Project Deliverable 5 – Testing

Test Plan

Prior to the implementation phase of this project, our team decided it would be best to do extensive manual testing of the backend game logic and android integration throughout the process. This was because the game itself relied on a random number generator in order to add randomly add cards from the deck to the players' hands. We had agreed to save automated unit testing and user testing for after we had completed the full implementation of the minimum viable product and use manual testing and code inspection during the development process.

One type of testing we did during the development process was manual testing, which included integration and end-to-end testing. For instance, after adding a couple lines of code, we tested to see if our product was still running and if the code worked as intended. This included extensive use of System calls such as prints to display many of the game actions on the console (including checking the cards in the player hand/computer hand, checking scores, checking if pairs were made, etc.). Additionally, after getting a basic product working, we tested various edge cases to see if we handled as much of the possibilities the user can run into while using the product as we can. After getting the product to run on an android phone, we did more manual testing on the speech aspect to see the various input words the built-in API would create, as this is necessary for handling homophones such as 'two', 'too' and 'to'.

As for integration testing, we similarly had to manually test to make sure each newly added component cooperated with the rest of the already existing system. This meant launching the app and using it many times to see if newly added components integrated as expected. This also called for the extensive use of Android Studio's Logcat when we began working specifically with the mobile app integration with the speech-to-text capability and of the backend game with the UI. In terms of the UI, we also manually tested to the system to make sure the app would transition to the appropriate screens/activities by simply making use of the speech-to-text component and testing out the app.

A final type of manual testing that we performed included end-to-end testing, which meant launching the app from start, and going through all the different activities and playing the whole game until game over, while testing various edge cases in what the user spoke. Overall, manual testing played an important role throughout all stages of the development process of our Go Fish mobile app. As for automated testing, we had decided to only do so on the backend game components and not the Android app itself (further details on automated testing below).

Technology

The main tool used for testing was Android Studio's run function. Using this, we were able to connect an Android device to our computers with USB and test the functionality of the game throughout the various steps of implementation. Whenever a new section of code was added to the game, the game would be run on the Android device and tested to see whether

the code worked as intended, as well as if the new code was compatible with the device (i.e. If code caused any crashes to occur).

For automated testing, we used Java's JUnit library to test the functionality of various units of code. It was the simplest method of implementing automated testing, as it allowed us to test each method separately to confirm that it works correctly, even for invalid parameters of inputs.

Description of the Files

The only automated testing file we made was PlayerTest.java, created using JUnit. It tests the Player class, as well as the Computer and User classes, which extend from the Player class and vary with only one method. The file tests each method in the Player class, including getSizeHand(), getScore(), getHand(), checkPairs(), getCard() [different versions in Computer and User], and goFish() [a verson specifically for testing was put into the Player class]. Each method has multiple tests to check multiple possible scenarios, especially methods such as checkPairs(), which create changes in the instance variables of the Player objects. One method, addToHand(Card c), which adds a specified card to the player's deck, was added to the Player class to assist with the unit tests.

As a result of the way we created the main Game class, there was no simple way to test the functionality of this class using JUnit testing. Instead, we decided manual testing was sufficient for this class.

Contributions:

- Ivan: 100%

User Testing

We asked a total of four people to test our product, three of which were told to play through a round without any assistance and one which was given a set of goals to achieve. Below is their feedback:

<u>No Assistance</u>

- Person A
    - Positives
        - Graphics and mechanics work well
    - Suggestions/Issues found
        - There should be a visible count of number of cards left in the deck
        - There should be some sort of timer to end the game if it takes too long
        - Each theme should have its own background
        - Voice recognition. heard "pair" instead of "pear" when playing with fruits
- Person B
    - Positives
        - Great concept

        - Clever way of making it feel like you're playing against a person instead of just an AI
    - Suggestions/Issues found
        - Shouldn't need to press the microphone button every single time
        - Asking in the form of "Do you have any jacks" doesn't work (plurals)
- Person C
    - Positives
        - The app provides a functional layout with an attractive color scheme
        - The diversity of categories that would not be readily accessible with physical playing cards allows for the player to have a unique experience with each game that is appropriate for both children and adults
        - Had a positive experience with the app and found it to be without much fault
    - Suggestions
        - Potential option to type in questions if voice option cannot be utilized
        - Audio feedback when a card is selected so children or those with speech impairments can hear and confirm their selection
        - A way to pause and save a current game session to continue later on

<u>Goal-oriented testing</u>

- Person D
    - Provided test script:
        - Open the how to play page
        - Choose the Animals theme
        - Ask for a card you don't have in your hand (I suggested "snake")
        - Play until you or the computer have 5 points
        - Exit and change the theme to Colors
        - Play to the end
    - Feedback:
        - Easy to find out how to navigate through app
        - Should be clearer about which "Go Fish" is for the player and which one is for the computer
        - The time between turns can feel too long sometimes