

**Incoming Exchange Student - Final Degree Project**

Erasmus  Techno  Other (specify):

**Degree course:** Grau en Enginyeria Informàtica Pla 2010

**Title:** e-Health tools supporting healthy habits for obese and diabetic kids

**Document:** Final Degree Project

**Student** (Name & Surname): SREYNOCH SOUNG

**EPS Advisor:** Dr. BEATRIZ LOPEZ

**Department:** Eng. Elèctrica, Electrònica i Automàtica

**Delivered on** (month/year): June/2016

# Acknowledgements

---

This dissertation would not have been possible without the guidance and the help of several individuals who in one way or another contributed and extended their valuable assistance in the preparation and completion of this project.

First and foremost, I would like to express my utmost and deepest gratitude to my parents who always provide me with all kinds of support and advice.

I would like to say thank to Erasmus Mundus TECHNO II project for providing me the scholarship. It's the best opportunity for me to be an exchange student in the University of Girona.

I would like to show my respect to Prof. Dr. Emili GARCÍA-BERTHOU, University of Girona academic officer of the TECHNO II project for his general management and cooperation with all the scholarship processes.

I would like to show my best regard to Dr. Beatriz LÓPEZ IBÁÑEZ, senior lecturer at the University of Girona and also as my advisor for her management of this research project, her priceless advices and her good communication with me.

I would like to say thank to Mr. Pablo GAY and Dr. Albert PLANAS, my tutors for helping me to solve some problems that I met during the internship, discussing for the good ideas and correcting my thesis.

I would like to say thank to Natalia MORDVANYUK for helping me to solve some technical problems during my implementation.

I would like to say thank to everyone in the eXiT group for your encouragement, kindness and friendship during my present in the group.

The last time, I would like to say thank to all the people who involve and help me to write this thesis and develop this project successfully.

# Table of contents

---

1. Introduction .....	1
1.1. Motivation .....	2
1.2. Purpose .....	2
1.3. Objective .....	3
2. Resources.....	3
2.1. Server side development resources .....	3
2.1.1. Programming.....	3
2.1.2. Tool .....	4
2.2. Mobile application development resources.....	7
2.3. Version control tool .....	8
2.4. Prototype design resource .....	9
2.5. Cloud storage.....	9
2.6. 3rd party libraries .....	10
3. Methodology .....	11
3.1. What's ICONIX ? .....	11
3.2. Phases of methodology.....	11
3.2.1. Analysis requirement .....	12
3.2.2. Preliminary design .....	12
3.2.3. Detail design .....	13
3.2.4. Implementation .....	14
4. Planning.....	16
5. System requirements .....	18
5.1. Functional requirements.....	18
5.2. Non-functional requirements.....	24

6. Studies and decision .....	25
6.1. Android .....	25
6.2. Existing applications .....	27
6.3. Spring security framework .....	31
7. Analysis and design system.....	33
7.1. Use case diagrams .....	33
7.1.1. Use case specification .....	37
7.2. Class diagrams.....	64
7.3. System architecture .....	65
7.4. Entity relationship diagram .....	67
7.5. Design GUI prototype .....	68
7.5.1. Kid GUI .....	68
7.5.2. Parent GUI .....	79
7.5.3. Doctor GUI .....	85
7.5.3. Admin GUI.....	92
8. Results and Implementation .....	97
8.1. Results .....	97
8.2. Project Structure .....	98
8.3. Problems and solutions.....	99
8.4. Spring MVC and security java configure.....	102
8.5. Using JSP - Standard Tag Library formatting tags.....	106
9. Conclusion.....	108
9.1. Achievement.....	108
9.2. Non-Achievement .....	109
9.3. Experiences .....	109
9.4. Perspective.....	109

9.5. Future works.....	110
10. Bibliography.....	111

# Table of figures

---

Figure 1: Spring Tool Suite logo .....	4
Figure 2: MAMP logo.....	4
Figure 3: MySQL logo.....	5
Figure 4: MySQLWorkbench logo .....	5
Figure 5: Pivotal tc Server logo.....	6
Figure 6: Google Chrome logo.....	6
Figure 7: Postman logo.....	6
Figure 8: Android Studio logo .....	7
Figure 9: Tablet Nexus 9.....	7
Figure 10: Genymotion logo.....	7
Figure 11: Bitbucket logo .....	8
Figure 12: SourceTree logo .....	8
Figure 13: Marvel logo .....	9
Figure 14: Dropbox logo .....	9
Figure 15: ICONIX Process.....	15
Figure 16: Mobile OS global market share.....	26
Figure 17: Spring security flow .....	32
Figure 18: Use case diagram for mobile .....	35
Figure 19: Use case diagram for web page.....	36
Figure 20: Class diagram.....	64
Figure 21: System architecture .....	66
Figure 22: Entity relational diagram.....	67
Figure 23: Login kid .....	68
Figure 24: Register kid .....	68
Figure 25: Confirm code doctor .....	69
Figure 26: Home page kid .....	69
Figure 27: Food empty dashboard .....	70
Figure 28: Select photos .....	70
Figure 29: Input food info.....	71
Figure 30: Upload food.....	71
Figure 31: Food dashboard.....	72
Figure 32: Input exercise info.....	72

Figure 33: Verify from watch .....	73
Figure 34: Exercise dashboard.....	73
Figure 35: Competition.....	74
Figure 36: Group.....	74
Figure 37: Message history.....	75
Figure 38: Chat .....	75
Figure 39: My progress.....	76
Figure 40: Setting .....	76
Figure 41: Notification .....	77
Figure 42: Menu .....	77
Figure 43: Profile kid.....	78
Figure 44: Login parent .....	79
Figure 45: Register parent .....	79
Figure 46: Kid code confirm.....	80
Figure 47: Parent Home page .....	80
Figure 48: Food info .....	81
Figure 49: Exercise info.....	81
Figure 50: Exercise schedule.....	82
Figure 51: Kid’s progress.....	82
Figure 52: Profile.....	83
Figure 53: Setting .....	83
Figure 54: Menu .....	84
Figure 55: Notification .....	84
Figure 56: Login doctor.....	85
Figure 57: List kids.....	86
Figure 58: Kid detail.....	87
Figure 59: Exercise target.....	88
Figure 60: Food target .....	89
Figure 61: New target.....	90
Figure 62: Doctor profile.....	91
Figure 63: Log in admin .....	92
Figure 64: List doctor and admin.....	93
Figure 65: Doctor detail.....	94
Figure 66: Admin profile .....	95
Figure 67: New user doctor or admin.....	96



# List of tables

---

Table 1: Planning .....	16
Table 2: Functional requirements .....	18
Table 3: Use case specification Register account app .....	37
Table 4: Use case specification Log in app .....	38
Table 5: Use case specification Modify account .....	38
Table 6: Use case specification Form kid team .....	39
Table 7: Use case specification Organize team match calendar .....	40
Table 8: Use case specification Input percentage of each food tag .....	40
Table 9: Use case specification Assign photo for validating .....	41
Table 10: Use case specification Validate food .....	42
Table 11: Use case specification Manage exercise schedule .....	43
Table 12: Use case specification Manage exercise activity.....	44
Table 13: Use case specification Validate the photo validation .....	45
Table 14: Use case specification Validate activity input from watch .....	46
Table 15: Use case specification Feed back to kid .....	46
Table 16: Use case specification Assess grad.....	47
Table 17: Use case specification Compute different% of food .....	47
Table 18: Use case specification Compute different% of activity.....	48
Table 19: Use case specification View kid process .....	49
Table 20: Use case specification View grad of all member.....	50
Table 21: Use case specification Manage chat to other kids .....	52
Table 22: Use case specification Delete food photo.....	52
Table 23: Use case specification Return result to team .....	53
Table 24: Use case specification Find the winner team .....	54
Table 25: Use case specification Log out .....	54
Table 26: Use case specification Register account on server side .....	55
Table 27: Use case specification Log in .....	56
Table 28: Use case specification Modify account .....	57
Table 29: Use case specification View kid process .....	57
Table 30: Use case specification Set recommend food to kid .....	58
Table 31: Use case specification Set recommend activity to kid.....	59
Table 32: Use case specification Generate kid process chart .....	60

Table 33: Use case specification Log out .....	60
Table 34: Use case specification Create account for doctor /admin.....	61
Table 35: Use case specification Deactivate account doctor/admin.....	62
Table 36: Use case specification Select/Search doctor/admin.....	62

# 1. Introduction

---

This report presents my final project of engineering of information class which is under the topic “e-Health tools supporting healthy habits for obese and diabetic kids”. I was doing this project in eXiT group, University of Girona, Spain for duration 6 months. It is a system which consists of an android application, a web application and a server implementation. This system is used to assist in diabetic treatments.

Obesity and diabetes is a main issue for many people, especially children. It is becoming a big concern in european and occidental countries. According to physicians of obese and diabetic children, there is a strong link between the diets and physical habits of the children and the evolution of those diseases. Physicians would like to be able to have a complete track of their (kid) patients’ stats (food intake, physical activity, glucose levels in the case of diabetes, etc.), nevertheless manually tracking all this information is tiring, boring and imprecise.

As a solution, we propose a framework where patients input their nutrition and exercise activity using an application in their mobile phone. The system can help tracking all this information and allowing physicians access to it through a web site. It is important to avoid patients getting tired of it and to encourage patients to tell the truth. So, serious games and gamification can help improving patient engagement.

To develop this system, Java Spring framework is used as backend side, which has been developed with Spring Tool Suite IDE. Also, an Android application is used on client side and developed using Android Studio. Moreover, MySQL is used for data management.

## 1.1. Motivation

---

The idea of helping children to recover from obese and diabetic is the main motivation to develop this project. I'm really happy to use my knowledge to create something useful for society. Especially, children should be cared and live with healthy life by having a proper nutrition and exercise activity.

Moreover, this is my final project to finish my bachelor degree. So I would like to have a good result from this work. This means I use all the knowledge that I've learnt during my academic years at university.

Last, I would like to learn android technology which is a famous mobile technology nowadays. Also, I try to figure out more about Spring framework .

## 1.2. Purpose

---

We would like to develop a mobile application to manage diabetes and/or obesity patients and to help them acquiring appropriate habits. We also want to gamify the application in order to encourage them to use the application and to do physical activity. The application aims to encourage kids to make physical activities and play sports instead of just playing a video game: we want to develop a cooperative, competitive and serious game in which patients are formed as teams to compete against other teams of patients in order to win a league. To win matches against other teams of patients, members of the teams will have to follow their doctor's recommendations. So, the use of mobile technology will help keeping track of the patient's status. Moreover, social networks will be used to encourage team members and to validate other participant.

## 1.3. Objective

---

There are two main objectives in this project. First, we need to create a web application which is used by doctors and system administrators. Through this web application, doctors can view information of their patients and set nutrition and exercise activity targets. Administrators can control doctor users.

Second, we need to develop an android application with two interfaces. One interface is used for kids to input their daily activity related to nutrition and exercise activity. The other one is used by the parents to view their kid's information, since parents are responsible to feed their kids properly. Parents also can encourage the kids to have good exercise activity.

## 2. Resources

---

There are some important IDEs and other tools are used in order to develop this system.

### 2.1. Server side development resources

---

This section describes the resources used like programming and some tools that have been used using to develop the backend system and web page.

#### 2.1.1. Programming

---

**Java:** To develop a backend system, [27] we chose java programming language which is used the implementation of the server side: controllers, domains and services. Java is used to create functions and to interact with the database. We decided to use java because it is a well-known programming language with a lot of libraries that is easy for use in the development. Moreover, Java has the notion OOP.

**JSP:** is used for creating the web page for doctors and admins. Java Server Pages [1] is a server-side programming technology that enables the creation of dynamic, platform-independent method for building Web-based applications. It helps to create dynamically generated web pages based on HTML, XML, or other document types. Released in 1999 by Sun Microsystems, JSP is similar to PHP and ASP, but it uses the Java programming language. JSP has access to the entire family of Java APIs, including the JDBC API to access enterprise databases.

### 2.1.2. Tool

---

**Spring Tool Suite version 3.7.3.RELEASE:** Used to develop the backend and web interface [15] for doctors and admins. It is an Eclipse-based development environment that is customized for developing Spring applications. It provides a ready-to-use environment to implement, debug, run, and deploy Spring applications and includes Git and Maven.



**Figure 1: Spring Tool Suite logo**

**MAMP(Mac, Apache, MySQL, PHP) version 3.0.7.3:** Web development environment [16] for creating a web applications with Apache, PHP and the MySQL database. It's more easy to manage databases because it comes with PHPMyAdmin.



**Figure 2:MAMP logo**

**MySQL:** Used for storing the data. We have chosen to use it as database [17] for the system because it is a freely available open-source RDBMS (Relational Database Management System) that has many advantages. It's a secure, scalable and efficient database. It uses SQL, which is a standard interactive programming language for getting information from and updating database. It allows us to make queries about the information in our database - data selection, insertion, updating, and locating.



**Figure 3: MySQL logo**

**MySQLWorkbench:** Used as unified visual tool [18] for database architecture and data modeling. It provides data modeling, SQL development, and comprehensive administration tools for server configuration, user administration and backup.



**Figure 4: MySQLWorkbench logo**

**Pivotal tc Server version 3.1.3:** Used to build and run web application. It's the best place [19] to build and run Spring Java applications, provides enterprise users with a drop-in replacement for Apache Tomcat. It provides a secure, supported, and extended Java application server based on and fully compatible with Apache Tomcat .



**Figure 5: Pivotal tc Server logo**

**Google Chrome browser Version 50.0.2661.94:** Used to test web application. It is an open source [20] program for accessing the World Wide Web and running Web-based applications.



**Figure 6: Google Chrome logo**

**Postman REST Client version 3:** REST Client for Chrome Browser [21] and it's used for testing APIs of the web application.



**Figure 7: Postman logo**

## 2.2. Mobile application development resources

---

**Android Studio version 2.0:** Used to develop [22] the android application. It is the official IDE for Android application development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance the productivity when building Android apps, such as: flexible gradle-based build system.



**Figure 8: Android Studio logo**

**Tablet Nexus 9 with android version 6.0.1:** Testing device.

**Genymotion version 2.6.0:** Used as emulator [23] for testing application. It's a fast third-party emulator that can be used instead of the default Android emulator.



**Figure 9: Tablet Nexus 9**



**Figure 10: Genymotion logo**

## 2.3. Version control tool

---

**SVN:** A central repository [24] used in software organization to keep track of changes made in files in the form of revisions.

**Bitbucket:** Also used to keep track of a git project [25]. It is a Git/Mercurial solution for professional teams. It collaborates with code using inline comments and pull requests. Bitbucket offers both commercial plans and free accounts. It offers free accounts with an unlimited number of private repositories which can have up to five users.



**Figure 11: Bitbucket logo**

**SourceTree:** Git client version control systems [26]. It facilitates the interaction with the Git repository.



**Figure 12: SourceTree logo**

## 2.4. Prototype design resource

---

**Marvel:** is a web application editor [2] allows to link all designs together, then add gestures and transitions to make prototype feel just like a real application or website.

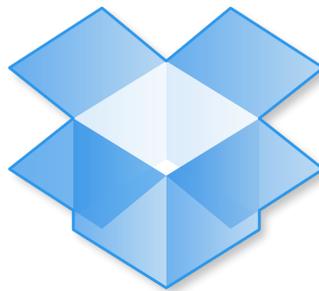


**Figure 13: Marvel logo**

## 2.5. Cloud storage

---

**Dropbox:** is used as a cloud storage service [28] to store all project documents such as requirement, prototype proposal, use case description and system architecture.



**Figure 14: Dropbox logo**

## 2.6.3rd party libraries

---

To help the implementation, we need to consider some important plugins and libraries for our project:

**Maven:** is plugged in to Spring Tool Suite. It is used to [29] generate files (\*.classpath, \*.wtpmodules and the .settings folder) for projects. There are some important commands:

`eclipse:configure-workspace` : is used to add the classpath variable M2\_REPO to Eclipse which points to your local repository and optional to configure other workspace features.

`eclipse:eclipse` : generates the Eclipse configuration files.

`eclipse:clean` : is used to delete the files used by the Eclipse IDE.

Maven also is used to synchronize project in team from repository.

**Bootstrap version 3.3.6:** is a free and open-source front-end library [3] for creating websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. It aims to ease the development of dynamic websites and web applications. It helps us with designing user interface easily.

**JQuery version 1.12.3:** is also included to our project [30] to simplify event, effects, and Ajax functions. It is a cross-platform JavaScript library designed to simplify the client-side scripting of HTML.

## 3. Methodology

---

To develop a more effective implementation, it is necessary to work with a methodology. There are a lot of development methodologies, but we should choose the most suitable one to current project. Since this is a research project and we need to change the requirement to follow the needs of users, ICONIX has been chosen as methodology for this development project.

### 3.1. What's ICONIX ?

---

ICONIX is a software development methodology [4] which predates both the Rational Unified Process (RUP), Extreme Programming (XP) and Agile software development. Like RUP, the ICONIX process is UML Use Case driven but more lightweight than RUP. Unlike the XP and Agile approaches, ICONIX provides sufficient requirement and design documentation, but without analysis paralysis. The ICONIX Process uses only four UML based diagrams in a four step process that turns use case text into working code.

ICONIX goals respond to changing requirements in a robust and timely manner, improving the design and architecture of a project without massively impacting its schedule and give customers exactly what they want from a project.e

### 3.2. Phases of methodology

---

There are some main phases of ICONIX that we need to follow [5] such as analysis requirement, analysis/preliminary design and review, detail design and review and implementation.

### 3.2.1. Analysis requirement

---

a. Functional requirements: They define what the system should be capable of doing. Depending on how the project is organized, either we'll be involved in creating the functional requirements or the requirements will be provided by a customer or a team of business analysts.

b. Domain modeling: Defining the problem space without ambiguities. This task consists on building a project glossary or a dictionary of terms that will be used in the project. Its purpose is to make sure everyone on the project understands the problem space in unambiguous terms. The domain model for a project defines the scope and forms the foundation on which to build use cases. The domain model also provides a common vocabulary to enable clear communication among members of a project team.

c. Behavioural requirements: Defines how the user and the system will interact (i.e., first-draft use cases). It starts with a GUI prototype (storyboarding the GUI) and identify all the use cases that we're going to implement, or at least come up with a first-pass list of use cases, which would reasonably change as we explore the requirements in more depth.

**Milestone 1: Requirements review,** Make sure that the use cases match the customer's expectations. It might be required to review the use cases in small batches, just prior to designing them. This vital step ensures that the requirements are sufficiently well understood by both the development team and the customer/users/project stakeholders.

### 3.2.2. Preliminary design

---

Preliminary design is an intermediate step between analysis and design.

a. Robustness analysis: Draw a robustness diagram (an "object picture" of the steps in a use case) and re-write the use case text. To get from use cases to detailed design (and then to

code), we need to link our use cases to objects. Robustness analysis helps us to bridge the gap between analysis and design by doing exactly that.

b. Update the domain model while we're writing the use case and drawing the robustness diagram. Here we will discover missing classes, correct ambiguities, and add attributes to the domain objects.

c. Name all the logical software functions (controllers) needed to make the use case work and re-write the first draft use cases.

### **Milestone 2: Preliminary Design Review (PDR)**

The Preliminary Design Review (PDR) session helps you to make sure that the robustness diagrams, the domain model, and the use case text all match each other. This review is the “gate- way” between the preliminary design and detailed design stages, for each package of use cases.

### **3.2.3. Detail design**

---

Preliminary design create the first prototype. This prototype most include all the functionals and behavioural requirements but it still requires for the refinement. Detailed design worries about efficiency in terms of execution times, network loading, and memory footprint, and is concerned with the reusability of code whenever possible.

a. Sequence diagramming: ICONIX Process uses the sequence diagram as the main vehicle for exploring the detailed design of a system on a scenario-by-scenario basis. In object-oriented design, a large part of building the system right is concerned with finding an optimal allocation of functions to classes. The essence of this is drawing message arrows on sequence diagrams and allowing a modeling tool to automatically assign an operation to the class of the target object that receives the runtime message.

b. Update the domain model while we're drawing the sequence diagram, and add operations to the domain objects. In this stage, the domain objects are domain classes, or entities, and the domain model should become a static model, or class diagram a crucial part of our detailed design.

c. Clean up the static model. This stage should have an extremely well-factored design that works within the real-world constraints of our project's requirements: application framework design, deployment topology, and so forth. There's just one last stop before we begin coding:

### **Milestone 3: Critical Design Review(CDR)**

The CDR helps us to achieve two important goals, before we begin coding the current use cases. It ensures that the "how" of detailed design matches up with the "what" specified in requirements. We can review the quality of our design in this stage.

#### **3.2.4. Implementation**

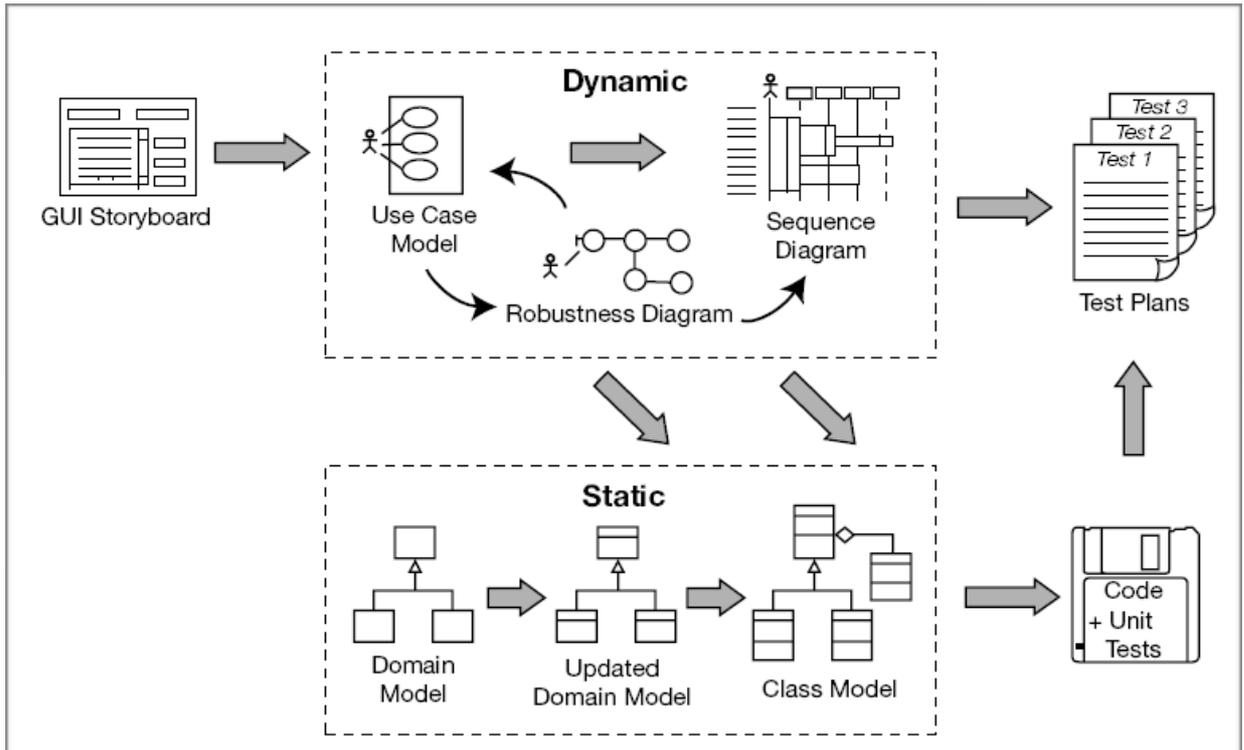
---

Once we've made the effort to drive a model from use cases through a detailed design, it would not be advisable to disregard the model and just start coding totally independent of the model that has been produced.

- a. Coding/unit testing: Write the code and its associated unit tested or viceversa.
- b. Integration and scenario testing: Base the integration tests on the use cases, so that you're testing both the basic course and the alternate courses.

c. Perform a Code Review and Model Update: The main purpose of the Code Review and Model Update milestone is to synchronize the code and the model before the next iteration begins. This ongoing effort to keep the design tight prevents entropy, or code rot, from setting in as more and more functionality is added to a complex system.

Figure 15 shows all the processes of each phase in the ICONIX methodology. [6]



**Figure 15: ICONIX Process**

## 4. Planning

---

The project was conducted over 22 weeks. It started the 1st January until 15th June 2016. I've been working on this project 8 hours a day from Monday to Friday. The proposed schedule is done after we have identified the main objective of this project and idea. This table below is shown about project plan.

**Table 1: Planning**

Task	Start Date	End Date
<b>Analysis requirement</b>		
Analysis functional requirements	01/01/16	15/01/16
Model domain	18/01/16	22/01/16
Analysis & design prototype	25/01/16	19/02/16
<b>Requirements review</b>		
<b>Analysis and preliminary design</b>		
Update requirement by the changes of user	29/02/16	04/03/16
Design use case diagram	07/03/16	18/03/16
Design class diagram	21/03/16	25/04/16
<b>Preliminary Design Review</b>		
<b>Detail design</b>		
Design Sequence diagramming	28/04/16	01/04/16
<b>Critical Design Review</b>		

Task	Start Date	End Date
<b>Implement</b>		
Model entity relationship diagram	04/04/16	08/04/16
Learn spring framework	01/04/16	22/04/16
Code security	08/04/16	29/04/16
Code back office and backend	02/05/16	30/05/16
Learn and Code android	21/03/16	30/05/16
Test	15/04/16	23/05/16
<b>Write Report</b>	04/04/16	07/06/16

## 5. System requirements

---

This section contains the description of project's requirements, both functional and non-functional requirement. After having discussed with physicians about the problems and studied about their objectives and needs, we tried to identify the functionalities that should be included in this project.

### 5.1. Functional requirements

---

This table is a list of functional requirement in our project.

**Table 2: Functional requirements**

1. Kids nutrition	<ul style="list-style-type: none"><li>- The physician prescribes a diet composed of the some food tags (Carbohydrates, Meat/Fish, Fruite/Vegetable) to kid.</li><li>- 5 times a day, the kid makes a photo with the smartphone camera or album photo about the food (dish) that she/he is going to eat. The photo is labelled with the tags. Each tag receiving the percentage expected to be on the dish.</li><li>- Each time, the kid can input only 3 photos of their meal.</li></ul>
-------------------	--

<p>2. Kids validation</p>	<ul style="list-style-type: none"> <li>- Once a day every kid receives a set of photos labelled from other kids in the social network. The kid should validate the contents of the labels</li> <li>- The owner of the received photos is unknown. They could come either from kids in the same team or adversary teams.</li> </ul>
<p>3. Parents validation</p>	<ul style="list-style-type: none"> <li>- Once a day, every parent receives a set of photos labelled from others kids in the social network. The parent should validate the contents of the labels.</li> <li>- The owner of the received photos is unknown. They could come either from kids in the same team that his son or daughter or form adversary teams.</li> </ul>
<p>4. Kid exercise</p>	<ul style="list-style-type: none"> <li>- Every kid defines a profile regarding its preferences about sports and the timetable they use to practice (supported by parents when under 12 years old).</li> <li>- The smart watch detects and confirms the exercise activity. Some activity could come with non-scheduled hours (as for example, playing soccer in the school playground).</li> </ul>

## 5. Kid assessment

- A procedure on the validated photos determine the different percentage of food ingested in every dish. Moreover, photos validated by parents should have a higher confidence.
- An aggregation measure computes the percentage of ingested food every day.
- A difference measure computes the deviation of the ingested food and the recommended food.
- Map the difference on a scale for nutrition fitness, from A to E (best, worst).
- An aggregation measure computes the amount of exercise performed every day.
- Map the difference on a scale for exercise fitness, from A to E (best, worst).
- The final assessment of the kid is performed based on a multi-criteria function. It is obtained in a scale from A to E. This is the kid state.
- Some feedbacks should be provided every day to the kid to assess about her state.

## 6. Social networks

- There are  $N$  obese [diabetic] kids managed by the same healthcare team and kid can chat to member in the same team.
- Teams among the  $N$  kids are generated based on the kid profile, trying to guarantee that:
  - There would be enough teams in each sport to set up a competition
  - There should be a certain satisfaction degree among the kids profile and the kid of sportive team assigned.
- The score of each team depends on the state values of its members:
  - The score is measured in a scale from A to E.
  - An aggregation function is used to compute the overall value of the team
    - An aggregation measure should contemplate that “at least the 75% of the team has an average value of C”.
- Validation of photos
  - 25% photos of a kid are validated by members of the same team
  - 50% photos are validated by members of other teams of the same competition
  - 25% photos are validated by members of other competitions
  - 50% photos are validated by parents (select random)

7. Games competition	<ul style="list-style-type: none"> <li>- Once a week (Sunday) all teams plays a match.</li> <li>- The winner of the match is the team with the highest score.</li> <li>- Solve the ties at random.</li> <li>- Return the results to the kids and parents in a friendly, fun visualization.</li> </ul>
8. Easy training	
9. Doctor requirements	<ul style="list-style-type: none"> <li>- Doctors are be able to review their patient’s information (Nutrition, Exercise activities)</li> <li>- Doctors can add new nutrition and exercise activities target to their patients</li> </ul>

## 10. User authorization

There are 4 types of user which are:

- Admin: User uses web page to create new doctor account or admin account. As an admin, she/he also can deactivate doctor account.

- Doctor: User uses username and password that are provided by an admin to login in to web page. Doctors can change their password and other info in account setting.

- Kid: User uses application on mobile to register their account. During the registration process, the kids can input a code key provided by their doctor so she/he can get nutrition and exercise targets from their doctor.

If kid doesn't have a code key from their doctor, she/he still can register as a normal kid. So, she/he will get a target for normal kid from system.

- Parent: User uses application on mobile to register their account. During the registration process, parent need to input a code key from their kid. If they don't have it, they can't register as a parent account.

## 5.2. Non-functional requirements

---

After listing all the functional requirements, so in this part we will talk about non-functional requirements. The non-functional requirements are requirements that are defined to improve our system. There are:

1. Verify kid's age.
2. Multi-languages for both Web page and Mobile application can switch language : English, Catalan and Spanish.
3. Provide the accessibility features and services for helping users navigate the web and mobile application more easily, including big menu icons, big font-size, and tidy design structure.
4. Provide the usability by making the system easier to use's a good user experience.
5. Make system more secure.

## 6. Studies and decision

---

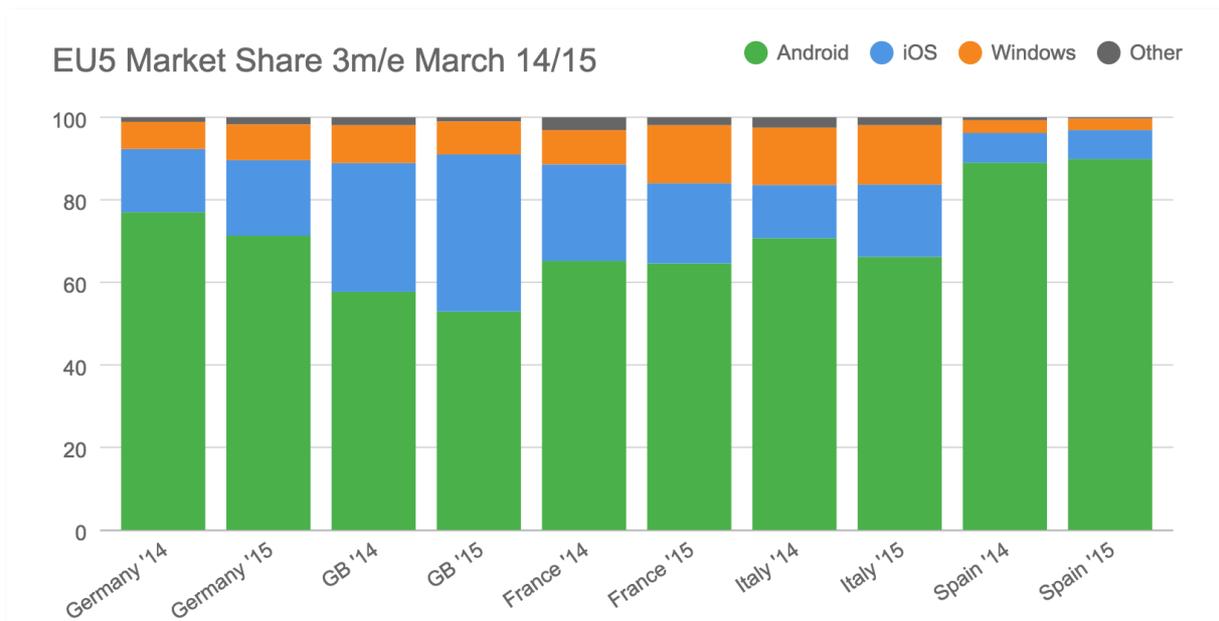
Before starting to develop this project, Some researches has been performed regarding technologies as well as some important frameworks that would be used in the project existing and some healthy habit applications. A description about the studies, the decision of choosing Android mobile operating system and Spring framework of java programming language is provided.

### 6.1. Android

---

Android is a mobile operating system (OS) currently developed by Google, based on the Linux kernel and designed primarily for touchscreen mobile devices such as smartphones and tablets. There are some main reasons that android has been chosen for our project.

Firstly, it is the current market leader in the world [8] as well as in Europe. With around 1 billion ( $10^9$ ) devices activated monthly, it gives developers and users the advantage of most probably having their devices supported. Latest research from Kantar Worldpanel ComTech on smartphone operating systems reveals that Android continues to make year-on-year gains in Great Britain (which excludes Northern Ireland), Germany, France, Italy and Spain. For the three months ending February 2016, Android accounted for 74.3% of the top five EU markets as of the end of February, rising 6.7% points year on year. The share of Apple iOS smartphones fell marginally by 1.8% points to 19.1% while Windows dropped 4.2% to 5.9%. There is also a graph of global market share from “androidauthority” website [9] to show about successful of android market in Europe. This graph is comparing of market share between 2014 and 2015 in five countries of Europe such as Germany, GB (Great Britain), France, Italy and Spain.



**Figure 16: Mobile OS global market share**

According to this graph, we found that there are many people are using android in Europe especially in Spain. Since the first target of our application is in Spain, and then Europe, developing this project on android should be a good decision.

Secondly, since we require kids to use our application on their phone or tablet we have to consider that Android has a good price. Also, in exercise section, we need them to wear a smart watch during their exercise activity. Then that smart watch will be connected to our application and send data to server. So, the kids require to have two devices: phone/tablet and smart watch. We found that android phones and android wears's price are reasonable compare to others and hence is more affordable for the parents.

According to these reasons, we can conclude that android is the best choice for our project.

## 6.2. Existing applications

---

There are some healthy habit applications were developed but It is not match to the requirement and the concept of our application. This is the list of all those applications:

A. Easy Eater 2: In Easy Eater, kids are responsible for naming and keeping a pet healthy and happy by feeding it the same foods they eat. Set in a magical forest, Easy Eater boasts a motley cast of characters that teach food groups and encourage food recognition. Kids learn that shrimp, tofu and nuts are proteins and that avocados are fruits, for example. Healthy choices earn "grub bucks(money)" to buy application accessories and real world prizes.

B. Eat and Move-O-Matic: This application helps kids to understand the relationship between food and exercise. The application compares the calories they eat with the time it takes to burn them off with activities that range from doing homework to dancing. With a colorful and engaging design that feels like a video game, Eat and Move-O-Matic offers ideas and tips for healthy alternatives to high-calorie foods like burgers and fries.

C. Healthy Heroes 1 & 2: Nutrition for Kids - (The application is full of bugs and the video commercial is unable to stop. The kid actor in the game must feed the monster, the monster just show the feeling happy or not when he eats the food )

D. As the Healthy Heroes in this game, kids are charged with saving the city of Yogopolis from Hungry Monsters. Through 36 levels of game play, kids fend off the Hungry Monsters with healthy foods like fruits and vegetables. Junk foods anger the monsters and prevent advancement. Kids learn to recognize healthy foods and eating habits throughout the game.

E. Perfect Picnic teaches food safety skills with a goal to create the safest picnic operation in the park. The game encourages players to wash hands, use a food thermometer to measure internal food temperatures, keep perishable foods at safe temperatures and keep preparation surfaces clean.

F. Smash Your Food – (there is an Android application “Yummy Smash Smash and Fast” . Player selects images that are not junk food to gain more points. This does not make the child eat better!) This popular application let's you smash food to see its actual sugar, salt and oil content by the numbers compared to what's recommended. Kids will enjoy smashing real images of a burger, imploding a can of cola and pounding a pizza to greasy smithereens. Application masters can unlock or buy new food fridges to keep the learning going.

G. Veggie Circus Farm – (Not for Android OS.)

H. Veggie Circus Farm helps children as young as two years old recognize vegetables. Led by Brianna the butterfly or Brian the bee, the application provides animated vegetable performances that teach kids how to pronounce the names of vegetables and basic nutrition benefits without the need to read.

I. Eat-And-Move-O-Metic teaches kids the relationship between taking calories in and burning them off through exercise and can be played as a web game.

J. Body Quest - Food of the Warrior. Auburn University, Alabama Cooperative Extension. Meet 6 superheroes who gain their superpowers from healthy food and drinks. This series of 6 apps/lessons teaches children about healthy foods with games.

K. Grow It-Know It. University of Nebraska-Lincoln. Teaches children where their food (and other simple items) comes from in agriculture.

L. Catch the Carrot. University of Illinois Extension. Answer trivia questions about different foods. Correct answers release a falling food that you must catch in your shopping cart.

M. Snack Planet. University of Nebraska-Lincoln. Player helps the robot to move quickly through the maze, pick healthy foods and reach the exit before time runs out. A warning though, is to steer clear of unhealthy snacks because they reduce the time and score. Work It Off. Humanitarian Free and Open Source. Teaches children the relationship between what they eat and the calories they burn.

N. Max's Plate. Merryweather Farms LLC allows kids to play with their food as they learn about the five food groups. Children can tap and drag each food image to the correct area of the plate to score points and progress through the game's three exciting levels. The game also includes a "My Plate" section that gives users a simple way to track their daily servings.

O. Frutas y verduras para niños – Touch the food image and find the couple. It's better for memory but not for healthy nutrition.

P. Poco YumYum comida para niños – Yum-Yum has to eat as many food items as possible in order to stay healthy and happy. The boy has the challenge of choosing the healthiest food for Yum-Yum: carrots, onions, cabbage, broccoli, milk or juice. You can even treat your lovely pet with some ice cream, lollipop, cupcake or pizza. If this cute character eat something inedible, becoming sick and sad. But when you eat delicious dishes, it is so happy that breaks into a dance! Help always hungry little Yum-Yum grow in a healthy monster.

Q. Hora de Comer: Diet Panda - Panda always fed with healthy food. If food has no energy and can not play minigames.

R. Emma breakfast -KIDS - In the educational game, the player will learn from many typical foods and drinks to choose the right one. With the help of Emma each day arises a healthy varied breakfast that could be well prepared on your real breakfast table.

S. EduKitchen-Kids Educational – Play only in English. This is a kitchen where if you squeeze on some food, this is pronounced in English. Once you have done and cooked food is recycling and pronounce "Let's recycle." In the menu you can eat junk food or normal food. The only way to know if food is healthy tightening tab healthy food.

T. Veggie Bottoms Lite – a slider funny pictures with descriptions. Like a picture book. Very tacky.

U. Sopa Hacedor – soup recipes. Encourage children to cook.

V. Awesome Eats -Sort, stack, pack and plate a rainbow of fresh-from-the-garden foods! And now recycle items after lunch! In each level you'll stack and sort fruits, veggies and whole grains across wacky contraptions to win stars and score big points! Be on the lookout for thieving birds and tap to shoo them away for extra points. You and your kids will unlock hours of challenging game play and get loads of healthy eating tips along the way.

W. Cocomong Season 2 - Our friend Cocomong is back with robocong who becomes stronger by eating fruits and vegetables.Cocomong and his friends are having a blast at fresh park and fresh world that are filled with fresh vegetables full of nutrients. However, Germ King and his army, created from global warming, plots to take over fresh world by lowering fresh Energy through Cocomong and his fresh world friends' bad eating habits. 26 fun and interesting stories unfold for your children to learn the preciousness of nature and form healthy eating habits.

## 6.3. Spring security framework

---

Spring security is a security framework that provides declarative security for Spring-based applications. Spring security provides a comprehensive security solution, handling authentication and authorization, at both the web request level and at the method invocation level. Based on the Spring framework, Spring security takes full advantage of dependency injection (DI) and aspect oriented techniques. [10] The project was started in late 2003 as ‘Acegi Security’ by Ben Alex, with it being publicly released under the Apache license in March 2004. The real power of Spring security is found in how easily it can be extended to meet custom requirements.

It provides some main features such as:

- Comprehensive and extensible support for both Authentication and Authorization

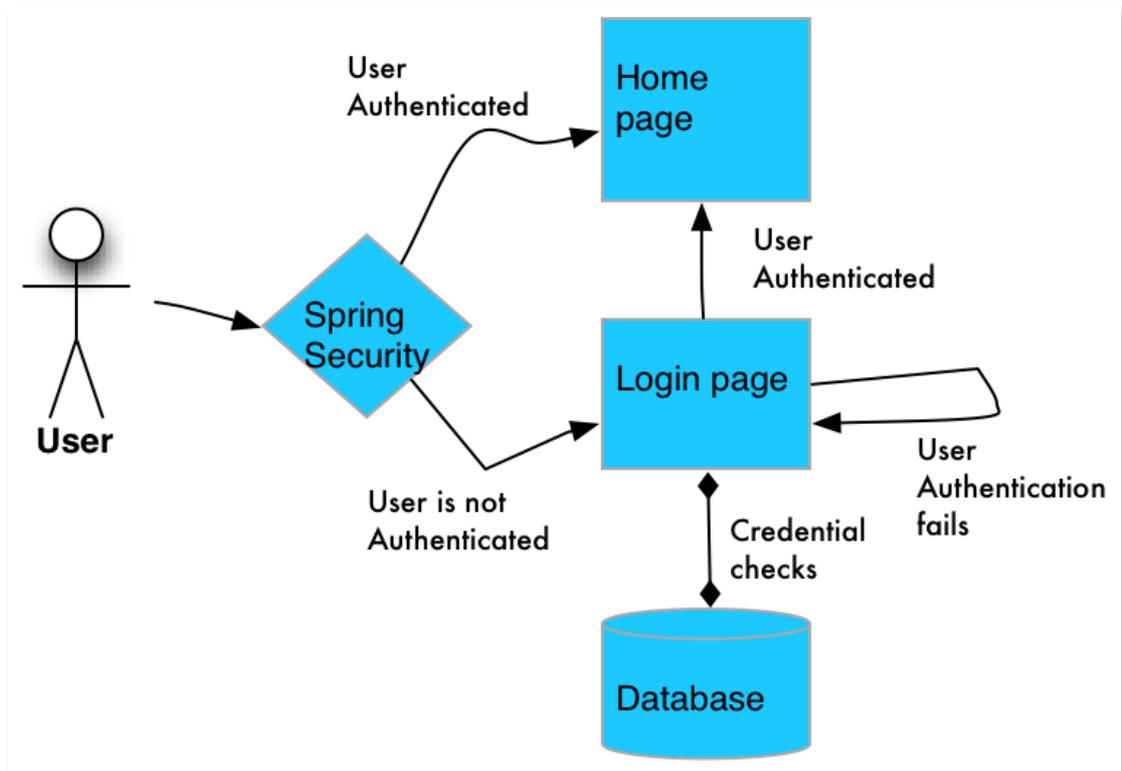
Authentication is the assurance that [11] the user is actually the user he is claiming to be, for example, when the user logs into any application and gives his credentials, he authenticates himself. At the authentication level, spring supports various authentication models such as Http Basic authentication or Form Based authentication among others.

Authorization is the assurance that the user is allowed to access only those resources that he is authorized to use. For example, in a corporate application, there are some parts of an application where only admin have access and to some parts all the employees have access. These access rules are determined by the access rights given to each user of the system. At the authorization level, spring targets three main areas: authorizing web request, authorizing whether methods can be invoked and authorizing access to individual domain object instances.

- Protection against attacks like session fixation, clickjacking, cross site request forgery (csrf)

- Servlet API integration and integration with Spring Web MVC.

The diagram from Figure 17 shows the flow process of spring security.



**Figure**

**17: Spring security flow**

“e-Health tools supporting healthy habits for obese and diabetic kids” is a medical system. So, medical information, or patient health information should be accessible by only their doctor. Moreover, every transaction of their information should be secured. That’s why we tried to make our system more secured. There are two major areas of application security are “authentication” and “authorization” (or “access-control”). These are the two main areas that spring security targets. So, we think that Spring security is matched to our system. We decide to integrate spring security with our Spring web MVC project.

## 7. Analysis and design system

---

This part describes in more details about of the application and user's role.

### 7.1. Use case diagrams

---

This application has four different interfaces according to user type. It is designed for three types of user such as "Kid", "Parent", "Doctor" and "Admin".

As shown in the Figure 18, Kid view: is on mobile application and it's provided to kids to use it. In this view, there are some main functions such as: register account, log in, modify account, log out, validate food, CRUD exercise schedule, view process, CRUD exercise activity, CRUD chat, view grad, delete photo, CRUD food info, input food photo, take photo and choose photo.

As shown in the Figure 18, Parent view: is on mobile application and it's provided to parents to use it. Parents can perform some functions such as: register account, log in, modify account, log out, validate food, CRUD kid's exercise schedule and view kid's process.

As shown in the Figure 19, Doctor view: is on web page of our system and it's provided to doctors to use it. Doctors can perform some function such as: log in, modify account, log out, CRUD target food for kid, CRUD target exercise for kid, view kid's process and search for kids.

As shown in the Figure 19, Admin view: is on web page of our system and it's provided to admins to use it. Admin scan perform some function such as: create new doctor or admin, search for any doctor or admin, deactivate doctor or admin, log in, modify account and logout.

There are also 2 other actors: System and Sytem timer, which perform some important functions.

As shown in the Figure 18 and 19, System: Form kid team, organize team match calendar, assign photo for validating and generate kid process chat.

As shown in the Figure 18, System timer: validate exercise activity from smart watch with the data that user input, validate the validation of photo, return result to every team, find the winner team, feedback to kid, assess grad, compute different percentage of food and compute different percentage of exercise activity.

With the description above, we can represent the use cases Figure 18 and 19. The different color of each every case defines its section because this system is divided in to different section such as social network, game, kid health trace and the game+kid health trace.

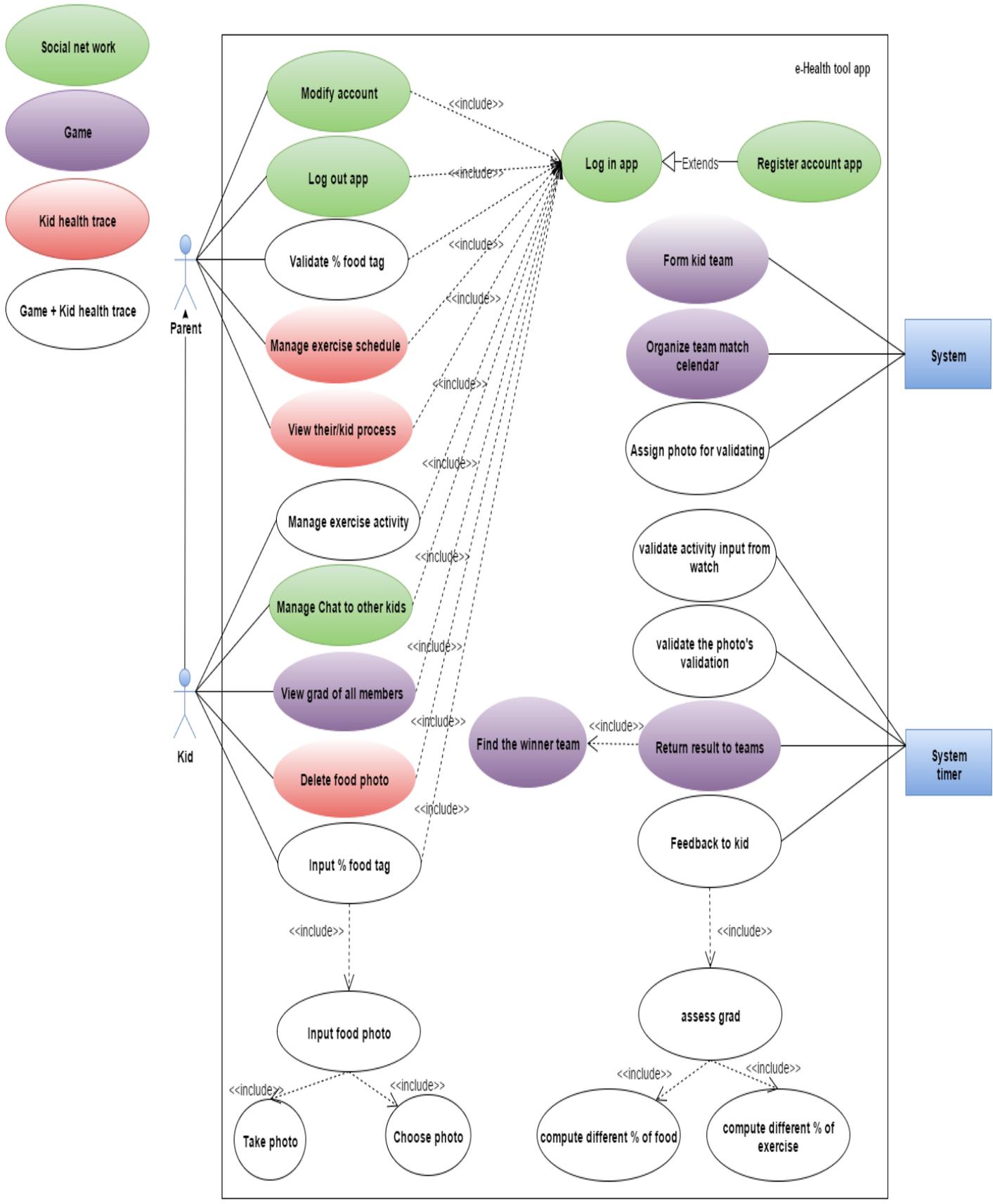


Figure 18: Use case diagram for mobile



**Figure 19: Use case diagram for web page**

### 7.1.1. Use case specification

In this path we will describes more specific of each use case.

#### - On mobile application

**Table 3: Use case specification Register account app**

Use Case File	
<b>ID</b>	1
<b>Name</b>	Register account app
<b>Description</b>	User wants to use this application on android mobile
Regular Flow	
<b>Actor</b>	Kid, Parent
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Actor has an android mobile</li> <li>2. Their device has network connection to server</li> </ol>
<b>Activation</b>	Actor click on the application icon to lunch it
<b>Description</b>	<ol style="list-style-type: none"> <li>1. App show up an information form(email, passwords, verify passwords, telephone, gender) <ul style="list-style-type: none"> <li>- Select type of user</li> <li>- Actor inputs all the information form</li> </ul> </li> <li>2. Actors click “create”</li> <li>3. App shows a successful message</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. Verify email or telephone number</li> <li>2. New user is created</li> <li>3. User can use this app, they are in their profile page</li> </ol>
Alternative Flow	
<b>Description</b>	<ol style="list-style-type: none"> <li>1. In step 4, Application shows an error message <ul style="list-style-type: none"> <li>- It’s not the correct email address</li> <li>- Passwords is not match</li> </ul> </li> </ol>

<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. User re-input the correct information</li> <li>2. Registration is success</li> </ol>
-----------------------	--

**Table 4: Use case specification Log in app**

<b>Use Case File</b>	
<b>ID</b>	2
<b>Name</b>	Log in app
<b>Description</b>	User wants to Log in to this application on android mobile
<b>Regular Flow</b>	
<b>Actor</b>	Kid, Parent
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Actor has an android mobile</li> <li>2. Their device has network connection to server</li> </ol>
<b>Activation</b>	Actor click on the application icon to lunch it
<b>Description</b>	<ol style="list-style-type: none"> <li>1. App show up an information form(username, passwords)</li> <li>2. Actor inputs the all information form</li> <li>3. Actor click “Ok”</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. User get in to the app</li> <li>2. They are in Home page and they can use the application</li> </ol>
<b>Alternative Flow</b>	
<b>Description</b>	<ol style="list-style-type: none"> <li>1. In step 3, Application shows an error message <ul style="list-style-type: none"> <li>- It’s not the correct email address or passwords</li> </ul> </li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. User re-input the information</li> <li>2. User can’t Log in</li> <li>3. User request new passwords</li> <li>4. App server send new passwords to their email address</li> </ol>

**Table 5: Use case specification Modify account**

<b>Use Case File</b>	
<b>ID</b>	3
<b>Name</b>	Modify account

<b>Description</b>	User wants to modify their account information
<b>Regular Flow</b>	
<b>Actor</b>	Kid, Parent
<b>Precondition</b>	1. Actor has been logged in/ just finish their registration new account
<b>Activation</b>	Actor click on modify button
<b>Description</b>	<ol style="list-style-type: none"> <li>1. App show up an information form (name, surname, age, weight, high, occupation)</li> <li>2. Actor inputs the all information form</li> <li>3. Actor click “Save”</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. Server update user’s information in database</li> <li>2. App refresh to get the updated data</li> </ol>
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 6: Use case specification Form kid team**

<b>Use Case File</b>	
<b>ID</b>	4
<b>Name</b>	Form kid team
<b>Description</b>	Team among N kids are generated base on the kid profile
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	1. There are N kids have registered in to application
<b>Activation</b>	System count the number of registered kid
<b>Description</b>	<ol style="list-style-type: none"> <li>1. System formed team with N kid</li> <li>2. Each kid is in only 1 team</li> </ol>
<b>Post condition</b>	1. Kids can post their food photo
<b>Alternative Flow</b>	
<b>Description</b>	1. There is not enough kid to form a team

<b>Post condition</b>	1. Kid wait until she/he can be fitted in any team
-----------------------	--

**Table 7: Use case specification Organize team match calendar**

<b>Use Case File</b>	
<b>ID</b>	5
<b>Name</b>	Organize team match calendar
<b>Description</b>	System will prepare a calendar for a match between 2 team
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	1. There are 2 team available to match
<b>Activation</b>	System find out that there are teams available to match
<b>Description</b>	<ol style="list-style-type: none"> <li>1. System find 2 teams which are the same number of member</li> <li>2. System find that those teams are not in any match</li> <li>3. System set the start date match and end date match</li> </ol>
<b>Post condition</b>	1. Kids who are in those two team get notification about their match
<b>Alternative Flow</b>	
<b>Description</b>	<ol style="list-style-type: none"> <li>1. There is not enough team to be matched</li> <li>2. The team is not available</li> </ol>
<b>Post condition</b>	1. Kids wait until there are available team

**Table 8: Use case specification Input percentage of each food tag**

<b>Use Case File</b>	
<b>ID</b>	6
<b>Name</b>	Input percentage of each food tag
<b>Description</b>	Kid user input the percentage of Fruits, Vegetables, Carbohydrates and Meat. But, firstly they need to input a food photo by taking new photo or choose from their library.
<b>Regular Flow</b>	
<b>Actor</b>	Kid

<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Kid is logged in to their account</li> <li>2. Kid is in a team</li> <li>3. Their team is matched</li> </ol>
<b>Activation</b>	Kid click button post photo
<b>Description</b>	<ol style="list-style-type: none"> <li>1. Kid chose photo from their phone library or take new photo</li> <li>2. Kid set % of each tag</li> <li>3. Kid click “Post”</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. Photo is sent to server</li> </ol>
<b>Alternative Flow</b>	
<b>Description</b>	<ol style="list-style-type: none"> <li>1. In step 1, Kid has uploaded the wrong photo</li> <li>2. In step 2, Kid set the wrong % of his food</li> <li>3. In step 3, the kid post is not successful because of his/her internet connection</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. Kid delete his post</li> <li>2. Kid re-modify the % of his food</li> <li>3. Kid click “re-try”</li> </ol>

**Table 9: Use case specification Assign photo for validating**

<b>Use Case File</b>	
<b>ID</b>	7
<b>Name</b>	Assign photo for validating
<b>Description</b>	<p>When system gets the input photo from kid user, it will assign that photo to other users for validating.</p> <ul style="list-style-type: none"> <li>- 25% to members of the same team</li> <li>- 50% to other teams of the same competition</li> <li>- 25% to members of other competitions</li> <li>- 50% to parents, selected at random</li> </ul>
<b>Regular Flow</b>	

<b>Actor</b>	System
<b>Precondition</b>	1. A photo was sent to server
<b>Activation</b>	System gets a new photo from a kid
<b>Description</b>	<ol style="list-style-type: none"> <li>1. System sent that photo to other user (kid and parent): <ul style="list-style-type: none"> <li>- 25% to members of the same team</li> <li>- 50% to other teams of the same competition</li> <li>- 25% to members of other competitions</li> <li>- 50% to parents, selected at random</li> </ul> </li> <li>2. Other users (kid and parent) get a notification with a photo to validate the percentage of each tag in that food photo</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. Other users (kid and parent) submit their validation to server</li> <li>2. Server save those result in database</li> </ol>
<b>Alternative Flow</b>	
<b>Description</b>	1. Other users don't validate the photo
<b>Post condition</b>	1. System will validate that photo with other methods

**Table 10: Use case specification Validate food**

Use Case File	
<b>ID</b>	8
<b>Name</b>	Validate food
<b>Description</b>	When user receives notification (group of photo) to validate a food photo from unknown source, He will open notification, which is a list of food picture and select YES/ NO on each photo. After that, He can submit that validation to server.
<b>Regular Flow</b>	
<b>Actor</b>	Parent, Kid

<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Server sends a photo to user for validating</li> <li>2. User gets a notification alert</li> <li>3. User clicks open the app</li> </ol>
<b>Activation</b>	User checks their notification and click
<b>Description</b>	<ol style="list-style-type: none"> <li>1. User views a list of photos</li> <li>2. User clicks YES or NO</li> <li>3. User clicks “submit” to submit their validation to server</li> </ol>
<b>Post condition</b>	1. That validation value is submitted to server database
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 11: Use case specification Manage exercise schedule**

<b>Use Case File</b>	
<b>ID</b>	9
<b>Name</b>	Manage exercise schedule
<b>Description</b>	Every kid defines a profile regarding its preferences about sports and the timetable that they use to practice (supported by parents when under 12 years old).
<b>Regular Flow</b>	
<b>Actor</b>	Parent, Kid
<b>Precondition</b>	1. Kid logs in to their account or Parent goes to their kid’s profile
<b>Activation</b>	User clicks on “Exercise schedule”

<b>Description</b>	<ol style="list-style-type: none"> <li>1. User clicks on “Exercise schedule”</li> <li>2. App goes to exercise schedule table</li> <li>3. User clicks “add” to add new activity to any date</li> <li>4. User selects Date (Monday-Sunday) <ul style="list-style-type: none"> <li>- User clicks “add activity” to select Time(available time) and Type of activity</li> </ul> </li> <li>5. User clicks “Save”</li> <li>6. App back to exercise schedule table</li> <li>7. User can clicks on any date to delete or modify activity</li> </ol>
<b>Post condition</b>	1. Exercise schedule of that kid is submitted to server
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 12: Use case specification Manage exercise activity**

<b>Use Case File</b>	
<b>ID</b>	10
<b>Name</b>	Manage exercise activity
<b>Description</b>	<p>Kid needs to input their exercise activity after they finish it.</p> <p>To confirm, they will have a watch device detects and exercise activity to server. Some activity could come with non-scheduled hours (as for example, playing soccer in the school playground...)</p>
<b>Regular Flow</b>	
<b>Actor</b>	Kid
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Kid logs in to their account</li> <li>2. He/she are in their profile page</li> </ol>
<b>Activation</b>	Click on “Share activity”

<b>Description</b>	<ol style="list-style-type: none"> <li>1. User clicks on “ Share activity ”</li> <li>2. App shows a activity form</li> <li>3. User selects a type of activity that they did</li> <li>4. User inputs total hours in of their activity</li> <li>5. User clicks “Synchronize” to get data from watch</li> <li>6. User clicks “Submit” to submit those data to server</li> </ol>
<b>Post condition</b>	1. Activity input from user and watch are submitted to server
<b>Alternative Flow</b>	
<b>Description</b>	1. In step 5, User has can't synchronize. There is a watch detected
<b>Post condition</b>	1. User will be able to submit without data from watch

**Table 13: Use case specification Validate the photo validation**

<b>Use Case File</b>	
<b>ID</b>	11
<b>Name</b>	Validate the photo validation
<b>Description</b>	<p>Every day, System will validate every food photo. System will compare value between input of the photo owner and the validation of other users. A procedure on the validated photos determine the different percentage of food ingested in every dish.</p> <ul style="list-style-type: none"> <li>- Photos are validated by parents should have a higher confidence</li> </ul>
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. There are data validation are submitted from user</li> <li>2. There is the value from the owner input</li> </ol>
<b>Activation</b>	System time will do it every day
<b>Description</b>	1. System selects data validation of each photo from database and compute what value should be for that photo(from other users)
<b>Post condition</b>	1. Save the final value of each photo to database
<b>Alternative Flow</b>	
<b>Description</b>	

<b>Post condition</b>	
-----------------------	--

**Table 14: Use case specification Validate activity input from watch**

<b>Use Case File</b>	
<b>ID</b>	12
<b>Name</b>	Validate activity input from watch
<b>Description</b>	Everyday system will validate all activities of user. It will compare value between owner input and watch input
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. There are data submitted from owner</li> <li>2. There are the value input from watch</li> </ol>
<b>Activation</b>	System time will do it every day
<b>Description</b>	<ol style="list-style-type: none"> <li>1. System selects data validation of each activity from database and compute what value should be for that activity(from watch)</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>2. Save the final value of each activity to database</li> </ol>
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 15: Use case specification Feed back to kid**

<b>Use Case File</b>	
<b>ID</b>	13
<b>Name</b>	Feed back to kid
<b>Description</b>	Every day, System will send a feed back to kid about their state after system has assessed grad to kid.
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Kid is assessed</li> </ol>
<b>Activation</b>	System time will do it every day

<b>Description</b>	<ol style="list-style-type: none"> <li>1. System selects grad of each kid from database</li> <li>2. System sends a notification of the grad to kid</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. Kid gets the notification</li> </ol>
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 16: Use case specification Assess grad**

Use Case File	
<b>ID</b>	14
<b>Name</b>	Assess grad
<b>Description</b>	<p>Every day, System will compute total grad for every kid.</p> <p>To assess total grad of a kid, system need to :</p> <ol style="list-style-type: none"> <li>1. Compute different% of food</li> <li>2. Compute different% of activity</li> </ol>
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	There are the result of nutrition and activity
<b>Activation</b>	System time will do it every day
<b>Description</b>	<ol style="list-style-type: none"> <li>1. System will calculate of both value</li> <li>2. Grad the result A-E</li> </ol>
<b>Post condition</b>	Save result to database
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 17: Use case specification Compute different% of food**

Use Case File	
<b>ID</b>	15
<b>Name</b>	Compute different% of food

<b>Description</b>	<ul style="list-style-type: none"> <li>- Every day, system will compute the different % of food :</li> <li>- An aggregation measure computes the percentage of ingested food every day.</li> <li>- A difference measure computes the deviation of the ingested food and the recommended food.</li> <li>- Map the difference on a scale for nutrition fitness, from A to E (best, worst).</li> </ul>
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	<ul style="list-style-type: none"> <li>- There are the value of food from owner input and other user's validation</li> <li>- There are the value of target food in database</li> </ul>
<b>Activation</b>	System time will do it every day
<b>Description</b>	1. System will calculate the different value of each food photo
<b>Post condition</b>	Save result to database
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 18: Use case specification Compute different% of activity**

Use Case File	
<b>ID</b>	16
<b>Name</b>	Compute different% of activity

<b>Description</b>	<p>Every day, system will compute the different % of activity :</p> <ul style="list-style-type: none"> <li>- An aggregation measure computes the amount of exercise performed every day.</li> <li>- Map the difference on a scale for exercise fitness, from A to E (best, worst).</li> <li>- The final assessment of the kid is performed based on a multi-criteria function. It is obtained in a scale from A to E. This is the kid state</li> </ul>
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	<ul style="list-style-type: none"> <li>- There are the value of activity from owner input and watch validation</li> <li>- There are the value of target activity in database</li> </ul>
<b>Activation</b>	System time will do it every day
<b>Description</b>	1. System will calculate the different value of each activity that user have shared
<b>Post condition</b>	Save result to database
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 19: Use case specification View kid process**

Use Case File	
<b>ID</b>	17
<b>Name</b>	View kid process
<b>Description</b>	Kid can view the process of themselves. Parent also can view their kid process. This process chart will show about the progress of their activity and nutrition that they have been doing compare to target value.

Regular Flow	
<b>Actor</b>	Kid, Parent
<b>Precondition</b>	1. User is logged in and has network connection
<b>Activation</b>	1. For kid, he/she clicks on My progress 2. For parent, he/she clicks on My child progress
<b>Description</b>	1. User goes to their home page 2. User clicks on “_progress ” button 3. App will show a progress chart, 4. If parent have more than one child, Application will show a select box for parent to choose their kid’s name.
<b>Post condition</b>	- User back to their home page
Alternative Flow	
<b>Description</b>	
<b>Post condition</b>	

**Table 20: Use case specification View grad of all member**

Use Case File	
<b>ID</b>	18
<b>Name</b>	View grad of all member
<b>Description</b>	Kid can view the process of his or her team, which has, note the process of each member in that chart.
Regular Flow	
<b>Actor</b>	Kid
<b>Precondition</b>	1. User is logged in and has network connection
<b>Activation</b>	1. User clicks My team

<b>Description</b>	<ol style="list-style-type: none"> <li>1. User goes to their home page</li> <li>2. User clicks on “My team” button</li> <li>3. App will show a progress chart of their team which has list the name of all member</li> </ol>
<b>Post condition</b>	- User goes back to their home page
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 21: Use case specification Manage chat to other kids**

Use Case File	
<b>ID</b>	19
<b>Name</b>	Manage chat to other kids
<b>Description</b>	Kid can chat to their team member
Regular Flow	
<b>Actor</b>	Kid
<b>Precondition</b>	1. User is logged in and has network connection
<b>Activation</b>	1. User clicks My team
<b>Description</b>	<ol style="list-style-type: none"> <li>1. User goes to their home page</li> <li>2. User clicks on “My team” button</li> <li>3. User clicks on “Member”</li> <li>4. App will list all member in team</li> <li>5. User clicks on any name of member</li> <li>6. App will show a chat page <ul style="list-style-type: none"> <li>- User can chat to their member</li> <li>- User can delete his chat</li> <li>- User can delete entire message</li> </ul> </li> </ol>
<b>Post condition</b>	- User goes back to their home page
Alternative Flow	
<b>Description</b>	
<b>Post condition</b>	

**Table 22: Use case specification Delete food photo**

Use Case File	
<b>ID</b>	20
<b>Name</b>	Delete food photo
<b>Description</b>	Kid can delete their food photo
Regular Flow	

<b>Actor</b>	Kid
<b>Precondition</b>	1. User is logged in and has network connection
<b>Activation</b>	1. User clicks option on that photo
<b>Description</b>	<ol style="list-style-type: none"> <li>1. User goes to their home page</li> <li>2. User clicks on “Option” button on any photo</li> <li>3. User chooses delete option</li> <li>4. App shows a message confirm to delete</li> <li>5. User clicks OK</li> </ol>
<b>Post condition</b>	<ul style="list-style-type: none"> <li>- App sends request delete to server</li> <li>- App deletes that photo</li> <li>- Server marks that photo status to isDeleted</li> </ul>
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 23: Use case specification Return result to team**

<b>Use Case File</b>	
<b>ID</b>	21
<b>Name</b>	Return result to team
<b>Description</b>	<ul style="list-style-type: none"> <li>- Once a week (Sunday) all teams will get the result of their match during that week</li> <li>- The winner of the match is the team with the highest score.</li> </ul>
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	1. There is a result of team match in database
<b>Activation</b>	1. System timer tells that it’s time to send result
<b>Description</b>	<ol style="list-style-type: none"> <li>1. System selects result from database</li> <li>2. System sends result to all kid in each team</li> </ol>

<b>Post condition</b>	- Kid gets notification about their match
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 24: Use case specification Find the winner team**

<b>Use Case File</b>	
<b>ID</b>	22
<b>Name</b>	Find the winner team
<b>Description</b>	Once a week, System will get calculate the total score and find the winner between each two team.
<b>Regular Flow</b>	
<b>Actor</b>	System
<b>Precondition</b>	1. There is any teams match
<b>Activation</b>	1. System timer tells that it's time to calculate the result
<b>Description</b>	1. System selects all the score of each photo and activity of each kid in the team from database  2. System calculates those score
<b>Post condition</b>	System saves score to database
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 25: Use case specification Log out**

<b>Use Case File</b>	
<b>ID</b>	23
<b>Name</b>	Log out
<b>Description</b>	User can log out from the application any time.
<b>Regular Flow</b>	
<b>Actor</b>	Kid, Parent
<b>Precondition</b>	1. User is logged in

<b>Activation</b>	1. User clicks log out
<b>Description</b>	1. User goes to their profile page 2. User clicks log out
<b>Post condition</b>	Application will be log out from the current state
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

- On web page

**Table 26: Use case specification Register account on server side**

<b>Use Case File</b>	
<b>ID</b>	1
<b>Name</b>	Register account on server side
<b>Description</b>	Doctor will have his own account to access to e-Health system
<b>Regular Flow</b>	
<b>Actor</b>	Doctor
<b>Precondition</b>	1. User has a web browser 2. Their device has network connection to server
<b>Activation</b>	User input web url in browser
<b>Description</b>	1. User goes to web site with an url 2. There is a Log in page 3. User clicks on a link to register new account 4. User inputs some information form (email, passwords, verify passwords, telephone, gender, specialist, hospital) 5. Actor inputs the all information form 6. Actors clicks “create” 7. App shows a successful message

<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. New user is created</li> <li>2. User can use this app, they are in their profile page</li> </ol>
<b>Alternative Flow</b>	
<b>Description</b>	<ol style="list-style-type: none"> <li>1. In step 4, Application shows an error message <ul style="list-style-type: none"> <li>- It's not the correct email address</li> <li>- Passwords is not match</li> </ul> </li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. User re-input the correct information</li> <li>2. Registration is success</li> </ol>

**Table 27: Use case specification Log in**

<b>Use Case File</b>	
<b>ID</b>	2
<b>Name</b>	Log in
<b>Description</b>	User wants to Log in to web site
<b>Regular Flow</b>	
<b>Actor</b>	Doctor, Admin
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. Actor has an web browser</li> <li>2. Their device has network connection to server</li> </ol>
<b>Activation</b>	User input web url in browser
<b>Description</b>	<ol style="list-style-type: none"> <li>1. There is a log in page show up an information form(email/telephone, passwords)</li> <li>2. Actor inputs the all information form</li> <li>3. Actor click "Ok"</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. User get in to the web site</li> <li>2. They are in Home page</li> </ol>
<b>Alternative Flow</b>	
<b>Description</b>	<ol style="list-style-type: none"> <li>1. In step 3, there is a pop-up shows an error message <ul style="list-style-type: none"> <li>- It's not the correct email address or passwords</li> </ul> </li> </ol>

<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. User re-input the information</li> <li>2. User can't Log in</li> <li>3. User requests new passwords</li> <li>4. Server sends new passwords to their email address</li> </ol>
-----------------------	--

**Table 28: Use case specification Modify account**

Use Case File	
<b>ID</b>	3
<b>Name</b>	Modify account
<b>Description</b>	User wants to modify their account information
Regular Flow	
<b>Actor</b>	Doctor, Admin
<b>Precondition</b>	1. Actor has been log in/ just finish their registration new account
<b>Activation</b>	Actor click on modify button
<b>Description</b>	<ol style="list-style-type: none"> <li>1. A web page shows up an information form</li> <li>2. Actor inputs the all information form</li> <li>3. Actor clicks "Save"</li> </ol>
<b>Post condition</b>	<ol style="list-style-type: none"> <li>1. Server update user's information in database</li> <li>2. Page reload to get the updated data</li> </ol>
Alternative Flow	
<b>Description</b>	
<b>Post condition</b>	

**Table 29: Use case specification View kid process**

Use Case File	
<b>ID</b>	4
<b>Name</b>	View kid process

<b>Description</b>	Doctors can view the process of kids who they are treating. This process chart will show about the progress of kid's activity and nutrition that they have been doing compare to target value.
<b>Regular Flow</b>	
<b>Actor</b>	Doctor
<b>Precondition</b>	1. User is logged in and has network connection
<b>Activation</b>	1. User clicks on kid progress progress
<b>Description</b>	<ol style="list-style-type: none"> <li>1. User goes to their home page</li> <li>2. User clicks on "kid progress " button</li> <li>3. App will show list name of kids</li> <li>4. Users click on any name</li> <li>5. There is a page with a progress chart of that kid</li> </ol>
<b>Post condition</b>	- User goes back to their home page
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 30: Use case specification Set recommend food to kid**

<b>Use Case File</b>	
<b>ID</b>	5
<b>Name</b>	Set recommend food to kid
<b>Description</b>	Doctor need to set a target food to each kid that they are treating.
<b>Regular Flow</b>	
<b>Actor</b>	Doctor
<b>Precondition</b>	1. User is logged in and has network connection
<b>Activation</b>	Click on set recommend food

<b>Description</b>	<ol style="list-style-type: none"> <li>1. User goes to their home page</li> <li>2. User clicks on “set recommend food” button</li> <li>3. App will show an input form (select kid, type of food, value of food, start date, end date) <ul style="list-style-type: none"> <li>- User clicks button add to add more info (, type of food, value of food)</li> </ul> </li> </ol>
<b>Post condition</b>	- User goes back to their home page
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 31: Use case specification Set recommend activity to kid**

<b>Use Case File</b>	
<b>ID</b>	6
<b>Name</b>	Set recommend activity to kid
<b>Description</b>	Doctor need to set a target activity to each kid that they are treating.
<b>Regular Flow</b>	
<b>Actor</b>	Doctor
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. User is logged in and has network connection</li> </ol>
<b>Activation</b>	User click on set recommend food
<b>Description</b>	<ol style="list-style-type: none"> <li>1. User goes to their home page</li> <li>2. User clicks on “set recommend activity” button</li> <li>3. App will show an input form (select kid, type of activity, value , start date, end date) <ul style="list-style-type: none"> <li>- User clicks button add to add more info (type of activity, value)</li> </ul> </li> </ol>
<b>Post condition</b>	- User back to their home page
<b>Alternative Flow</b>	

<b>Description</b>	
<b>Post condition</b>	

**Table 32: Use case specification Generate kid process chart**

Use Case File	
<b>ID</b>	7
<b>Name</b>	Generate kid process chart
<b>Description</b>	System will generate a kid process chart to present the progress of a kid.
Regular Flow	
<b>Actor</b>	System
<b>Precondition</b>	There is a request to see kid process
<b>Activation</b>	User clicks to see the process chart
<b>Description</b>	<ol style="list-style-type: none"> <li>1. System selects all data of a kid from database</li> <li>2. System uses that data to present a chart</li> </ol>
<b>Post condition</b>	
Alternative Flow	
<b>Description</b>	
<b>Post condition</b>	

**Table 33: Use case specification Log out**

Use Case File	
<b>ID</b>	8
<b>Name</b>	Log out
<b>Description</b>	User can log out from web site any time.
Regular Flow	
<b>Actor</b>	Doctor, Admin
<b>Precondition</b>	1. User is logged in
<b>Activation</b>	1. User clicks log out

<b>Description</b>	<ol style="list-style-type: none"> <li>1. User goes to their pro file page</li> <li>2. User clicks log out</li> </ol>
<b>Post condition</b>	Application will log out from the current state
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 34: Use case specification Create account for doctor /admin**

<b>Use Case File</b>	
<b>ID</b>	9
<b>Name</b>	Create account for doctor /admin
<b>Description</b>	User can crate new account for doctor or admin.
<b>Regular Flow</b>	
<b>Actor</b>	Admin
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. User is logged in</li> </ol>
<b>Activation</b>	<ol style="list-style-type: none"> <li>1. User clicks button add new user</li> </ol>
<b>Description</b>	<ol style="list-style-type: none"> <li>1. User is at home page</li> <li>2. User clicks button add new user</li> <li>3. User chooses type of user <ul style="list-style-type: none"> <li>- If user chooses the doctor option, he/she need to complete some information such as: Hospital name, Username, Surname, Name, Telephone, Email and temporary password.</li> <li>- If user choose the admin option, he/she need to complete some information such as: Username, Surname, Name, Telephone, Email and temporary password.</li> <li>- User clicks button create</li> <li>- New user account is created</li> </ul> </li> </ol>

<b>Post condition</b>	New user account is created and page redirect back to home page
<b>Alternative Flow</b>	
<b>Description</b>	<ol style="list-style-type: none"> <li>1. User click button create <ul style="list-style-type: none"> <li>- Validation Error message: Username or Email or Telephone is already token.</li> <li>- password is not match</li> </ul> </li> </ol>
<b>Post condition</b>	User goes back to the data field and change to the new one.

**Table 35: Use case specification Deactivate account doctor/admin**

<b>Use Case File</b>	
<b>ID</b>	10
<b>Name</b>	Deactivate account doctor/admin
<b>Description</b>	User can deactivate doctor account or other admin account.
<b>Regular Flow</b>	
<b>Actor</b>	Admin
<b>Precondition</b>	<ol style="list-style-type: none"> <li>1. User is logged in</li> </ol>
<b>Activation</b>	<ol style="list-style-type: none"> <li>1. User clicks on any user that he/she want to deactivate.</li> </ol>
<b>Description</b>	<ol style="list-style-type: none"> <li>1. User is at home page</li> <li>2. User clicks on an user</li> <li>3. User clicks deactivate</li> </ol>
<b>Post condition</b>	That user account is set status to deactivate and page redirect back to home page
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	

**Table 36: Use case specification Select/Search doctor/admin**

<b>Use Case File</b>	
<b>ID</b>	11
<b>Name</b>	Select/Search doctor/admin
<b>Description</b>	User can search for any doctor or admin by their name
<b>Regular Flow</b>	
<b>Actor</b>	Admin
<b>Precondition</b>	1. User has logged in
<b>Activation</b>	1. User inputs doctor's name or admin's name in the search input bar
<b>Description</b>	1. User inputs doctor's name or admin's name in the search input bar  2. User clicks search or press Enter  3. If there're some results found, it'll be show in the table list.
<b>Post condition</b>	Table list in page home will refresh with the result
<b>Alternative Flow</b>	
<b>Description</b>	
<b>Post condition</b>	



## 7.3. System architecture

---

The system architecture describes the processes and communication between server and client is shown in the Figure 21. Here, the architecture is divided into different important blocks. Each block handle s different tasks.

On the server side:

- Game engine: It defines the logic of the game. It is a block to calculate the score, set the rule, define calendars for matching, form teams, find the winner and return the result.
- Social Network: This block will manage user accounts and profiles, relation between users and user authentication.
- Communications: A gate for communication between server and client. It allows both sides to transfer data and signal.
- Medical target setting: This block defines methods to set up a target medical, activity for each kid. Doctor inputs medical targets, else a default target is set.
- Medical “follower” function: It traces the different value between target value & the real value that kid input.
- Validation block: It is used to validate the data that kid and validator actor input. We could find out the most realizable data among all the validated data from user.
- Server DB defines the data structure for storing all information of this system.
- Web Interface: A web site that used to input target food & activity for each kid by doctor. User can view the process of any kid with this web page al well.

On client side:

- Game block: It manages how user upload photos, set percentage of food tag, receive photos to validate. It also represents users in a team, processes chats of the team and user grad.
- Social Network: It manages user names, passwords, log in session, chat with other users and user profile.
- Communication: It is a way for client request to server or sends data, signal to server.
- Monitoring: It monitors when user input data (food, activity).
- Sensor data gathering: It is a function that can get data from watch and save in to Mobile DB with its sensor.
- Mobile DB: It is a database in mobile phone used to store small data that is frequently used.
- App interface : An android application has user-friendly interface and easy to train for new user.

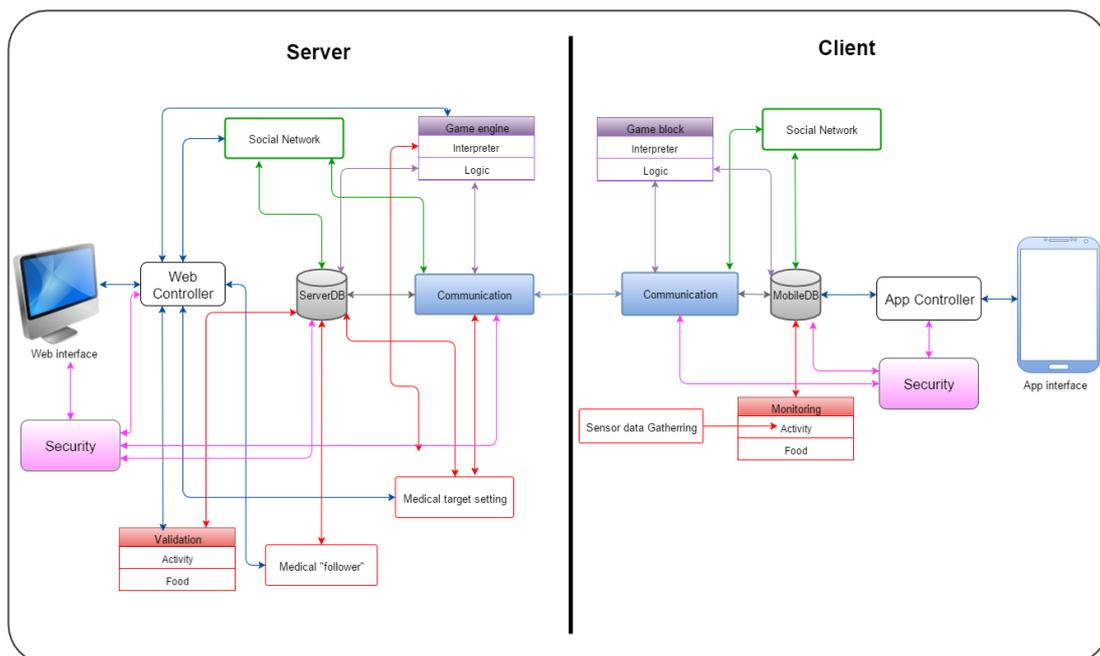


Figure 21: System architecture

## 7.4. Entity relationship diagram

There are 27 tables in the database that store different object information of this project on server side. This is the entity relationship diagram of all the tables that represent relation between tables and its cardinality. It also shows all the properties of each table which is also contains primary key and foreign key.

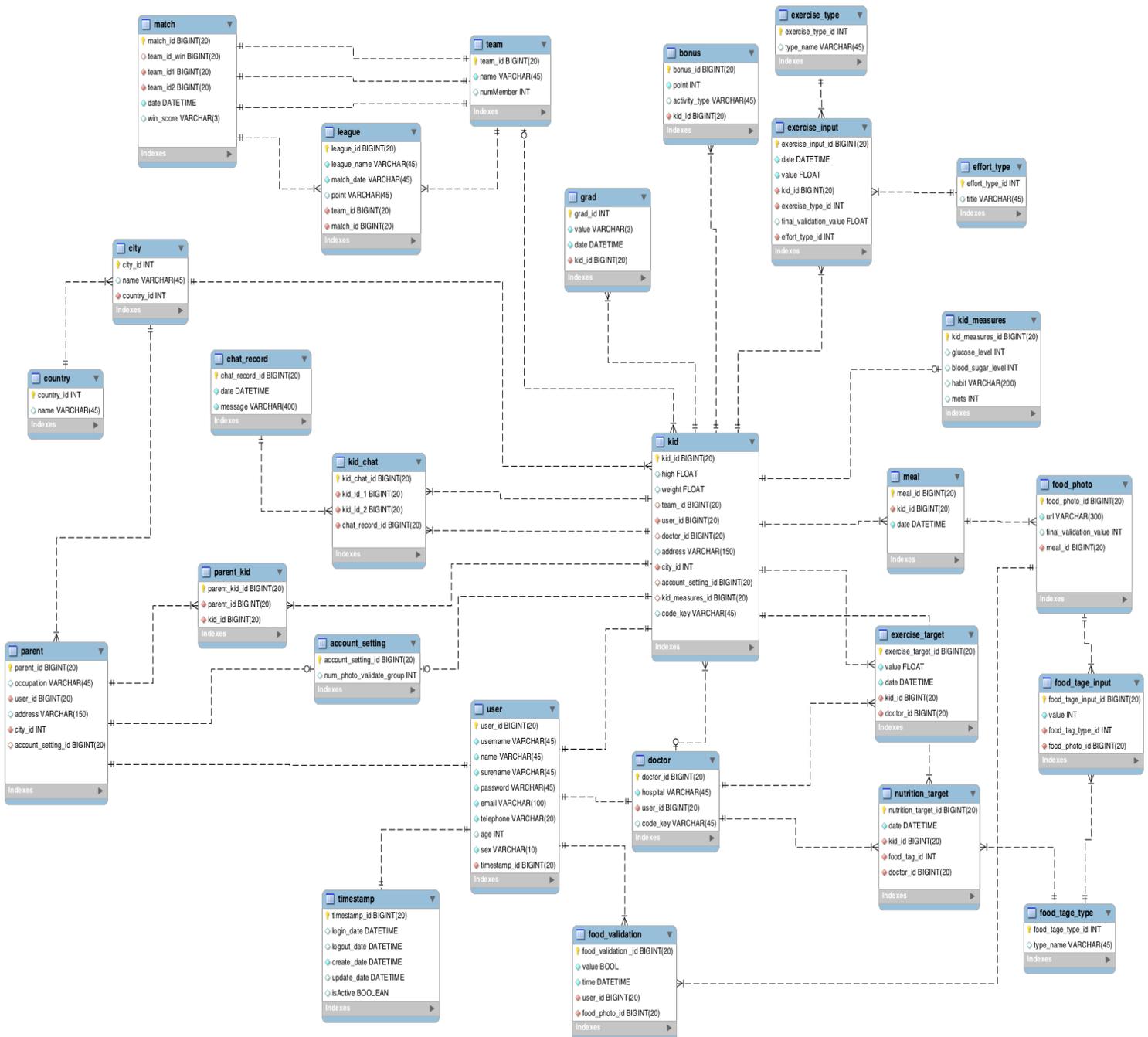


Figure 22: Entity relational diagram

## 7.5. Design GUI prototype

---

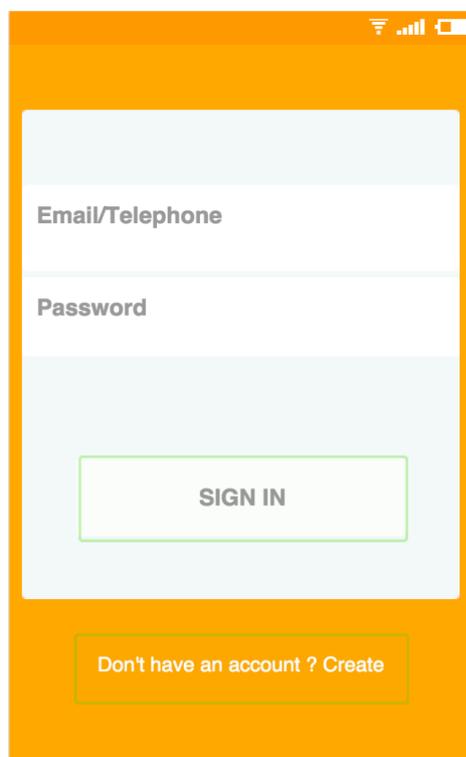
A project prototype has been designed to represent the global idea and requirement from users before starting to code anything. After finishing this prototype, it was presented to physician and to get some recommendation improvements and some changes. After that, the discussed changes were included. This section will show all the GUI prototypes both in mobile application (Kid and Parent) interface and web application (Doctor and Admin) interface.

### 7.5.1. Kid GUI

---

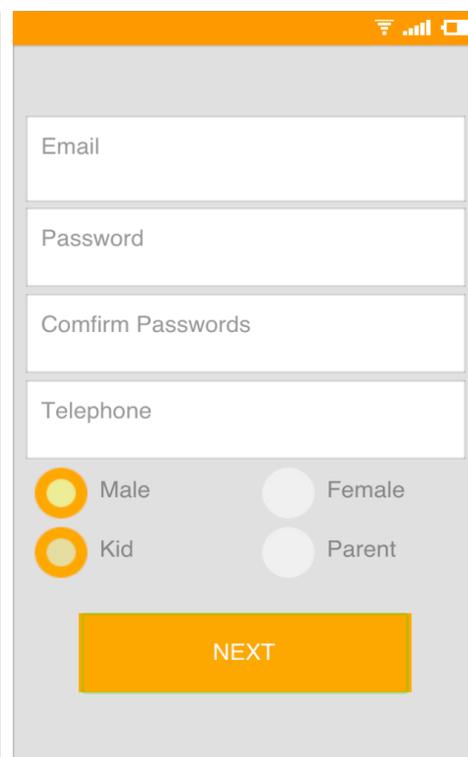
Figure 23 is the login screen. The kid can input username and password to login into their account. If they don't have it yet, they also can crate new account from this screen.

Figure 24 is the registration screen that allows kid to create new account by inputting some information.



The login screen for a kid features a light blue background with a white form area. At the top, there are icons for signal strength, Wi-Fi, and battery. The form contains two input fields: 'Email/Telephone' and 'Password'. Below the form is a green 'SIGN IN' button. At the bottom, there is a link that says 'Don't have an account ? Create'.

**Figure 23: Login kid**

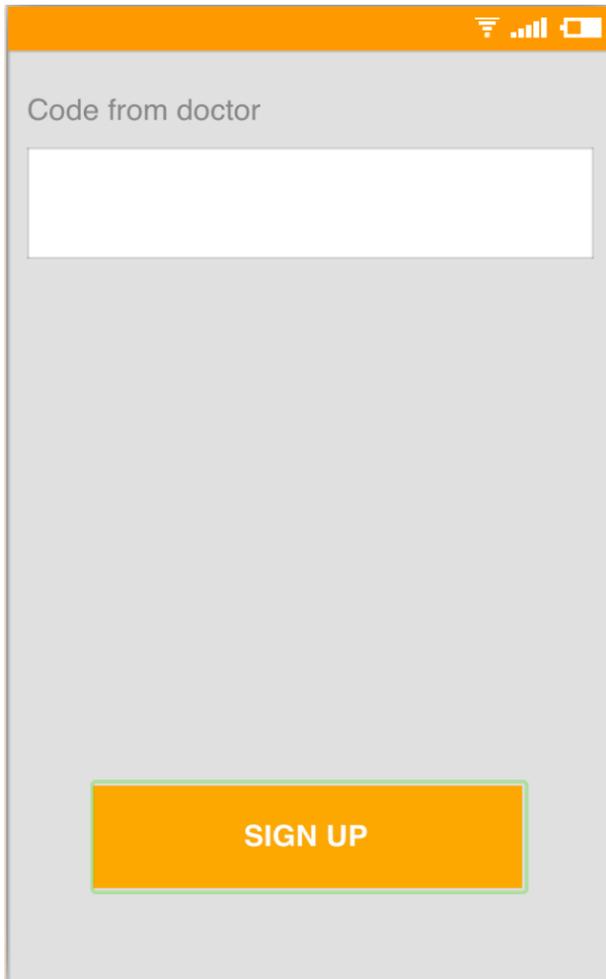


The registration screen for a kid features a light gray background with a white form area. At the top, there are icons for signal strength, Wi-Fi, and battery. The form contains four input fields: 'Email', 'Password', 'Comfirm Passwords', and 'Telephone'. Below the form are four radio button options: 'Male', 'Female', 'Kid', and 'Parent'. At the bottom, there is a yellow 'NEXT' button.

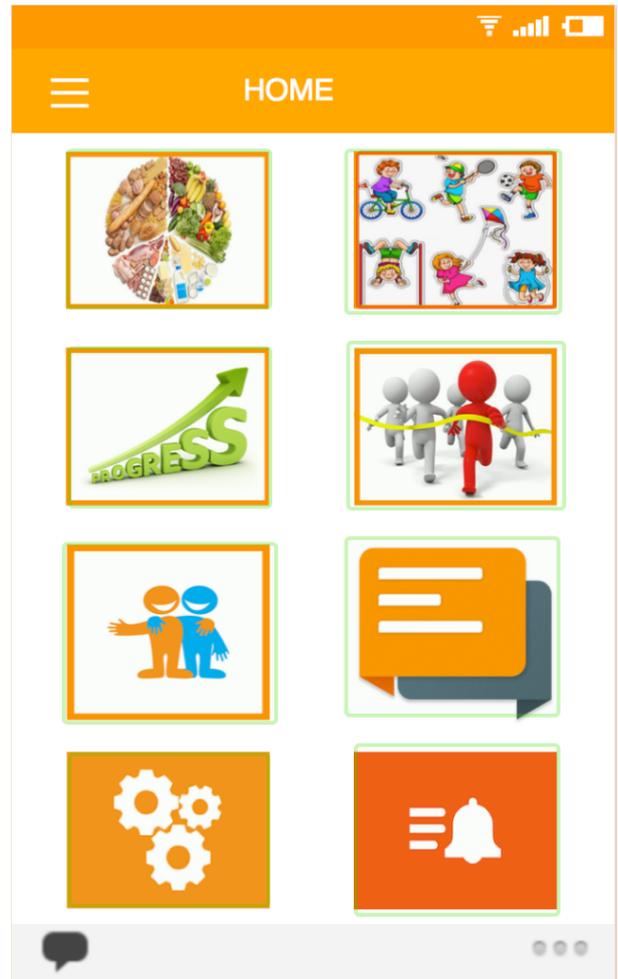
**Figure 24: Register kid**

Figure 25 is also a part of registration form for the user kid. The kid needs to input a code from his/her doctor in order to get connection with the doctor.

Figure 26 is the home screen for kid user. There are 8 menus like food, exercise, progress, competition, team, messenger, setting and notification.



**Figure 25: Confirm code doctor**

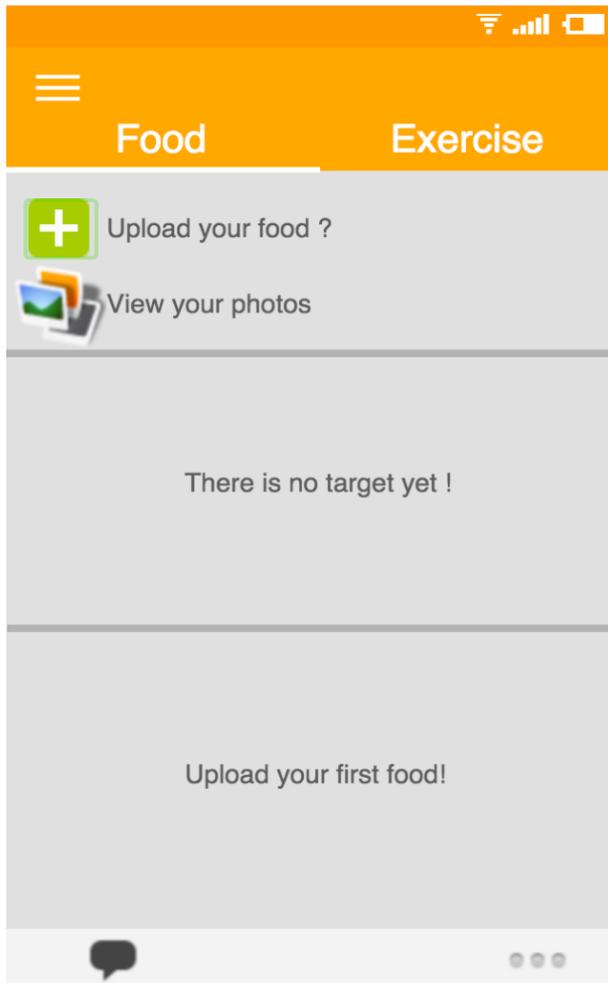


**Figure 26: Home page kid**

Figure 27 is the food dashboard without food records. It's navigated from Food Menu.

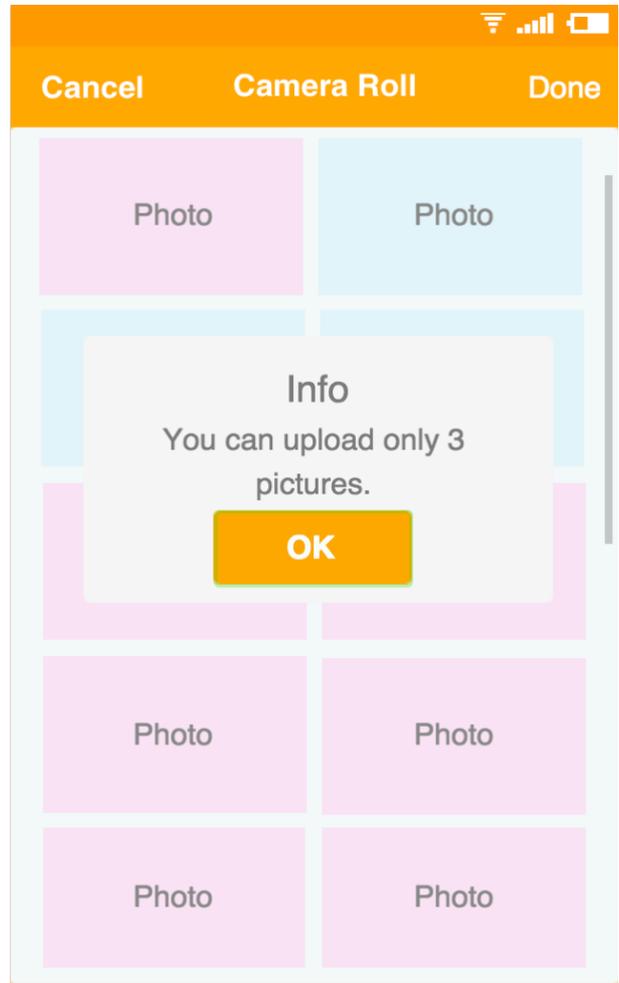
There are two main option in this page: upload new food and view all the food photos.

Figure 28 is the food photo screen. After the kid clicks on button "Upload food", there



will be this screen and he/she can select or take 3 photos maximum from their device.

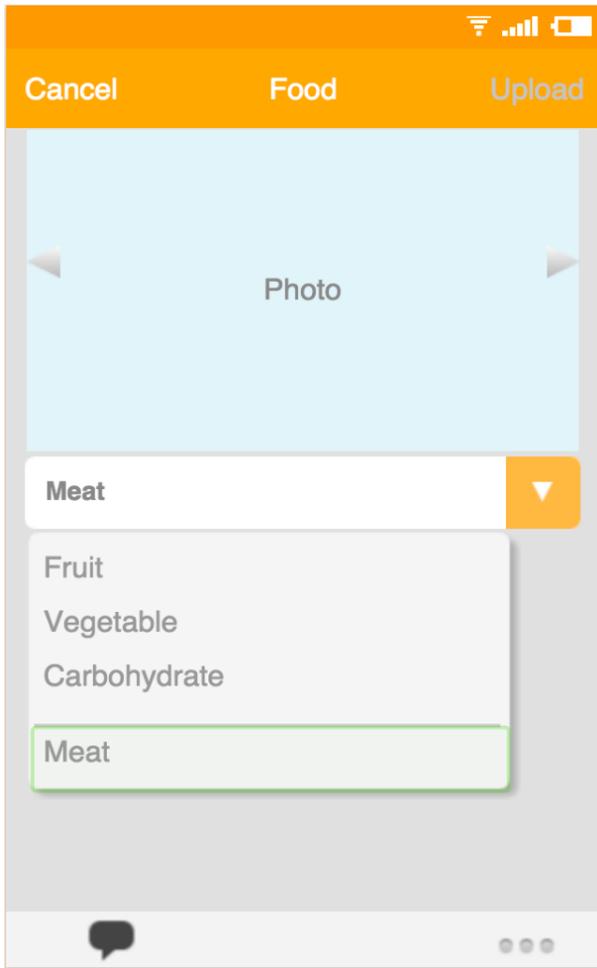
**Figure 27: Food empty dashboard**



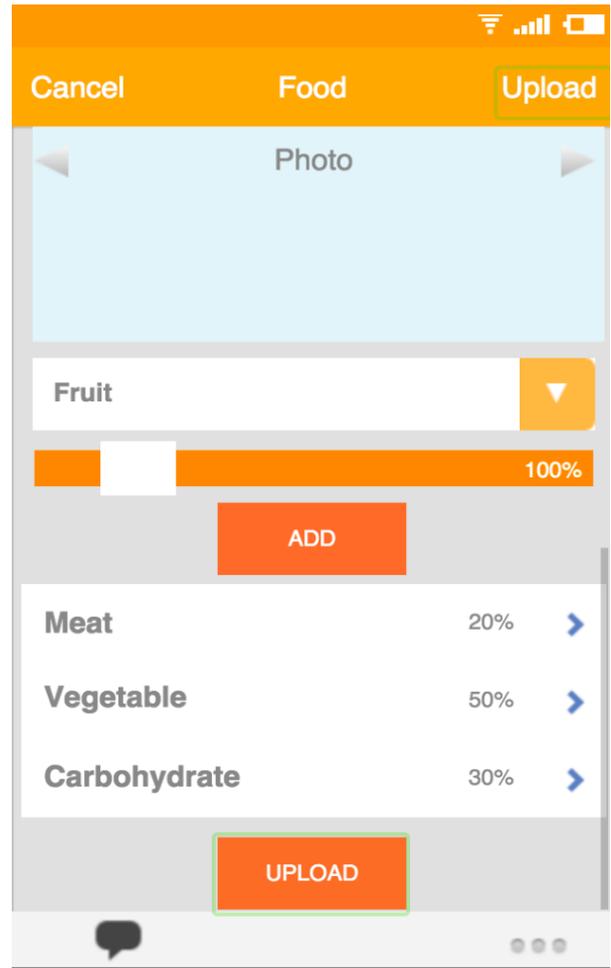
**Figure 28: Select photos**

Figure 29 is the input food information screen. This screen will be appeared after the kid selected the photos. He/she also can input the percentage of each food category of each photos.

Figure 30 is still in the food information screen but it's the complete information and the kid can click upload to send it to server side.



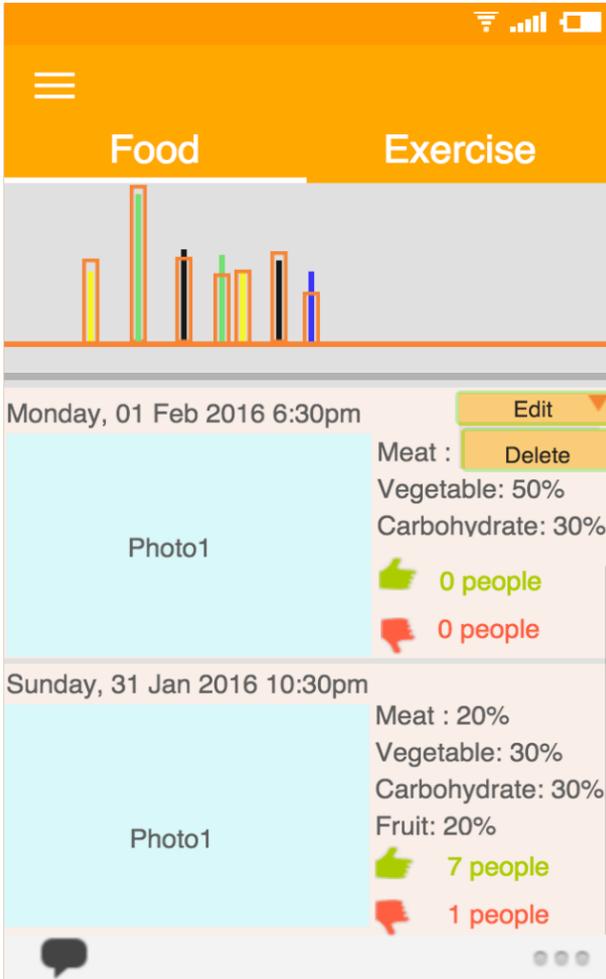
**Figure 29: Input food info**



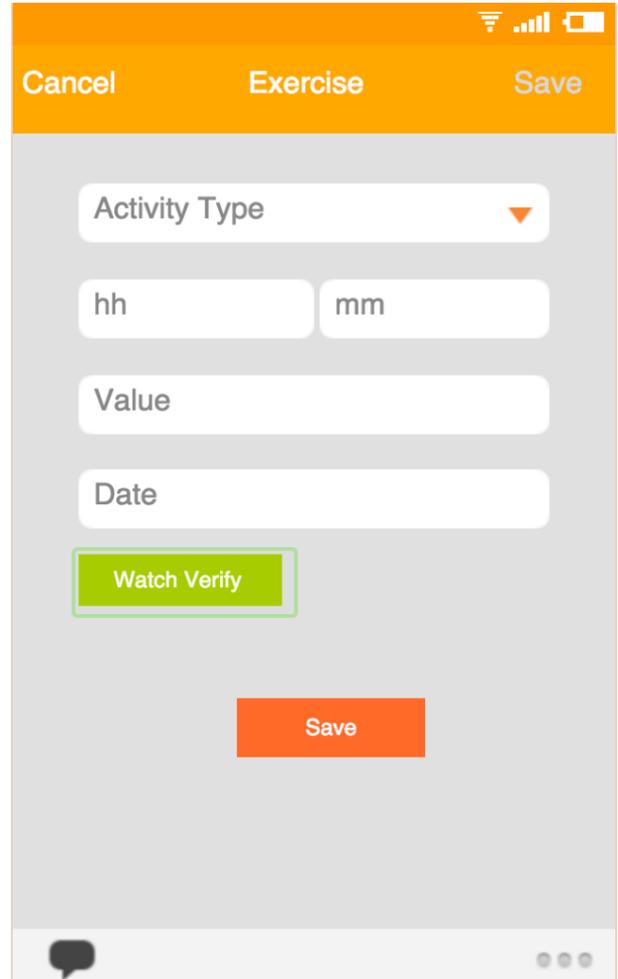
**Figure 30: Upload food**

Figure 31 is the food dashboard with some food records and a chart showing the status of the kid's input. The kids can also edit and delete their food record.

Figure 32 is the exercise input screen. The kids can input the information of their exercise activity that he/she just did it.

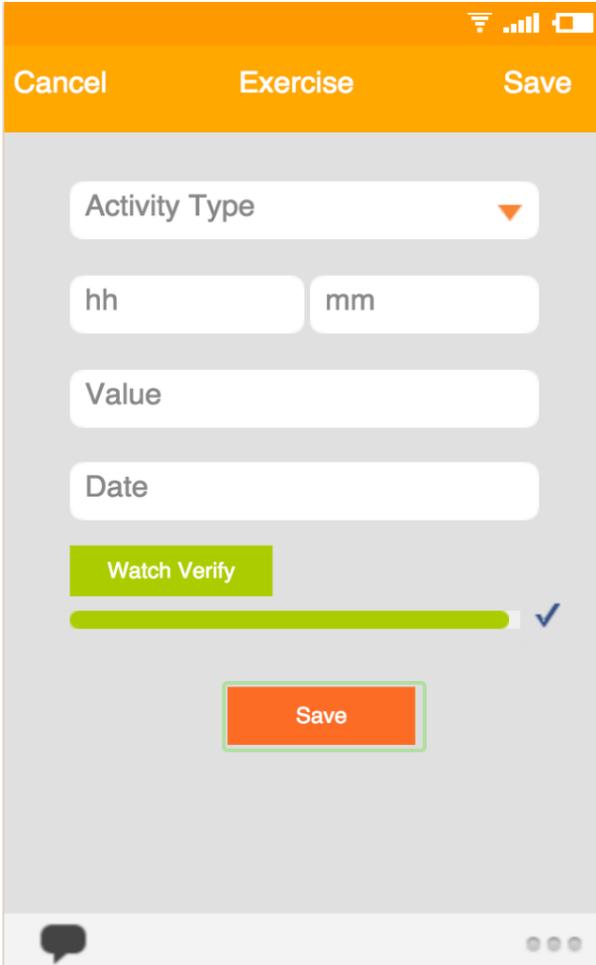


**Figure 31: Food dashboard**

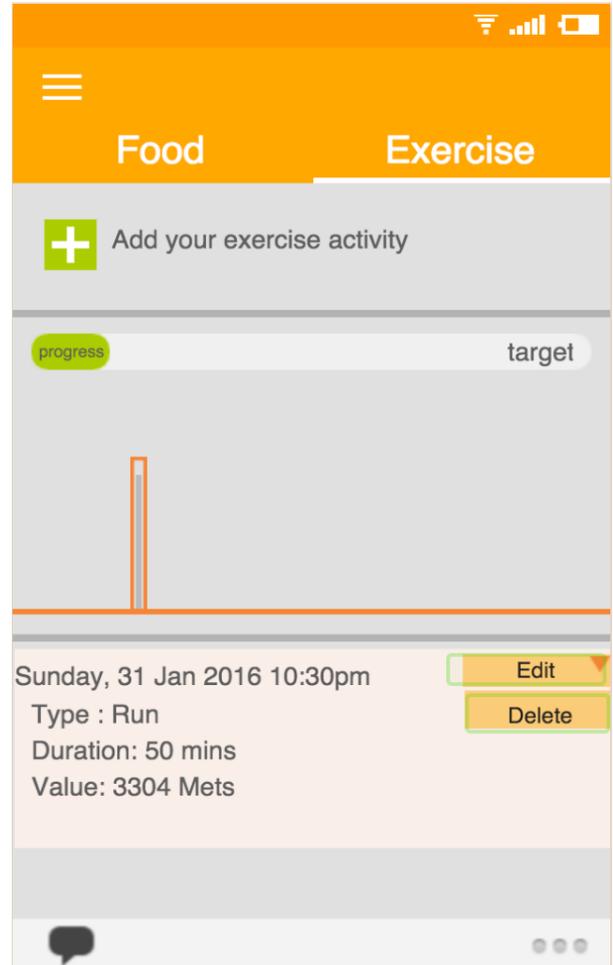


**Figure 32: Input exercise info**

Figure 33 is still in the exercise information input but it's verified with the smart watch that the kid is wearing. The application will start to synchronize the data from the smart watch when the kid click button "Watch Verify".



**Figure 33: Verify from watch**



**Figure 34: Exercise dashboard**

Figure 35 is the Competition screen. The Kid can get here from the Competition menu. He/She can see the information about the rank score of the teams are in the same competition and see the match calendar for the next match.

Figure 36 is the Group screen. The Kid can get here from My Group menu. He/She can see the grad and the status of food and exercise activity of all the members in his/her team. He/She also can click on any team member to send a message to them.

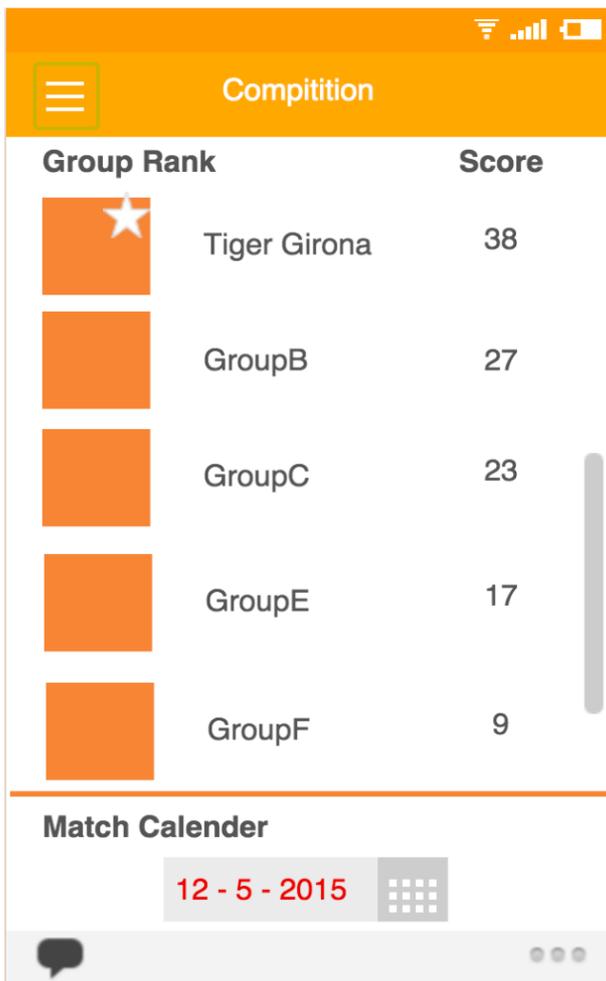


Figure 35: Competition

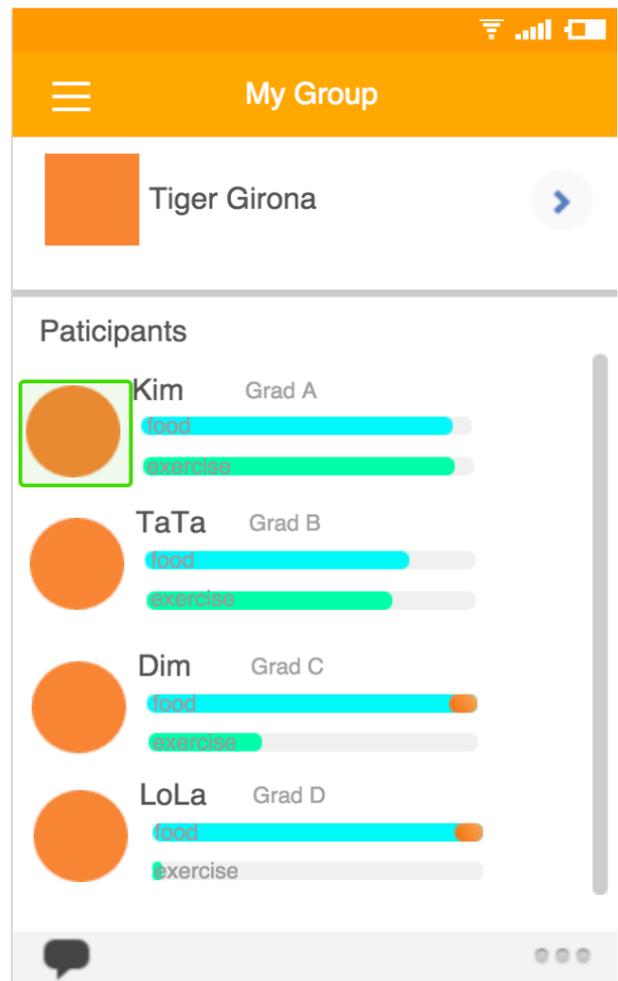
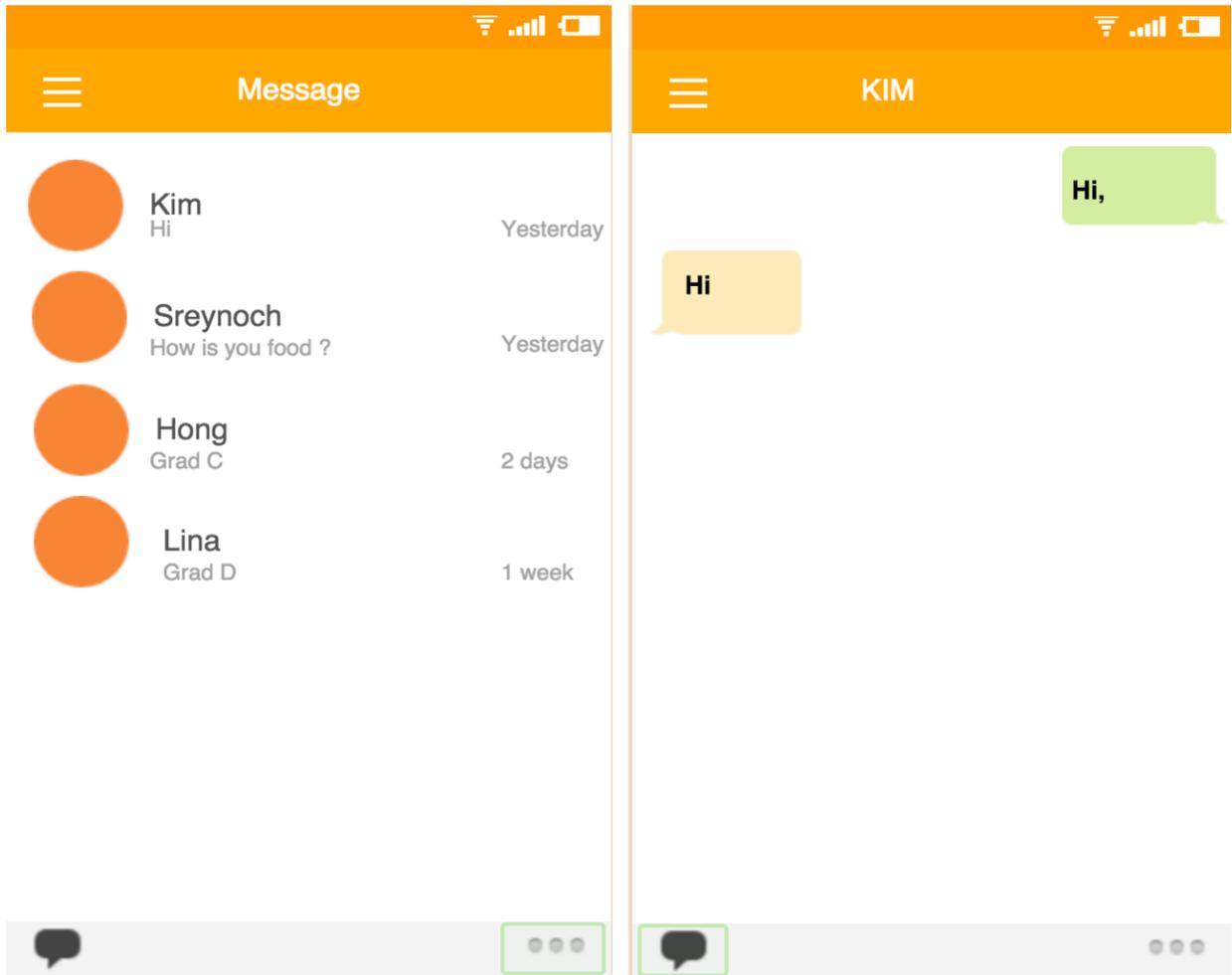


Figure 36: Group

Figure 37 is the message history screen. It shows all the histories of chat that the kid has been sent the message to their team member. The kid can get this screen from Messenger menu.

Figure 38 is the chat screen which shows all the conversation between the kid with another kid.

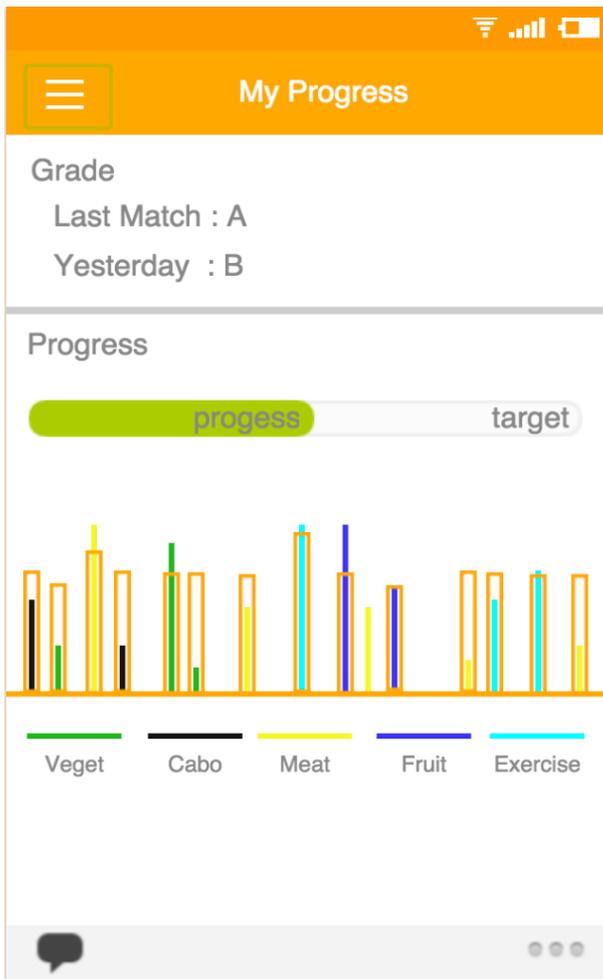


**Figure 37: Message history**

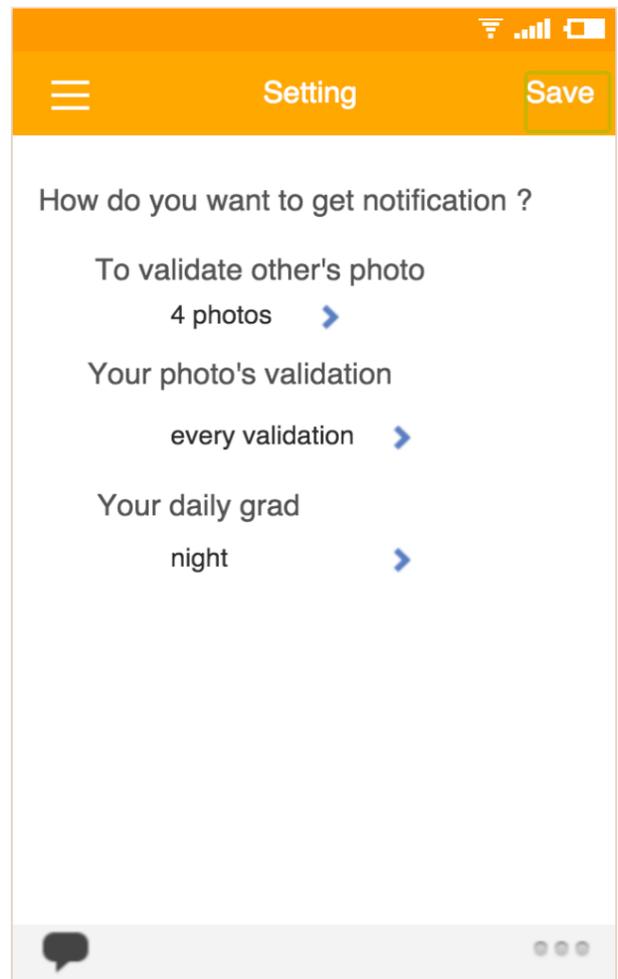
**Figure 38: Chat**

Figure 39 is My progress screen which is from My progress menu and it shows the kids grad and the progress status of their food and exercise.

Figure 40 is Setting screen which is from Setting menu and it allows the kid to set some information.



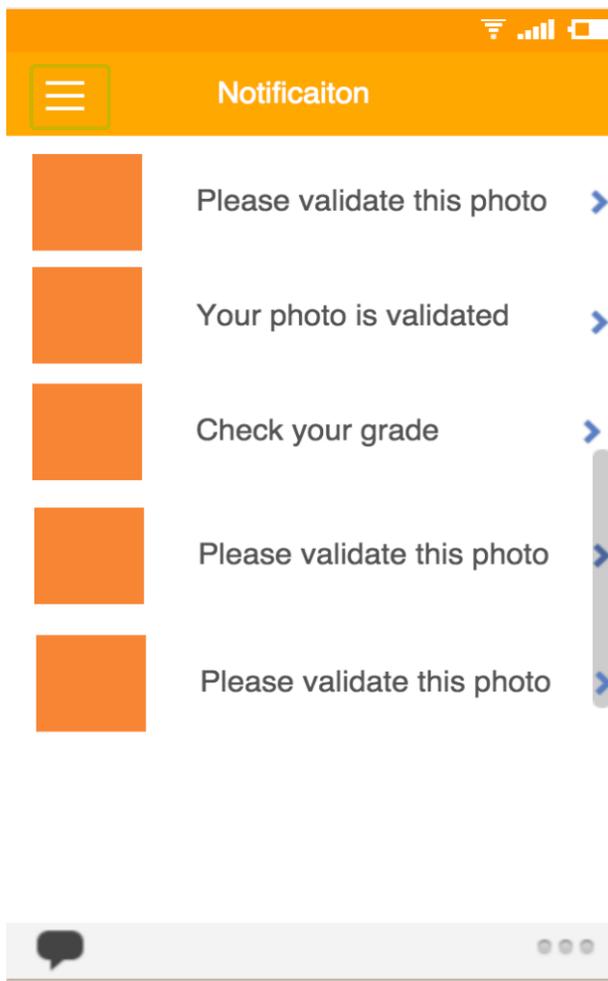
**Figure 39: My progress**



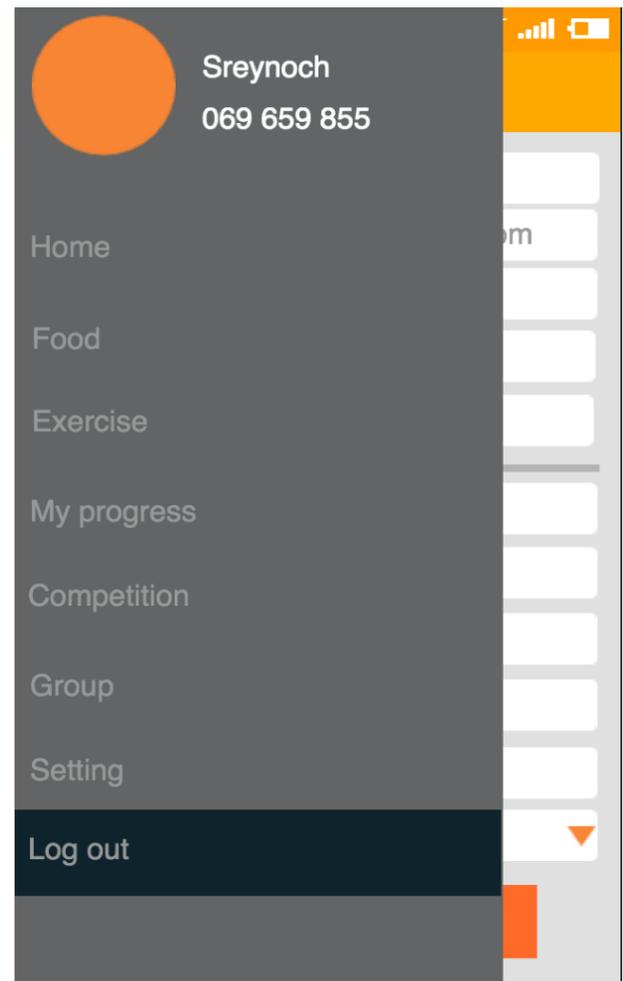
**Figure 40: Setting**

Figure 41 is Notification screen. All the notifications from the server are presented here. The kid can click on any notification then it will be navigated to another page related to that notification. He/She can get here from Notification menu and from the option menu on the corner of the screen.

Figure 42 is the another slide menu from left side when there's a gesture from the the left screen.

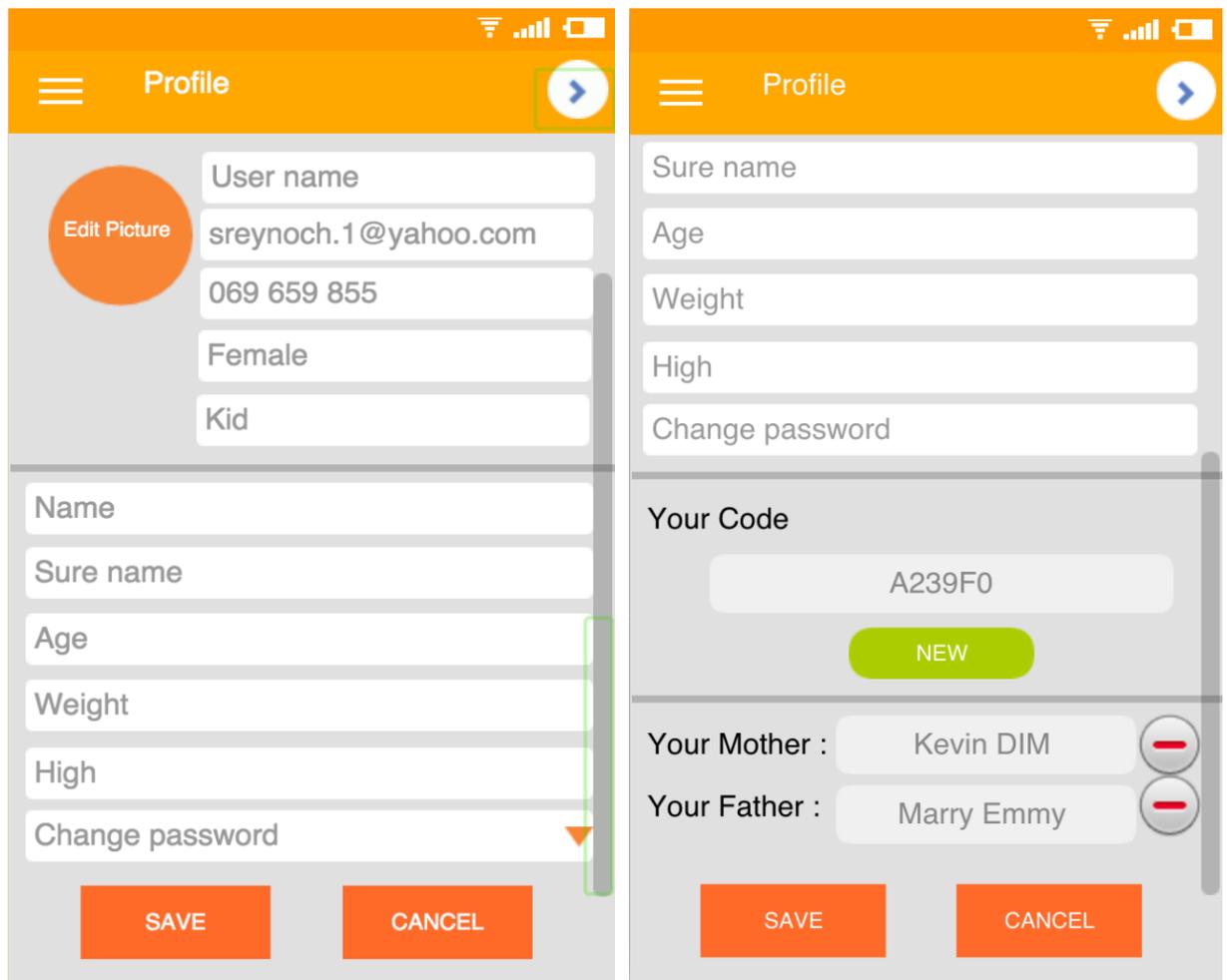


**Figure 41: Notification**



**Figure 42: Menu**

Figure 43 is the profile kid screen where is the kid can update their information.



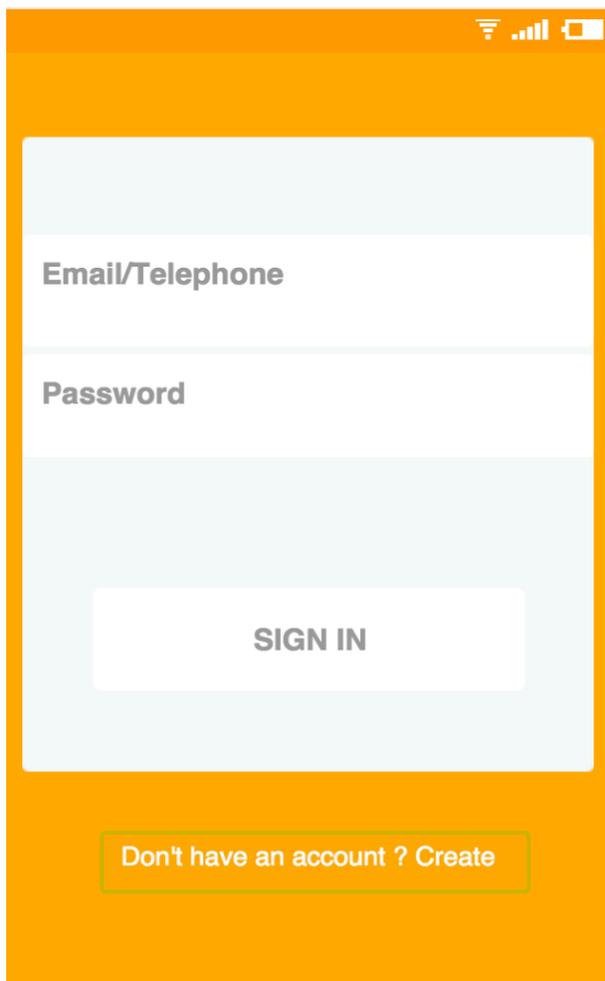
**Figure 43: Profile kid**

## 7.5.2. Parent GUI

---

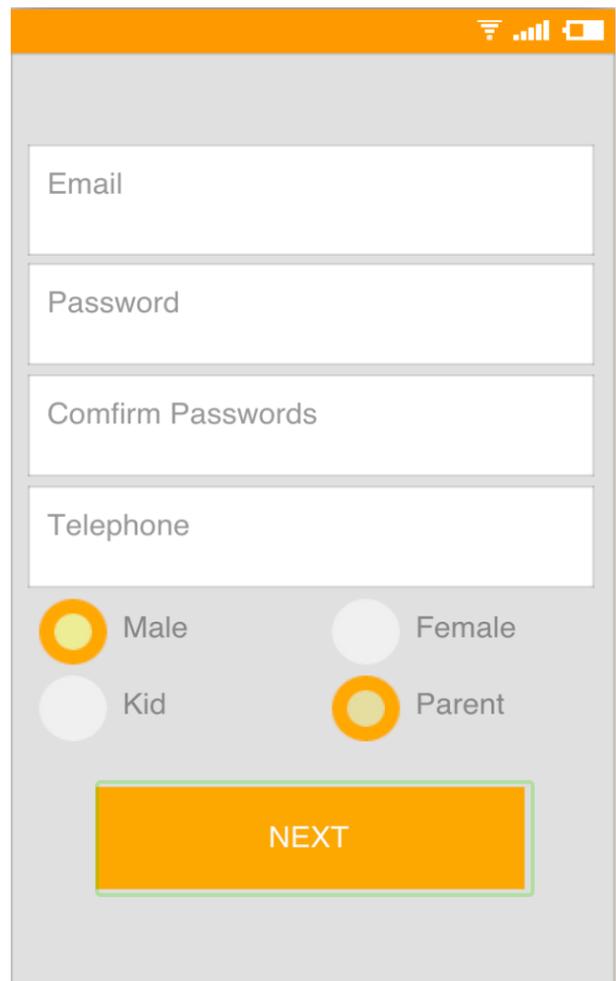
Figure 44 is the login screen. The parent can input username and password to login into their account. If they don't have it yet, they can create new account from this screen.

Figure 45 is the registration screen which allows parent to input their information and chose the option parent in order to create new user account.



The login screen features a light blue background with a white sign-in box. At the top, there are two input fields labeled "Email/Telephone" and "Password". Below these fields is a white button with the text "SIGN IN". At the bottom of the screen, there is a link that says "Don't have an account ? Create". The status bar at the top shows signal strength, Wi-Fi, and battery icons.

**Figure 44: Login parent**

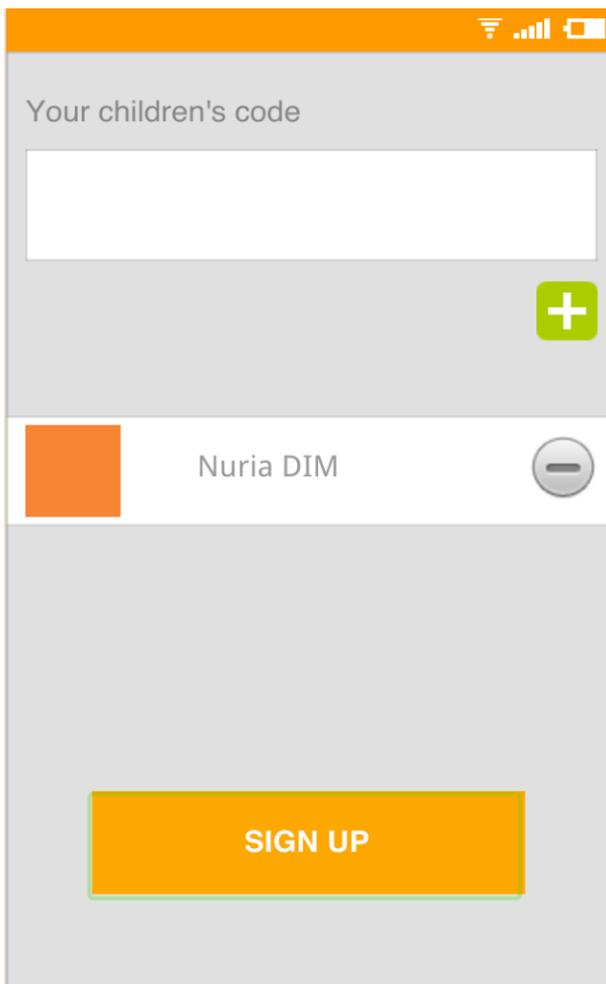


The registration screen has a light gray background. It contains four input fields: "Email", "Password", "Comfirm Passwords", and "Telephone". Below the fields are four radio button options: "Male" (selected), "Female", "Kid", and "Parent" (selected). At the bottom, there is a large orange button with the text "NEXT". The status bar at the top shows signal strength, Wi-Fi, and battery icons.

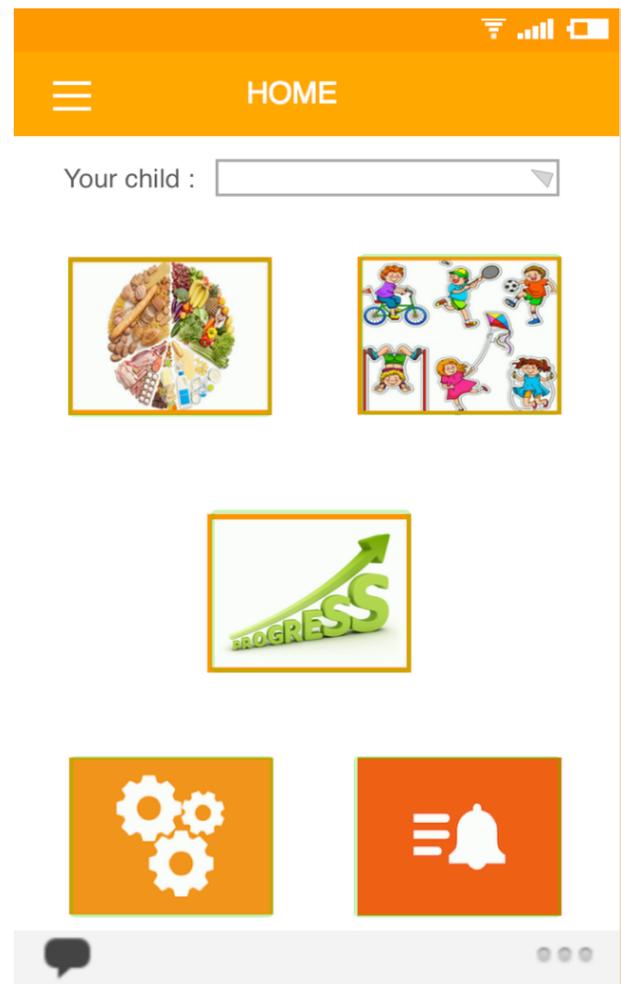
**Figure 45: Register parent**

Figure 46 is also in the registration account screen but it requires the parents to input their children's code. He/She also can add more than one child by clicking on button "add new".

Figure 47 is the home screen for the parent user. There is the selection box for selecting the children in order to see the information of that kid. Moreover, there are 5 menus like food, exercise, progress, setting and notification.



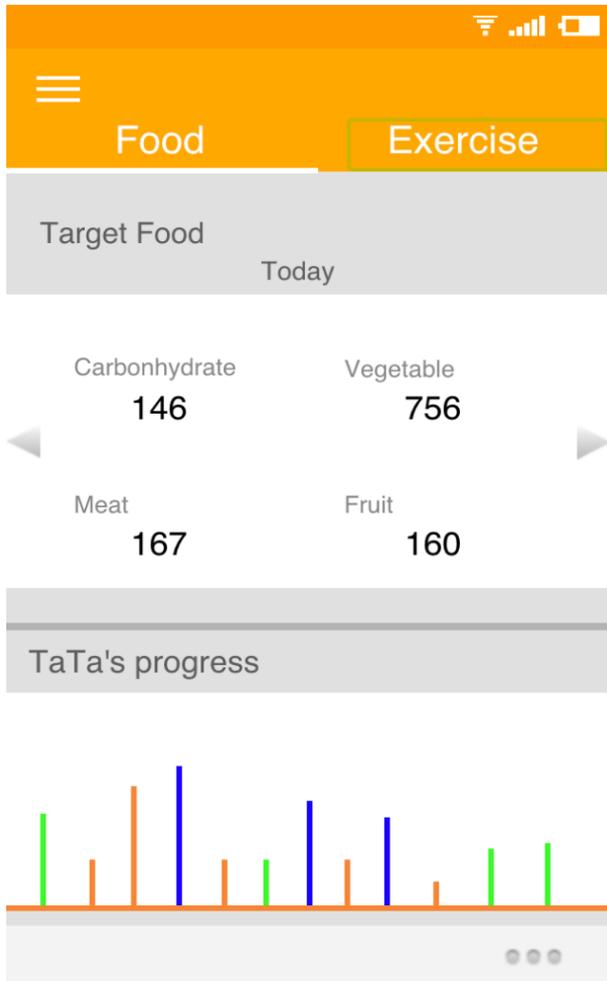
**Figure 46: Kid code confirm**



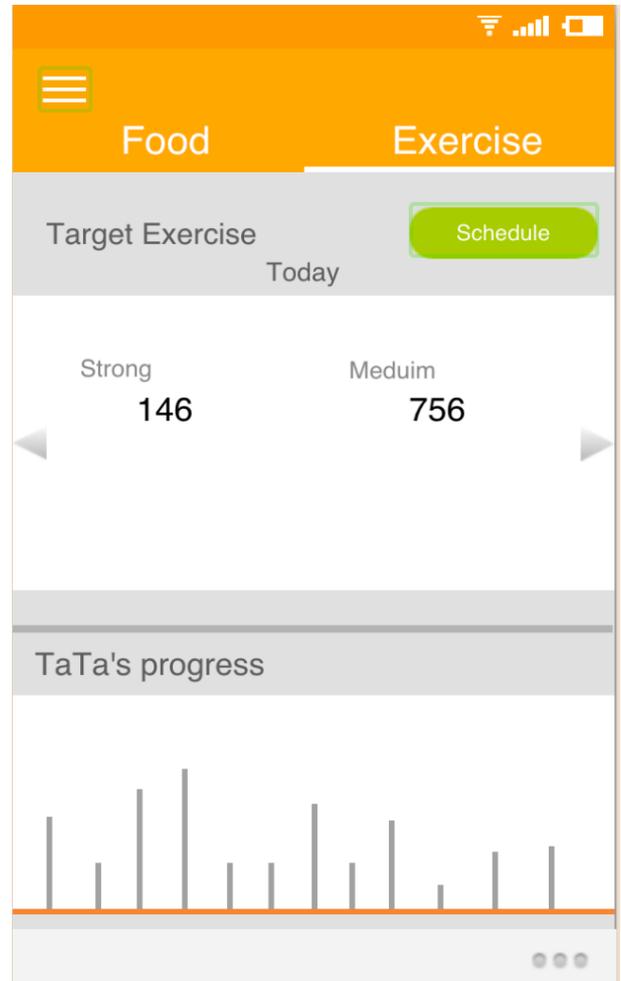
**Figure 47: Parent Home page**

Figure 48 is the food information screen which shows the information of the food that there kid uploaded.

Figure 49 is the exercise information screen which shows the information of the exercise activity of their kid.



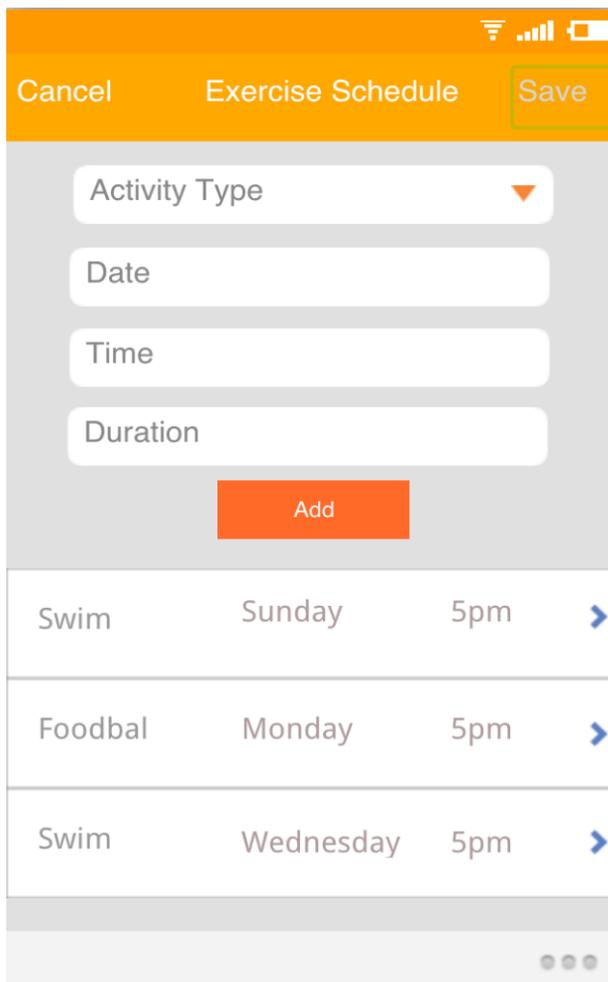
**Figure 48: Food info**



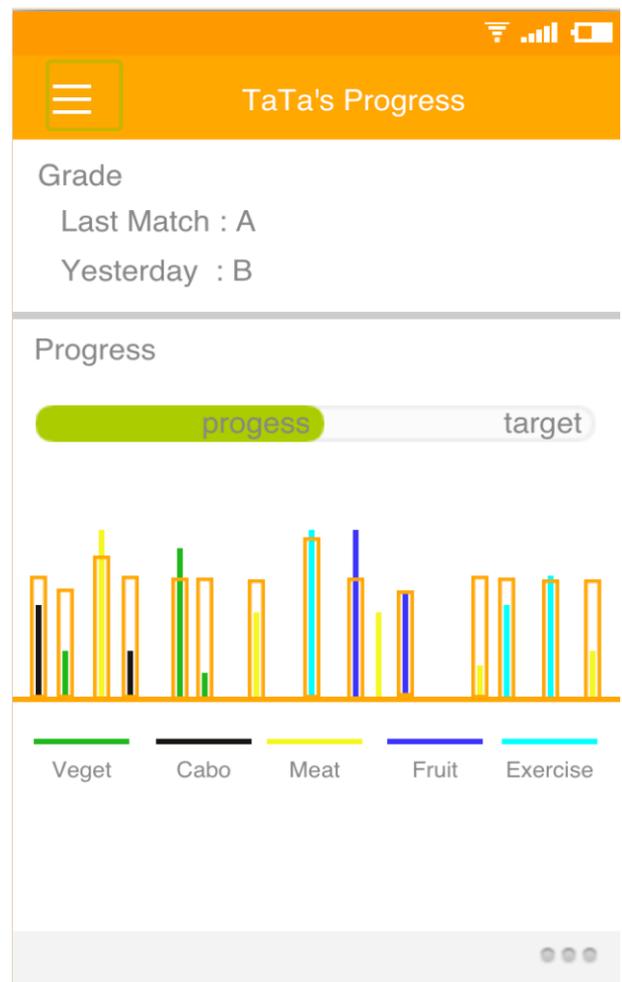
**Figure 49: Exercise info**

Figure 50 is the exercise schedule screen that is navigated from the exercise screen. It is where the parent can set the exercise schedule of their kid.

Figure 51 is the kid's progress screen. The parent can view their kid grade, progress bar and the graph of the input information from the kid compare to the target from the physician.



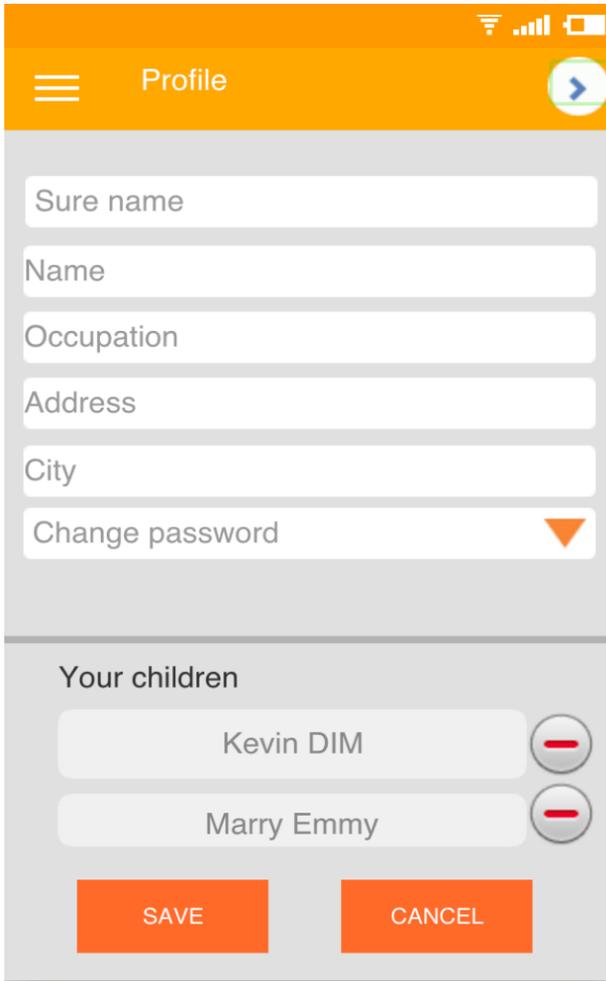
**Figure 50: Exercise schedule**



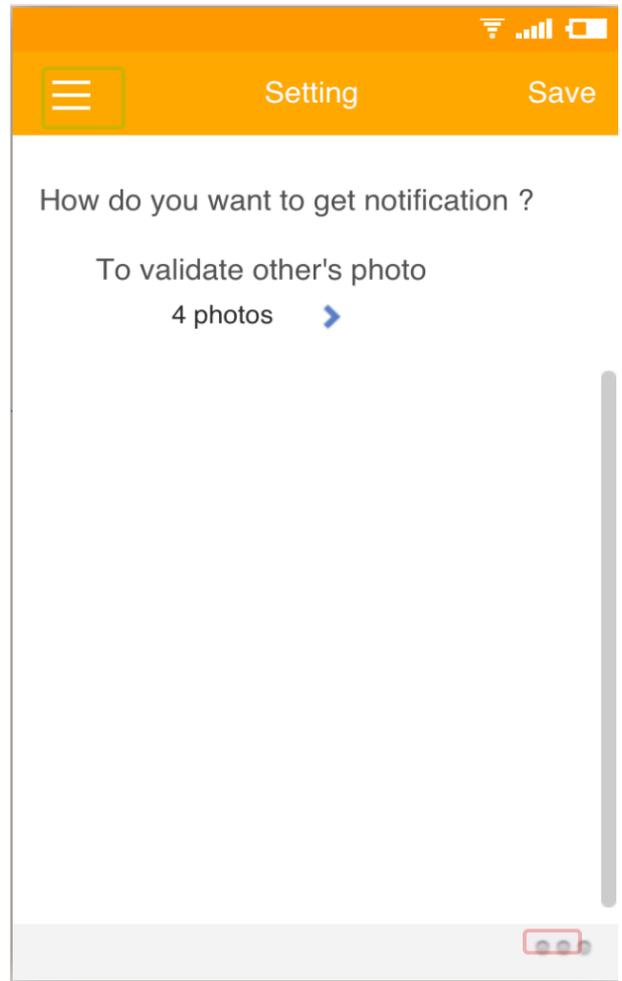
**Figure 51: Kid's progress**

Figure 52 is the parent profile screen where is the parent can update their information.

Figure 53 is the setting screen where the parent can update some information related to that notification.



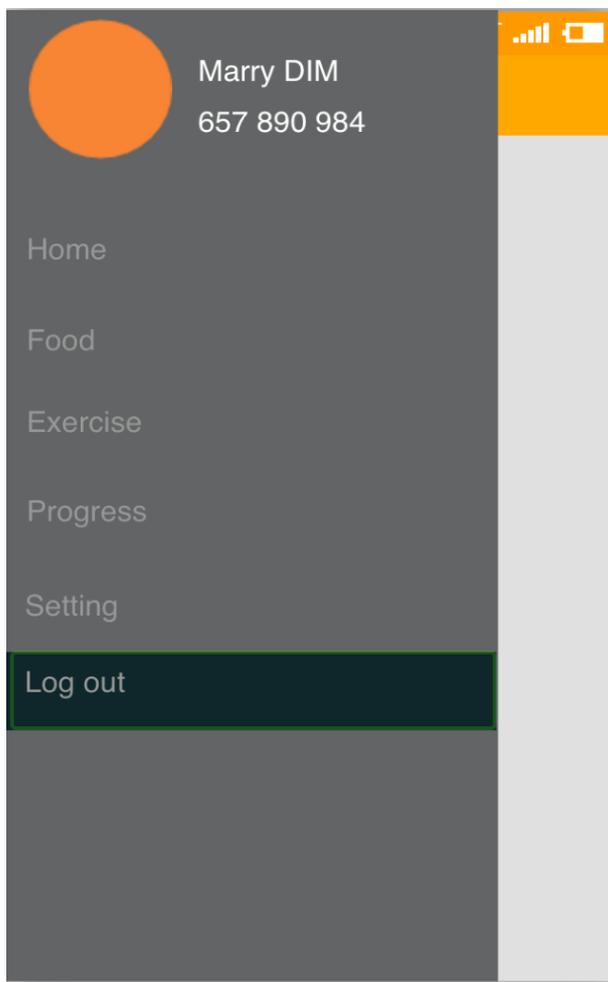
**Figure 52: Profile**



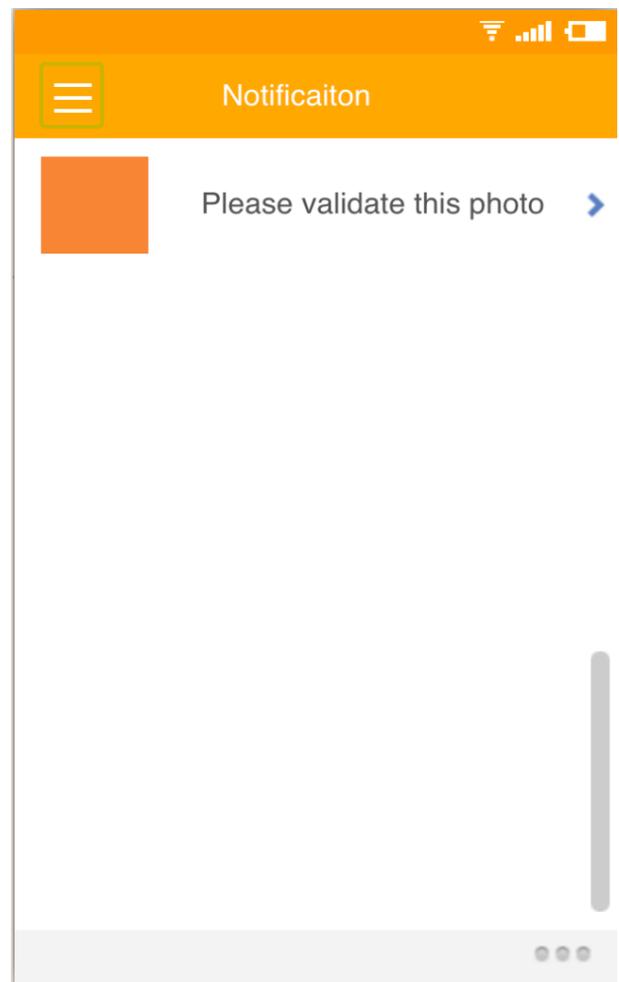
**Figure 53: Setting**

Figure 54 is the another slide menu from left side when there's a gesture from the the left scree.

Figure 55 is Notification screen. All the notifications from server are presented here. The parent can click on any notification then it will be navigated to another page related to that notification. He/She can get here from Notification menu and from the option menu on the corner of the screen.



**Figure 54: Menu**

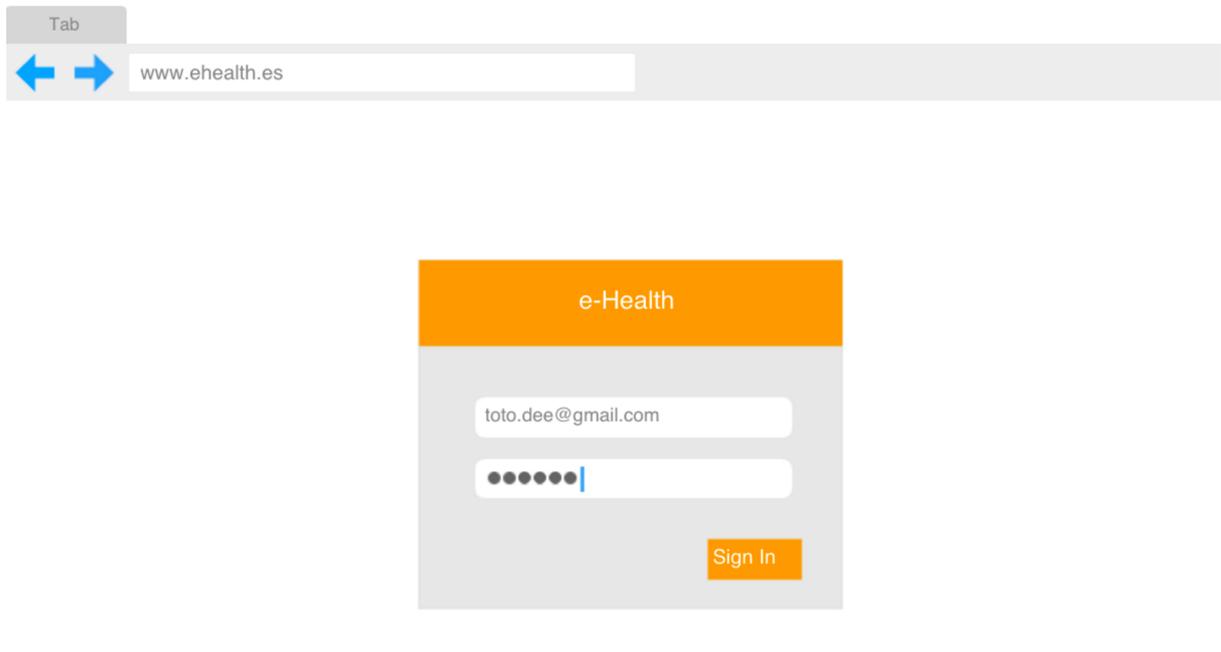


**Figure 55: Notification**

### 7.5.3. Doctor GUI

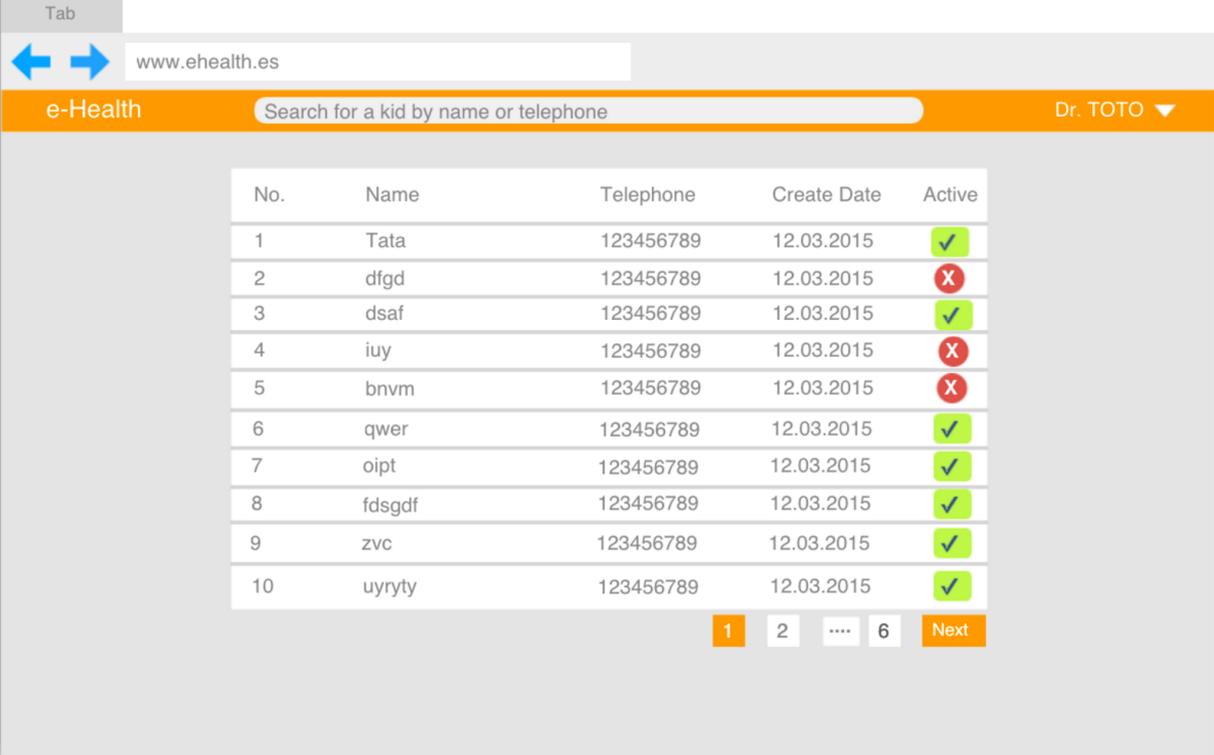
---

Figure 56 is the login page for user doctor. He/she can input username and password to access in to his/her account.



**Figure 56: Login doctor**

Figure 57 is the home page of doctor where he/she can see the list of their patients (kid), search for the patient on the search bar on the top of the page and go to their profile by click on his/her name on the top right of the page.

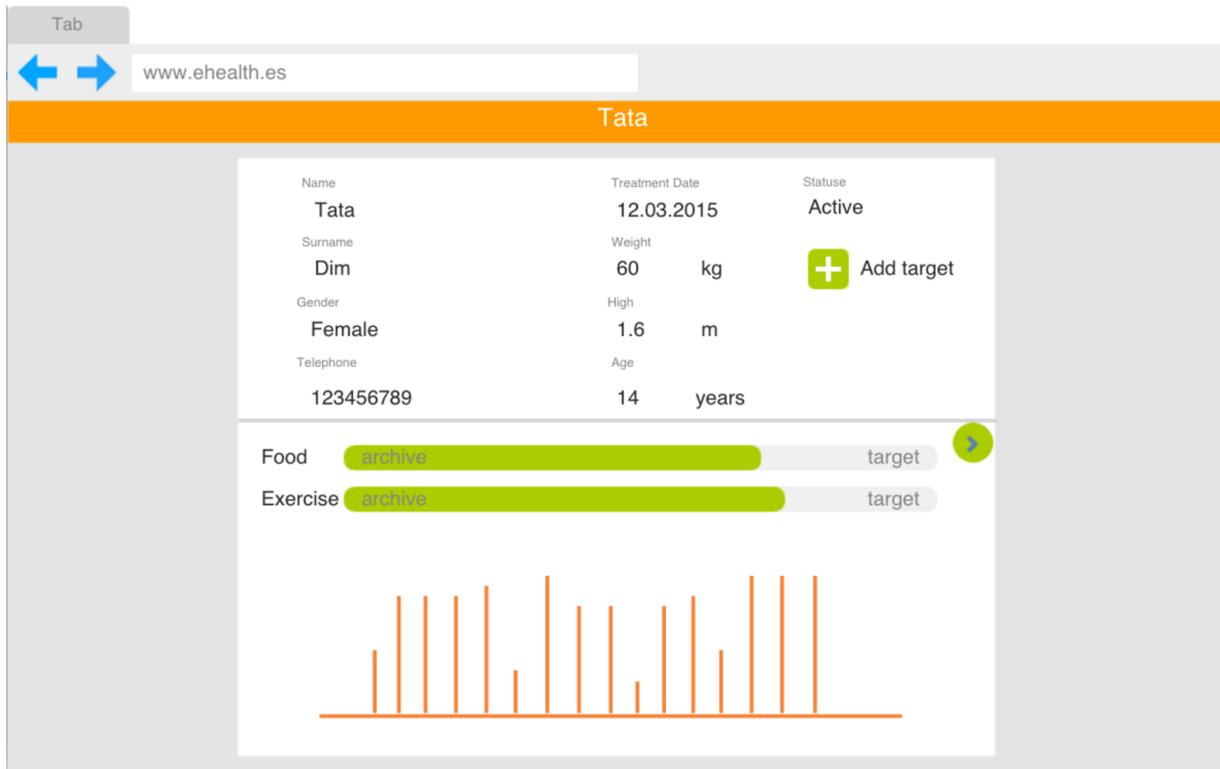


The screenshot shows a web browser window with the address bar displaying 'www.ehealth.es'. The page header includes 'e-Health' on the left, a search bar with the text 'Search for a kid by name or telephone' in the center, and 'Dr. TOTO' with a dropdown arrow on the right. Below the header is a table with 10 rows of patient data. The 'Active' column contains green checkmarks for rows 1, 3, 6, 7, 8, 9, and 10, and red 'X' marks for rows 2, 4, and 5. At the bottom right of the table, there are pagination controls showing '1' (highlighted), '2', '...', '6', and 'Next'.

No.	Name	Telephone	Create Date	Active
1	Tata	123456789	12.03.2015	✓
2	dfgd	123456789	12.03.2015	✗
3	dsaf	123456789	12.03.2015	✓
4	iuy	123456789	12.03.2015	✗
5	bnvm	123456789	12.03.2015	✗
6	qwer	123456789	12.03.2015	✓
7	oipt	123456789	12.03.2015	✓
8	fdsgdf	123456789	12.03.2015	✓
9	zvc	123456789	12.03.2015	✓
10	uyryty	123456789	12.03.2015	✓

**Figure 57: List kids**

Figure 58 is the kid detail page that shows the detail information of the kid. It's navigated from the home page when the doctor click in any kids's name in the list. In this page, doctor can add new target to the kid by clicking on Add target button. Moreover he/she also can see the status bar and progress graph of the kids nutrition and exercise activity.



**Figure 58: Kid detail**

Figure 59 is the exercise target list page that shows all the list of exercise targets that contains the value of each category and the duration. The doctor also can edit and delete the target by clicking on the detail button on each row of the list.

	Exercise	Food
	Effort type : Medium	130 mets
	From: 12 - 5 - 2014	To: 15 - 5 - 2014
	Effort type : Strong	200 mets
	From: 16 - 5 - 2014	To: 20 - 5 - 2014
✓	Effort type : Strong	200 mets
	From: 16 - 5 - 2014	To: 20 - 5 - 2014
✓	Effort type : Strong	200 mets
	From: 16 - 5 - 2014	To: 20 - 5 - 2014
✓	Effort type : Strong	200 mets
	Effort type : Medium	140 mets
	From: 16 - 5 - 2014	To: 20 - 5 - 2014

**Figure 59: Exercise target**

Figure 60 is the food target list page that shows all the list of food targets that contains the value of each category and the duration. The doctor also can edit and delete the target by clicking on the detail button on each row of the list.

Tata	
Exercise	Food
Vegetable : 200 Fruit: 100 From: 16 - 5 - 2014	Carbohydrate : 210 Meat: 165 To: 20 - 5 - 2014
Vegetable Fruit: 100 From: 16 - 5 - 2014	Carbohydrate : 210 Meat: 165 To: 20 - 5 - 2014
<input checked="" type="checkbox"/> Vegetable Fruit: 100 From: 16 - 5 - 2014	Carbohydrate : 210 Meat: 165 To: 20 - 5 - 2014
<input checked="" type="checkbox"/> Vegetable Fruit: 100 From: 16 - 5 - 2014	Carbohydrate : 210 Meat: 165 To: 20 - 5 - 2014

**Figure 60: Food target**

Figure 61 is the new target page that allows the doctor to set a new target for kid. He/She can set both exercise and food target in this page by inputting the information and chose the duration then he/she can click save to save in to database and send to the kid.

The screenshot shows a web browser window with the address bar displaying 'www.ehealth.es'. The page title is 'Tata'. The main content area is divided into two sections: 'Exercise' and 'Food'.  
**Exercise Section:**  
- Row 1: Effort type: medium, 130 mets  
- Row 2: Effort type: strong, 200 mets  
- Row 3: Effort type: normal, 200 mets  
- Date range: From 12-5-2014 To 12-5-2014  
**Food Section:**  
- Row 1: Vegetable: 130, Carbohydrate: 130  
- Row 2: Fruit: 130, Meat: 130  
- Date range: From 12-5-2014 To 12-5-2014  
A red 'Save' button is positioned at the bottom right of the form.

**Figure 61: New target**

Figure 62 is the doctor profile page that allows doctor to update his information and to get his/her code key.

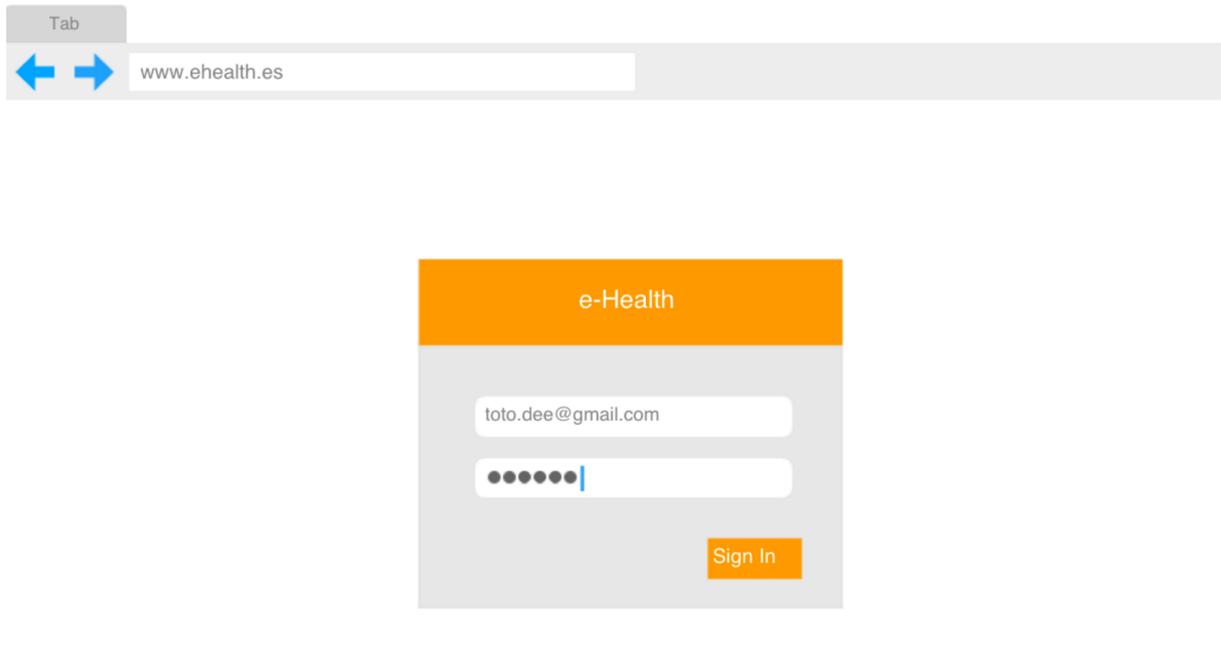
The screenshot shows a web browser window with the address bar displaying 'www.ehealth.es'. The page header is orange and contains the text 'e-Health' and 'Hello Dr. TOTO!'. The main content area is a white form titled 'Clinicia Girona'. The form has a placeholder for a profile picture (an orange square). Below the picture are five input fields: 'Username', 'Surname', 'TOTO', 'email', and 'Telephone'. To the right of these fields are two 'New passwords' fields, each containing six asterisks. Below the passwords is a 'Your Code' field containing 'N20T58'. There is a green 'New Code' button and an orange 'UPDATE' button at the bottom right of the form.

**Figure 62: Doctor profile**

### 7.5.3. Admin GUI

---

Figure 63 is the login page for user admin. He/she can input username and password to access in to his account.



**Figure 63: Log in admin**

Figure 64 is the home page of admin where he/she can see the list of doctors and other admins. He/she can also search for the patient on the search bar on the top of the page, create new doctor or admin account by clicking on the new button. Moreover, he/she can go to their profile by clicking on his/her name on the top right of the page.

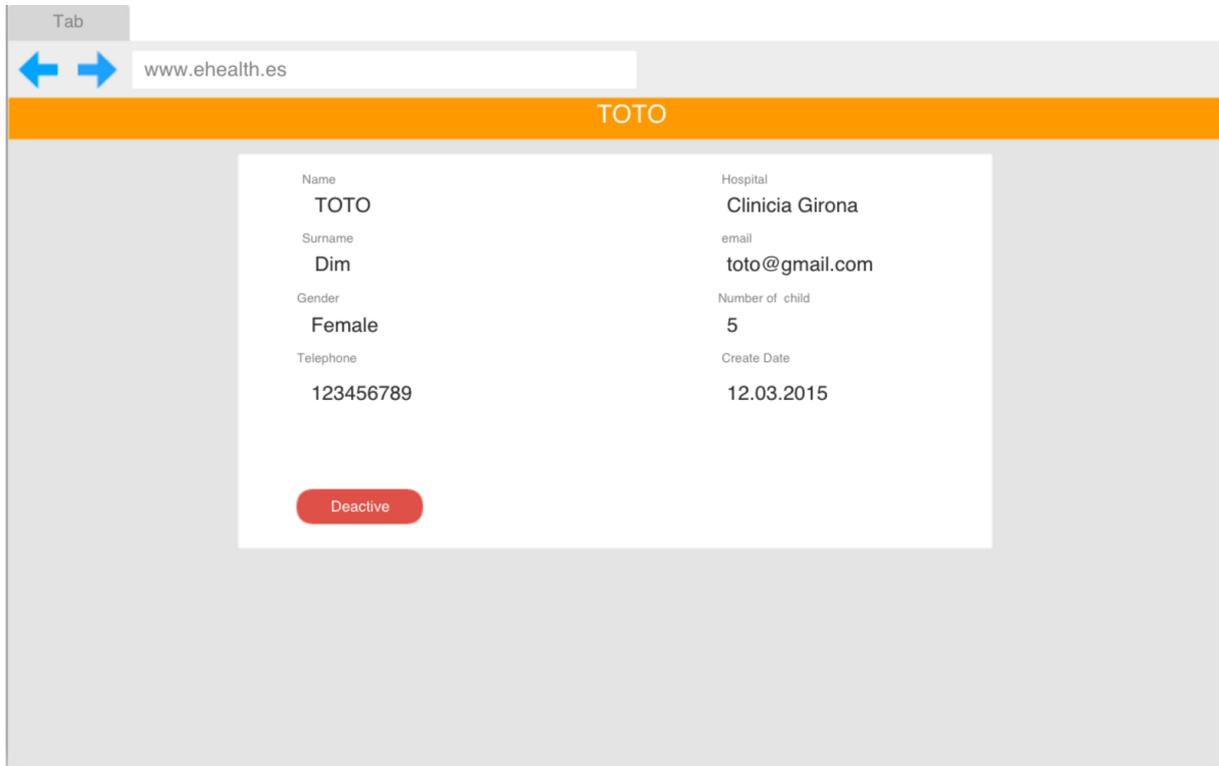
The screenshot shows a web browser window with the URL www.ehealth.es. The page header includes the text 'e-Health' and a search bar labeled 'Search for a doctor by name or telephone'. On the right side of the header, there is a dropdown menu labeled 'Admin'. Below the header, there is a section for 'Add new user' with a green plus icon and a blue button labeled 'Doctor'. The main content area features a table with the following data:

No.	Name	Telephone	Hospital
1	TOTO	123456789	Clinicia Girona
2	dfgd	123456789	darter
3	dsaf	123456789	Clinicia Girona
4	iuy	123456789	utyutyu
5	bnvm	123456789	Clinicia Girona
6	qwer	123456789	Clinicia Girona
7	oipt	123456789	iyuir
8	fdsgdf	123456789	Clinicia Girona
9	zvc	123456789	ioiuo
10	uyryty	123456789	Clinicia Girona

At the bottom right of the table, there is a pagination control with buttons for '1', '2', '...', '6', and 'Next'.

**Figure 64: List doctor and admin**

Figure 65 is the doctor detail page that shows the detail information of the doctor. It's navigated from the home page when the admin click in any doctor's name in the list. In this page, admin can deactivate or activate this doctor account.



**Figure 65: Doctor detail**

Figure 66 is the admin profile page that allows the admin to update his/her information.

The screenshot shows a web browser window with the address bar displaying "www.ehealth.es". The page has an orange header with "e-Health" on the left and "Admin" on the right. The main content area is a white form with a grey border. At the top left of the form is an orange square representing a profile picture. Below it are five input fields: "Admin", "Nuria", "Lin", "lin@gmial.com", and "Telephone". To the right of these fields is a section titled "New passwords" with two input fields, each containing six asterisks. At the bottom right of the form is an orange button labeled "UPDATE".

**Figure 66: Admin profile**

Figure 67 is the creating new user account page. Admin has right to create account for doctor and another admin so he/she can input the information and choose the account type then click create button.

The screenshot shows a web browser window with the address bar displaying 'www.ehealth.es'. The page title is 'New User' and the header is 'e-Health'. The form is centered and contains the following fields and options:

- Username
- Surname
- Hospital name
- Telephone
- email
- temp passwords (two fields, each with six asterisks)
- User Type:  Doctor,  Admin
- Create button

**Figure 67: New user doctor or admin**

## 8. Results and Implementation

---

### 8.1. Results

---

After having the reviewed prototype from physician, the implementation has been started.

All the use cases in web application also some use cases in mobile application are developed.

- Web application:

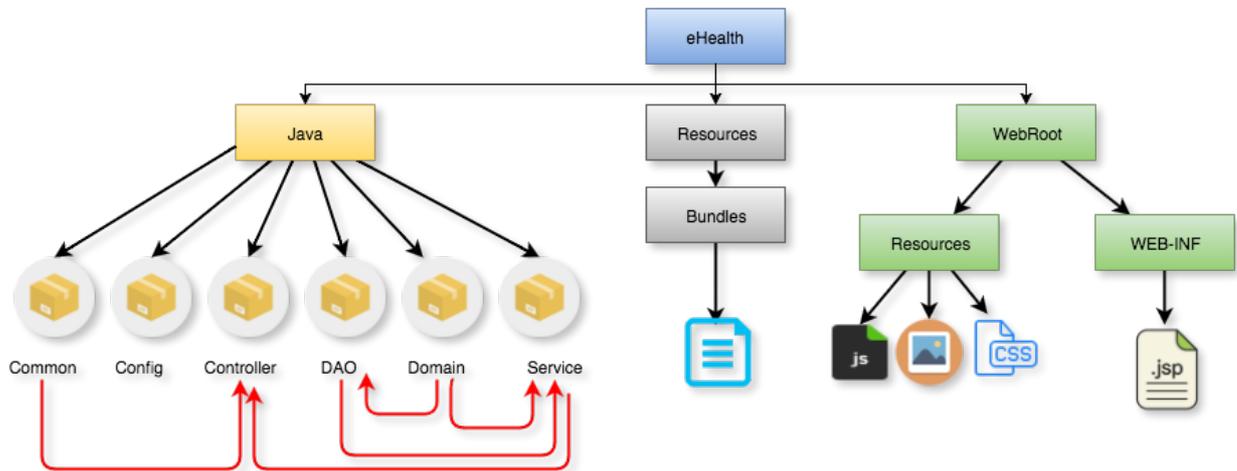
- Create account for doctor and admin
- Modify account
- Log in
- Log out
- Deactivate doctor and doctor account
- Select, Search for doctor or admin
- Manage kids nutrition target
- Manage kids exercise target
- Select, Search for kid
- Generate chart of kids process
- View chart of kids process

- Mobile application

- Register account
- Log in
- Log out
- Input food photo
- Take food photo
- Choose food photo and input % food tag

## 8.2. Project Structure

For managing code (creating, reading, updating and deleting) easily and avoid the code duplication, functionalities are separated in different package. There are some important packages and folders in the project as it is shown in the Figure 68 below.



**Figure 68: Project Structure**

- Package “config”: This package is used to store security method and some configuration in project.
- Package “common”: This package is used to store method use in common. For example, method generate object respond to rest API, method generate code key.
- Package “controller” : It stores java files with functions that control the communication between services and views.
- Package “domain” : It stores java object files which contain the properties that match with each column of the database table and it also stores getter and setter methods.
- Package “DAO” : It stores java files that extend from AbstractJpaDao and implement query method to do transaction with the database.
- Package “service”: It store abstract and logical methods that are used in the controller.
- WebRoot folder: It stores two folders, one is Resources for storing some asset file like image, CSS file and javascript file. In WEB-INF folder, it stores JSP file that contain of HLML code, use for design user interface.

## 8.3. Problems and solutions

---

### - Return Json object

In this project, it is required to return Json object for the Spring MVC. Annotation `@ResponseBody` is used to indicate a method that the return value should be bound to the web response body. Anyway, we obtained HTTP return code status 406 when using the Spring version 4.1.0.RELEASE. In pom.xml file, there were included jackson dependencies from “org.codehaus.jackson”, an older version of Jackson like below:

```
<dependency>
  <groupId>org.codehaus.jackson</groupId>
  <artifactId>jackson-core-asl</artifactId>
  <version>1.9.13</version>
</dependency>
<dependency>
  <groupId>org.codehaus.jackson</groupId>
  <artifactId>jackson-mapper-asl</artifactId>
  <version>1.9.13</version>
</dependency>
```

If a lower version of spring was used (like 4.0.7 or 3.2.11), it worked as normal. After doing some researches, we found that Spring version 4.1.0.RELEASE works well with “com.fasterxml.jackson”. So, we changed it by removing those two dependencies to:

```
<jackson.version>2.5.1</jackson.version>

<dependency>
  <groupId>com.fasterxml.jackson.jaxrs</groupId>
  <artifactId>jackson-jaxrs-base</artifactId>
  <version>${jackson.version}</version>
</dependency>

<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>${jackson.version}</version>
</dependency>
```

By doing so, using “com.fasterxml.jackson” solved the problem.



The problem solved by using the annotation `@JsonIdentityInfo`:

```
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "countryId")
```

```
@JsonIdentityInfo(generator = ObjectIdGenerators.PropertyGenerator.class, property = "cityId")
```

Object Identity information is used [12] for determining how to serialize/deserialize properties value to/from JSON (and other data formats) and it is based on existence of the `@JsonIdentityInfo` annotation. It can be used on classes (to indicate that properties of that type should have such feature enabled) as well as on individual properties (to support cases where the type itself can not be annotated; or to use different id generation sequence).

#### - Unwanted the property of child class

After solving the problem above, another problem arised. When trying to retrieve data of parent class, it serialized all the properties of the child class.

The solution consisted on using the annotation `@JsonIgnoreProperties` and `@JsonIgnore`. So it is required add `@JsonIgnoreProperties` to the class and `@JsonIgnore` to the method getter of the property that It is not desired. All properties that have `@JsonIgnore` will be ignored in the serialization.

```
@JsonIgnoreProperties({"hibernateLazyInitializer", "handler"})
```

```
@JsonIgnore
public Set<City> getCities() {
    if (cities == null) {
        cities = new java.util.LinkedHashSet<ehealth.domain.City>();
    }
    return cities;
}
```

`@JsonIgnoreProperties` is an annotation at the class level and it expects that the properties to be excluded would be explicitly indicated in the form of a list of strings.

## 8.4. Spring MVC and security java configure

---

This section will show the configuration and integration of Spring MVC 4, Spring Data JPA with Hibernate and SpringSecurity using JavaConfig. The Spring Framework supports integration with Hibernate, [13] Java Persistence API (JPA) and Java Data Objects (JDO) for resource management, data access object (DAO) implementations, and transaction strategies. For example, for Hibernate there is first-class support with several convenient IoC features that address many typical Hibernate integration issues. It can configure all of the supported features for O/R (object relational) mapping tools through Dependency Injection. They can participate in Spring's resource and transaction management, and they comply with Spring's generic transaction and DAO exception hierarchies. The recommended integration style is to code DAOs against plain Hibernate, JPA, and JDO APIs.

First, it is necessary to configure all the dependencies in pom.xml.

```
<!-- Hibernate -->

<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-core</artifactId>
  <version>4.1.4.Final</version>
</dependency>

<dependency>
  <groupId>org.hibernate.common</groupId>
  <artifactId>hibernate-commons-annotations</artifactId>
  <version>4.0.1.Final</version>
</dependency>

<dependency>
  <groupId>org.hibernate.javax.persistence</groupId>
  <artifactId>hibernate-jpa-2.0-api</artifactId>
  <version>1.0.1.Final</version>
</dependency>

<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-entitymanager</artifactId>
  <version>4.1.4.Final</version>
</dependency>

<dependency>
  <groupId>org.hibernate</groupId>
  <artifactId>hibernate-validator</artifactId>
  <version>4.2.0.Final</version>
</dependency>
```

Then, the database connection properties is configured.

```
private Properties getHibernateProperties() {
    Properties prop = new Properties();
    prop.put("hibernate.format_sql", "true");
    prop.put("hibernate.show_sql", "true");
    prop.put("hibernate.dialect", "org.hibernate.dialect.MySQL5Dialect");
    return prop;
}

@Bean(name = "dataSource")
public BasicDataSource dataSource() {

    BasicDataSource ds = new BasicDataSource();
    ds.setDriverClassName("com.mysql.jdbc.Driver");
    ds.setUrl("jdbc:mysql://localhost:8889/ehealthdb");
    ds.setUsername(" ");
    ds.setPassword(" ");
    return ds;
}
```

And finally the general resource management is configured. By using spring application contexts to handle the location and configuration of Hibernate Session Factory instances, JPA Entity Manager Factory instances, JDBC DataSource instances, and other related resources. So it is easy to manage change and safe handling of persistence resources. Moreover, using Hibernate generally needs to use the same Hibernate Session to ensure efficiency and proper transaction handling.

```
@Bean
public LocalContainerEntityManagerFactoryBean entityManagerFactory() {
    LocalContainerEntityManagerFactoryBean em = new LocalContainerEntityManagerFactoryBean();
    em.setDataSource(dataSource());
    em.setPackagesToScan(new String[] { "ehealth.domain" });

    JpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
    em.setJpaVendorAdapter(vendorAdapter);
    em.setJpaProperties(getHibernateProperties());

    return em;
}
```

Another important configuration is `WebSecurityConfigurerAdapter`. The `@EnableWebSecurity` annotation and `WebSecurityConfigurerAdapter` work together to provide web based security by extending `WebSecurityConfigurerAdapter`.

- It requires the user to be authenticated prior to accessing any URL within the application
- It create a user with the username “user”, password “password”, and role of “ROLE\_USER”
- Enables HTTP Basic and Form based authentication

```
@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Autowired
    @Qualifier("userDetailsService")
    UserDetailsService userDetailsService;

    @Autowired
    public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService).passwordEncoder(passwordEncoder());
    }
    @Bean
    public AuthenticationSuccessHandler successHandler() {
        return new RedirectLoginSuccessHandler();
    }
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/admin/**").access("hasRole('ADMIN')")
            .antMatchers("/Country").permitAll()
            .antMatchers("/Kid_ListKids/**").access("hasRole('DOCTOR')")
            .antMatchers("/NutritionTargetInput/**").access("hasRole('DOCTOR')")
            .antMatchers("/ExerciseTargetInput/**").access("hasRole('DOCTOR')")
            .antMatchers("/RigisterKid").permitAll()
            .and()
        .httpBasic()
            .and()
        .formLogin()
            .loginPage("/login").failureUrl("/login?error")
            .usernameParameter("username")
            .passwordParameter("password")
            .successHandler(successHandler())
            .and().logout().logoutSuccessUrl("/login?logout")
            .and().csrf()
            .and().exceptionHandling().accessDeniedPage("/403");
    }
}
```

## - Authentication Success Handler

In the project, there are different types of interface for different users. After a user log in into system successfully, there will determine the URL after login and perform a redirect to that URL.

In Spring, there's an excited interface "AuthenticationSuccessHandler" to handle this things. We just need to create our custom class and custom that interface. This implementation is going to determine the URL to redirect the user to after login based on the role of the user:

```
RedirectLoginSuccessHandler.java
1 package ehealth.config;
2
3 import java.io.IOException;
13
14 public class RedirectLoginSuccessHandler implements AuthenticationSuccessHandler {
15
16     @Override
17     public void onAuthenticationSuccess(HttpServletRequest httpServletRequest, HttpServletResponse httpServletResponse,
18         Authentication authentication) throws IOException, ServletException {
19
20         RedirectStrategy redirectStrategy = new DefaultRedirectStrategy();
21         String url="";
22         if (authentication.getAuthorities().toString().equals("[ROLE_DOCTOR]")){
23             url = "/Kid_ListKids?username=" + authentication.getName();
24         }
25         else if (authentication.getAuthorities().toString().equals("[ROLE_ADMIN]")){
26             url = "/Doctor_ListDoctors?username=" + authentication.getName();
27         }
28         redirectStrategy.sendRedirect(httpServletRequest, httpServletResponse,url);
29     }
30 }
```

## 8.5. Using JSP - Standard Tag Library formatting tags

---

This web application that has been developed in Catalan and Spanish language. So it is necessary to find a good solution to solve the problems with changing language. We can use JSP - Standard Tag Library (JSTL) formatting tags to help us in this problem. It is required to use a message to display a name as label or button name etc... So when a change of language is required, changing the message property is enough.

The JavaServer Pages Standard Tag Library (JSTL) is a collection of useful JSP tags [14] which encapsulate core functionalities common to many JSP applications. JSTL has support for common, structural tasks such as iteration and conditionals, tags for manipulating XML documents, internationalization tags, and SQL tags. It also provides a framework for integrating existing custom tags with JSTL tags. The JSTL tags can be classified, according to

their functions, into following JSTL tag library groups that can be used when creating a JSP page: Core Tags, Formatting tags, SQL tags, XML tags, JSTL Functions.

The JSTL formatting tags are used to format and display text, the date, the time, and numbers for internationalized Web sites.

It's easy to use this existing library of JSP. The image below shows how to use it.

First, we need a properties file to store those text.

```
kid-resources.properties
17 # Kid
18 kid.title=Kid
19 kid.kidid.title=KidId
20 kid.kidid.help=Enter KidId
21 kid.high.title=High
22 kid.high.help=Enter High
23 kid.weight.title=Weight
24 kid.weight.help=Enter Weight
25 kid.address.title=Address
26 kid.address.help=Enter Address
27 kid.codekey.title=CodeKey
28 kid.codekey.help=Enter CodeKey
```

Then, we can call it to use in .JSP file by importing JST library and set Bundle property.

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<fmt:setBundle basename="bundles.user-resources" />
```

Finally, we can use it like in code below: and result

```
<div id="contentarea">
  <div id="tablewrapper">
    <table id="listKidsTable" class="table table-hover results">
      <thead>
        <tr>
          <th class="thead">No</th>
          <th class="thead"><fmt:message key="user.name.title" /></th>
          <th class="thead"><fmt:message key="user.surename.title" /></th>
          <th class="thead"><fmt:message key="user.telephone.title" /></th>
          <th class="thead"><fmt:message key="city.title" /></th>
          <th class="thead"><fmt:message key="kid.connect.title" /></th>
        </tr>
      </thead>
```

## 9. Conclusion

---

The internship at the University of Girona went well. Knowledge and experiences that have been learnt from school are used to develop the project, including research into new technologies, indications from tutor help to improve the development. In resume, the project is very interesting that provides a lot of experiences of project development, problem solving and how to build web application and android application.

Finally, for almost six months of system development, although the application is not finished completely but almost all the necessary features are done and work very well. Moreover, the project's requirements are already analysed to match with what user need. With the experiences of this internship, it shows that more than study theory at school, practice in the real project is very important for the professional lives.

This part will explain the conclusion of the project which includes the achievement and non-achievement. In addition, the strong point, weak point and experience are presented in this part.

### 9.1. Achievement

---

After finishing and testing the project already we can resumed that the project is not finished completely. However, we get a web application, which has a user friendly interface, and it's can be used by doctor and admin. Main functions such as security part, user authentication, a web application have been done. All the use cases of actor doctor and admin like log in, modify account, logout, CRUD food target for kid, CRUD exercise target for kid, view kid process, search/select kid, create account doctor/admin, search doctor/admin are

complete. Moreover, there are some use cases on mobile application for actor kid and parent like log in, register account and input food photo's information that already completed.

With all these achievements, there are some strong points such as the application has been developed with the necessities functionalities of backend and designed with a simple and understandable interface. In addition, the project has a well organised structure which is easy for another to continue. It's also work well when we have an interaction with the database like insert data to database invalid and the performance of the application is acceptable.

## 9.2. Non-Achievement

---

We achieved some parts of the project but there are still remain some parts that could not be finished. It doesn't goes well as the plan that has been set in the beginning. Moreover, there is no more time and the project size is big so it's very challenge to handle it a lone. All these reasons, most of the use cases on the mobile side such as functionality of exercise activity, game competition and social are not yet done.

## 9.3. Experiences

---

After finishing this project we learnt many experiences. We now have more knowledge of java programming language, Spring MVC, Spring Security, Android development and understand the methodology of project development. It also teaches how to search for the new technology, technique and documents to solve problem and application development.

## 9.4. Perspective

---

In the future, if I have other projects I will use all the experience that I get from my internship in these six months to apply on it. I will spend much more time to study about

framework that I use in this project, because it's very useful. I hope that I can finish all functions that remain on mobile application.

## 9.5. Future works

---

Since it's not a finished project so there are some remain task need to be completed in the future. The future works includes implementation all the functions on mobile application, fix few bugs of the finished function and research about smart watch connection with mobile and how to retrieve data from it.

## 10. Bibliography

---

- [1] JavaServer Pages. *JavaServer(TM) Pages Implementation*. <https://jsp.java.net/>. (Last access date: 01 June 2016)
- [2] Mavel. <https://marvelapp.com/> (Last access date: 20 May 2016)
- [3] Bootstrap (4 June 2016). *Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web*. <http://getbootstrap.com/>(Last access date: 03 June 2016)
- [4] Wikipedia (4 April 2015). *ICONIX*. <https://en.wikipedia.org/wiki/ICONIX> (Last access date: 03 June 2016)
- [5] Doung, R. & Matt, S. (2007). *Use Case Driven Object Modeling with UML: Introduction to ICONIX Process*. United State of America. Apress. 1-20
- [6] Sureshkumarveluswamy (29th July 2010). *ICONIX Process for Agile Software development*. <https://sureshkumarveluswamy.wordpress.com/2010/07/29/iconix-process-for-software-development/> (Last access date: 03 June 2016)
- [8] Kantarworldpanel (02 September 2015). *Android Share Loss Continues in Europe “Big Five” Markets*. <http://www.kantarworldpanel.com/global/News/Android-Share-Loss-Continues-in-Europe-Big-Five-Markets> (Last access date: 05 June 2016)
- [9] Androidauthority (2016). *Is Android losing ground to iOS in Europe?*. <http://www.androidauthority.com/samsung-s-health-update-compare-steps-friends-heart-rate-695455/> (Last access date: 07 June 2016)

- [10] Spring Security (6 March 2016). *Spring Security*. [https://en.wikipedia.org/wiki/Spring\\_Security](https://en.wikipedia.org/wiki/Spring_Security) (Last access date: 03 June 2016)
- [11] Monjism. *Application Security Areas*. <https://monjism.wordpress.com/tag/authorization/> (Last access date: 03 June 2016)
- [12] Fasterxml. *Feature: Object Identity (Or Object Identifier / Object Id)*. <http://wiki.fasterxml.com/JacksonFeatureObjectIdentity> (Last access date: 03 June 2016)
- [13] Spring.io (2016). *Object Relational Mapping (ORM) Data Access*. <http://docs.spring.io/spring/docs/current/spring-framework-reference/html/orm.html> (Last access date: 05 June 2016)
- [14] Tutorialspoint (2016). *JSP - Standard Tag Library (JSTL) Tutorial*. [http://www.tutorialspoint.com/jsp/jsp\\_standard\\_tag\\_library.htm](http://www.tutorialspoint.com/jsp/jsp_standard_tag_library.htm) (Last access date: 05 June 2016)
- [15] Spring.io (2016). *Spring Tool Suite™*. <https://spring.io/tools> (Last access date: 05 June 2016)
- [16] Mamp (2016). *MAMP: My Apache - MySQL - PHP*. <https://www.mamp.info/en/> (Last access date: 05 June 2016)
- [17] MySQL (2016). *MySQL*. <https://www.mysql.com/> (Last access date: 05 June 2016)
- [18] MySQL (2016). *MySQL Workbench*. <https://www.mysql.com/products/workbench/> (Last access date: 05 June 2016)
- [19] VMWare (2016). *Pivotal tc Server*. <https://www.vmware.com/products/vfabric-tcserver/overview> (Last access date: 05 June 2016)
- [20] Google, Chrome. *Get a fast, free web browser*. <https://www.google.com/chrome/browser/desktop/> (Last access date: 05 June 2016)
- [21] Google. *Postman*. <https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdgghehcdcbncdddomop?hl=en> (Last access date: 05 June 2016)

[22] Developer android. *Meet Android Studio*. <https://developer.android.com/studio/intro/index.html> (Last access date: 05 June 2016)

[23] Genymotion. *Speed up your development lifecycle with an easy, accessible, and effective testing and collaboration tool*. <https://www.genymotion.com/> (Last access date: 05 June 2016)

[24] Subversion. *Apache™ Subversion®*. <https://subversion.apache.org/> (Last access date: 05 June 2016)

[25] Bitbucket. *Code, Manage, Collaborate*. <https://bitbucket.org/> (Last access date: 05 June 2016)

[26] Atlassian. *Harness the power of Git and Hg in a beautifully simple application*. <https://www.atlassian.com/software/sourcetree> (Last access date: 05 June 2016)

[27] Java. <https://www.java.com/en/> (Last access date: 05 June 2016)

[28] Dropbox. <https://www.dropbox.com> (Last access date: 05 June 2016)

[29] Maven. *Welcome to Apache Maven*. <https://maven.apache.org/>(Last access date: 05 June 2016)

[30] JQuery. <http://jqueryui.com/>(Last access date: 05 June 2016)

[31] JQuery. <http://jqueryui.com/>(Last access date: 05 June 2016)

---