|  |  |
| --- | --- |
|  | **MINISTRY OF EDUCATION AND TRAINING** |

**FPT UNIVERSITY**

|  |
| --- |
| **Capstone Project Document** |
| **Remindful Refrigerator Application** |

|  |  |
| --- | --- |
| **Group 2** | |
| **Group members** | **Nguyễn Hoàng Giang – SE61275**  **Trương Thanh Lâm – SE61257**  **Nguyễn Lê Hoàng Thiện – SE61277**  **Phan Hoàng Giáp – SE61347** |
| **Supervisor** | **Kiều Trọng Khánh** |
| **Ext. Supervisor** | **N/A** |
| **Capstone Project Code** | **RRA** |

- Ho Chi Minh City, January 04, 2016 -

Table of Contents

[Table of Contents 3](#_Toc441487142)

[List of Tables 5](#_Toc441487143)

[List of Figures 6](#_Toc441487144)

[Definitions, Acronyms and Abbreviations 7](#_Toc441487145)

[A. REPORT NO. 1: Introduction 8](#_Toc441487146)

[1. Project Information 8](#_Toc441487147)

[2. Introduction 8](#_Toc441487148)

[3. Current Situation 8](#_Toc441487149)

[4. Problem Definition 8](#_Toc441487150)

[5. Proposed Solution 9](#_Toc441487151)

[5.1. Feature functions 9](#_Toc441487152)

[5.2. Advantages 9](#_Toc441487153)

[5.3. Disadvantages 9](#_Toc441487154)

[6. Functional Requirements 9](#_Toc441487155)

[7. Roles and Responsibilities 10](#_Toc441487156)

[B. REPORT NO. 2: Software Project Management Plan 11](#_Toc441487157)

[1. Problem Definition 11](#_Toc441487158)

[1.1. Name of this Capstone Project 11](#_Toc441487159)

[1.2. Problem Abstract 11](#_Toc441487160)

[1.3. Project Overview 11](#_Toc441487161)

[1.3.1. Current Situation 11](#_Toc441487162)

[1.3.2. The Proposed System 11](#_Toc441487163)

[1.3.2.1. Mobile Application 12](#_Toc441487164)

[1.3.2.2. Website 12](#_Toc441487165)

[1.3.3. Boundaries of the System 12](#_Toc441487166)

[1.3.4. Future Plan 12](#_Toc441487167)

[1.3.5. Development Environment 13](#_Toc441487168)

[1.3.5.1. Hardware requirements 13](#_Toc441487169)

[1.3.5.2. Software requirements 13](#_Toc441487170)

[2. Project Organization 14](#_Toc441487171)

[2.1. Software Process Model 14](#_Toc441487172)

[2.2. Roles and Responsibilities 14](#_Toc441487173)

[2.3. Tools and Techniques 16](#_Toc441487174)

[3. Project Management Plan 16](#_Toc441487175)

[3.1. Software Development Life Cycle 16](#_Toc441487176)

[3.2. Phase Details 17](#_Toc441487177)

[3.2.1. Phase 1: Requirement Analysis and Definition 17](#_Toc441487178)

[3.2.2. Phase 2: System and Software Design 17](#_Toc441487179)

[3.2.3. Phase 3: Implementation and Unit Testing 18](#_Toc441487180)

[3.2.4. Phase 4: Integration and System Testing 18](#_Toc441487181)

[3.2.5. Phase 5: Operation and Maintenance 18](#_Toc441487182)

[3.3. All Meeting Minutes 19](#_Toc441487183)

[4. Coding Convention 19](#_Toc441487184)

[C. REPORT NO. 3: Software Requirement Specification 20](#_Toc441487185)

[1. User Requirement Specification 20](#_Toc441487186)

[1.1. Guest Requirement 20](#_Toc441487187)

[1.2. Member Requirement 20](#_Toc441487188)

[1.3. Staff Requirement 20](#_Toc441487189)

[1.4. Scheduler Requirement 20](#_Toc441487190)

[2. User Requirement Specification 20](#_Toc441487191)

[2.1. External Interface Requirement 20](#_Toc441487192)

[2.2. System Overview Use Case 21](#_Toc441487193)

[2.3. List of Use Case 21](#_Toc441487194)

List of Tables

[Table 1 - Roles and Responsibility 10](#_Toc443521783)

[Table 2 - Hardware Requirements for Server 13](#_Toc443521784)

[Table 3 - Hardware Requirements for Mobile 13](#_Toc443521785)

[Table 4 - Software Requirements 14](#_Toc443521786)

[Table 5 - Roles and Responsibilities Details 16](#_Toc443521787)

[Table 6 - Tools and Techniques 16](#_Toc443521788)

[Table 7 - Software Development Life Cycle 17](#_Toc443521789)

[Table 8 - Phase 1: Requirement gathering and analysis 17](#_Toc443521790)

[Table 9 - Phase 2: System design 18](#_Toc443521791)

[Table 10 - Phase 3: Implementation 18](#_Toc443521792)

[Table 11 - Phase 4: Testing 18](#_Toc443521793)

[Table 12 - Phase 5: Deployment of System 19](#_Toc443521794)

List of Figures

[Figure 1 - Waterfall model 14](#_Toc443521795)

[Figure 2 - System Overview Use Case 21](#_Toc443521796)

[Figure 3 - <Staff> Overview Use Case Diagram 22](#_Toc443521797)

[Figure 4 - <Staff> Input new website 22](#_Toc443521798)

[Figure 5 - <Staff> Set parser configuration 25](#_Toc443521799)

[Figure 6 - <Staff> Parse manually 28](#_Toc443521800)

[Figure 7 - <Staff> View logs 31](#_Toc443521801)

[Figure 8 - <Scheduler> Overview Use Case 33](#_Toc443521802)

[Figure 9 - <Scheduler> Parse automatically 33](#_Toc443521803)

[Figure 10 - <Scheduler> Create notification 36](#_Toc443521804)

[Figure 11 - <Scheduler> Send notification 38](#_Toc443521805)

[Figure 12 - <Scheduler> Match food 40](#_Toc443521806)

[Figure 13 - <Scheduler> Send suggestion 42](#_Toc443521807)

[Figure 14 - Conceptual Diagram 45](#_Toc443521808)

Definitions, Acronyms and Abbreviations

|  |  |
| --- | --- |
| Name | Definition |
| RRA | Remindful Refrigerator Application |
| BA | Business Analysis |
| DEV | Developer |
| SRS | Software Requirements Specification |
| ERD | Entity-Relationship Diagram |

1. REPORT NO. 1: Introduction
   1. Project Information

* Project name: Remindful Refrigerator Application
* Project code: RRA
* Product type: mobile application + web
* Start date: 04/01/2016
* End date:
  1. Introduction

In this document, we introduce a solution for tracking food in refrigerator. Almost every family has a refrigerator with a lot of food inside. But they cannot remember everything, especially petty things such as expiry date of food. Based on our analysis and our own experiences, we think of a solution for every people to manage their food easier.

Remindful Refrigerator Application (RRA) is built to solve current problem. RRA is convenient to manage information of all food in fridge, checking which food is expiring and notify to the user so that they can use it earlier instead of throwing it away. We also suggest some dishes which can be made with what the user have in their fridge.

This document also describes our working process in four months, includes our perspective in the system, component design, detailed core workflows. We hope our solution will help resolve the problems that everyone can face to.

* 1. Current Situation

About user’s behavior: Many people prefer to buy a lot of food by once and store them in their fridges for saving time. This behavior causes them to get confused about which food/ingredients they are having, their expiry date and how to cook with them.

Current solutions in the market: Samsung introduced a smart refrigerator product, which uses Android to store inside food’s information. However, those features are attached into Samsung’s own product, which means user have to buy a whole new fridge if they want to use the management function. Besides, its interface is inconvenient for users, as the inputting keyboard is huge and slowing down typing. One more problem is users will have difficulties interact with the product from distance, as they cannot bring their fridge along with them.

* 1. Problem Definition

There are disadvantages of current situation:

* With a lot of food in the fridge, people sometimes do not remember or care about the expiration but keep using those food. Using expired food may affect to people heath, and even if they do not use them, it also be a waste.
* People may be confused with a lot of food and ingredients. They do not know what or how to cook with those food.

About the product of SAMSUNG:

* Advantages:
* Friendly interface.
* Have some additional features such as Music and Video player.
* Disadvantages:
* Not flexible: user must buy the whole fridge to use the provided features.
* Wasting energy: the large tablet integrated with the fridge require more energy to work.
* The system is just a demo.
  1. Proposed Solution

According to the growth of smart phone, we proposed a solution is to build a mobile application called “Remindful Refrigerator Application” (or shortly RRA). RRA includes a mobile app and an admin panel with following functions:

* + 1. Feature functions

RRA provides these following core functions:

* Add food: user can add food information to database.
* Search dishes: according to food that user have in fridge, they can manually search for appropriate dishes when they need.
* Make notification: with expiring food, system will send notification to user.
* Suggest dishes: with expiring food, system will give some dishes that user can make with that food.
  + 1. Advantages
* User can easily keep track on their food. They can know which food should be used first.
* User can get suggestion about dishes can be cook with their food.
* Application is compatible with most of Android smartphone.
* Flexible: application can be used dependently from fridges.
  + 1. Disadvantages
* There are many competitive in market.
  1. Functional Requirements

Functional requirements are listed as below:

* Mobile app:
* New account/Login
* Add new food
* Get notification
* Get suggestion
* Search dishes
* Add dishes to favorite list
* Admin panel:
* Input new resources website
* View report
* Scheduler:
* Send notification
* Make suggestion
* Parse recipes
* Create report
  1. Roles and Responsibilities

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No | Full name | Role | Position | Contact |
| 1 | Kiều Trọng Khánh | Project manager | Supervisor | khanhkt@fpt.edu.vn |
| 2 | Nguyễn Hoàng Giang | Developer | Leader | giangnhse61275@fpt.edu.vn |
| 3 | Trương Thanh Lâm | Developer | Member | lamttse61257@fpt.edu.vn |
| 4 | Nguyễn Lê Hoàng Thiện | Developer | Member | thiennlhse61277@fpt.edu.vn |
| 5 | Phan Hoàng Giáp | Developer | Member | giapphse61347@fpt.edu.vn |

Table - Roles and Responsibility

1. REPORT NO. 2: Software Project Management Plan
   1. Problem Definition
      1. Name of this Capstone Project

* Official name: Remindful Refrigerator Application
* Vietnamese name: Tủ lạnh tiện lợi
* Abbreviation: RRA
  + 1. Problem Abstract

To support users to keeping track of their food, we concentrate on one thing familiar with people nowadays - their smart phones. We provide a mobile application that give users a helpful method to manage food in their fridges, as well as those foods’ expired date. The application also aids users by reminding them about incoming expired date in case they forget. In addition, we develop a web site as an alternative using way for the application in some situations.

Moreover, the application can suggest some dishes based on the ingredients in the fridge. It also provides users with recipes if they are interested in a certain one. To support making such suggestions, we have a team that continuously looking for good dishes from many culinary websites.

* + 1. Project Overview
       1. Current Situation

Some problems encountered in this project:

* User’s behavior: Users may forget to update data of their food when changes occur. Example: buying or using some food
* Rarity of food: Some food may be rare or in short use, hardly finding matching dishes.
* Variety of users: People have different favorite tastes. The work of finding dishes to satisfy all of them is very difficult.
* Gathering data: culinary websites have different interfaces and structures, which makes gathering data more difficult.
  + - 1. The Proposed System

We will build a mobile application as the main functions for users. The application consumes web services from the server to perform its assignments. The application can cache data such as viewed dishes, favorite dishes and personal settings.

We also develop a web site with some same functions from the mobile application to make an alternative. This helps ensuring the availability of the system.

The system has a scheduler service to perform “scheduled” tasks such as notify users and parse information periodically.

We have an internal site for staff team to gather data about dishes and recipes from culinary websites.

* + - * 1. Mobile Application

Mobile Application is the mainly part used by users. It has the following core functions:

* Manage food and their expired date
* Scan barcode with camera for quick adding new food
* Notification about expiring food
* Suggest dishes and view recipes
* Caching data for using without internet connection
  + - * 1. Website

Web Site is an essential part of the system. The user’s site provides some functions like the mobile application, while the staff’s site performs gathering data task.

* For users:
  + Manage food
  + Suggest dishes and view recipes
* For staffs:
  + Get data from culinary sites
    - 1. Boundaries of the System

This system can:

* Allows users to manage food that are put in their refrigerator.
* Notify users about expiring food.
* Suggest dishes to user based on the food user currently have, using data gathered from culinary websites.
  + - 1. Future Plan

Current system has simple managing and suggesting functions. With further research, the system can apply high techniques of data mining to develop expanded suggestion functions, such as:

* Nutrition scaled dishes: suggest dishes with a balanced or suitable nutrition ingredients for different meals of a day (breakfast, lunch, dinner), or for different types of people (diet, growing children).
* Food and news: suggest news about food for user.
  + - 1. Development Environment
         1. Hardware requirements
* For Server

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum** | **Recommended** |
| Internet Connection | Cable, Wi-Fi (4 Mbps) | Cable, Wi-Fi (8 Mbps) |
| Operating System | Winder Server 2008 | Winder Server 2008 |
| Computer Processor | Intel® Xeon® 1.4GHz | Intel® Xeon® Quad Core (12M Cache, 2.50 Ghz) |
| Computer Memory | 1GB RAM | 2GB or more |

Table - Hardware Requirements for Server

* For Mobile

|  |  |  |
| --- | --- | --- |
| **Windows** | **Minimum** | **Recommended** |
| Internet Connection | Wi-Fi or 3G (4 Mbps) | Wi-Fi or 3G (8 Mbps) |
| Operating System | Android 4.2.2 | Android 4.4.2 |
| Computer Processor | Cortex-A7 Dual-Core 1.3GHz | Cortex-A7 Dual-Core 1.3GHz |
| Computer Memory | 1GB RAM | 2GB or more |

Table - Hardware Requirements for Mobile

* + - * 1. Software requirements

|  |  |  |
| --- | --- | --- |
| **Software** | **Name/Version** | **Description** |
| Operating system | Windows 7 or higher | Operating system and platform for development |
| Environment | Java EE 6 | Specification for developing |
| Modeling tool | StarUML 2.0 | Used for data modeling |
| IDE | Netbeans 7.4 or 8.0.1  Android Studio 1.5.1 | Programming tools |
| DBMS | Microsoft SQL Server 2008 | Used to create and manage the database for the system |
| Source control | TortoiseSVN 1.8.2 | Used for source control |
| Web Browser | Chrome 42 or above  Internet Explorer 9 or above | Testing browser |
| Android Emulator | Genymotion 2.6.0 | Android emulator used for testing mobile application |

Table - Software Requirements

* 1. Project Organization
     1. Software Process Model



Figure - Waterfall model

Reference: SOFTWARE ENGINEERING 9th Edition, by Ian Sommerville.

The Waterfall Model is a linear-sequential process used in software development. This model requires each phase to be fully completed before the next phase can begin. This model is used for small project, which has no uncertain requirements. We choose waterfall model for RRA project because of these following reasons:

* RRA is a small project, which must be done in a short time (14 weeks) with only four members.
* Team members are inexperience since no one has not done any real project, so that all requirements must be well-defined and stay stable for team to work better. Other models (such as Agile) are not appropriate for us because of the limited skills.
* All project aspects are clear to team members and hence every member can understand the whole system instead of their parts only.
* Team meetings are held frequently to keeps the coherence of the team as well as the project.
  + 1. Roles and Responsibilities

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Full name** | **Role in Group** | **Responsibilities** |
| **1** | Kiều Trọng Khánh | Project manager | * Specify user requirements * Control the development process * Give out technique and business analysis support |
| **2** | Nguyễn Hoàng Giang | Team leader, BA, DEV, Tester | * Managing process * Designing database * Clarifying requirements * Prepare documents * GUI design * Create test plan * Coding * Testing |
| **3** | Trương Thanh Lâm | Team member, BA, DEV, Tester | * Clarifying requirements * Prepare documents * GUI Design * Designing database * Coding * Create test plan * Testing |
| **4** | Nguyễn Lê Hoàng Thiện | Team member, BA, DEV, Tester | * Clarifying requirements * Prepare documents * GUI Design * Designing database * Coding * Create test plan * Testing |
| **5** | Phan Hoàng Giáp | Team member, BA, DEV, Tester | * Clarifying requirements * Prepare documents * GUI Design * Designing database * Coding * Create test plan * Testing |

Table - Roles and Responsibilities Details

* + 1. Tools and Techniques

|  |  |
| --- | --- |
| **Tool/Technique** | **Name/Version** |
| Front-end | HTML, CSS5, JavaScript, jQuery, Boostrap |
| Back-end | JavaEE, Servlet, JSP |
| Database Management System | Microsoft SQL Server 2008 R2 Express |
| Mobile | Android Studio 1.5.1 |
| Web Server | Apache Tomcat 7.0.42 |
| Scheduler Framework | Quartz Scheduler |
| Parser API | jSoup, html Unit |

Table - Tools and Techniques

* 1. Project Management Plan
     1. Software Development Life Cycle

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Phase** | **Description** | **Deliverables** | **Resources needed** | **Dependencies and Constrains** | **Risks** |
| Requirement Analysis and Definition | - Collect requirements.  - Identify and clarify requirements. | - Software requirements specification. | 10 man-days | N/A | - Missing requirements.  - Unclear scope of project.  - Lack of member share of understand. |
| System and Software Design | - Architectural design  - Interface design  - Component design  - Database design | - System architecture  - Database specification  - Interface specification  - Component specification | 15 man-days | Depend on “Requirement Analysis and Definition” | - Lack of experience.  - Not fulfil requirements. |
| Implementation and Unit Testing | - Implement system design into real system. | - Web application.  - Mobile application on Android.  - Physical database. | 60 man-days | Depend on “System and Software design” | - Mistake while implementing. |
| Integration and System Testing | - Execute the system with test cases  - Fix bugs | - Test report. | 15 man-days | Depend on “Implementation and Unit Testing” | - Lack of experience.  - Lack of test case. |
| Operation and Maintenance | - Deploy on web server and mobile. | - Installation guide.  - User manual. | 10 man-days | Depend on “Integration and System Testing” | - Lack of experience. |

Table - Software Development Life Cycle

* + 1. Phase Details
       1. Phase 1: Requirement Analysis and Definition

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| 1. Collect requirements | Find which systems currently provide similar service, their strength and weakness. | GiangNH, LamTT, ThienNLH, GiapPH |
| 2. Identify and clarify main functions | Define which main function system should provide. | GiangNH, LamTT, ThienNLH, GiapPH |
| 3. Requirements specification | Define the requirements in details.  Write SRS document. | GiangNH, LamTT, ThienNLH, GiapPH |
| 4. Requirement validation | Check the validity of the requirements and complete SRS document. | GiangNH, LamTT, ThienNLH, GiapPH |

Table - Phase 1: Requirement gathering and analysis

* + - 1. Phase 2: System and Software Design

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| 1. Architectural design | Define the overall structure of the system.  Define the main components, the relationships between them and how they are distributed. | GiangNH, LamTT, ThienNLH, GiapPH |
| 2. Interface design | Define the interfaces between system components. | GiangNH, LamTT, ThienNLH, GiapPH |
| 3. Component design | Design how each component will operate. | GiangNH, LamTT, ThienNLH, GiapPH |
| 4. Database design | Design the system data structures and how these are to be represented in a database. | GiangNH, LamTT, ThienNLH, GiapPH |

Table - Phase 2: System design

* + - 1. Phase 3: Implementation and Unit Testing

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| 1. Implement web back-end | Code functions for web application and web services for mobile to consume. | LamTT |
| 2. Implement web front-end | Code GUI for web application | LamTT |
| 3. Implement mobile application | Code GUI and functions for mobile application | GiangNH |
| 4. Implement scheduler | Code functions of scheduler | GiapPH |
| 5. Implement parser | Code parser | ThienNLH |
| 6. Unit testing | Write test cases and run to ensure that code meets the designs and behaves as expected. | GiangNH, LamTT, ThienNLH, GiapPH |

Table - Phase 3: Implementation

* + - 1. Phase 4: Integration and System Testing

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| 1. Component testing | Test each component independently  Fix bugs | GiangNH, LamTT, ThienNLH, GiapPH |
| 2. System testing | Test the whole system.  Fix bugs | GiangNH, LamTT, ThienNLH, GiapPH |
| 3. Acceptance testing | Testing with customer data to check that the system meets the customer’s needs. | GiangNH, LamTT, ThienNLH, GiapPH |

Table - Phase 4: Testing

* + - 1. Phase 5: Operation and Maintenance

|  |  |  |
| --- | --- | --- |
| **Task** | **Description** | **Author** |
| 1. Maintain the system | Maintain current system | N/A |

Table - Phase 5: Deployment of System

* + 1. All Meeting Minutes

All meeting minutes are saved at:

<https://github.com/Giang94/CapstoneProject_SmartRefrigerator/tree/master/Doc/Meeting%20Minutes>

* 1. Coding Convention

Using Java coding convention to develop website, parser, web services and mobile application.

Conventions:

* Declaration:
* One declaration per line.
* Declarations should be placed only at the beginning of blocks.
* Naming:
* Variable and method names are in mixed case, with first letter of each internal word capitalized except first word.
* Method names should be verbs.
* Class names should be nouns, in mixed case with first letter of each internal word capitalized.
* Constant names should be all uppercase with words separated by underscore.
* Indentation:
* Avoid lines longer than 80 characters.

References: <http://www.oracle.com/technetwork/java/codeconventions-150003.pdf>

1. REPORT NO. 3: Software Requirement Specification
   1. User Requirement Specification
      1. Guest Requirement

Guest is a person who doesn’t have access to the system. Guest can only use two functions in the system. To use others functions, guest must login. These are two functions guest can use:

* Register: create a new account so that guest can login to the system.
* Login: after login, guest will have a role in the system and can use appropriate functions.
  + 1. Member Requirement

Member is a person who logged in the system. Member is the main user of the system and can use some function in the system. A member can use these following functions after logged into the system with member account:

* Manage food:
* Add new food
* Edit existing food
* Remove food
* Search dish
* View recipe
* Add dish to favorite list
  + 1. Staff Requirement

Staff is a person with permission to manage some aspect of the system. Staff can use these functions after logged into the system with staff account:

* Add new website
* View log
* Parse recipe
  + 1. Scheduler Requirement

Scheduler is a part of RRA allows system to do some functions automatically. These automatic functions are:

* Parse recipe
* Match dish
* Send notification
* Write log
  1. User Requirement Specification
     1. External Interface Requirement
     2. System Overview Use Case

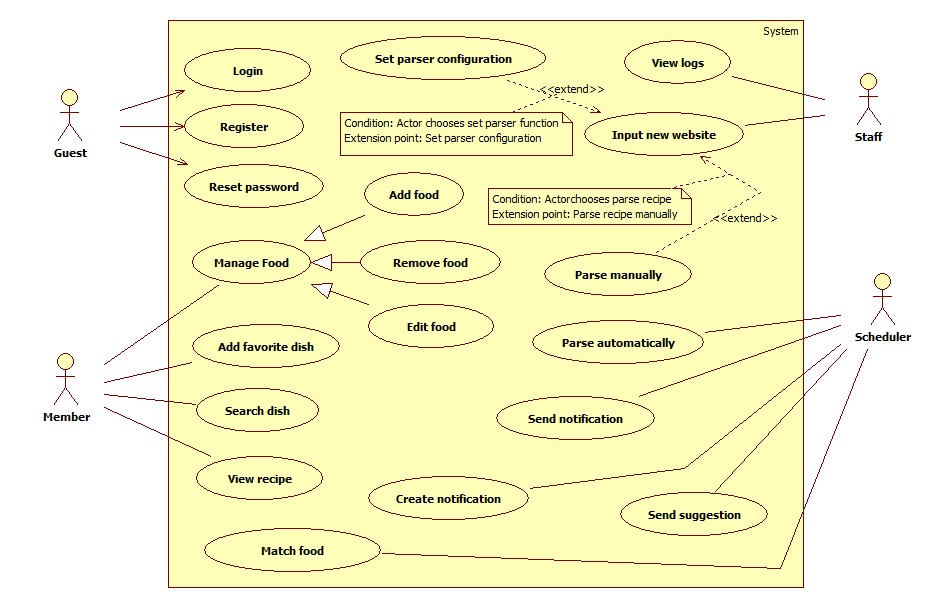


Figure - System Overview Use Case

* + 1. List of Use Cases
       1. <Guest> Overview Use Cases
          1. <Guest> Login (UC\_01)
          2. <Guest> Register (UC\_02)
          3. <Guest> Reset password (UC\_03)
       2. <Member> Overview Use Cases
          1. <Member> Add food (UC\_04)

**Use Case Diagram**

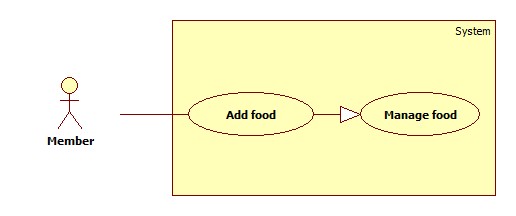


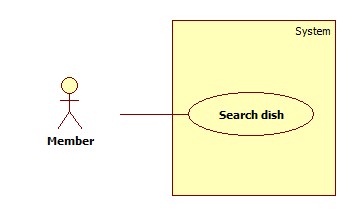
Figure - <Member> Add food

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_04** | | | |
| **Use Case No.** | 04 | **Use Case Version** | 1.0 |
| **Use Case Name** | Add food | | |
| **Author** | Trương Thanh Lâm | | |
| **Date** | 25/01/2016 | **Priority** | High |
| **Actor:**  Member  **Summary:**  This use case allows members to add food. This use case is applied for both mobile application and web interface.  **Goal:**  Member can use this method to add food to his/her fridge after acquiring food in reality.  **Triggers:**  Member sends Add food command  **Preconditions:**  Actor must logged in with role Member  **Post Conditions:**  **Success**: New food is added to the system  **Fail**: Error message is shown  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Member accesses Add food view | System shows view and requires new food information:  Name: free text input, required, length [2,30]  Quantity: free text input, integer value (0,1000]  Unit: choose from list  Expired date: date, required, must be later than or equals current date  Notification date: number, required. | | **2** | Member inputs new food information  [Alternative 1] |  | | **3** | Member sends Add food command | New food is added to the system  [Exception 1]  [Exception 2]  [Exception 3] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | **1** | Member selects Scan barcode option  Member uses camera to scan barcode  Member inputs remaining information: Quantity, Unit, Expired date, Notification date | System activates camera to scan food’s barcode  System recognizes barcode and fills food name |   **Exception**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | **1** | Member inputted food name with inappropriate length | System shows error message: “Food name must be between 2 and 30 characters.” | | **2** | Member inputted quantity with a value less than 0 or greater than 1000 | System shows error message: “Quantity must be greater than 0 and less or equals to 1000.” | | **3** | Member inputted expired date sooner than current date | System shows error message: “Expired date must be later or equals current date.” |   **Relationships: N/A**  **Business Rules:**  Unit can be kilogram, gram, can, bottle, piece, box... or user manually input.  The default value of notification day is 3 days  After adding, food has status “Pending” | | | |

* + - * 1. <Member> Remove food (UC\_05)
        2. <Member> Edit food (UC\_06)
        3. <Member> Add favorite dish (UC\_07)
        4. <Member> Search dish (UC\_08)

**Use Case Diagram**



**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_08** | | | |
| **Use Case No.** | 08 | **Use Case Version** |  |
| **Use Case Name** | Search dish | | |
| **Author** | Trương Thanh Lâm | | |
| **Date** | 25/01/2016 | **Priority** | High |
| **Actor:**  Member  **Summary:**  This use case allows members to search dishes by name. This use case is applied for both mobile application and web interface.  **Goal:**  Member can use this method to search for dishes. After inputting dish name, some suggested dishes are displayed.  **Triggers:**  Member changes text in search field  **Preconditions:**  Actor must logged in with role Member  **Post Conditions:**  Success: Search result is shown  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Member accesses Search dish view | System shows view and requires search information:  - Search text: free text input, required, length [2,50] | | **2** | Member inputs dish name | System searches dishes based on name and displays result  [Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | **1** |  | If no record is matched, system displays message “No record is matched.” |   **Exception: N/A**  **Relationships:**  Add favorite dish  View recipe  **Business Rules:**  Searching by comparing search text and dish name. If dish name contains search text, the dish is selected.  Search result is sorted by founded order.  Search result is displayed as a table contains dishes’ names, short description and options for Member to Add favorite dish, View recipe | | | |

* + - * 1. <Member> View recipe (UC\_09)
      1. <Staff> Overview Use Cases

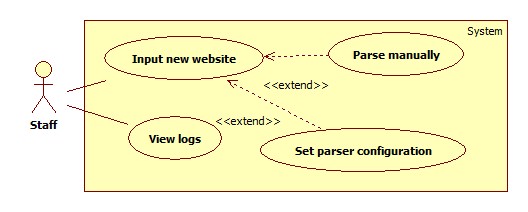


Figure - <Staff> Overview Use Case Diagram

* + - * 1. <Staff> Input new website (UC\_10)

**Use Case Diagram**

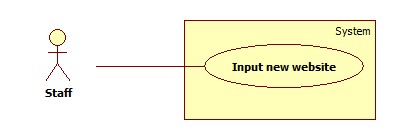


Figure - <Staff> Input new website

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_10** | | | |
| **Use Case No.** | 10 | **Use Case Version** | 1.0 |
| **Use Case Name** | Input new website | | |
| **Author** | Nguyễn Lê Hoàng Thiện | | |
| **Date** | 21/01/2016 | **Priority** | High |
| **Actor:**  Staff.  **Summary:**  This use case helps staff input new culinary website in order to set parser configuration or parse recipe.  **Goal:**  Input new culinary website and load its resource to view.  **Triggers:**  Staff sends “Input new website” command.  **Preconditions: N/A**  **Post Conditions:**  **Success**: The content of input website is loaded.  **Fail**: Show error messages.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Staff goes to Input new website view. | System shows view and requires information from staff:  URL of new website: free text input, required, length [10,250] | | 2 | Staff inputs information |  | | 3 | Staff sends command to load the input website. | System connects to the input website, shows all resources of the website to view.  [Exception 1]  [Exception 2] |   **Alternative Scenario: N/A**  **Exception**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | Staff sends Load new website command with invalid input data.  The input URL is not long enough or longer than 250 characters. | System will show error message: “The URL is too short or too long. Please try again” | | 2 | The system cannot connect to input website.  The given URL is not available. | System will show error message: “Cannot load website. Please check the URL and try again” |   **Relationships:** Set parser configuration, Parse manually.  **Business rules:**  The contents of input website will be showed in the staff’s view.  The contents of website includes: text, images, etc. | | | |

* + - * 1. <Staff> Set parser configuration (UC\_11)

**Use Case Diagram**

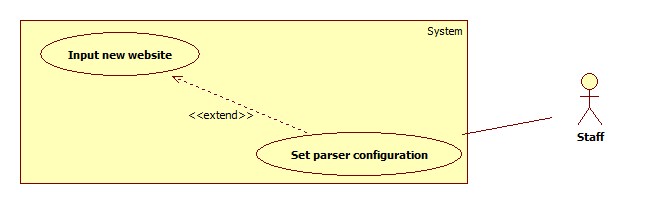


Figure - <Staff> Set parser configuration

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_11** | | | |
| **Use Case No.** | 11 | **Use Case Version** | 1.0 |
| **Use Case Name** | Set parser configuration | | |
| **Author** | Nguyễn Lê Hoàng Thiện | | |
| **Date** | 21/01/2016 | **Priority** | High |
| **Actor:**  Staff.  **Summary:**  This use case allows staff to set configuration of input website for auto parsing resources.  **Goal:**  Set configuration for each website for auto parsing.  **Triggers:**  Staff sends “Set configuration” command.  **Preconditions:**  Staff passes inputting new website stage.  **Post Conditions:**  **Success**: New configuration file is inserted to storage.  **Fail**: Show error message.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Staff goes to Set parser configuration view. | System loaded all contents of input website to view.  System provided the wizard for staff to get data from input website. | | 2 | Staff chooses each necessary element for recipe. | System gets XPath of chosen element and shows a pop-up contain XPath and data every chosen element.  System shows notification when staff completes each element of recipe.  System stores XPath into temporary storage. | | 3 | Staff finishes choosing and sends “Preview” command.  [Alternative 1] | System compares all XPaths to get common XPath for each element.  System opens a new view and shows a table containing final data of recipe parsed from XPaths. | | 4 | Staff sends update new configuration command.  [Alternative 2]  [Alternative 3] | The configuration file is updated in to system’s storage. |   **Alternative Scenario**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Staff sends “Preview” command before completing choosing elements or staff doesn’t choose enough the necessary elements. | System shows the warning message: “You do not choose enough of the necessary elements. Please continue choosing.” | | 2 | Staff closes the preview view. | System closes the preview view. The data is unchanged. | | 3 | Staff sends reset command. | All gotten data are cleared. The preview view is closed. |   **Exception: N/A**  **Relationships:** Input new website.  **Business rules:**  Staff must click the elements in the order: Recipe name – Introduction – Recipe image – Material – Recipe step.  With Recipe name, Introduction and Recipe image, staff just need click one time for each element.  With Materials and Steps, because they have more than one item, so that staff must clicks 3 times for every needed element.  If the XPath of Materials or Steps does not have the same rule, staff must parse that recipe manually.  When staff finishes updating parser configuration file, system will update “PARSER CONFIGURATION FILE”. If the input website’s configuration is already existed, this will be updated. If it’s not existed, it will be inserted to the configuration file with the given structure.  **File parser configuration structure:** PARSER CONFIGURATION FILE  File name: parserconfig.xml  Time updated file: {Created date}, {Create time}  <?xml version="1.0" encoding="UTF-8"?>  <configurations>  <configuration site="[site\_name\_1]">  <recipename>…</recipename>  <introduction>…</introduction>  <recipeImage>…</recipeImage>  <material>…</material>  <recipestep>…</recipestep>  </configuration>  ...  <configuration site="[site\_name\_n]">  ...  </configuration>  </configurations> | | | |

* + - * 1. <Staff> Parse manually (UC\_12)

**Use Case Diagram**

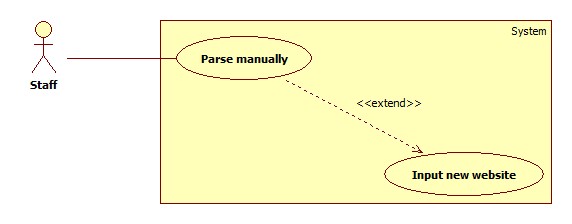


Figure - <Staff> Parse manually

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_12** | | | |
| **Use Case No.** | 12 | **Use Case Version** | 1.0 |
| **Use Case Name** | Parse manually | | |
| **Author** | Nguyễn Lê Hoàng Thiện | | |
| **Date** | 21/01/2016 | **Priority** | High |
| **Actor:**  Staff.  **Summary:**  This use case helps staff parse a new recipe manually.  **Goal:**  Parse a new recipe to system’s storage from the input website.  **Triggers:**  Staff sends “Parse manually” command.  **Preconditions:**  User logged into the system with Staff role.  Staff passes inputting new website stage.  **Post Conditions:**  **Success**: The new recipe is inserted to storage.  **Fail**: Show error messages. Storage is unchanged.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Staff goes to Parse manually view. | System loaded all resource of input website to view. | | 2 | Staff clicks each necessary element for recipe. | System gets XPath of clicked element and shows a pop-up contain XPath and data every element-click.  System storage XPath into temporary storage. | | 3 | Staff sends “Preview” command.  [Alternative 1] | System opens a new view and shows a table containing final data.  [Exception 1] | | 4 | Staff sends Save command.  [Alternative 2]  [Alternative 3] | The parsed data is inserted to the system’s storage. |   **Alternative Scenario**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Staff sends “Preview” command before completing choosing elements or staff doesn’t choose enough the necessary elements. | System shows the warning message: “You do not choose enough of the necessary elements. Please continue choosing.” | | 2 | Staff closes the preview view. | System closes the preview view. The data is unchanged. | | 3 | Staff sends reset command. | All gotten data are cleared. The preview window is closed. |   **Exception**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | System cannot parse information from one or some XPath. | System will show error message for each XPath that cannot be parsed: “This XPath cannot be parsed.” |   **Relationships:** Input new website.  **Business rules:**  Staff must click the elements in the order: Recipe name – Introduction – Recipe image – Materials – Recipe steps.  When system show the table containing the information parsed, staff can edit this table. Staff can do that when there are some XPaths cannot be parsed.  Recipes data must be validated before saved to database.  If the parsed recipe exist in storage, update the recipe. | | | |

* + - * 1. <Staff> View logs (UC\_13)

**Use Case Diagram**

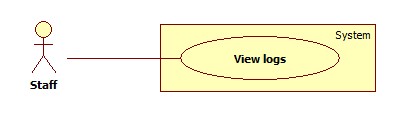


Figure - <Staff> View logs

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_13** | | | |
| **Use Case No.** | 13 | **Use Case Version** |  |
| **Use Case Name** | View logs | | |
| **Author** | Nguyễn Lê Hoàng Thiện | | |
| **Date** | 21/01/2016 | **Priority** | Normal |
| **Actor:**  Staff  **Summary:**  This use case allows staff view logs from systems.  **Goal:**  Staff can view logs created by system.  **Triggers:**  Staff sends “View logs” command.  **Preconditions:**  User logged into the system with Staff role.  **Post Conditions:**  **Success:** Logs’ state are changed from “pending” to “read”.  **Fail:** The logs’ status will not be changed.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | 1 | Staff goes to View logs view. | System shows view.  System loads all logs with state “pending” to the table of view.  [Exception 1] | | 2 | Staff sends “Complete” command.  [Alternative 1] | System changes states of selected log from “pending” to “read”. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | 1 | Staff sends “Complete all” command. | System changes states of all logs from “pending” to “read” |   **Exception**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | 1 | System cannot read the log file. | System will show error message: “Cannot read log file.” |   **Business rules**   * The new log is created with state “pending”. When it’s read, its state changes from “pending” to “read”. * Logs are shown descending by created date. | | | |

* + - 1. <Scheduler> Overview Use Case

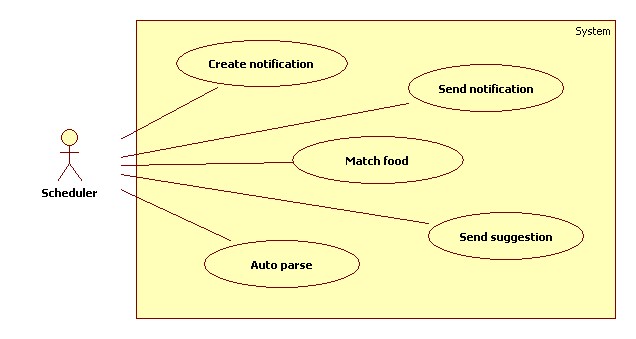


Figure - <Scheduler> Overview Use Case

* + - * 1. <Scheduler> Parse automatically (UC\_14)

**Use Case Diagram**

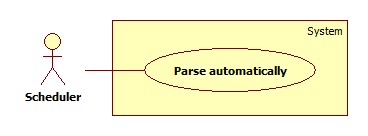


Figure - <Scheduler> Parse automatically

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_14** | | | |
| **Use Case No.** | 14 | **Use Case Version** | 1.0 |
| **Use Case Name** | Parse automatically | | |
| **Author** | Phan Hoàng Giáp | | |
| **Date** | 19/01/2016 | **Priority** | High |
| **Actor:**  Scheduler.  **Summary:**  This use case allows scheduler to parse resource automatically from culinary websites at specified time.  **Goal:**  Scheduler can get resources from culinary websites.  **Triggers:**  The time hit configured time.  **Preconditions:**  Configuration files have been created.  Parse time has been configured.  **Post Conditions:**  **Success**: New data is inserted to storage. Log file is generated.  **Fail**: Log file is generated.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Server checks the current time. If it hits configured time, parse process starts. | Send request to the chosen websites.  Fetch data from the response based on configuration files.  Data is saved to temporary storage.  Validate data [Exception 1].  Fetched link resource is not exist yet, insert to storage [Alternative 1].  Generate log file. |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | **1** | The fetched link resource is already in the storage. | Update its information. |   **Exception**   |  |  |  | | --- | --- | --- | | **No** | **Cause** | **System Response** | | **1** | Data is invalid | Generate log file. |   **Relationships:** N/A  **Business Rules:**  Parse time can be configured by staff or it is set as default at 0 o’clock on Monday.  At configured time, system will send request to parsed websites.  System fetch data and insert to storage.  Recipes data must be validated before saved to database.  Valid data: parsed resource is correct with the configuration files  If fetched resource exists in storage, update its information.  **Log file structure:** AUTO PARSE PROCESS LOG FILE  Filename: parserecipe.log  Created date: , Create time:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | STT | Website parse | Elapsed time | Total recipes received | Insert successful | Insert failed | |  |  |  |  |  |  | |  |  |  |  |  |  |   Total elapsed time:  Total parsed recipes: | | | |

* + - * 1. <Scheduler> Create notification (UC\_15)

**Use Case Diagram**

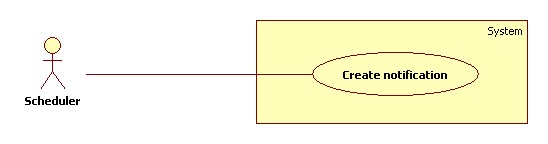


Figure - <Scheduler> Create notification

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_15** | | | |
| **Use Case No.** | 15 | **Use Case Version** | 1.0 |
| **Use Case Name** | Create notification | | |
| **Author** | Phan Hoàng Giáp | | |
| **Date** | 19/01/2016 | **Priority** | High |
| **Actor:**  Scheduler  **Summary:**  This use case allows scheduler to create notifications for all users about all expiring foods in their refrigerator at specified time.  **Goal:**  Scheduler can create notifications.  **Triggers:**  The time hit configured time.  **Preconditions:**  The creating notification time has been configured.  **Post Conditions:**  **Success**: New notifications are created. Log file is generated.  **Fail**: Log file is generated.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Server checks the current time. If it hits configured time and scheduler status is “Ready”, creating notification process starts. | System gets all foods information of all users from storage:   * Food name * Food expiring day * Number of notification day   Create notification.  Insert notifications to storage | | **2** |  | Scheduler status is changed from “Ready” to “Notification created”. |   **Alternative Scenario:** N/A  **Exception:** N/A  **Relationships:** N/A  **Business Rules:**   * Every day at 0 o’ clock, scheduler status is changed from “Matching finished” to “Ready”. * After creating notification process, scheduler status is changed to from “Ready” to “Notification created”. * Creating notification time can be configured by staff or it is set as default at 0 o’clock. * System will check all food expiring day of all users. * Conditions for creating notification are:   + Remaining time = Expire day - Current day ≤ Number of notification day.   + Number of Notification day is set by user for each food. If user does not set up yet, it is set as system default. * When a new notification is created, notification’s status is “Pending”.   **Log file structure:** CREATE NOTIFICATION PROCESS LOG FILE  File name: createnotification.log  Created date: , Create time:   |  |  |  | | --- | --- | --- | | STT | User account | Number of notification created | |  |  |  | |  |  |  |   Total elapsed time:  Total notifications created: | | | |

* + - * 1. <Scheduler> Send notification (UC\_16)

**Use Case Diagram**

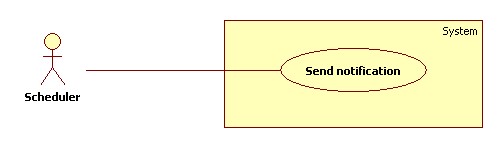


Figure - <Scheduler> Send notification

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_16** | | | |
| **Use Case No.** | 16 | **Use Case Version** | 1.0 |
| **Use Case Name** | Send notification | | |
| **Author** | Phan Hoàng Giáp | | |
| **Date** | 19/01/2016 | **Priority** | High |
| **Actor:**  Scheduler  **Summary:**  This use case allows scheduler to send notifications for all users about all expiring foods in their refrigerator at specified time.  **Goal:**  Scheduler can send notifications to user.  **Triggers:**  The time hits configured time.  **Preconditions:**  Notifications was created.  Notification time has been configured.  **Post Conditions:**  **Success**: Notifications are sent to user. Log file is generated.  **Fail**: Log file is generated.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Server checks the current time. If it hits notification time, sending notification process starts. | System sends notification messages to account of users.  [Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | **1** | System sends notification fail. | System will try to resend notification. |   **Exception:** N/A  **Relationships:** Create notification  **Business Rules:**  Scheduler had created notifications. At notification time, system sends notifications to account of all users. Users can set their own notification time.  Sent messages are notifications in “Pending” status. After sending, notification’s status is changed from “Pending” to “Finish”.  The scheduler will retry sending maximum three times.  **Log file structure:** SEND NOTIFICATION PROCESS LOG FILE  File name: sendnotification.log  Created date: , Create time:   |  |  |  | | --- | --- | --- | | STT | User account | Number of notification sent | |  |  |  | |  |  |  |   Total elapsed time:  Total sent notifications: | | | |

* + - * 1. <Scheduler> Match food (UC\_17)

**Use Case Diagram**

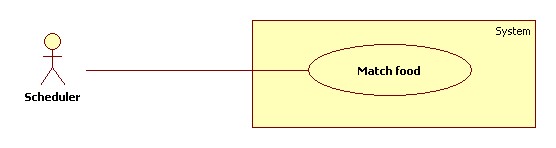


Figure - <Scheduler> Match food

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE - <UC Number>** | | | |
| **Use Case No.** |  | **Use Case Version** | 1.0 |
| **Use Case Name** | Match food | | |
| **Author** | Phan Hoàng Giáp | | |
| **Date** | 19/01/2016 | **Priority** | High |
| **Actor:**  Scheduler  **Summary:**  This use case allows scheduler to match food in user’s refrigerator to suggest dishes at specified time.  **Goal:**  Scheduler can match food in refrigerator.  **Triggers:**  The time hit configured time.  **Preconditions:**  Matching food time has been configured.  **Post Conditions:**  **Success**: Log file is generated.  **Fail**: Log file is generated.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Server checks the current time. If it hit configured time and scheduler status is “Notification created”, matching process starts. | System gets all foods name of all users from storage.  System matching foods with recipes in database.  Create suggestion with matched recipes.  Insert suggestion to storage.  Generate log file. | | **2** | Scheduler status is changed from “Notification created” to “Matching finished”. |  |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | **1** | If a food is not contains in any recipe | Create log file. |   **Exception:** N/A  **Relationships:** N/A  **Business Rules:**   * Matching food time can be configured by staff or it is set as default at 0 o’clock. * After matching food process, scheduler status is changed from “Notification created” to “Matching finished”. * System will check all foods in all users account. * In the matching algorithm, the results must be order by points. Points are given by these rules:   + 1 point for each food which not expiring.   + 2 points for each expiring food. * When a new dish suggestion is created, its status is “Pending”.   **Log file structure**: MATCHING FOOD PROCESS LOG FILE  File name: matchfood.log  Created date: , Create time:   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | STT | User account | Elapsed time | Total dishes received | Matching successful | Matching failed | |  |  |  |  |  |  | |  |  |  |  |  |  |   Total elapsed time:  Total dish suggestion: | | | |

* + - * 1. <Scheduler> Send suggestion (UC\_18)

**Use Case Diagram**

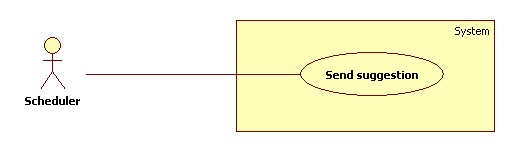


Figure - <Scheduler> Send suggestion

**Use Case Specification**

|  |  |  |  |
| --- | --- | --- | --- |
| **USE CASE – UC\_18** | | | |
| **Use Case No.** | 18 | **Use Case Version** | 1.0 |
| **Use Case Name** | Send suggestion | | |
| **Author** | Phan Hoàng Giáp | | |
| **Date** | 19/01/2016 | **Priority** | High |
| **Actor:**  Scheduler  **Summary:**  This use case allows scheduler to send suggest dishes to all users at specified time.  **Goal:**  Scheduler can send suggest dishes to all users.  **Triggers:**  The time hit configured time.  **Preconditions:**  Dishes were suggested.  Sending suggestion time has been configured.  **Post Conditions:**  **Success**: Dishes information is sent to user. Log file is generated.  **Fail**: Log file is generated.  **Main Success Scenario:**   |  |  |  | | --- | --- | --- | | **Step** | **Actor Action** | **System Response** | | **1** | Server checks the current time. If it hits configured time, sending suggestion process starts. | System sends suggestion messages to account of all users.  [Alternative 1] |   **Alternative Scenario:**   |  |  |  | | --- | --- | --- | | **No** | **Actor Action** | **System Response** | | **1** | System sends notification fail. | System will try to resend notification. |   **Exception:** N/A  **Relationships:** N/A  **Business Rules:**  Scheduler had matched food. At sending suggestion time, system sends suggestions to account of all users. Users can set their own sending time.  Sent messages are suggestions in “Pending” status. After sending, dishes suggestion’s status is changed from “Pending” to “Finish”.  The scheduler will retry sending maximum three times.  **Log file structure**: SEND SUGGESTION PROCESS LOG FILE  File name: sendsuggestion.log  Created date}, Create time:   |  |  |  |  | | --- | --- | --- | --- | | STT | User account | Elapsed time | Number of sent suggestion | |  |  |  |  | |  |  |  |  |   Total elapsed time: | | | |

* 1. Software System Attribute
     1. Usability
     2. Reliability
     3. Availability
     4. Security
     5. Maintainability
     6. Portability
     7. Performance
  2. Conceptual Diagram



Figure - Conceptual Diagram

**Data Dictionary**

|  |  |
| --- | --- |
| **Entity Name** | **Description** |
| User | Abstract entity describes a user in system |
| Staff | Contain the staff information |
| Member | Contain the member information |
| Fridge | Contain all food of a member |
| Recipe | Contain the recipe information |
| Material | Contain the material information |
| Step | Contain the step of a recipe |
| Step Image | Contain an URL of image of a step |
| Notification | Contain the notification information |

1. REPORT NO. 4: Software Design Description
   1. Design Overview

* This document describes the technical and user interface design of RRA System. It includes the architectural design, the detailed design of common functions and business functions and the design of database model.
* The architectural design describes the overall architecture of the system and the architecture of each main component and subsystem.
* The detailed design describes static and dynamic structure for each component and functions. It includes class diagram, class explanations and sequence diagrams for each use cases.
* The database design describes the relationships between entities and details of each entity.
* Document overview:
* Section 2: gives an overall description of the system architecture design.
* Section 3: gives component diagrams that describes the connection and integration of the system.
* Section 4: gives the detail design description which includes class diagram, class explanation and sequence diagram to details the application functions.
* Section 5: describes screen design.
* Section 6: describes a fully attributed ERD.
* Section 7: describes algorithms.
  1. System Architectural Design



* + 1. Web application architectural description

In Web Application, the system is developed using MVC architecture style. We decided to use this architecture for Web Application because of these following advantages:

* MVC separates the Model, View and Controller.
* Web Application includes the Web services (public API for mobile application), with the MVC architecture, we can use the business code in Web service without repeat the code.
* With the separation of Model, View and Controller, we only need to modify each part without affected to the other.

This project follows MVC architecture with following components:

* Servlet (Controller) is the parts of the application that acts like event handler to handles user interaction. This part may invoke changes on the model and view.
* HTML/JSP (View) is the part that renders the model into user interface. From View, user can input data and send command to interact with the system.
* Model contains the data that is retrieved according to commands from the controller and displayed on the view.
  + 1. Mobile application architectural description

Mobile application developed as a native application that runs on Android platform. The architecture of mobile application includes:

* Activity: is used to call web services or do function to interact with data.
* Layout: provides user interface for user to interact with the system.
* Model: similar to the model in web application. Model is used for storing data.
  1. Component Diagram
  2. Detailed Description
     1. Class diagram

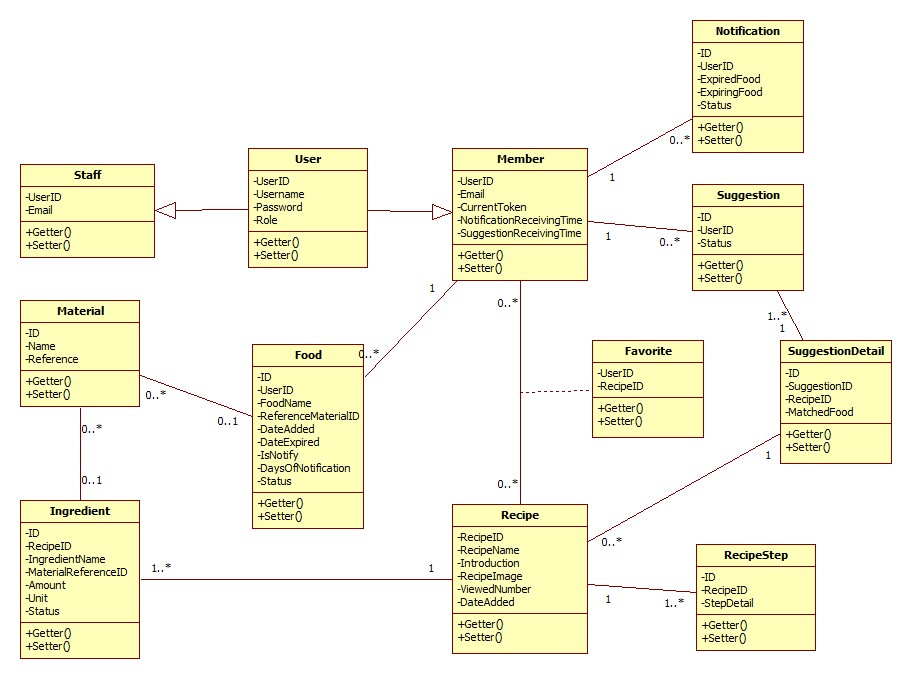


Figure - Class Diagram

|  |  |  |
| --- | --- | --- |
| Class dictionary: describe class | | |
| **Class name** | **Mapping column with conceptual diagram** | **Description** |
| Material | Material | This is the material dictionary, contains material dictionary. |
| Food | Refrigerator | Contains the information of user’s food. |
| Recipe | Recipe | Contains the information of recipe |
| RecipeStep | RecipeStep | Contains the information of a step of a recipe. |
| Ingredient | Ingredient | Contains the information of ingredient of a recipe. |
| User | User | Contains the information of a user. |
| Staff | Staff | Contains the information of a staff. This class is extended from user. |
| Member | Member | Contains the information of a member. This class is extended from user. |
| Favorite | Favorite | This is an association class from User and Recipe, represents a recipe that favored by a user. |
| Suggestion | Suggestion | Contains the information of a suggestion that will be sent to a user. |
| SuggestionDetail | SuggestionDetail | Contains the information of a suggestion detail. |
| Notification | Notification | Contains the information of a notification that will be sent to a user. |

Table - Class dictionary

* + 1. Class diagram explanation
    2. Interaction diagram
  1. Interface
     1. Component interface
     2. User interface design
  2. Database design
     1. Entity relationship diagram
     2. Data dictionary
  3. Algorithms