Assignment 2

Team number: 77
Team members

Name	**Student Nr.**	**Email**
Yaxin Li	*2781906*	*y.li16@student.vu.nl*
Seoyeon Kim	*2796797*	*s.y.kim2@student.vu.nl*
Hyeonjee Kim	*2796710*	*h.j.kim2@student.vu.nl*
Benjamin (Ahn Jaemin) de Vries	*2624258*	*b.d.de.vries@student.vu.nl* *Ahn-jaemin@protonmail.com(GitHub)*

Format: We style the name of each class in bold, whereas the attributes, operations, and associations as underlined text, objects are in italic.

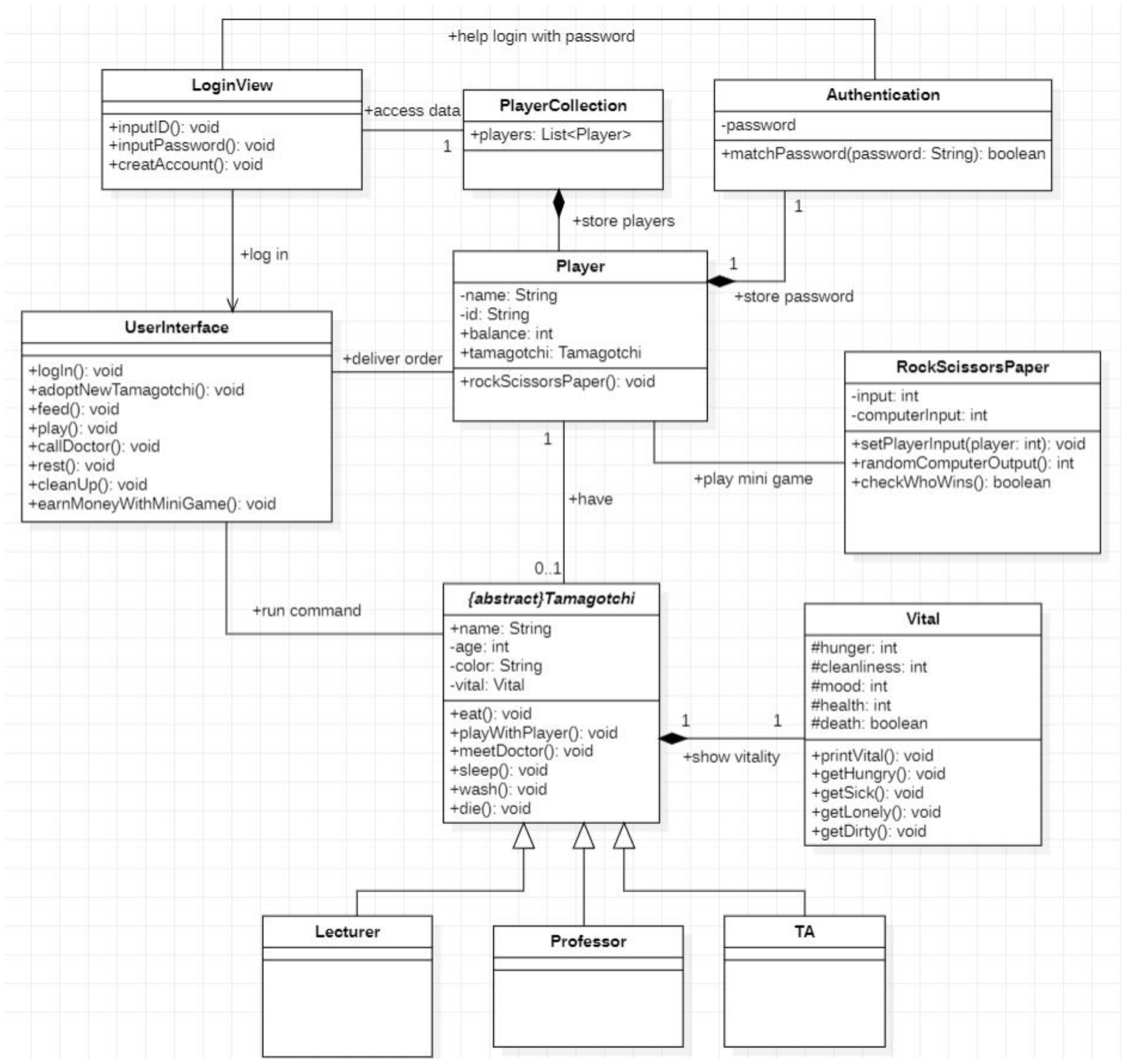
Summary of changes from Assignment 1

Author(s): Yaxin Li

Page	*Column*	*Changes*
3	*Overview*	*Templates' description removed. Pictures removed cause we shouldn't use them without author's permissions.*
4	*Functional features*	*Moscow format implemented and more detailed description.*
6	*Quality requirements*	*Quality attribute corrected and description more specified as suggested.*
7	*Time log*	*Time log updated*

Class diagram

Author(s): Hyeonjee Kim



This diagram shows the basic structure of our software for 'My Tamagotchi'. This consists of **'UserInterface'**, '**LoginView**', '**PlayerCollection**', '**Player**', '**Authentication**', '**RockScissorsPaper**', '**Tamagotchi**', '**Lecturer**', **'Professor**', '**TA**', and **'Vital'**.

First of all, the class '**UserInterface**' is where the main function would be operated. This class is made to separate the place where the user commands take place from the real execution. Users will start with 'login()' operation, which is related to the '**LoginView**' class. Also, the main buttons which will be shown in real user interface like 'feed', 'play', 'callDoctor', 'rest', 'cleanUp' and 'earnMoneyWithMiniGame' that users will use while growing their tamagotchi are implemented as operations here. If the Tamagotchi dies, 'adoptNewTamagotchi()' operation would add a new Tamagotchi for the player. Because UserInterface interacts with both player and tamagotchi, associations with '**Player**' class and '**Tamagotchi**' class work.

In the '**LoginView**' class, there are two operations about entering id and password.This class is independent in order to make **UserInterface** clearer and make the login process secure. When the user writes the id while operating 'inputID()', the operation would find whether there is a player with the id. If a corresponding player exists, the user writes the password while operating ‘inputPassword()’ and then the ‘matchPassword()’ in the ‘**Authentication**’ class would be activated. **LoginView** has an attribute of ‘PlayerCollection’ to match a corresponding player, so the association exists on this diagram. If the id or password doesn’t match the existing data, User can try logging in again or make a new account by ‘creatAccount()’ operation. This operation would add a new player in the ‘**PlayerCollection**’.

‘**PlayerCollection**’ is a class that contains a bunch of Players as a list just like the name.

‘**Player**’ has its own name, id, **Authentication**, and balance. Balance is the amount of money that the player has. Also, there is a **Tamagotchi** as an attribute. When some operations like ‘feed() in ‘**UserInterface**’ are activated, the balance of ‘**Player**’ is reduced. Players can earn money by ‘rockScissorsPaper()’ when triggered by ‘earnMoneyWIthMiniGame()’ operation in ‘**UserInterface**’.

‘**Authentication**’ is for saving the player’s password independently and securely. During the login process, ‘matchPassword(String)’ checks whether the given password is the same as the real password.

‘**RockScissorsPaper**’ is for the rock scissors paper game for earning money. After the user sets its input by ‘setPlayerInput()’ operation, ‘randomComputerOutput()’ operation sets the opponent’s input randomly. Then, ‘checkWhoWins()’ checks who won by comparing inputs.

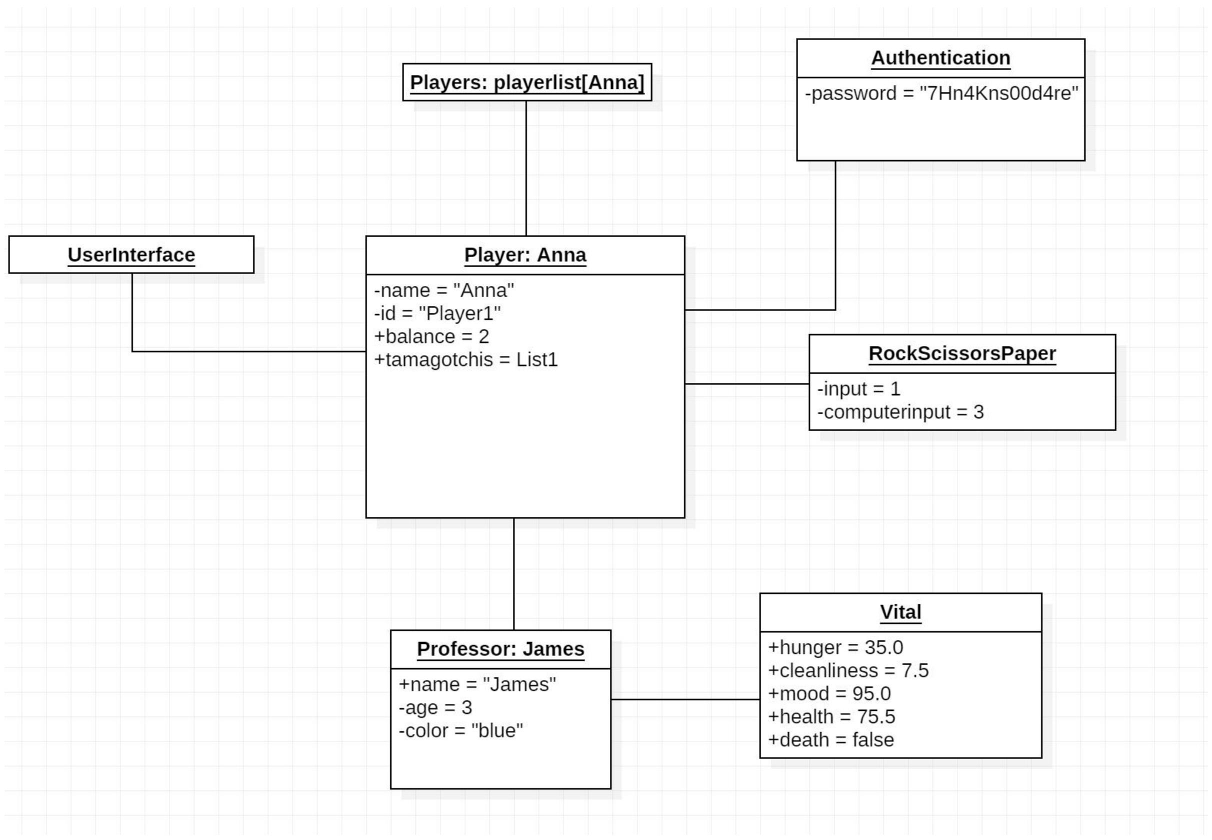
‘**Tamagotchi**’ is an abstract class inherited by ‘Lecturer’, ‘Professor’, and ‘TA’. Tamagotchi has its own name, age, and color. Each Tamagotchi has its own ‘**Vital’** which is directly relevant to its death. When the user executes some operations like ‘feed()’, ‘play()’, ‘callDoctor()’, ‘rest()’, and ‘cleanUp()’, the corresponding operations in ‘**Tamagotchi**’ class work. The operation ‘eat()’ which is operated during the ‘feed()’ operation lowers the ‘hunger’ of ‘**Vital**’. Likewise, ‘playWithPlayer()’ raises ‘mood’, ‘meetDoctor()’ raises ‘health’, ‘sleep()’ raises ‘mood’ and ‘health’, and ‘wash()’ raises ‘cleanliness’. The operation ‘die()’ checks whether the ‘death’ in ‘**Vital**’ turns into true and makes the Tamagotchi die.

‘**Lecturer**’, ‘**Professor**’, and ‘**TA**’ are subclasses of the abstract ‘**Tamagotchi**’ class. They have their own characteristics like different ranges of vitality.

‘**Vital**’ is composed in Tamagotchi. This class operates independently in order to make ‘**Tamagotchi**’ class clear. It has attributes like ‘hunger’, ‘cleanliness’, ‘mood’, ‘health’. Also, the ‘death’ is a boolean attribute to check whether the vitals are bad enough to die. The operation ‘printVital()’ is to show the Tamagotchi vital to the user. ‘getHungry()’, ‘getSick()’, ‘getLonely()’, and ‘getDirty()’ are the operations which change the vital attributes to become worse as time goes by.

Object diagram

Author(s): Benjamin de Vries



This figure represents the state of the system during execution. The situation is after the player has created an account, which has been added to the playerlist, and after a Tamagochi has been named and created. Now we are in game.

There is no LoginView object, because the Login screen has already been passed by the Player. The current player's name is "Anna" with id: "Player1". Her balance (financial) is 2, because she has already played the minigame 'RockScissorsPaper' a few times and won. Now she is in the state of playing another game, as you can see her input is 1 (representing Rock), with the computer's input being 3 (representing Paper).

Anna's Tamagochi is of the Professor type, whose name is James. James is age 3 as he has not been created long ago, and his color is blue.

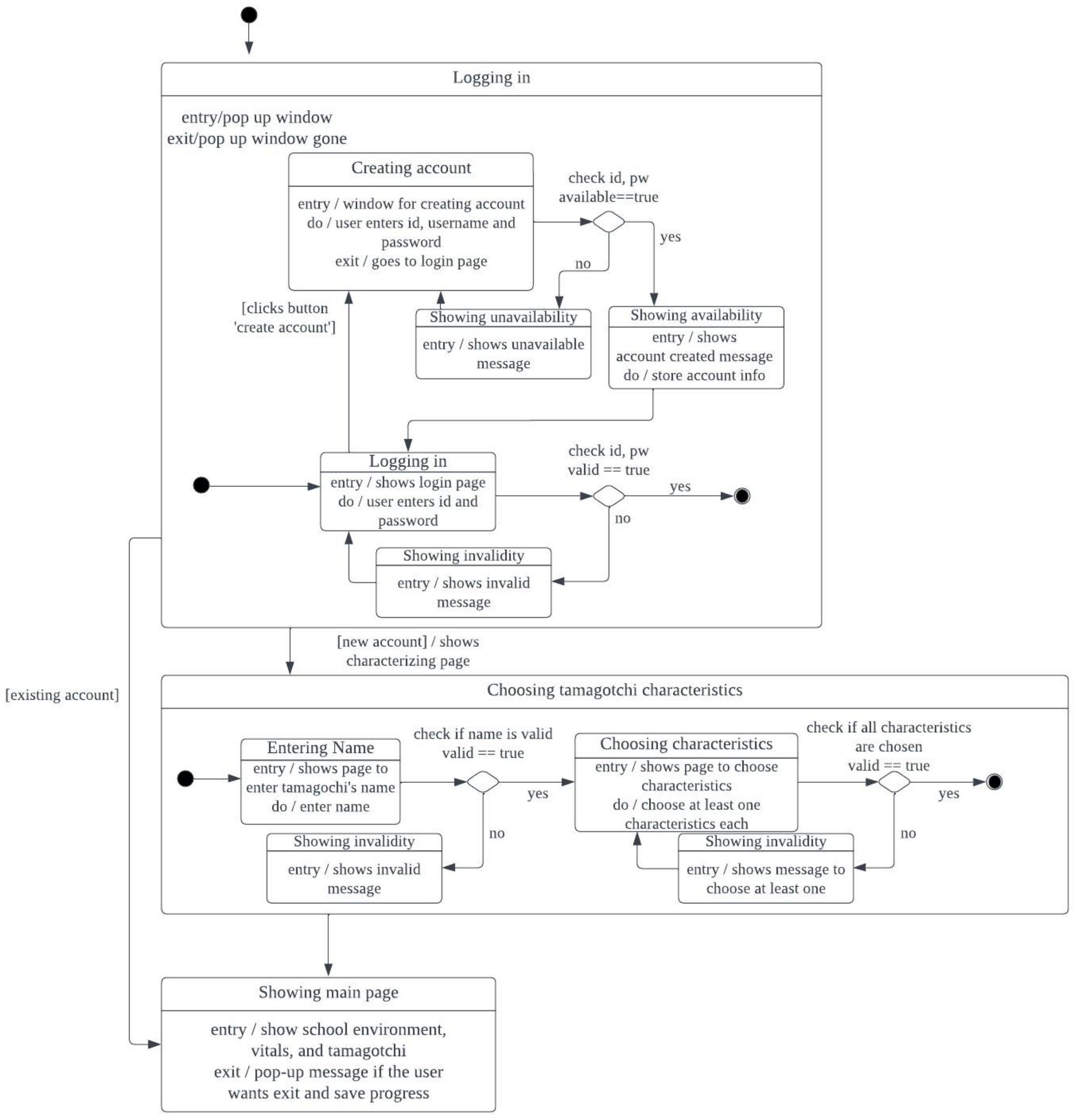
We can see his vitals on the right. In a range from 0.0 to 100.0, he is a little bit hungry, with a score of 35.0. The professor seems to be quite unclean, because his cleanliness score is only 7.5. It seems like he must be washed soon. But his mood is pretty good nonetheless, with a score of 95.0. His health is okay, but not optimal, with a score of 75.5 out of 100.0. And obviously he has not died at this point, since he is in the middle of playing RockScissorsPaper.

On the top right, we can see the player's password, which has been encrypted.

Finally, the UserInterface object exists on the left, but there are no parameters, as it has only functions, and the information to be displayed as meters exists in the Vital object.

State machine diagrams

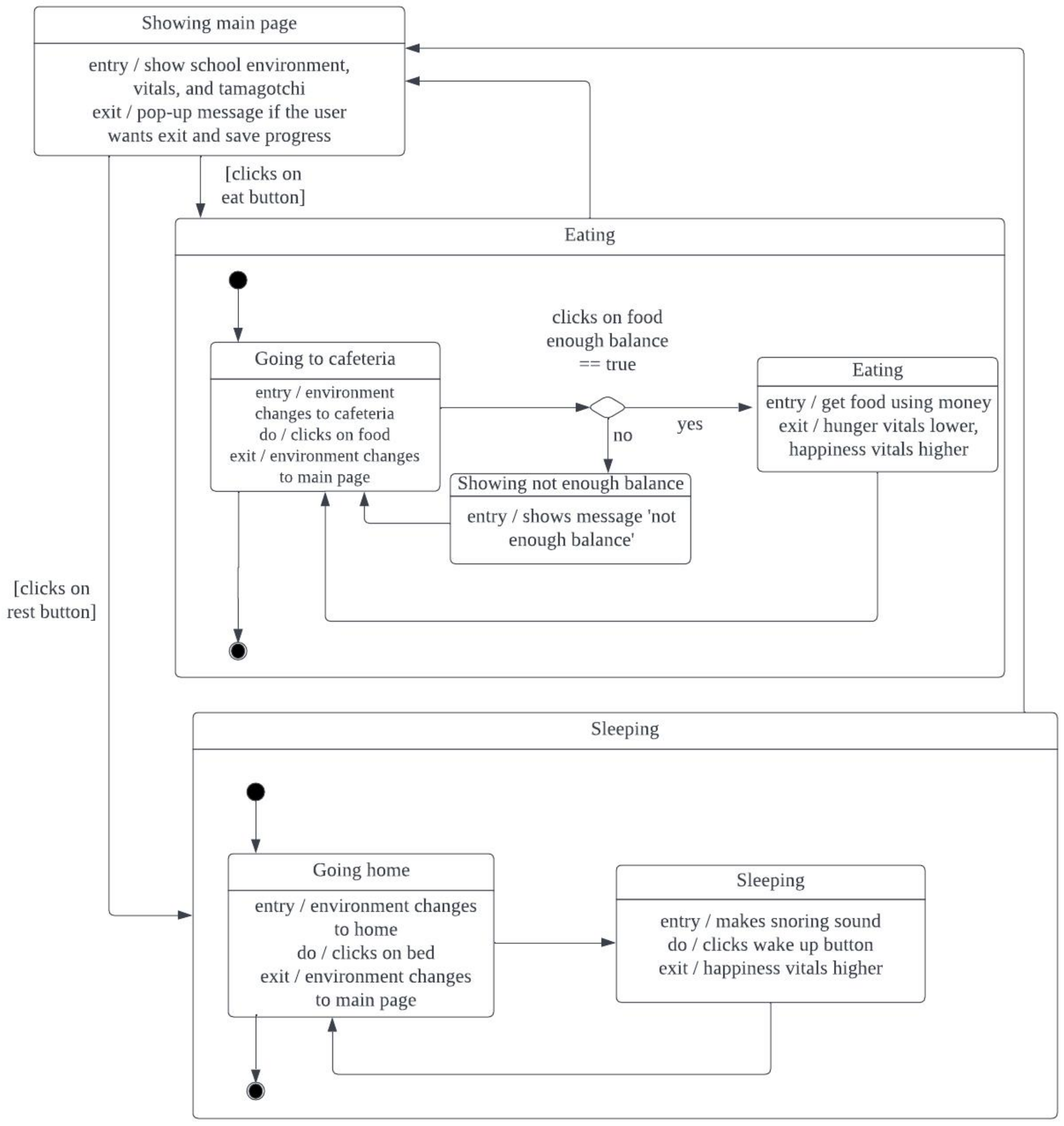
Author(s): Seoyeon Kim



Logging in
entry/pop up window
exit/pop up window gone
Creating account
entry / window for creating account
do / user enters id, username and password
exit / goes to login page
check id, pw available==true
yes
no
[clicks button 'create account']
Showing unavailability
entry / shows unavailable message
Showing availability
entry / shows account created message
do / store account info
Logging in
entry / shows login page
do / user enters id and password
check id, pw valid == true
yes
no
Showing invalidity
entry / shows invalid message
[new account] / shows characterizing page
[existing account]
Choosing tamagotchi characteristics
Entering Name
entry / shows page to enter tamagochi's name
do / enter name
check if name is valid valid == true
yes
no
Showing invalidity
entry / shows invalid message
Choosing characteristics
entry / shows page to choose characteristics
do / choose at least one characteristics each
check if all characteristics are chosen valid == true
yes
no
Showing invalidity
entry / shows message to choose at least one
Showing main page
entry / show school environment, vitals, and tamagotchi
exit / pop-up message if the user wants exit and save progress

The first state machine diagram represents the starting point of the MyTamagotchi program - creating an account and logging in, characterizing the tamagotchi and finally getting to the main page. First when the user enters the 'Login' state, a pop up window is displayed to login. If the user is new to the game and creates an account it checks the entered id and password to see if it is available for the user. If it is available, it goes to the login state. After the id and password is checked, it exits the state if it is valid.

If the user just made a new account, the characterizing page is shown and the user can enter the name and choose the color, type and characteristics. In each process the entered name and chosen characteristic is checked to see if it is valid, and if it isn't it shows the invalidity with a pop-up message. After this state is exited, it enters the Main Page which is the idle state for the game.


Showing main page
entry / show school environment, vitals, and tamagotchi
exit / pop-up message if the user wants exit and save progress
[clicks on eat button]
[clicks on rest button]
Eating
Going to cafeteria
entry / environment changes to cafeteria
do / clicks on food
exit / environment changes to main page
clicks on food enough balance == true
yes
no
Eating
entry / get food using money
exit / hunger vitals lower, happiness vitals higher
Showing not enough balance
entry / shows message 'not enough balance'
Sleeping
Going home
entry / environment changes to home
do / clicks on bed
exit / environment changes to main page
Sleeping
entry / makes snoring sound
do / clicks wake up button
exit / happiness vitals higher

The second state machine diagram is about the eating and sleeping process. The eating process resembles the feeding process in the original Tamagotchi game, but in our version the professor or the TA goes to the cafeteria to pick up food and eat. It starts at the main page, which is the idle state. If the user clicks on the 'eat' button, the tamagotchi goes to the cafeteria and clicks on the food. If there is enough balance, the tamagotchi gets food using the money, followed by a change in the vitals and goes back to the cafeteria state. If the user exits the cafeteria, it goes back to the idle state.

At the main page, if the vitals get lower the game sends an alert, for instance a sound to indicate that the tamagotchi is tired. In this case the tamagotchi can go home for a short nap or a sleep by clicking on the ‘rest’ button. It enters the Going home state and the environment changes to home - when the user clicks on the bed it enters the Sleeping state and goes to sleep. On the entry of this state it makes a snoring sound and stays at the state until the user clicks on the wake up button. As it exits the sleeping state the vitals gets higher. As the user exits from home, it goes back to the idle state.

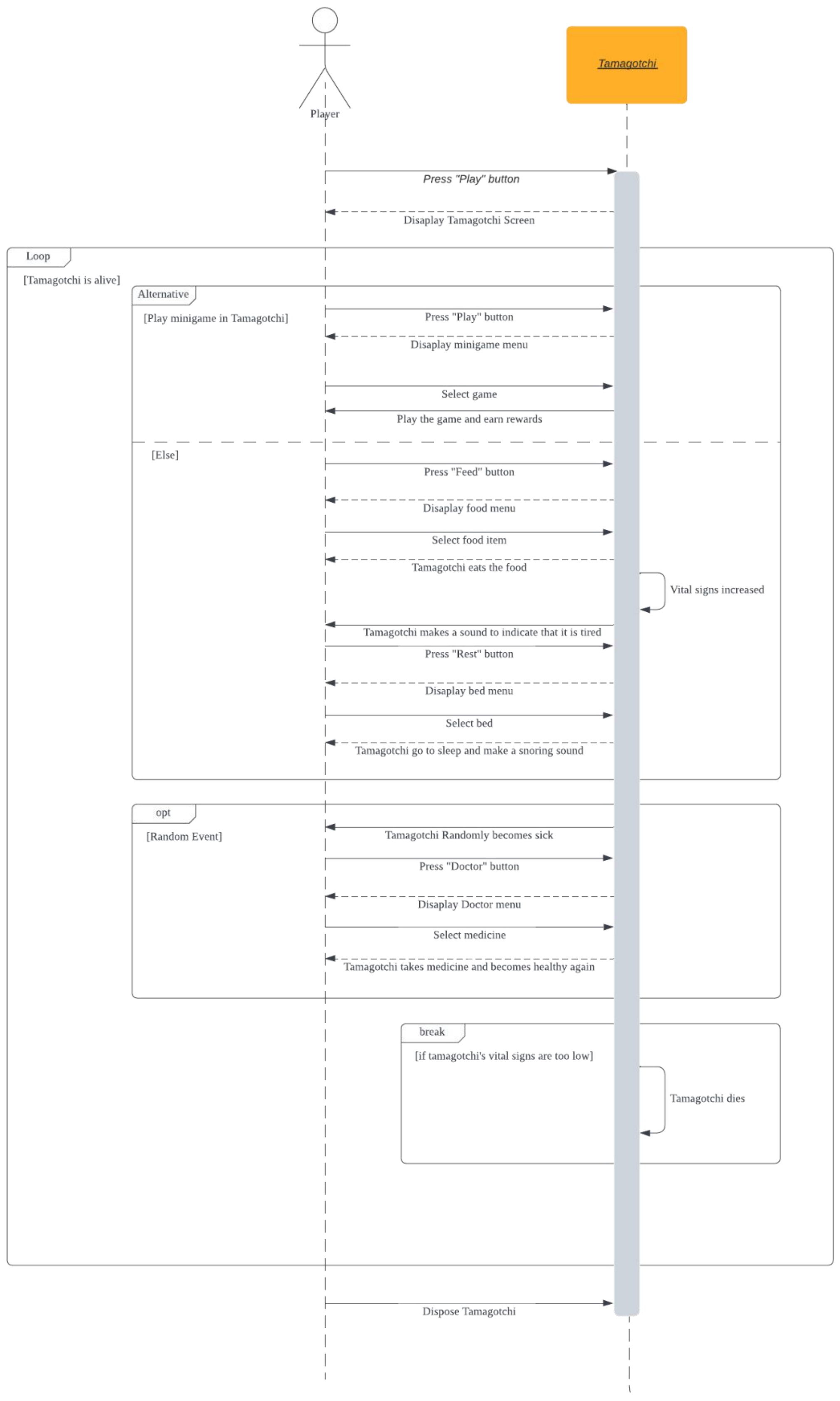
Other activities such as playing games, giving lectures, etc. has the similar relationship with the idle state. When the user clicks on a button on the main page, it enters the specific state and after the state is finished, the vitals gets lower or higher. If the user exits the state, it goes back to the idle state, which is the main page.

Sequence diagrams

Author(s): Yaxin Li

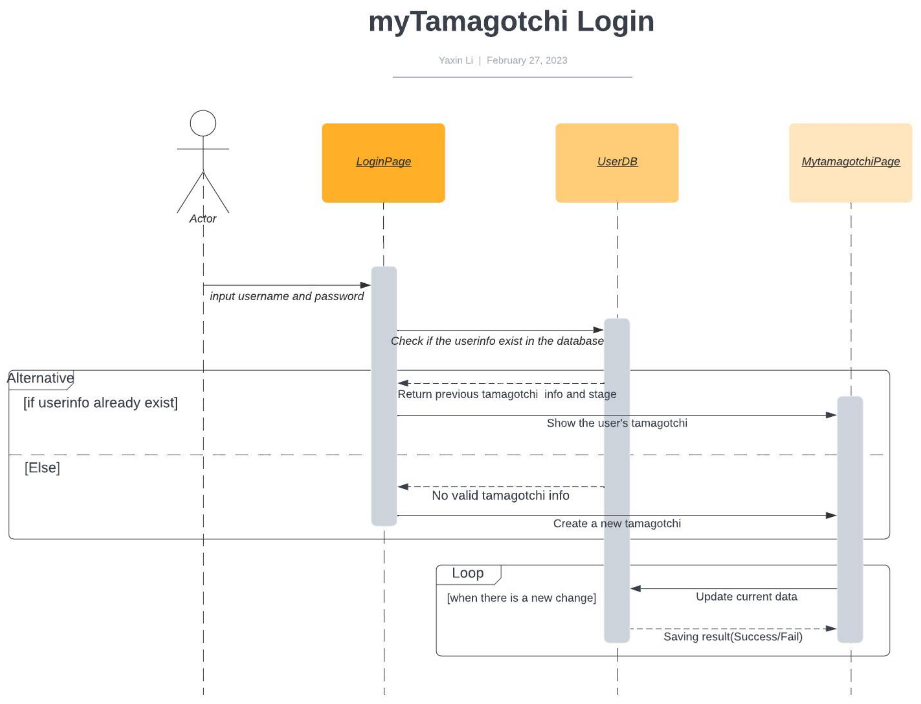
Play with Tamagotchi

Yaxin Li | March 9, 2023



In this sequence diagram, the player interacts with *the Tamagotchi* by pressing buttons on the Tamagotchi screen. So objects are *Player* and *Tamagotchi*. *The Tamagotchi* responds to *the player*'s actions by displaying menus, playing sounds, and performing actions like eating, playing games, and going to sleep. The diagram is continuously monitoring the vital signs, the game's performance in a loop once it starts and it will break when the vital signs are too low , in other words when tamagotchi dies.

Apart from the general description, the alt combined fragment in our diagram is used to represent the parallel message sequence for playing mini games with Tamagotchi. This fragment shows that *the player* can choose to tap on the "Play" button and select a game to play with the Tamagotchi at any time, without affecting the rest of the message sequence. You may also notice the opt combined fragment. That is used to express a random event where the Tamagotchi becomes sick. If this event occurs, *the player* can select the "Doctor" button and give the Tamagotchi medicine to make it healthy again. If the random event does not occur, the player can continue playing with the Tamagotchi without any interruption.


myTamagotchi Login
Yaxin Li | February 27, 2023
Actor
LoginPage
UserDB
MytamagotchiPage
input username and password
Check if the userinfo exist in the database
Alternative
[if userinfo already exist]
Return previous tamagotchi info and stage
Show the user's tamagotchi
[Else]
No valid tamagotchi info
Create a new tamagotchi
Loop
[when there is a new change]
Update current data
Saving result(Success/Fail)

This sequence diagram mainly describes the login operation of our user. This would be an optional part in our actual implementation. The three objects in the picture are the landing page, the database and the game page.

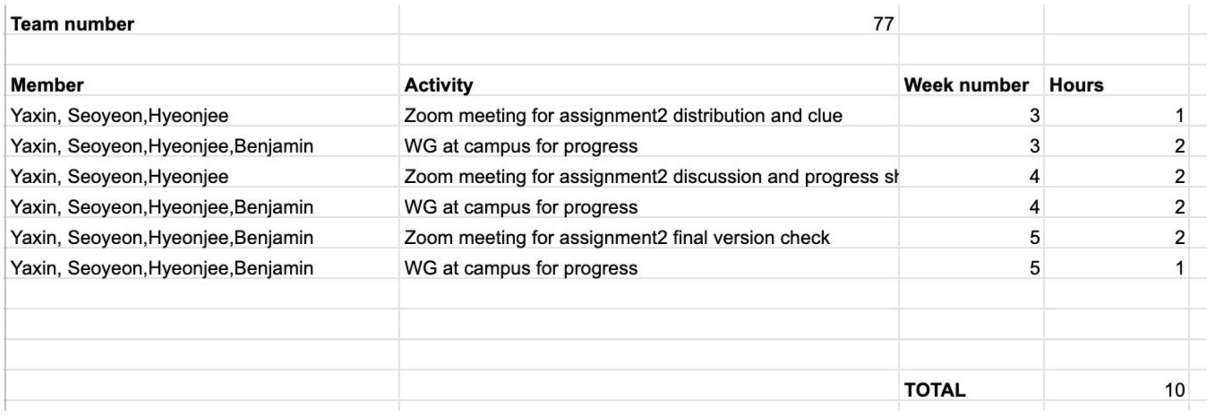
The user enters the user name and password in the login interface, that means, a message including the user name and password is passed to *the login interface*, and then *the login interface* transmits this information to *the database* for retrieval.

If the user's information exists in the database, in addition to the user name and password, the information in the database should also cover the saved player's last game progress. In this case, *the database* will return the information of the last game to *the login interface*, and then pass it to *the game interface*. In the meantime, the user will be redirected to it. At this point, the user can continue from the last game.

In another case, *the database* returns an error message, the user goes to *the game interface* to start a new game, and their username and password information is saved as a new entry in the database.

The under loop indicates that any changes in the game will update the database information immediately to prevent data loss. The database always gives feedback on whether the data is saved successfully or not.

Time logs



Team number	77		
Member	**Activity**	**Week number**	**Hours**
Yaxin, Seoyeon,Hyeonjee	Zoom meeting for assignment2 distribution and clue	3	1
Yaxin, Seoyeon,Hyeonjee,Benjamin	WG at campus for progress	3	2
Yaxin, Seoyeon,Hyeonjee	Zoom meeting for assignment2 discussion and progress sh	4	2
Yaxin, Seoyeon,Hyeonjee,Benjamin	WG at campus for progress	4	2
Yaxin, Seoyeon,Hyeonjee,Benjamin	Zoom meeting for assignment2 final version check	5	2
Yaxin, Seoyeon,Hyeonjee,Benjamin	WG at campus for progress	5	1
		TOTAL	10