Tong-EATS: A Card Game

Prepared by:
Antonio G. Callanta III
Christian Jerald Y. Chan
Faine Mabry A. Fonseca
Carlo D. Mendoza
Carlos Enrique P. Nava

Submitted on: December 11, 2017

CS 123 B Introduction to Software Engineering

A. Project Description

Tong-EATS is a simple 3–4 player card game software intended for children (ages 6+), but can also be played by adults. The game would be a mobile application that uses the internet to connect the players to each other. It will also have a database for the ranking system, which stores the ranking of each player based on their points and player statistics. The purpose of the game is to teach children proper nutrition in a fun and entertaining way. More specifically, the things that the players of the game should learn are the following: (1) different basic categories of healthy food (go, grow glow), and (2) balancing these three kinds of food properly.

Below are the formalized mechanics of the game:

A. This game is made for 3 - 4 players. Each of them gets five (5) cards at the start.
B. First person to drop a card decides the “meal” to be made by the players. Each card defines the meal components it requires, and meal options are written on the base card.
C. The players then drop a valid card on their turn. Valid cards have weight less than or equal to the remaining total weight in each category.
D. If the player has no valid card, he / she passes and draws a card from the deck.
E. If a player puts down a card and all others pass, the player drops another card to start a next meal.
F. Also, if a player completes a previous meal, he / she starts a new one by playing another card.
G. The first person to play his / her entire hand wins!

The game will be implemented as an android application. Players will be able to play with other players by connecting via the Internet. Players will be able to make an account that would hold their ranking. As a mobile application, it will accessible to parents who want to teach their children about nutrition in a different yet entertaining way. Moreover, the gameplay is fast-paced yet repeatable to cater to children’s short attention-span and competitiveness.

B. Team Organization

The team organization that the group used to produce this system in the limited time given is the Agile Process Team. They chose this type of team organization so that the team members could work at different times as small groups of two or three (since the team as a whole does not have a lot of mutual free time) but all the members know the functions of the code that each member makes. Aside from this, pair programming can allow the pairs to learn from each other and correct each other's mistakes.

The programmers who developed this software are Antonio "Tomy" Callanta III, Mabry Fonseca, Carlo Mendoza, and Carlos "Chino" Nava. Programmers were paired together and were treated as one unit. Each unit worked on a different task but the scope and requirements were discussed beforehand. This was to ensure that the code made by the two pairs complement each other once combined together. Once both units were done with their code, the other unit formed test cases for the code of the other unit. Afterwards, the code was combined and another set of tasks were given to each unit.

The documentation officer for the software is Christian Chan. The Documentation Officer had two jobs. His first job was to remind and make sure that the programmers are producing code based on the agreed upon requirements. He routinely checked upon both units and used this gathered information as basis whether both set of codes will match each other. This was when both units have yet to exchange codes and develop test cases. He acted to minimize the problem and make the code specific. His second job was to be an effective communicator between pairs. All reports and updates were sent to him so he can relay information to the other group while each unit is working in parallel.

Since the group chose to do Agile Processing, they held stand-up and follow-up meetings whenever possible. The stand-up meetings took only 15-20 minutes each for each member to raise their concerns and give a brief explanation of their progress. On the other hand, follow-up meetings were used to address the problems and get ideas from each other on how to solve the said problem. Aside from this, they held online conversations to keep each member in check and see if progress is being made. These short yet multiple meetings allowed each group member to be constantly updated despite not having a lot of mutual free time.

C. Functional and Non-Functional Requirements

FUNCTIONAL SPECIFICATION		
FUNCTIONAL REQUIREMENT	DESCRIPTION	PRIORITY
1. Login System	A feature that allows the player to log into the game using his or her account	1
2. Main Menu a. Start new game b. Join existing game c. View player statistics d. Manage options e. Get help f. Exit game	A feature that allows the player to navigate to different panels of the game	1
3. Multiplayer Lobby	A feature that allows players to view the other players' names while waiting for the game to start, and exit the multiplayer lobby	1
4. Tutorial	A feature that allows players to learn the mechanics of the game through an interactive simulation	2
5. Card Game a. View Player Hand	A feature that allows a player to see the current	1

b. View Current Meal Pile c. View Opponents' Remaining Hand Sizes	menu (and thus, see the card restraints), see their current hand of cards, select a card or pass on their turn, view the card selections of other players, the game's progression, and view the remaining number of cards on each player's hand	
6. Scoreboard	A feature that allows a player to view his or her opponents' names and their corresponding scores	3

NON-FUNCTIONAL SPECIFICATION		
NON-FUNCTIONAL REQUIREMENT	DESCRIPTION	PRIORITY
1. Security	The system should be secured with a login feature. Each account has a different profile and player statistic. This security allows them to play progressively with a single account.	1

2. Game Documentation	The game system should be well documented in every aspect. The code, as well as the game itself, should be easily understandable with minimal verbal explanation.	2
3. User Interface	The user interface should be clean, understandable, and easy to use for players. Game restrictions (such as which cards can and can't be played) should be clear to players.	1
4. Platform	The system should be in the form of a mobile application.	1

D. Iteration Plan

ITERATION	PLANNED	ACTUAL
1	● Create main menu ● Create initial prototype of game logic and randomization of selection ● Basic game assets	● Procure primary game assets ● Procure main menu prototype ● Procure final game screens ● Formalize

	and card skeleton	mechanics

2	• Update game script to include player actions
	• Implement basic AI opponents
	• Update game script to include win conditions and game resolution
	• Finalize all card values for the 30 playable cards and implement them in the game
		• Finalize menu screens
		• Procure card concepts
		• Procure cards assets
		• Procure turn scripts:
		  ○ Player and AI turns
		  ○ Card selection
		  ○ Win condition

3	• Implement sounds • Implement animation • Implement game aftermath / result, i.e. wins, losses, etc. • Implement help function • (optional) Implement multiplayer	• Create use case diagram • Input card values to pile • Input card images to pile • Scale game to better fit • Player pick requirements • AI picking following requirements • Resetting meal when no one can play • Debug history of actions

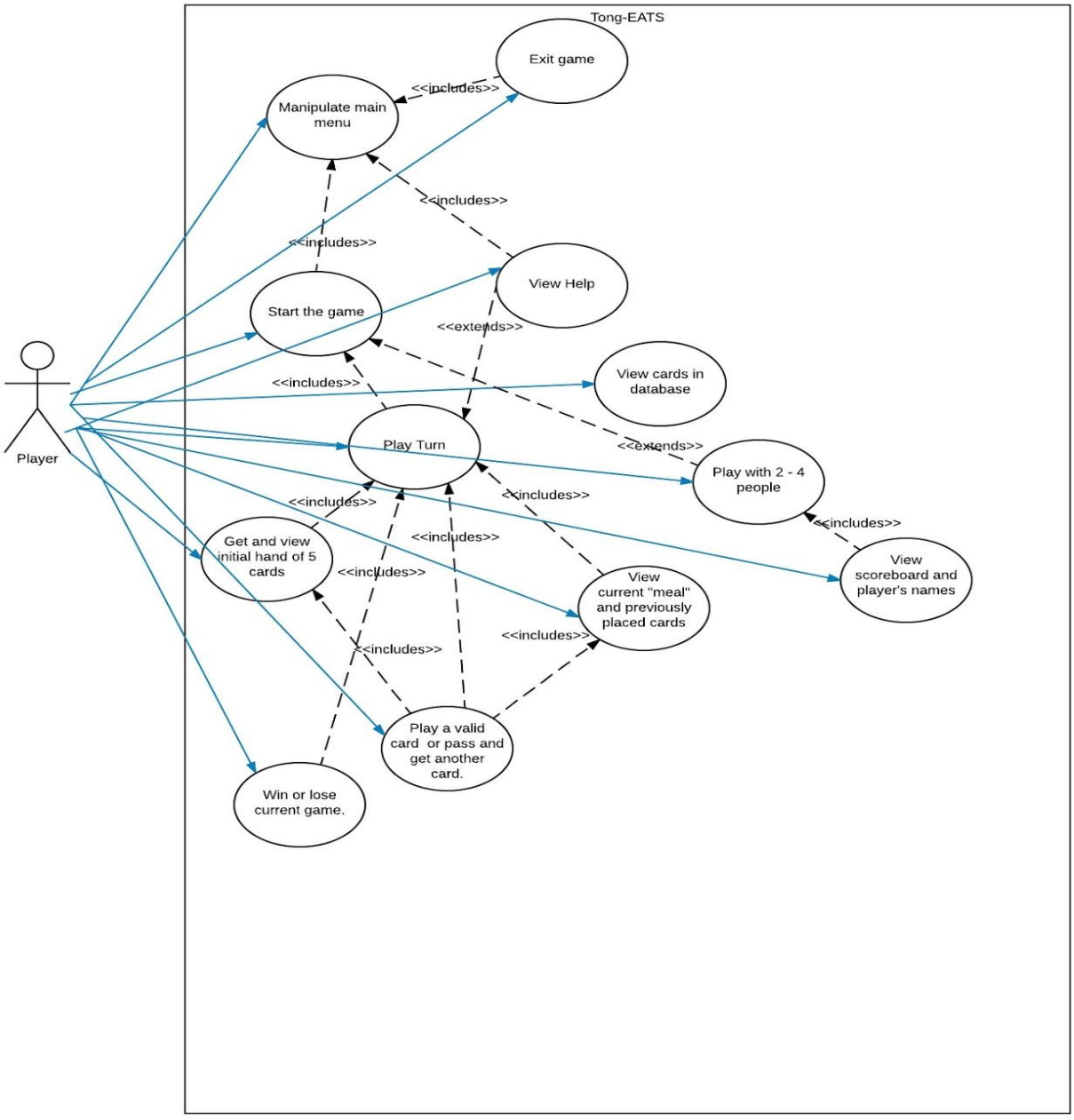
		• Prototype of card select confirmation
4	• Implement sounds and music • Animate assets • Implement WIN/LOSE progression • Implement repeated play (“play again / back to main menu”) • Implement and fill up help function • (optional) implement multiplayer system	• Implement passing and drawing from deck • Added more cards to game • Delayed dropping of cards • Implement WIN/LOSE progression • Implement repeated play (“play again / back to main menu”) • Implement and fill up help function • Implement onscreen keyboard for player name input

E. Requirements Models

A. ERD - N/A

B. DFD - N/A

C. Use Case Diagram



D. Use Case Description - N/A

E. Activity Diagram - N/A

F. Working Code

(see GitLab URL: https://gitlab.discs.ateneo.edu/CS123_B3/Tong-EATS.git)

G. Lessons Learned

A. Tomy Callanta

Creating functional software requires detailed planning. I realized that having the end in mind is very important in committing to a task such as this, and dividing this end goal to smaller and more manageable tasks makes its implementation more clearer to the team. The group often caught itself in point of development where a certain use case was not foreseen or that the task was too complex to finish in the time we set for ourselves. Holding meetings and communicating these issues to each other resolved them very quickly.

B. Christian Chan

I like creating these documents to hone my writing and design in documentation skills. However, I also feel that I have not done my job fully well because of not showing up at the meetings. Also, I feel that being a documentation officer slightly defeats the purpose of being a CS major: to learn programming and business at the same time. However, I am willing to learn where I lack.

C. Mabry Fonseca

There are steps in software engineering, most especially requirements design, that may seem unnecessary and tedious but it turns out to be useful. It allows the developers to keep their design and development on track. We had a tendency to focus on the optional or less prioritized features which made it seem like we did not have progress in our development. However, we simply did not list these features in the User Stories or it was a feature to be implemented in the latter half of the development. Therefore, it helps to make sure that

development goes through the desired process in order to meet the set requirements on the documented deadline or date.

D. Carlo Mendoza

Although the Agile Process Team uses a democratic approach which allows free flow of information and enables the members to do whatever they want with the code, it may be ineffective sometimes since the workload tends to be divided among its members according to their skill level. If we were to choose a team organization where there is one in charge of directing the work and monitoring the members, then maybe we can get more things done in a disciplined way.

E. Chino Nava

I should really learn Git. We should all really learn Git. Also, breaking down the project into multiple parts to be worked on by different people and integrating after is harder than I expected before starting the project.