Fun Tama

Design Document

Development Team (Team 9)

Jiaping Qi | Fangzhou Lin | Qi Zhang | Shayin Feng | Chi Luo

Table Of Contents

Purpose:

Nowadays people are stressed out after a long time of working or studying. A study shows that pets are effective for stress relief. Moreover, people with pets feel less lonely than those who don't have pets. Raising a pet in real life costs a lot of money and attention and requires owner's responsibilities as well. Some people also could not have a pet because of some personal reasons such as allergy. Thus, we came up with an idea that a digital and virtual pet may solve this problem. As a result, people can enjoy these digital pets with fewer efforts and costs.

The functional requirements of this application will include the following items:

- **Pet system:** we are going to build a virtual pet for the user. So the pet system is the main system of the project. Pet's model is a big part of the project. As a virtual pet, it should be able to walk, run, jump and so on. Pet's AI will make the decision what to do next. Pet's conditions will be represented by numbers. Pets have 4 conditions, which are hunger, happiness, health and cleanness. User's actions (eg. using items to feed, entertain or cure animals) can impact pet's conditions.
    - As a user:
        - I would like to see the pet on the screen.
        - I would like to see the pet's condition menu on the top-left corner.
        - I would like to see my pet walk, jump or have other action.
        - I would see my action have an impact on my pet.
        - I would like to see the pet growing up.

- **AR control system:** Fun Tama is a Augmented Reality project. The system is going to understand user's actions. Users are going to use their bodies in the game as controller. Users can also select menu, check condition and buy stuff in a in-game shop by their hand in the air.

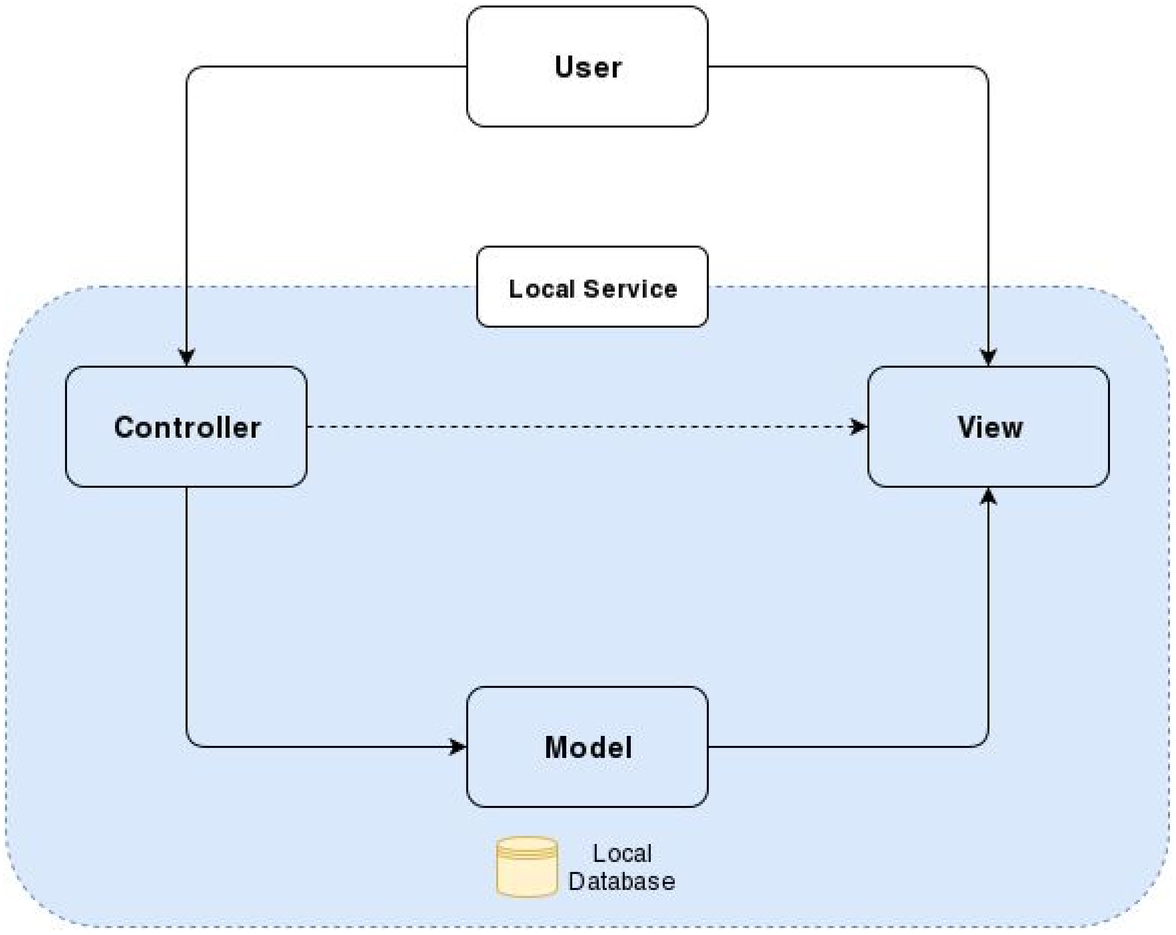
- As a user:
    - I would like to have a menu on the top-right side of the screen.
    - I would like to select the items in the menu.
    - I would like to check the condition on the top-left side of the screen.
    - I would like to buy goods in in-game shop.

- **Support system:** The game will be supported by some subsystems, such like a system to take the effect of the item that user uses, a system to record time passed since the game starts, in-game shop, currency system and a system to keep the game status when user quits and resume the game when user comes back.
    - As a user:
        - I would like to see the effect of the item I used.
        - I would have different items to use.
        - I would like to get coins from the pet.
        - I would like to resume the game anytime I want.

Design Outline:

Our application is an offline game which user could keep a virtual pet. It is a local application which utilizes a client-server model and the local server implements MVC architecture.

High-Level Structure of the system:

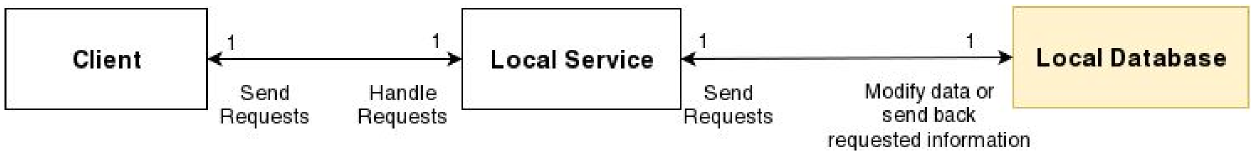


We choose Model-View-Controller pattern as the design pattern and follows the design of the above diagram. Since we are implementing an application using local service and database, client will send a request to controller classes and then controller classes will interact with the model that encapsulated data in the user's local database. The change of data in model will update and be displayed on View component.

The purpose of each component and interactions between components:

- **Model**: Stores the user data, responds to requests from controller and create and update view.
- **View**: View will be updated after the data being added to, modified in or retrieved from model and display the information to the client.
- **Controller**: Controller classes receive requests from client and obtain/modify data from/in Model component.

Our application also use a client-server model:



- **Client:** User will interact with local services through client. Client will send requests to backend and display the pet's information to users.
- **Local Service:** Local service will handle the requests sent from client and interact with local DB. It will also do the calculations, data validation check and apply different logics in the local service.
- **Local Database:** Local database will store the pet's and user's data and responds to the requests from local services including data modification, data retrieving and adding data.

Design Issues:

1. Which depth sensor camera to use.
    a. Option 1: Microsoft Kinect
    b. Option 2: ZED stereo Camera
    c. Option 3: DUO mini lx

   We decided to use Microsoft Kinect for three reasons. Firstly, Microsoft Kinect has a median price which is acceptable for a school project. Secondly, the Kinect is great for medium-range tracking of multiple skeletons and faces. At last, what's most important, Kinect has a great SDK supported by Microsoft.

2. Which platform to use.
    a. Option 1: Windows
    b. Option 2: OS X
    c. Option 3: Linux/Unix

   We decided to use Windows. Since Kinect is a Microsoft product. It's SDK is written in C#. C# is the system language of windows, just like C to Linux. So Using Windows will provide our team a great convenience on the project.

3. Using what to control the game.
    a. Option 1: AR cards
    b. Option 2: hands
    c. Option 3: Game controler

   We decided to choose option 2. AR cards will increase the cost of the project and since this is a AR project, observably using Game controller to control the game is not a good idea.

4. What is the view of game.
   a. Option 1: first-person
   b. Option 2: third-person

   We decided to use third-person view in this game. Because players will be able to see themselves and the surrounding in the screen if we use the third-person view. We think it is much more comfortable to control the game and interact with it.

5. Currency system.
   a. Option 1: Having a currency Class
   b. Option 2: User class includes a coin attribute.

   We decided to use option 2. Since Fun Tama is a single-player game, we don't want the game to have a very complex currency system. In this way, we can keep a number under the user class to record the number of coins, which is easy to maintain.

6. Death of pets.
   a. Option 1: pets won't die anyhow
   b. Option 2: pets die if one of the attributes is below a limit

   We decided to use option 2. Because it makes the game more realistic and more challenging. And the users could be much more responsible for their pets. Otherwise, they might stop playing the game and leave their pets for a very long time since nothing will happen to the pets.

7. Online game or offline game.
   a. Option 1: online and multi-player game
   b. Option 2: single-player game, online server and database
   c. Option 3: single-player game, storing all data locally

We decided to use option 3. Our game is mainly about the interaction between players and their own pets. It is not necessary to be an online multi-player game. Also, using online server and database will increase the cost of our project. It is much more convenient for us to implement local service than online server and database.

8. Same initial values for the pet's attributes or different values.
    a. Option 1: use same values for different pets
    b. Option 2: generate different values for different pets

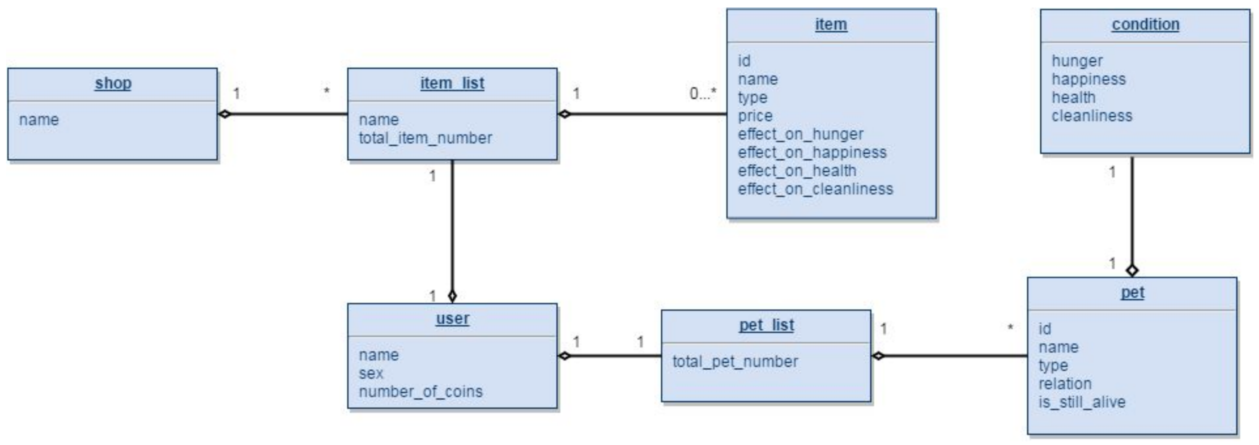
    We decided to use option 2. Since the main purpose of this game is to give a chance for user to have a digital pet. To be more real, random values for different pet's attributes should be better.

9. If more than more person in the range of sensor.
    a. Option 1: all people the sensor found are able to control it
    b. Option 2: only one person can control it

    We decided to use option 2. We think it will make the game very messy if two or more people can control the game at the same time. We choose to allow only one player to play it at one time, and the one who is the first to enter the playing range or the first recognized by program will be able to control the game.

Design Detail:

1. Class Design



2. Class Detail

a. User

- User class contains current user's information including name, sex and number_of_coins.
- Each attribute in user class contains get() and set() function for convenient use.
- Each user has a item_list (inventory) and a pet_list.

b. Pet_list

- Pet_list class contains the total number of the pet belongs to the current user.
- A pet_list may have multiple pets.

c. Pet

- Pet class saves information for a single pet, including id, name, type, relation (to user) and is_still_alive.
- For each attribute in pet class, get() and set() method are available.
- Each pet have a condition class related.

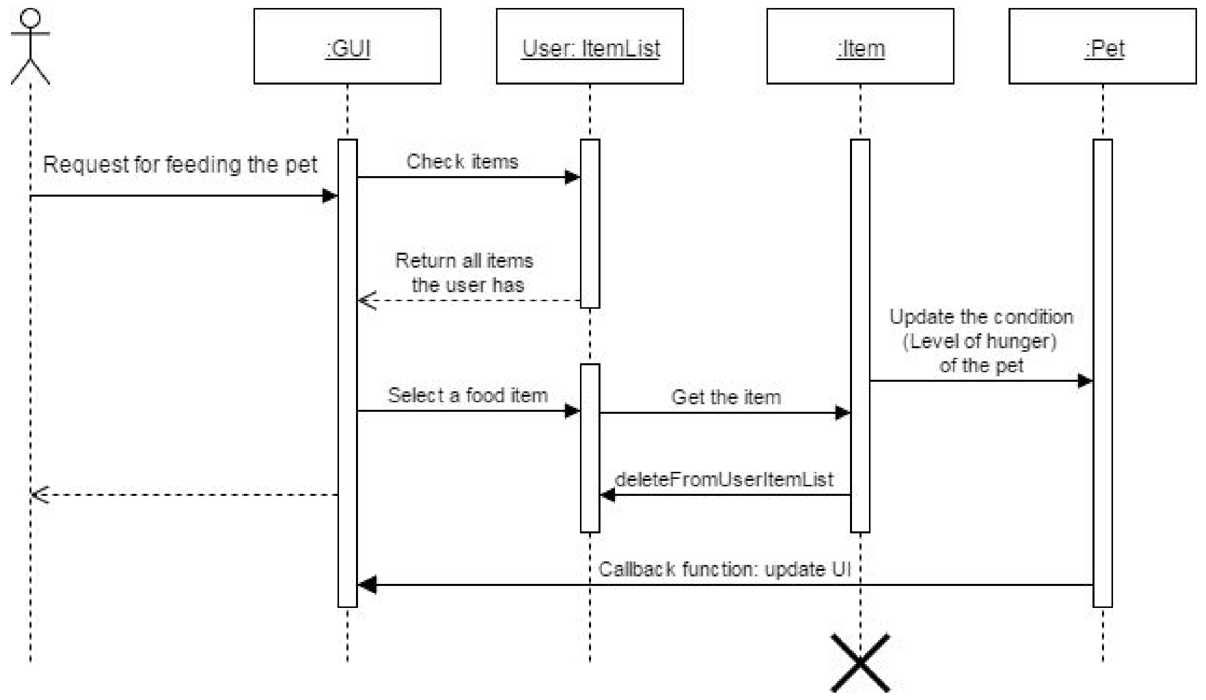
d. **Condition**
    - Condition class contains 4 attributes for a pet: hunger, happiness, health and cleanliness.

e. **Shop**
    - Shop class only contains the name of the shop as the attribute.
    - It is also related to several item_list class.

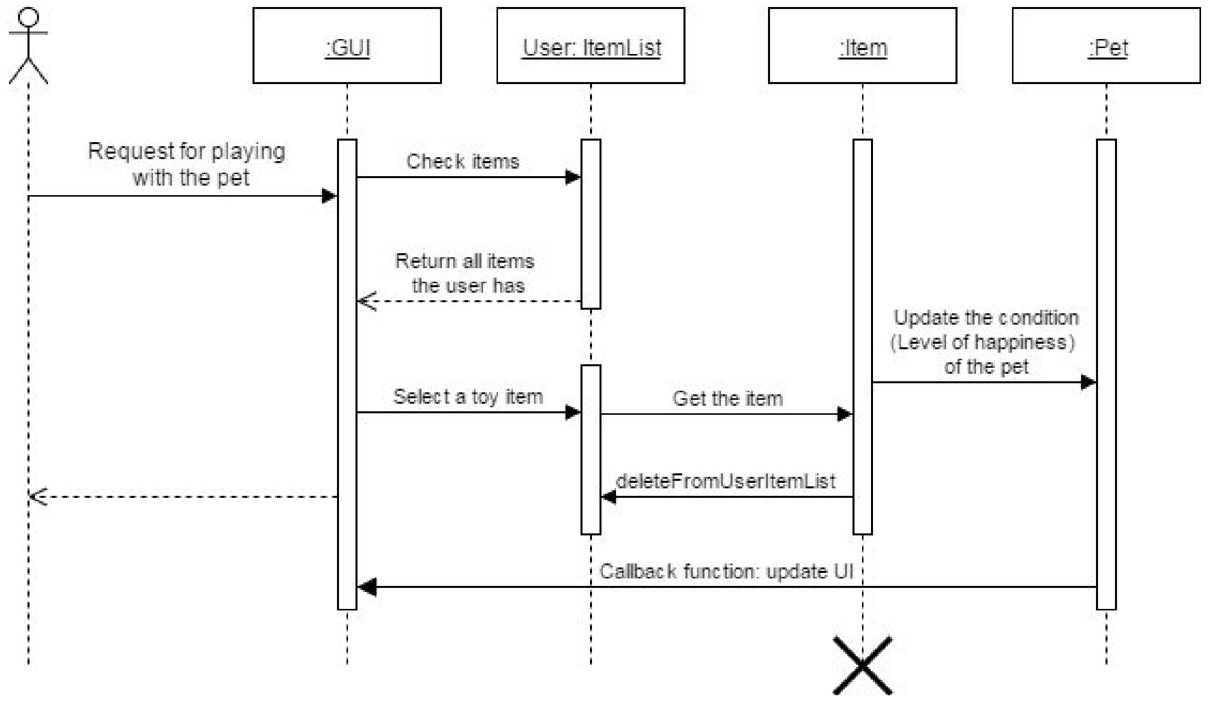
f. **Item_list**
    - Each item_list contains the name of the list (category) and the total_item_number of this list.
    - For total_item_number attribute in the item_list class, fetch() method is available.

g. **Item**
    - Item class contains the following attributes: id, name, type, price, effect_on_hunger, effect_on_happiness, effect_on_health, effect_on_cleanliness.
    - Each item has corresponding get(), set() and use() methods.

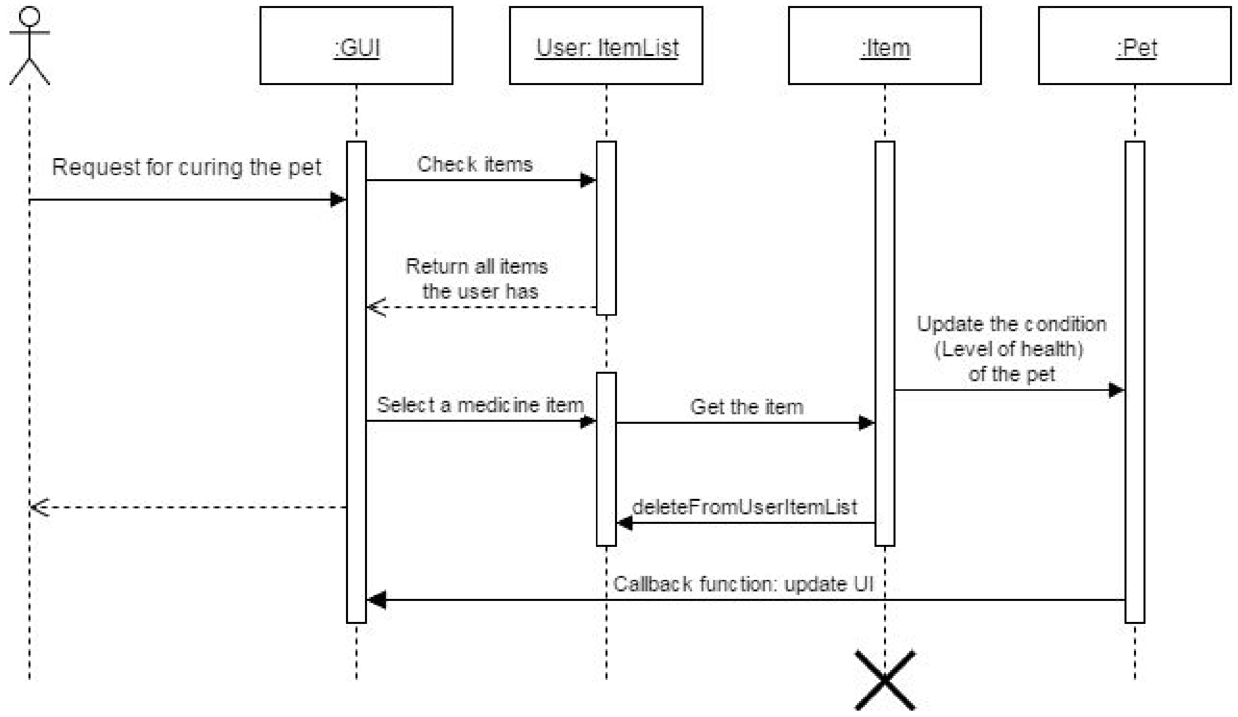
3. Sequence Diagram


:GUI
User: ItemList
:Item
:Pet
Request for feeding the pet
Check items
Return all items the user has
Update the condition (Level of hunger) of the pet
Select a food item
Get the item
deleteFromUserItemList
Callback function: update UI

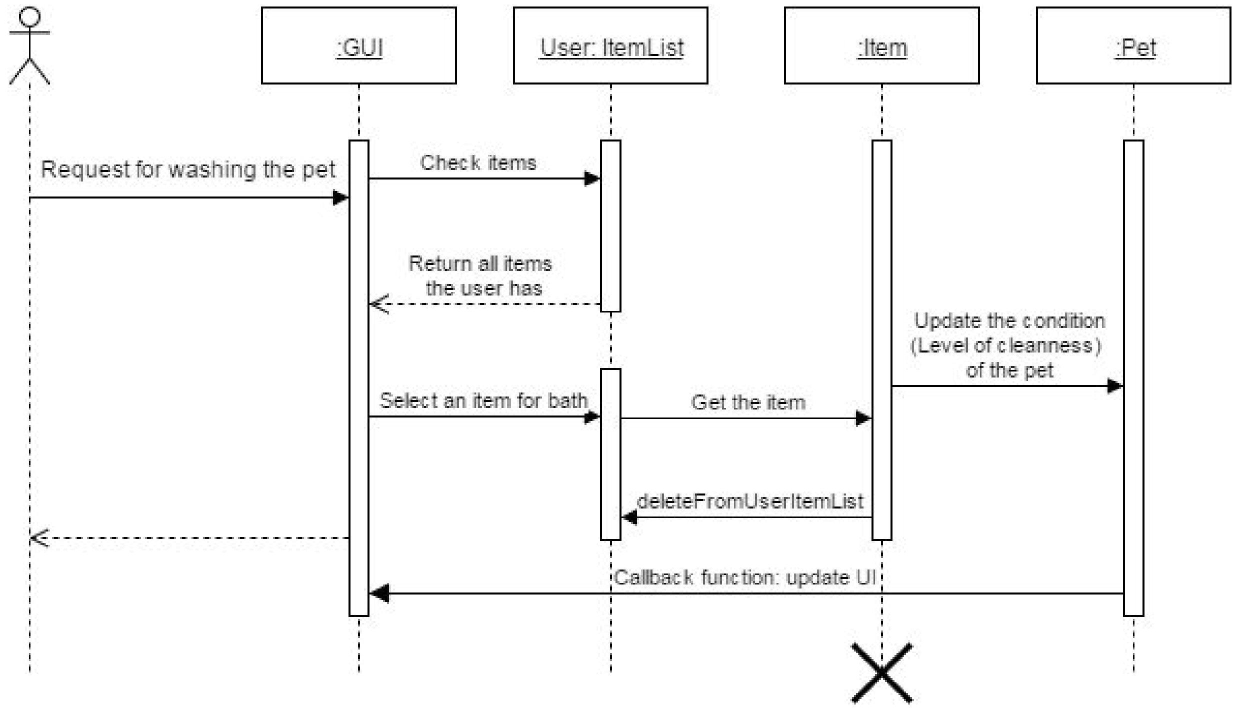
(1) Feed the pet


:GUI
User: ItemList
:Item
:Pet
Request for playing with the pet
Check items
Return all items the user has
Update the condition (Level of happiness) of the pet
Select a toy item
Get the item
deleteFromUserItemList
Callback function: update UI

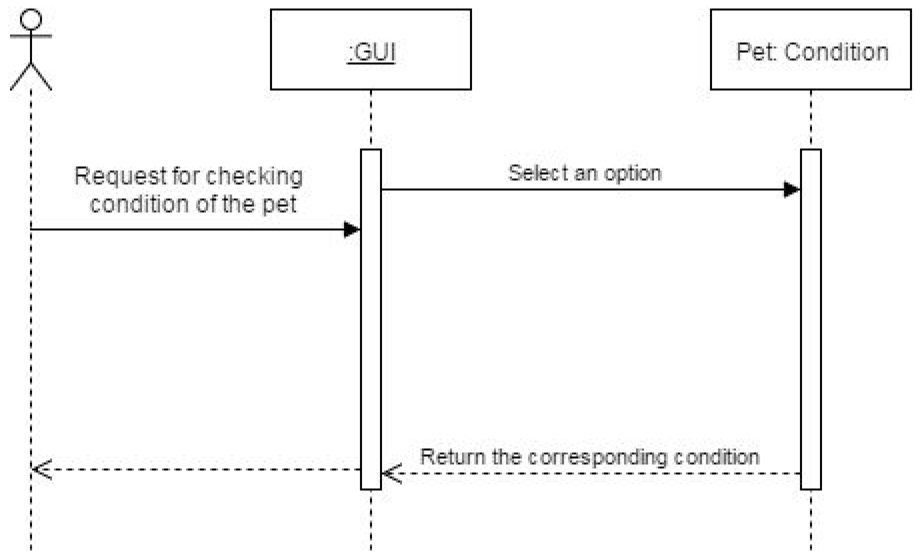
(2) Play with the pet


:GUI
User: ItemList
:Item
:Pet
Request for curing the pet
Check items
Return all items the user has
Update the condition (Level of health) of the pet
Select a medicine item
Get the item
deleteFromUserItemList
Callback function: update UI

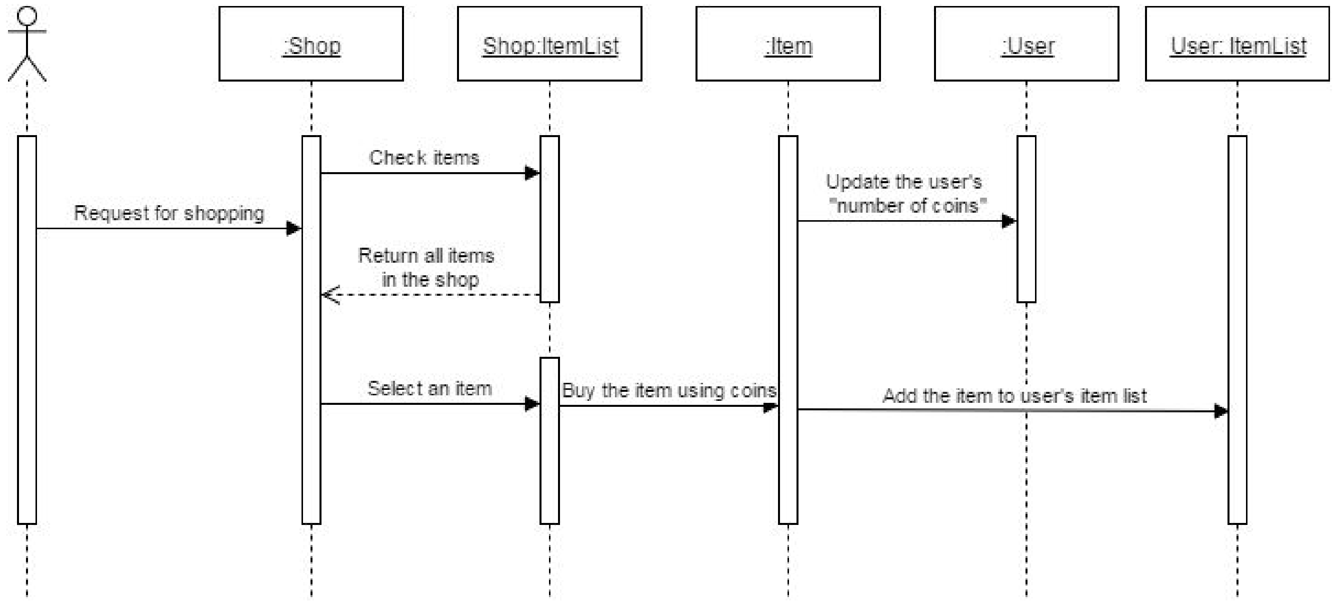
(3) Cure the pet


:GUI
User: ItemList
:Item
:Pet
Request for washing the pet
Check items
Return all items the user has
Update the condition (Level of cleanness) of the pet
Select an item for bath
Get the item
deleteFromUserItemList
Callback function: update UI

(4) Wash the pet

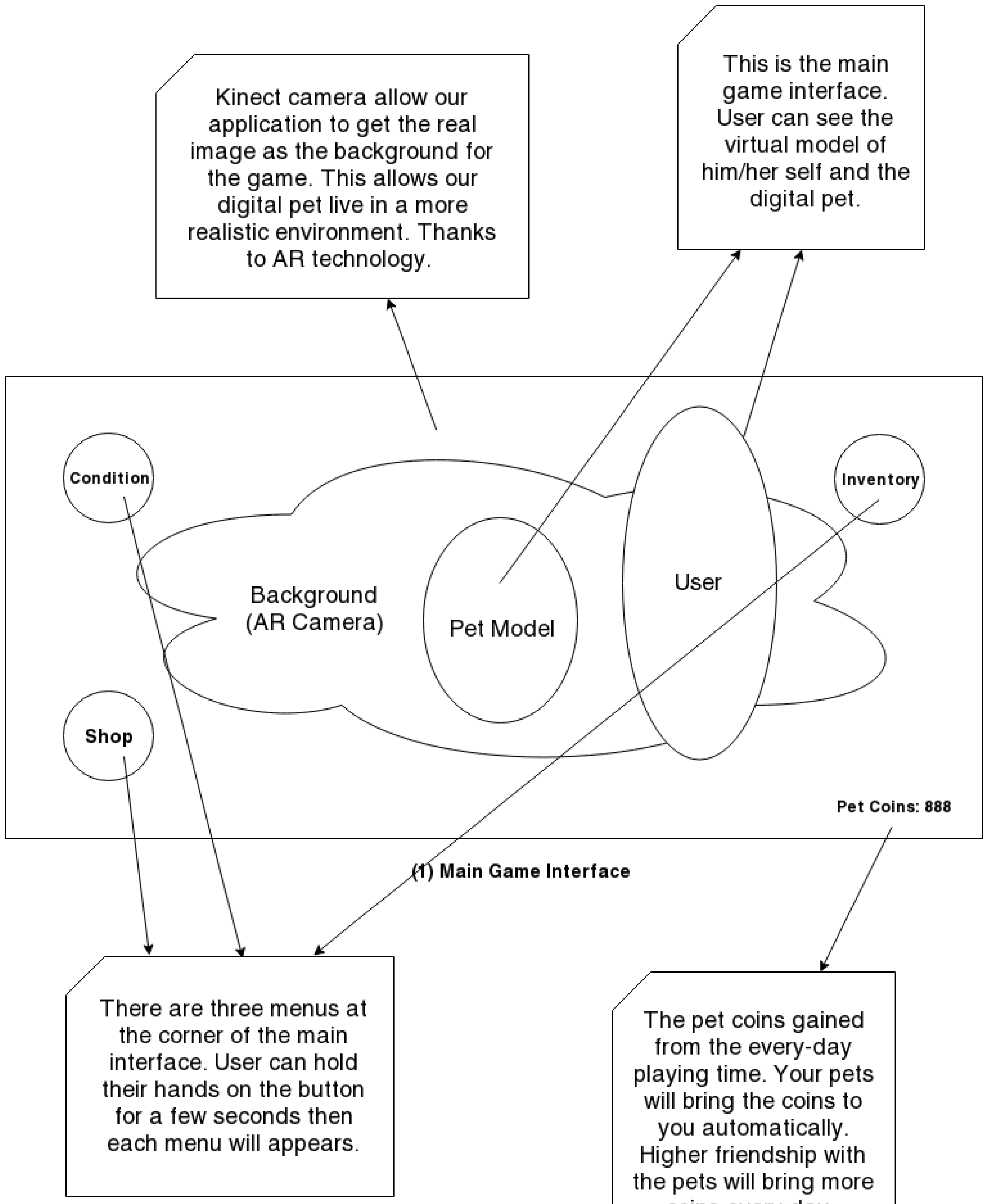

:GUI
Pet: Condition
Request for checking condition of the pet
Select an option
Return the corresponding condition

(5) Check the condition



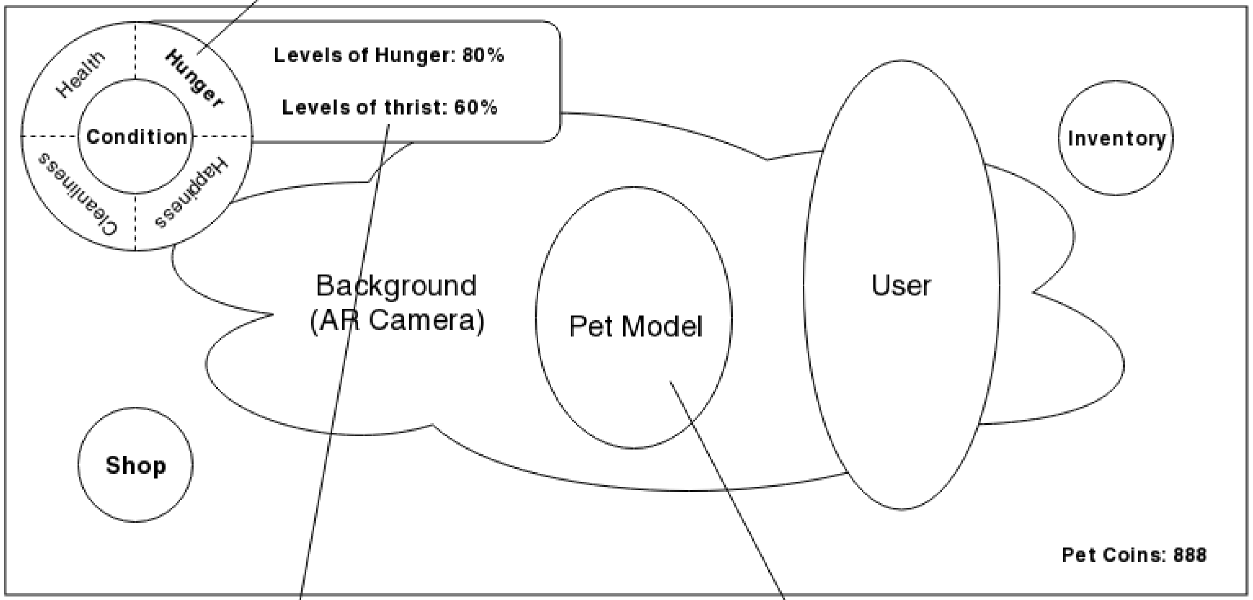
(6) Shopping

4. UI Mockups


Kinect camera allow our application to get the real image as the background for the game. This allows our digital pet live in a more realistic environment. Thanks to AR technology.
This is the main game interface. User can see the virtual model of him/her self and the digital pet.
Condition
Inventory
Background (AR Camera)
Pet Model
User
Shop
Pet Coins: 888
There are three menus at the corner of the main interface. User can hold their hands on the button for a few seconds then each menu will appears.
The pet coins gained from the every-day playing time. Your pets will bring the coins to you automatically. Higher friendship with the pets will bring more coins every day.

(1) Main Game Interface

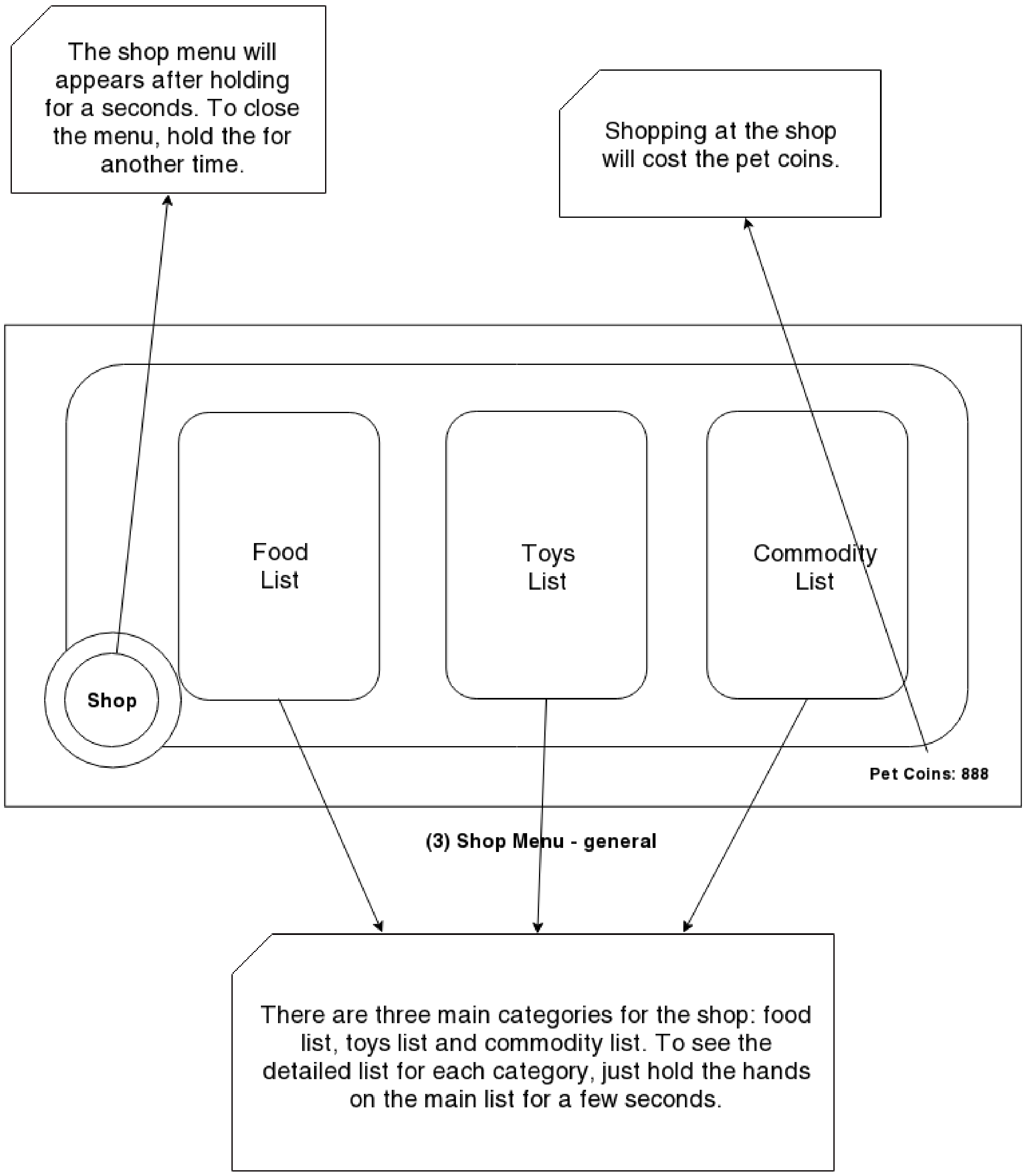
After holding the hands at the condition menu button, the condition menu will disappear. There are four different sub-menus for condition: Health, Hunger, Cleanliness and Happiness. This four data for pets reflex the current condition for your pets. To check the detailed information for each condition, hold the hands at each button.



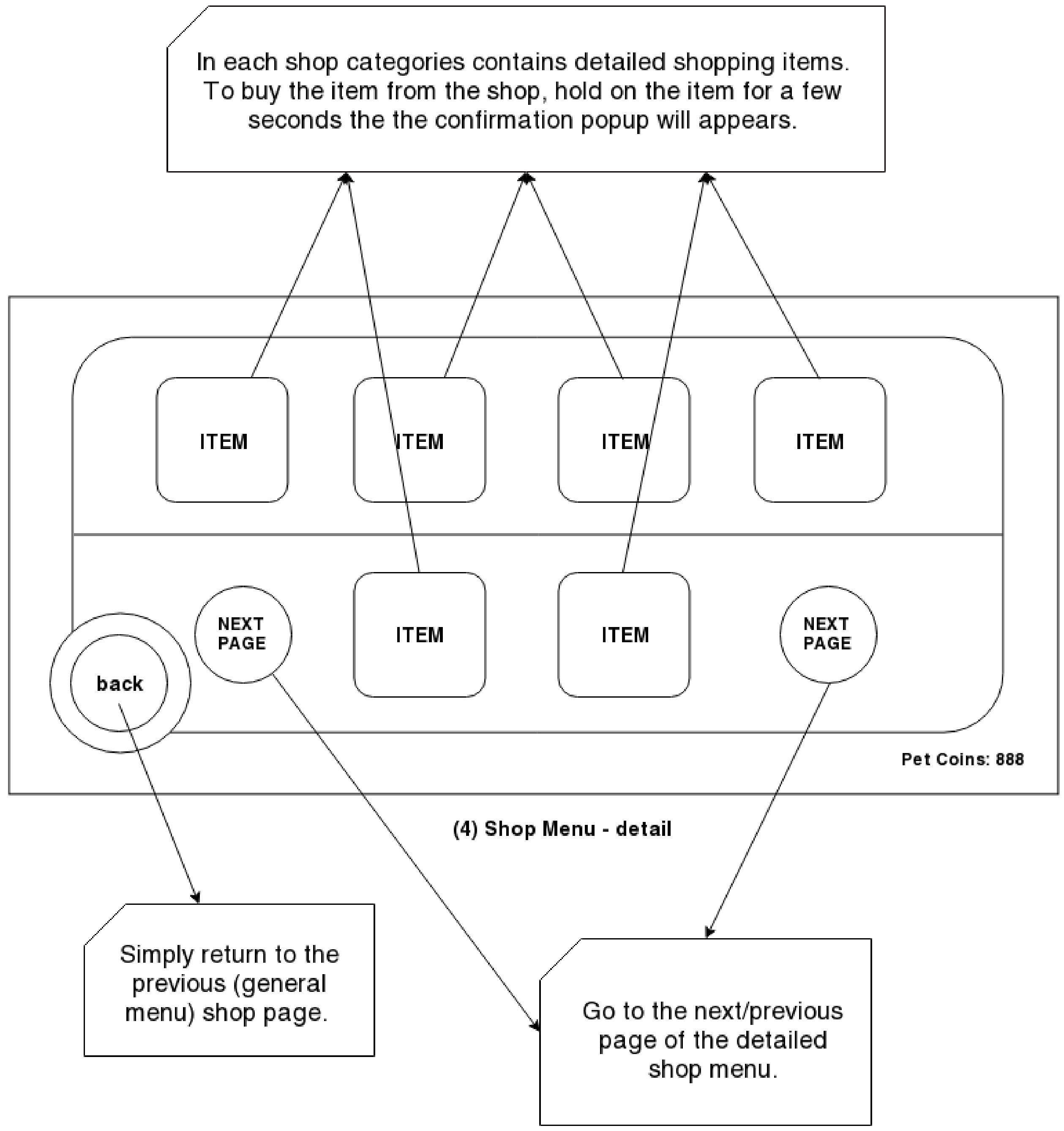
(2) Condition Menu

This is the detailed information for Hunger condition. If the value is too low, this means that user need to take more care of the pets.

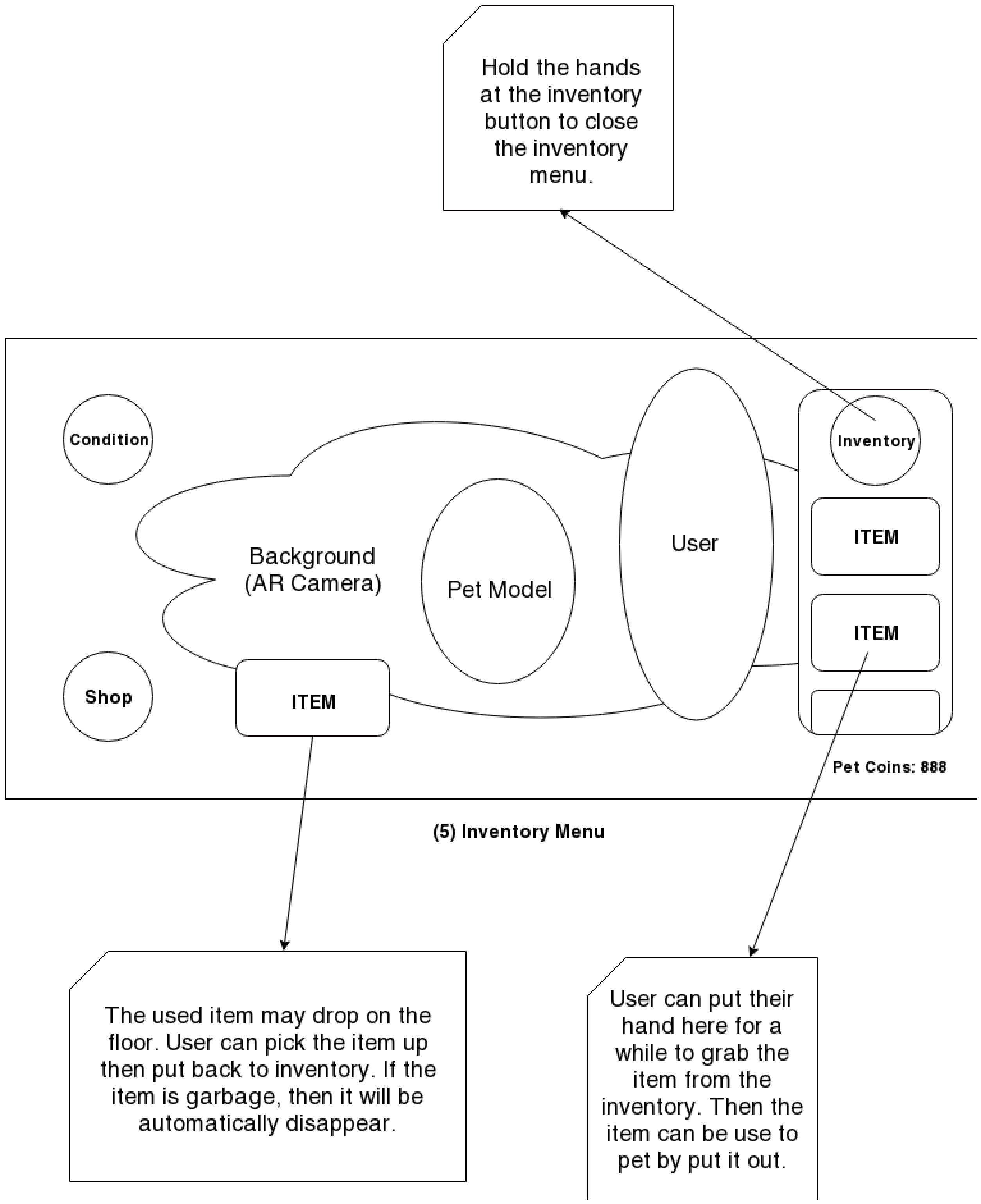
The value of the condition will reflect on the model of the pet. For example, if the health level for the pet is low, the pet model will looked sick and the action for the pet will become slow.


The shop menu will appears after holding for a seconds. To close the menu, hold the for another time.
Shopping at the shop will cost the pet coins.
Food List
Toys List
Commodity List
Shop
Pet Coins: 888
There are three main categories for the shop: food list, toys list and commodity list. To see the detailed list for each category, just hold the hands on the main list for a few seconds.

(3) Shop Menu - general


In each shop categories contains detailed shopping items. To buy the item from the shop, hold on the item for a few seconds the the confirmation popup will appears.
ITEM
ITEM
ITEM
ITEM
NEXT PAGE
ITEM
ITEM
NEXT PAGE
back
Pet Coins: 888
Simply return to the previous (general menu) shop page.
Go to the next/previous page of the detailed shop menu.

(4) Shop Menu - detail


Hold the hands at the inventory button to close the inventory menu.
Condition
Inventory
Background (AR Camera)
Pet Model
User
ITEM
ITEM
Shop
ITEM
Pet Coins: 888
The used item may drop on the floor. User can pick the item up then put back to inventory. If the item is garbage, then it will be automatically disappear.
User can put their hand here for a while to grab the item from the inventory. Then the item can be use to pet by put it out.

(5) Inventory Menu