Second Iteration Report

Team

TYTS

Members

Xi Lin
Lixing Zhang lz2636
Ruolin Jia
Tianyang Yang
Benhao Chen

Github

https://github.com/hreatx/TYTS

CI files & directory

Pre-commit config: TYTS/.pre-commit-config.yaml
Pre-commit report : TYTS/CI_report/pre_commit_report
Post-commit config : TYTS/.travis.yml
Post-commit report: TYTS/CI_report/post_commit_report

Change on our CI

Previously, we do static analysis in both pre-commit and post-commit phase. We find this redundant. We decide that the smelly code should be prevented to be committed to the shared repo. Therefore, we remove the static analysis from post-commit and we only do in pre-commit. Post-commit is responsible for the unit test.

Test Plan

1. TestEventController

Directory:TYTS/testBuddingWidget.py

Test Suit 1.1
In this unit test, we test the money value. The money can be updated by users' smile event. We intend to test whether our software can handle different money values and disallows illegal values updated by users
The input is the current amount of money.

Input: The updated money

Equivelent Class A: Valid Value

Boundary Condition:
Value >= 0

Example:
1,0.5,100

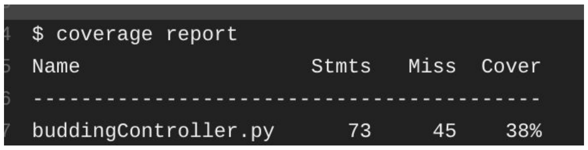
Expected Output:
This amount is valid and the money can be updated

Equivelent Class B: invalid Value
Boundary Condition:
Value < 0

Example:
-1, -1.5, -100

Expected Output:
This amount is invalid and the money should be set to 0.

Code Coverage Report
We measure the test coverage on buddingController file. This is the file that is tested by our unit test class.



```
$ coverage report
Name                   Stmts   Miss  Cover
----------------------------------------
buddingController.py      73     45    38%
```

2. TestBuddingWidget

TYTS/testEventController.py

Test Suit 2.1
Each budding's image is indexed by an integer. It can update the image of Budding by incrementing the index if the user clicks the Budding. In this unit test, we test the index is never out of bound.

If we have 10 images for Budding. The index is 0..9.

Input: Current Image Index

<u>Equivalent Class A: Indices except for the last index</u>

Boundary Condition:
Index < len(all_images) - 1

Examples:
If we have 10 images with indices from 0..9. This class contains:0..8 (Except for 9)

Expected Output:
The index is increased by 1

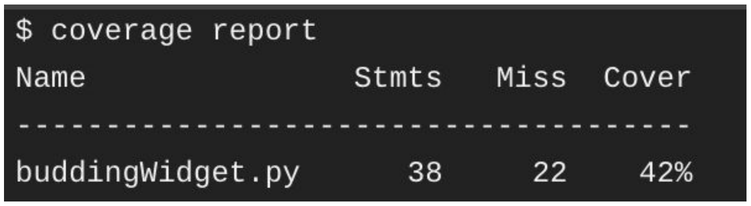
<u>Equivalent Class B: The last index</u>

Boundary Condition:
Index == len(all_images) - 1

Examples:
If we have 10 images with indices from 0..9. This class contains only 9

Expected Output:
The index is reset to 0

Code Coverage Report
We measure the test coverage on buddingWidget



```
$ coverage report
Name                Stmts   Miss  Cover
---------------------------------------
buddingWidget.py       38     22    42%
```

Future Testing Plan

We will test the store in the system in the next iteration

Use Case Documents

Use Case 1 - Create Account

Title:
Create Account

Role
Player

Description:
The user can create an account in order to log in and play the game

Basic Flow

1The system represents the account sign up window
User input his username and password
If the username satisfied that: the username is not yet used by other users and the username and password must have at least one characters, go to AF 1
If the username does not satisfy the condition, go to AF2

Alternative Flow
AF1

1. Promote the success message
2. End the use case

AF2

1. Reject the sign-up
2. Go back to step 1 in the basic flow

Post Condition
The user's account is created in the database.

Use Case 2 - Sign in

Title
Sign in

Role
Player

Description:
The user can sign in with his account

Precondition
The user has an account stored in the database

Basic Flow

1. The system presents the account sign-in window
2. User input his username and password
3. If the username and password match one account in the database, go to AF1
4. If the username or password is incorrect, go to AF2

Alternative Flow
AF1

1. Promote the success message
2. Present the game main window
3. End the use case

AF2

1. Reject the sign-in
2. Go back to step 1 of the basic flow

Post Condition
The user signs into the system

Use Case 3 - Smile

Title
Smile

Role
Player

Description:
The user can smile to the camera to make virtual money

Precondition
The user has signed in the system.

Basic Flows

1. The user clicks the Smile button
2. The system turns on the camera
3. The user smiles to the camera
4. The system turns off the camera
5. The system captures the user's smile and computing the smile score
6. If the smile score is greater than 0.5, go to AF 1
7. If the smile score is below 0.5. go to AF 2
8. End the use case

Alternative Flow
AF1

1. The system increases the user's cash by $10

AF2

1. The system decreases the user's cash by $10

Post Condition
User's cash is changed

Use Case 4 - Change Picture

Title
Smile

Role
Player

Description:
The user can click Budding to change its appearance

Precondition
The user has signed in the system.

Basic Flows

1. The user clicks the animation of Budding
2. If the user is level 1, a sad Budding should replace the current Budding
3. If the user is level 2, a neutral Budding should replace the current Budding
4. If the user is level 3 or above, a happy Budding should replace the current Budding
5. End the use case

Alternative Flow
NA

Post Condition
Budding’s appearance is changed

Use Case 5 - Play Music

Title
Music

Role
Player

Description:
The user can click on “voice” to play or pause music

Precondition
The user has logged in the system.

Basic Flows
1. The user clicks the animation of Budding
2. If the user is level 1, a sad music should be played
3. If the user is level 2, a neutral music should be played
4. If the user is level 3 or above, a happy music should be played
5. End the use case

Alternative Flow
NA

Use Case 6 - Store

Title
Store

Role
Player

Description
The player can access the store and use the virtual money to buy food to feed Budding

Frequency of use
All players will use the store to feed their budding pet because it is the only option

Triggers
The player clicks the “store” button

Precondition
The user has an account and the user sign in to the account

Basic Flow
1. A pop-up window appears and the player can see all the items in the store (see AF1)
2. The user clicks the corresponding button to purchase the item (see EX1)
3. The money is deducted from the user's balance and the energy of budding increase
4. If the current energy is larger than 100, the budding get a level up (see EX2)
5. If the player clicks the "store" button or the red cross on the left-up corner, the pop-up window disappears.

Alternative Flow
AF1
1. The player wants to close the store without trying to buy anything
2. Go to step 5

Exception Flow
EX1 The player doesn't have enough money
1. The transaction will be denied and the status of Budding will not change

EX2 The budding reaches the maximum level
1. The transaction will be denied and the status of Budding will not change

Post Condition
The status of Budding of that player is saved to the database

Additional Description

We use Microsoft facial detection API, which needs a subscription key. We stored the key in a local file called "secret.json", but not uploaded it onto the GitHub. We can provide the "secret.json" file if needed.