

|  |
| --- |
| Requirement Specification |
| A program assisting healthily getting fat |



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **제출일** | 2018. 4. 8. |  | **전공** | 컴퓨터공학부 |
| **과목** | 소프트웨어공학 |  | **학번** | 20141405 |
| **담당교수** | 송인식 |  | **이름** | 최형준 |

**Product Description**

This is a **program** that **assists low weight people healthily getting fats**. **Low weight people** have a greater chance to **suffer from diseases** such as pneumothorax than others. On the internet or in app markets, **all fitness programs focus on losing weights**. Through providing useful information to get fat, this program will **uniquely help low weight people**.

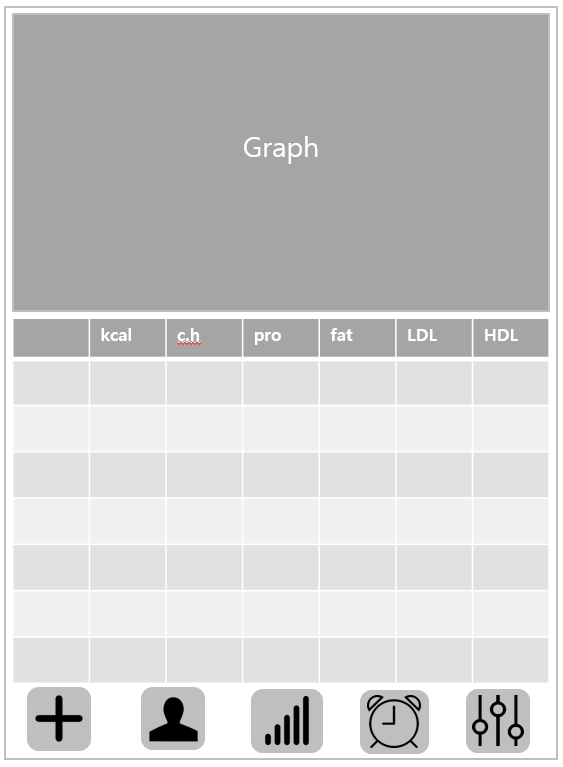
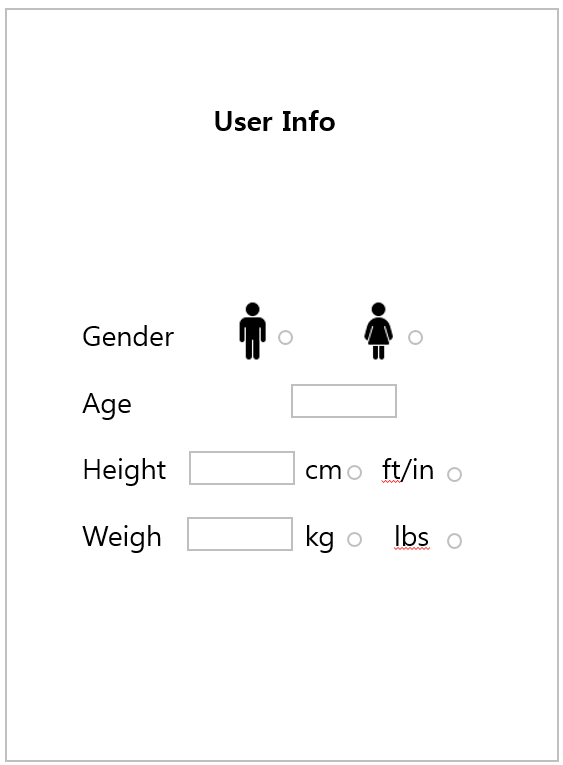
**- Functional Requirement**

|  |  |
| --- | --- |
| **Functions** | **Explanation** |
| **1. Obtaining & Saving user’s personal data** | A user types his or her personal information such as age, weight, height, and sex. |
| **2. Show user’s daily consumed calorie and nutrition** | A user inputs his or her daily eaten food, and the program calculates the total calorie and nutrition data. |
| **3. Check whether a newly added food is appropriate** | If the total calorie or nutrition a user newly inserted food into the program, the program will show up a warning. |
| **4. Alarm call for a regular meal time** | For healthily getting fat, the program keep tracks of user’s meal time, and alarms the user when a recommended meal time is arrived. |
| **5. Show a user weekly consumed data** | Provide weekly consumed calorie and nutrition data with a graph by user’s daily consumed data saved in a local database. |

**- Non-functional requirement**

|  |  |
| --- | --- |
| **Product Requirements** | **Explanation** |
| **Performance Requirement** | Response time: less than 1s |
| **System Requirement** | After the version Window 7.1 |

**UI(User Interface) Diagrams**



**Use Cases**

* **Case 1(Check whether a newly added food is appropriate)**

|  |
| --- |
| Initial assumption:  One user wants to know his own metabolism to be stout. So he give inputs which are his weight, height, gender and age to program accurately. So his metabolism amount is calculated well. And he eat some food now. |
| Normal:  User types food name with specified format. Then program gets nutrient and calories information of that food from database. And it checks user’s metabolism amount ,calories that he ate until that time. With algorithm we made, it sends information whether it is proper food to be stout and if it is correct, adds to today’s diet list. Also it gives some over-amount nutrients that time. And asks user whether he(or she) adds diet to list. |
| When program finishes:  Program adds food to the list of user’s today diet. And show it’s calorie and nutrient information in main screen. |
| What can go wrong:  If user types food’s name in wrong format, program can’t find food from it’s database, so it can return message that database don’t have information about that food thought there is food in the database. |
| Concurrent activities:  While program adds food to the list, it calculates total amount of that days’ diet, show info in graph. |

* **Case 2(Alarm call for a regular meal time)**

|  |
| --- |
| Initial assumption:  A user input how many times he wants to take a meal and the time he wants to eat. |
| Normal:  Our program will show small box that presents message in user’s window with alarm.  (ex “It’s time to take a meal. Menu: rice 300g, kimchi stew…”) |

* **Case 3(Show user’s daily consumed calorie and nutrition)**

|  |
| --- |
| Initial assumption : Personal information and Daily diet uploaded by user is correct. |
| Normal : The user prompted to select type of meal they ate and to choose certain menu.  Then, program calculates total calorie and nutrients and compares with recommended calories and recommended intakes of nutrient. Finally, program displays total calories and nutrients, and suggest users to consume calories or nutrients more or less respectively. |
| What can go wrong : It is possible to user can’t find their menu on DB. In this case, user can request developers to add a new menu to DB. |
| Other activities : user’s daily diet saved user’s diet history. |
| System state on completion : Information displays on screen and user’s daily diet add user’s diet history. |

**Process Description**

* **Software Tools**

|  |  |  |
| --- | --- | --- |
| **Name** | **Feature** | **Usage** |
| **Java Swing** | Java Library | Provide GUI functions in Java |
| **Jsoup** | Java Library | Offer crawling functions in Java |
| **Eclipse** | Java IDE | Edit and compile Java codes |

* **Schedule(Pair programming)**

|  |  |
| --- | --- |
| Week | To-Do |
| Week 6 | Oh: Define functions of the program.  Kwon: Define non-functions of the program.  Lee: Write user scenario(.show user’s daily consume)  Choi.H: Write user scenario.(check whether food is appropriate)  Choi.P : risk summary & process description  Kim: UI design |
| Week 7 | Oh,Kwon:Java Swing(UI) design proposal  Choi.H,Kim:make Algorithm for Obtaining & Saving user’s personal data  Choi.P,Lee: documentation for the process of week. |
| Week 8 | Mid-term Examination |
| Week 9 | Oh,Kwon: implement Java Swing(UI) for the function1(obtaining&saving)  Choi.H,Kim:make Algorithm for crawling information from web  Choi.P,Lee: documentation for the process of week. |
| Week 10 | Oh,Kwon: implement Java Swing(UI) main screen  Choi.H,Kim:make Algorithm for Check whether a newly added food is appropriate  Choi.P,Lee: documentation for the process of week. |
| Week 11 | Oh,Kwon: implement Java Swing(UI) graph, sync button with functions  Choi.H,Kim:make Algorithm for Alarm call for a regular meal time  Choi.P,Lee: documentation for the process of week. |
| Week 12 | Oh,Kwon: implement Java Swing(UI) lists ,sync button with functions  Choi.H,Kim: optimize algorithms for speed  Choi.P,Lee: documentation for the process of week. |
| Week 13 | Integrate all programs and fix errors |
| Week 14 | Get feedback and test all functions. |
| Week 15 | Final Demonstration |
| Week 16 | Final Examination |

**Risk Summary**

|  |  |
| --- | --- |
| **Risk** | **Detail** |
| **Illegal attempts on personal user data** | Personal data can be attacked by hackers. |
| **User types food’s name in illegal format** | Program can’t find information of food though it is in the database. |
| **User types wrong information** | Result of program can go wrong. |