://docwiki.embarcadero.com/RADStudio/Rio/en/Creating_an_Android_App
- https://www.embarcadero.com/starthere/xe7/mobdevsetup/android/en/running_your_android_application_on_an_android_device.html
- https://www.embarcadero.com/products/delphi
- https://www.mysql.com/
- http://docwiki.embarcadero.com/RADStudio/Tokyo/en/Mobile_Tutorial:_Connecting_to_an_Enterprise_Database_from_a_Mobile_Client_(iOS_and_Android)

Overview

- This is the fridge application that helps people to cook with their fridge contents.
- It makes people to answer "What will I cook today?" question easier.
- It makes people to discover different tastes without too much effort.

Current System

We will put the complete version of it to Google Play/App Store as soon as our project is finished.

Proposed system

Overview

This section provides a functional overview of the system. This will again be properly be divided into two parts.

Functional Requirements

- This application can work on Android, IOS and PC.
- Every user can input their own foods on their fridges.
- When the application is installed, users can create their own accounts.
- Users shall be able to display recipes without registering.
- Users who are enrolled can choose any categories between breakfast, lunch and dinner.

Non Functional Requirements

Usability

- Every user should be able to have different ID.
- Users can make a choice between different repasts and different recipes.
- System has to show the video to the user if the recipe has a video.

Reliability

- Components of the project code will be tested alongside the implementation phase to ensure that they are functional.
- To check if the program is running on PC, Android and IOS devices, we will test it with various users and programmers.

Performance

- Recipes supposed to be fit in their own repasts and the match ought to be reasonable.
- Registered users' fridges' contents would be saved on database and they also would be editable.

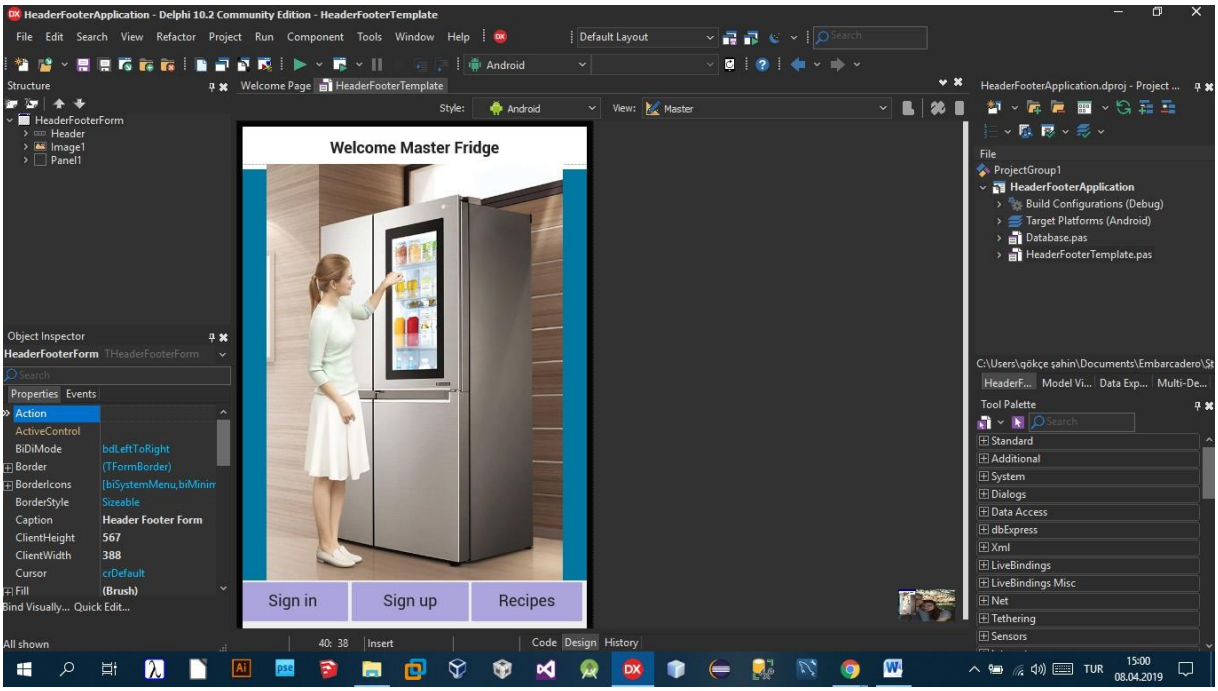
Supportability

- The application must not be platform dependent, i.e., it should be able to run on any platform supporting DELPHI.

Implementation

- Project will be implemented in DELPHI.
- All project graphical user interfaces will be created using a DELPHI Embarcadero editor.

Interface



Packaging

- Application packaging is a process of binding the relevant files and components to build a customized application for a customer.

Legal

The project will be active after taking the copyright and putting it in the market.

- If two or more people are registered as joint proprietors and all of them are deceased, then the Application by legal personal representative (APR) form must be used. The applicant(s) must be the executor(s) or administrator(s) of the last deceased joint proprietor.

System models

Scenarios

To get a recipe

- User logs in
- Meal selected
- Food options are displayed

For the registration

- User logs in
- Add nutrient
- View food categories

Use case model

Name:	Log In
Actor:	USER
Entry Conditions:	Application is running. SYSTEM is waiting user to enter the user name and password.
Flow of Events:	1. USER enters her/his user name and password. 2. SYSTEM verifies the information or displays LOGIN ERROR. 3. Meals are shown.
Exit Conditions:	SYSTEM is now in a new state.

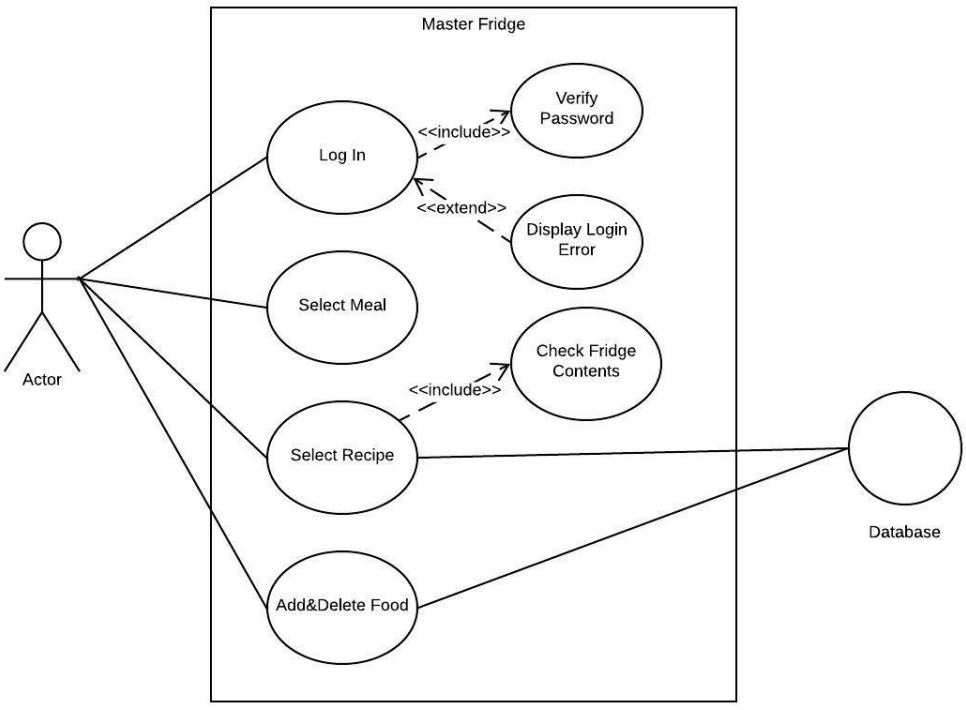
Name:	Verify Password
Actor:	USER
Entry Conditions:	SYSTEM is currently in progress.
Flow of Events:	1. SYSTEM verifies user's name and password. 2. Meals are shown.
Exit Conditions:	Now, system waits user to select meal. SYSTEM is now in a new state.

Name:	Display Login Error
Actor:	USER
Entry Conditions:	SYSTEM is currently in progress.
Flow of Events:	1. SYSTEM doesn't verify user's name and password. 2. Display error is shown.
Exit Conditions:	Now, system asks user to log in again.

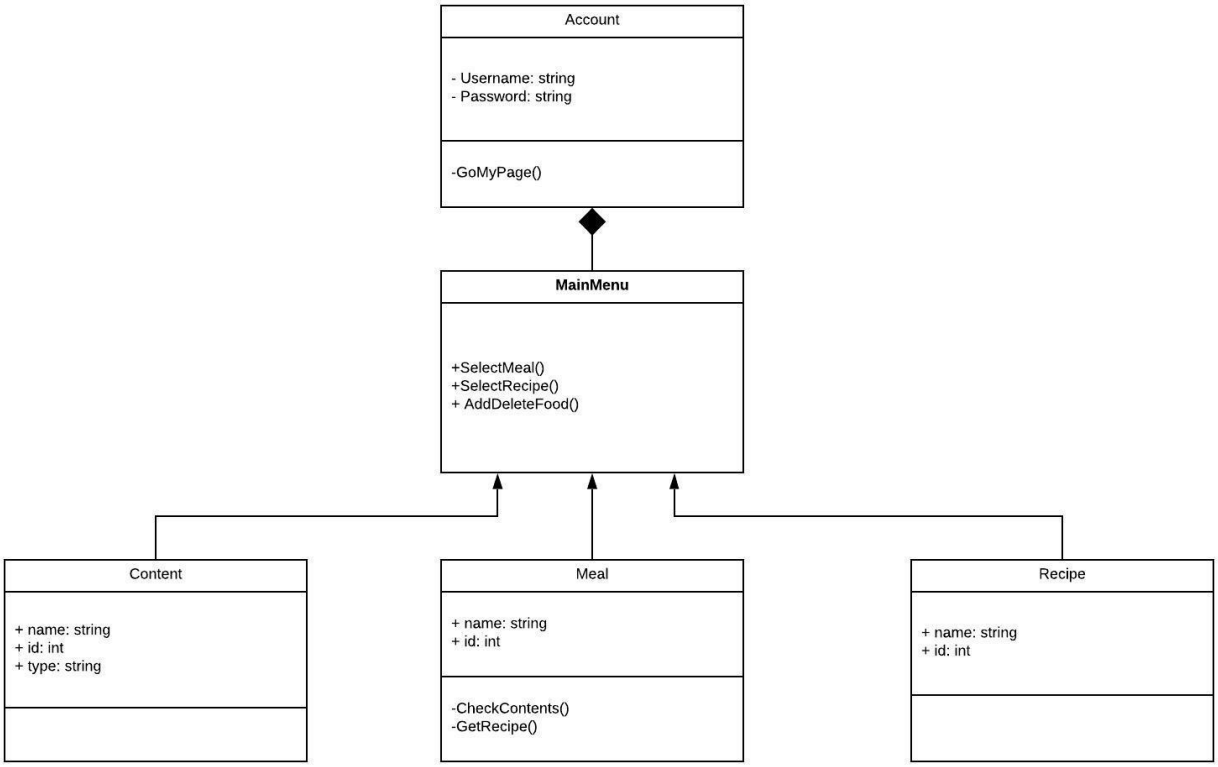
Name:	Select Meal
Actor:	USER
Entry Conditions:	SYSTEM is waiting user to select the meal that is needed to be prepared.
Flow of Events:	1. USER selects the meal that s/he wants to prepare. 2. Recipes are shown.
Exit Conditions:	System is waiting user to select the recipe that s/he can cook due to fridge contents.

Name:	Select Recipe
Actor:	USER
Entry Conditions:	SYSTEM is waiting user to select recipe.
Flow of Events:	1. USER selects the recipe that s/he wants to cook. 2. Recipes are shown.
Exit Conditions:	System is showing the recipe writing. If recipe has a video it also shows the video of its.

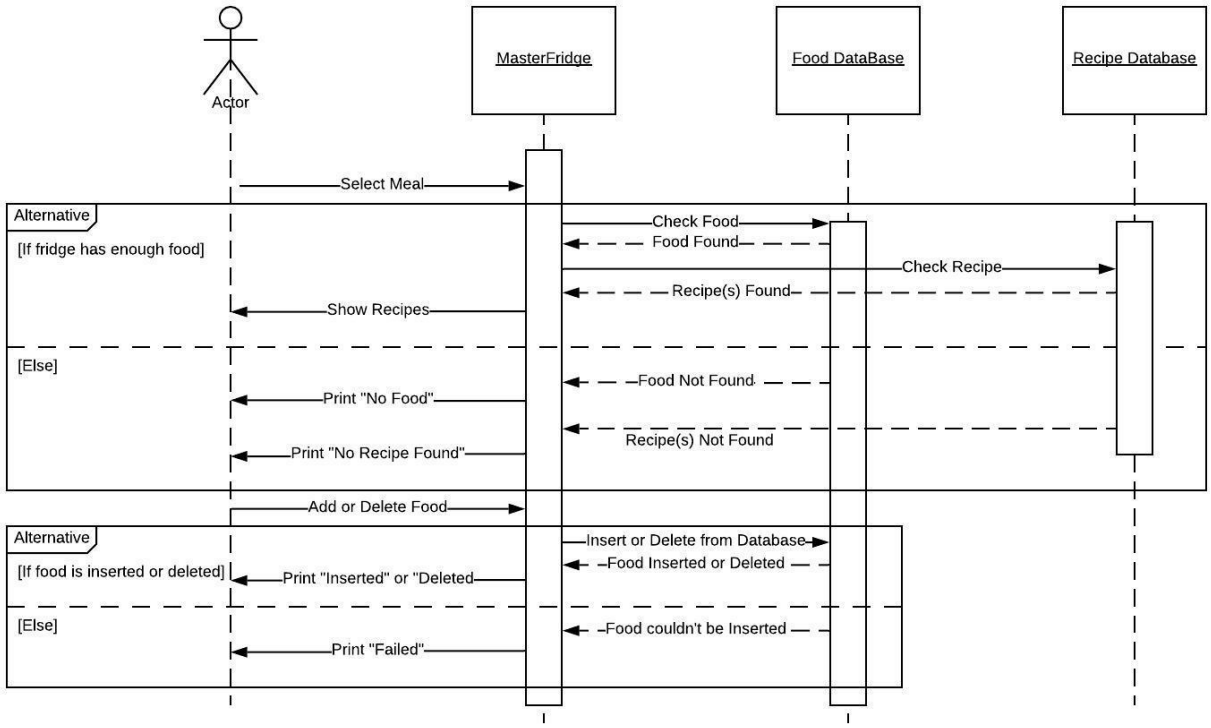
Name:	Add/Delete Food
Actor:	USER
Entry Conditions:	SYSTEM is waiting user to select the foods that are going to be added or deleted.
Flow of Events:	1. USER selects the food that are going to be added Or deleted. 2. Foods will be added or deleted form fridge database.
Exit Conditions:	System saves and updates the data.



Object model



Dynamic model



User interface-navigational paths and screen mock-ups

Glossary

- abstraction - The classification of phenomena into concepts. See also modeling.
- access control list - A representation of the access control matrix in which legitimate rights are represented as a list of (actor, operation) pairs attached to the class that is controlled.
- actor - External entity that needs to exchange information with the system. An actor can represent either a user role or another system.
- analysis - An activity during which developers ensure that the system requirements are correct, complete, consistent, and unambiguous. Analysis produces the analysis model.
- API - See Application Programmer Interface.
- argument - Value passed along with a message.
- base class - See superclass.
- branch - A concurrent development path under configuration management.
- bug - See fault.
- candidate key - In a database management system, a set of attributes that could be used as a primary key.
- change set - A set of deltas indicating the differences between two configurations.
- class diagram - UML notation representing the structure of the system in terms of objects, classes, attributes, operations, and associations. Class diagrams are used to represent object models during development.
- communication - An activity during which developers exchange information, either synchronously or asynchronously, and either spontaneously or according to a schedule.
- designer - See object designer.
- developer - Any role that is concerned with specifying, designing, and constructing subsystems. Examples of development roles include the analyst, the system architect, the object designer, and the implementor.
- end user - A role of the people who will use the delivered system.
- event - A relevant occurrence in the system. An event is an instance of an event class. Examples of events include a stimulus from an actor, a time-out, or the sending of a message between two objects.
- exit condition - A condition that are satisfied after the completion of a use case.
- implementation - An activity during which developers translate the object model into code.

- object design - An activity during which developers define custom objects to bridge the gap between the analysis model and the hardware/software platform. This includes specifying object and subsystem interfaces, selecting off-the-shelf components, restructuring the object model to attain design goals, and optimizing the object model for performance. Object design results in the object design model.
- participant - Any person involved with a software development project.
- Requirements Analysis Document (RAD) - The document describing the analysis model.
- sequence diagram - UML notation representing the behavior of the system as a series of interactions among a group of objects. Each object is depicted as a column in the diagram. Each interaction is depicted as an arrow between two columns. Sequence diagrams are used during analysis to identify missing objects, attributes, or relationships. Sequence diagrams are used during object design to refine the specification of classes. See also communication diagram.
- system design - An activity during which developers define the system design model, including the design goals of the project, and decompose the system into smaller subsystems that can be realized by individual teams. System design also leads to the selection of the hardware/software platform, persistent data management strategy, global control flow, access control policy, and boundary condition strategies.
- task model - A model of work for a project represented as tasks and their interdependencies.
- team - A set of participants who work on a common problem in a project.
- test case - A set of inputs and expected results that exercises a component with the purpose of causing failures.
- use case - A general sequence of interactions between one or more actors and the system. See also scenario.
- use case diagram - UML notation used during requirements elicitation and analysis to represent the functionality of the system. A use case describes a function of the system in terms of a sequence of interactions between an actor and the system. A use case also includes entry conditions that must be true before executing the use case and the exit conditions that are true at the completion of the use case.