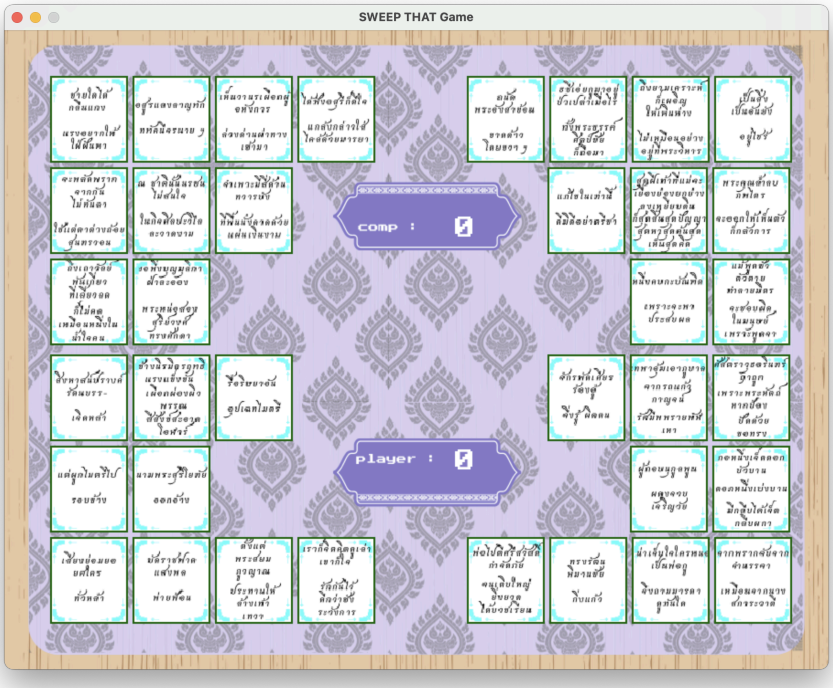
Sweep that !

1. Project Overview

Sweep That! is an immersive memory-matching game that blends traditional Thai poetry with engaging gameplay. Players test their memory by matching beautifully designed cards featuring excerpts from classic Thai literary works such as นิราศภูเขาทอง *(Nirat Phukhao Thong),* ลิลิตตะเลงพ่าย *(Lilit Taleng Phai),* พระอภัยมณี *(Phra Aphai Mani), and* นิราศเมืองแกลง *(Nirat Mueang Klaeng). The game features a streamlined 36-card layout for faster-paced matches, where players compete against the computer to collect all 18 cards assigned to their side.*

This project's game will contain 36 cards in total, 18 for each side of players. After the game ends it will show the stats of the player in this round along with the stats of the player all the round before and analyze it at which point you need to develop, such as you need to sweep the card more quickly! Or maybe you need to memorize the card and their position more correctly.

Example of the board :



2. Project Review

Card matching game : card matching game has a not so similar but useful system to Karuta game, so I can use this game's project as a reference and develop it into thai Karuta card game 'Sweep that!'.

3. Programming Development

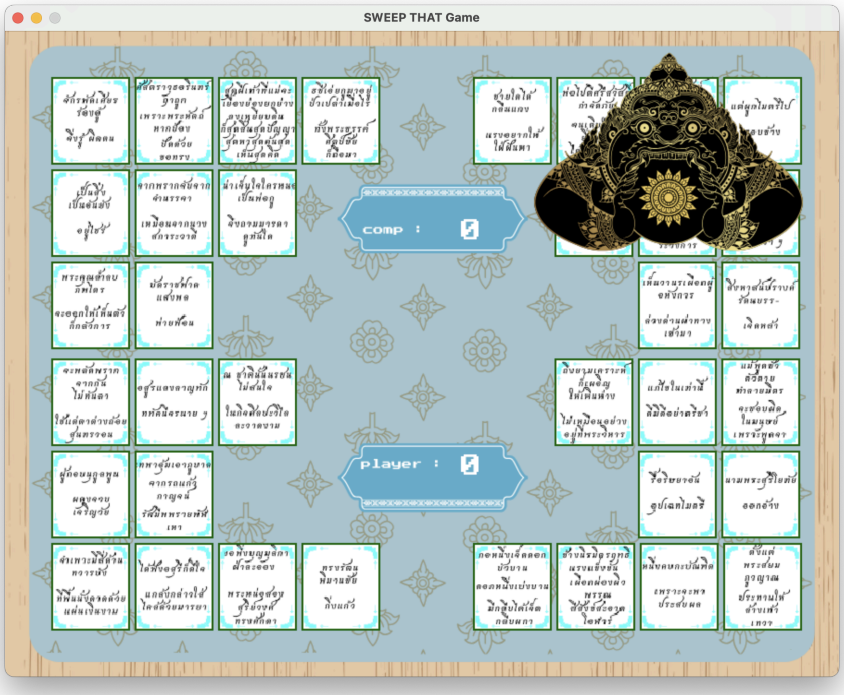
3.1 Game Concept

Sweep That! reinvents the traditional Japanese Karuta card game with a uniquely Thai twist, blending fast-paced memory gameplay with the beauty of classic Thai poetry. Players engage in an exciting battle of wits, matching cards featuring iconic Thai literary works like นิราศภูเขาทอง *(Nirat Phukhao Thong),* ลิลิตตะเลงพ่าย *(Lilit Taleng Phai),* พระอภัยมณี *(Phra Aphai Mani), and* นิราศเมืองแกลง *(Nirat Mueang Klaeng).*

Core Gameplay:

- ***36-Poem Card Deck****: A carefully curated selection of Thai poems replaces the original 50-card setup, optimized for balanced and strategic play.*
- ***Head-to-Head Memory Duel****: Players compete against the computer to be the first to "sweep" all 18 of their assigned cards from the board.*
- ***Audio-Visual Matching****: Listen to narrated poem excerpts and race to identify and select the corresponding card before your opponent.*
- ***Two Distinct Modes****:*
    - ***Normal Mode****: Pure memory challenge, match poems against the clock.*
    - ***Hard Mode****: Features Rahu (*ราหู*), a mythical serpent that devours random position of the board, forcing players to rely on their memory as some part of the board is invisible.*

Example of RAHU (reference from ราหูอมจันทร์ -> ราหูอมการ์ด*) :*



The selling point of this game is the adaptation of Karuta to thai styles card game. This game is a great way to learn about Thai poetry while having fun and improving your reaction time.

3.2 Object-Oriented Programming Implementation

1. *sweepthat_classes.py*
    - *Board class : this class will be used to create the board of the game and place the cards and return the card info for data collection.*
    - *Sound class : this class will use to collect all poem's reading sound.*
    - *Piece class : this class will store all card information and draw the cards on the board.*

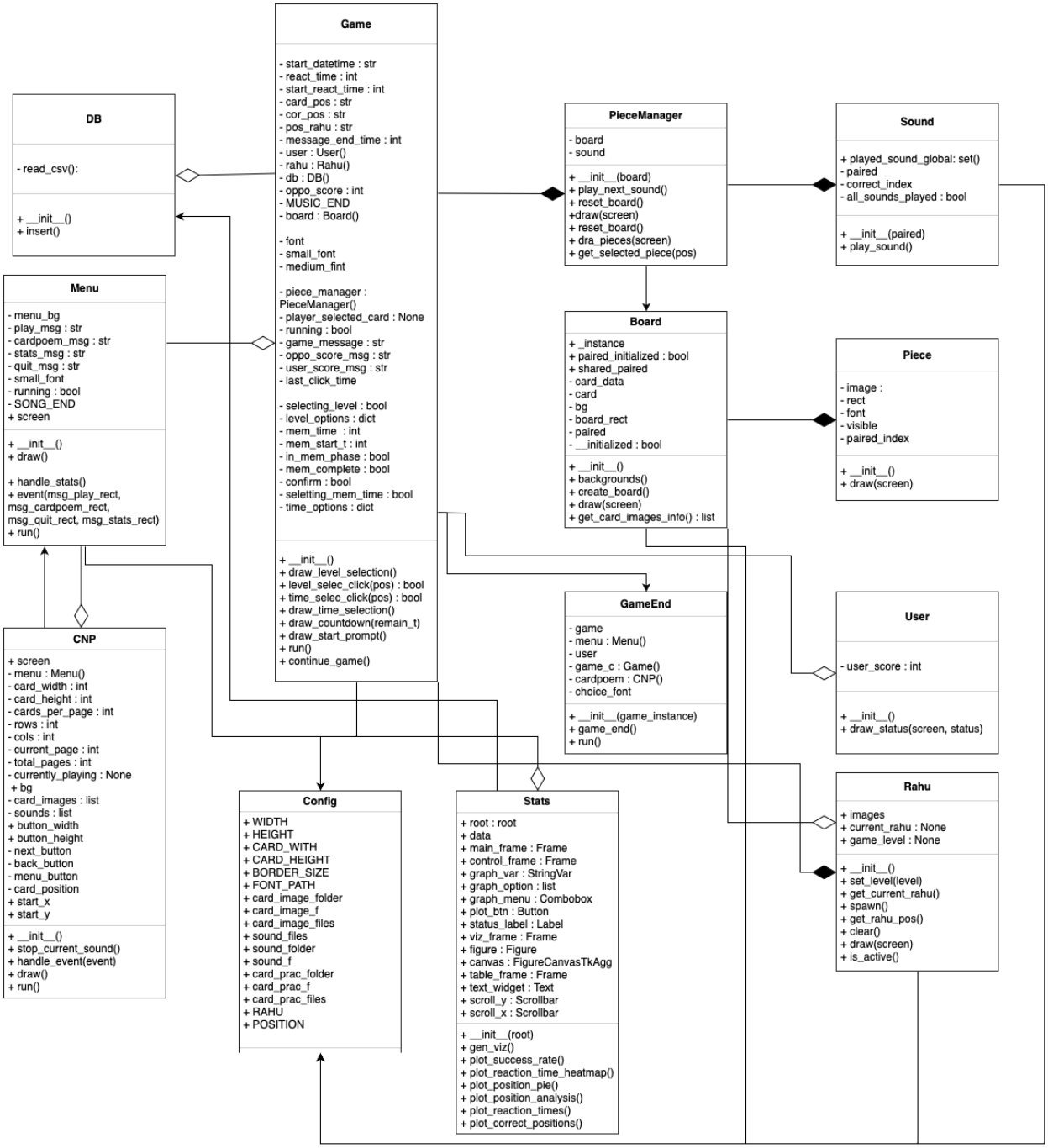
- *User class : The User class is used to store and display user UI that will show the score of the user from 18 to 0.*
- *Rahu class : this class will contain all information about rahu in hard mode such as send the position of the rahu to store in csv file, spawn, clear and draw the rahu in random position.*
- *PieceManager class : this class is responsible for managing the card and board, resetting the board for new games, playing the next sound and returning the index of the correct card.*

2. *sweepthat_config.py*
    - *Config class : Config class will store all basic game data such as size of the board, size of the card or the card's thai poem to use in Card class.*
3. *sweepthat_game.py*
    - *Game class : Game class is the main class of this sweep that! Game. This class will be used to activate the game with the run function along with a lot of helper functions such as draw_countdown that will draw the memorized time that user selects, draw_level_selection that will draw the level for the user to select.*
4. *game_end.py*
    - *GameEnd class : this is the class that will display the ending of the game, either the winner is user or opponent, and give users a choice if they will start a new game, go to the menu page or listen to the poem to practice in card and poem choice.*
5. *sweepthat_menu.py*
    - *Menu class : The menu class will be the class that represents the menu window and leads users to the different widget based on their choice, Play, Card and Poem, Stats or Quit.*
6. *Sweepthat_cardpoem.py*
    - *CNP class : The CNP class will be responsible in everything about the card and poem page.*
7. *Sweepthat_stats.py*

- *Stats class : The stats class is a tkinter that displays the statistical graph such as histogram, heat map or pie chart in the different analysis topic.*

8. *Sweepthat_db.py*
    - *DB class : the db class will be responsible for all csv and data parts.*

For all of the images and sound, it will be stored in 1. Card images folder 2. Sound folder 3.backgrounds 4.asset


Game
- start_datetime : str
- react_time : int
- start_react_time : int
- card_pos : str
- cor_pos : str
- pos_rahu : str
- message_end_time : int
- user : User()
- rahu : Rahu()
- db : DB()
- oppo_score : int
- MUSIC_END
- board : Board()
- font
- small_font
- medium_fint
- piece_manager : PieceManager()
- player_selected_card : None
- running : bool
- game_message : str
- oppo_score_msg : str
- user_score_msg : str
- last_click_time
- selecting_level : bool
- level_options : dict
- mem_time : int
- mem_start_t : int
- in_mem_phase : bool
- mem_complete : bool
- confirm : bool
- seletting_mem_time : bool
- time_options : dict
+ __init__()
+ draw_level_selection()
+ level_selec_click(pos) : bool
+ time_selec_click(pos) : bool
+ draw_time_selection()
+ draw_countdown(remain_t)
+ draw_start_prompt()
+ run()
+ continue_game()
DB
- read_csv():
+ __init__()
+ insert()
PieceManager
- board
- sound
+ __init__(board)
+ play_next_sound()
+ reset_board()
+draw(screen)
+ reset_board()
+ dra_pieces(screen)
+ get_selected_piece(pos)
Sound
+ played_sound_global: set()
- paired
- correct_index
- all_sounds_played : bool
+ __init__(paired)
+ play_sound()
Menu
- menu_bg
- play_msg : str
- cardpoem_msg : str
- stats_msg : str
- quit_msg : str
- small_font
- running : bool
- SONG_END
+ screen
+ __init__()
+ draw()
+ handle_stats()
+ event(msg_play_rect, msg_cardpoem_rect, msg_quit_rect, msg_stats_rect)
+ run()
Board
+ _instance
+ paired_initialized : bool
+ shared_paired
- card_data
- card
- bg
- board_rect
- paired
- __initialized : bool
+ __init__()
+ backgrounds()
+ create_board()
+ draw(screen)
+ get_card_images_info() : list
Piece
- image :
- rect
- font
- visible
- paired_index
+ __init__()
+ draw(screen)
CNP
+ screen
- menu : Menu()
- card_width : int
- card_height : int
- cards_per_page : int
- rows : int
- cols : int
- current_page : int
- total_pages : int
- currently_playing : None
+ bg
- card_images : list
- sounds : list
+ button_width
+ button_height
- next_button
- back_button
- menu_button
- card_position
+ start_x
+ start_y
+ __init__()
+ stop_current_sound()
+ handle_event(event)
+ draw()
+ run()
GameEnd
- game
- menu : Menu()
- user
- game_c : Game()
- cardpoem : CNP()
- choice_font
+ __init__(game_instance)
+ game_end()
+ run()
User
- user_score : int
+ __init__()
+ draw_status(screen, status)
Config
+ WIDTH
+ HEIGHT
+ CARD_WITH
+ CARD_HEIGHT
+ BORDER_SIZE
+ FONT_PATH
+ card_image_folder
+ card_image_f
+ card_image_files
+ sound_files
+ sound_folder
+ sound_f
+ card_prac_folder
+ card_prac_f
+ card_prac_files
+ RAHU
+ POSITION
Stats
+ root : root
+ data
+ main_frame : Frame
+ control_frame : Frame
+ graph_var : StringVar
+ graph_option : list
+ graph_menu : Combobox
+ plot_btn : Button
+ status_label : Label
+ viz_frame : Frame
+ figure : Figure
+ canvas : FigureCanvasTkAgg
+ table_frame : Frame
+ text_widget : Text
+ scroll_y : Scrollbar
+ scroll_x : Scrollbar
+ __init__(root)
+ gen_viz()
+ plot_success_rate()
+ plot_reaction_time_heatmap()
+ plot_position_pie()
+ plot_position_analysis()
+ plot_reaction_times()
+ plot_correct_positions()
Rahu
+ images
+ current_rahu : None
+ game_level : None
+ __init__()
+ set_level(level)
+ get_current_rahu()
+ spawn()
+ get_rahu_pos()
+ clear()
+ draw(screen)
+ is_active()

3.3 Algorithms Involved

- *Rule-based Logic : This game uses a rule-based logic algorithm based on traditional Karuta game's logic.*
- *Event-driven mechanics : Karuta game is the reaction game which really fits with event-driven mechanics, with actions triggered by events as the reciter reading a poem, a player clicking a card, or the game reaching its end. These events are handled using event listeners and callback functions.*

4. Statistical Data (Prop Stats)

4.1 Data Features

In the 'database.csv' csv file, we will track 5 features.

1. *Date - start time*
2. *<react_time> time that user use to click the card (If the opponent(computer) is the one who click it, the data will be 0.00)*
3. *<card_pos> position of the correct card (top left, middle left, bottom left, top right, middle right, bottom right)*
4. *<cor_idx> index of the correct card.*
5. *<clicked_pos> position of the clicked card (same as position of the card).*
6. *<clicked_idx> index of the clicked card.*
7. *<rahu_pos> position of the rahu (same as position of the card)*
8. *<result> Result of that round (Correct, Incorrect or Opponent Correct.)*

From the above, the data that we have tracked will be calculated into the user analysis to make the user improve themselves ex.If the position of the card often on the top right, means that the user needs to pay attention on the other side of the board more, or if the position of rahu is the same as position of the correct card and the result of user is correct means that user is can really remember the position of the card.

3.2 Data Recording Method

All the game's statistical data will be stored in a CSV file with date and start time, which can be used to calculate the average time that user clicks the card in each round.

3.3 Data Analysis Report

Statistical Measures

- *Mean and Median: Calculate the average time to sweep cards and total time per round to understand typical player behavior.*
- *Frequency Distribution: Analyze how often each poem is quickly or slowly clicked to identify patterns in player knowledge.*

Presentation

- *Histogram :*
    1. *The frequency of the correct result in hard mode (RAHU appear) compared to the result in normal mode (no RAHU appear)'*
    2. *Specific card and the frequency that user can't click that card correctly (click another card or can't click the card in time at all)*
- *Heat map : Represent the top 9 longest reaction times that users use.*
- *Table : Show the position of RAHU vs the position of the correct card and success rate.*
- *Line graph : Compare the average time that the user sweeps in each round.*
- *Pie chart : Position of the correct card that users often overlook.*

	Feature Name	*Graph Object*	*Graph Type*	*X-axis*	*Y-axis*
Frequency of correct result in hard mode compare to normal mode	*rahu_pos, result*	*Comparison of success rates between modes.*	*Histogram*	*Rahu_pos (hard mode and normal mode)*	*Frequency*
Most clicked cards.	*clicked_idx*	*To see which card user tends to remember the most.*	*Heatmap*	*-*	*-*
The poem of the correct card compares to the poem of the clicked card. (In the incorrect case)	*result, cor_idx, clicked_idx*	*To see what's the difference between these 2 cards that makes the user confused and choose different cards.*	*Table*	*round*	*Correct poem, clicked poem*
Position of rahu compare to position of clicked card and position of correct card	*rahu_pos, cor_pos, clicked_pos*	*To see if the position of rahu affects the clicked card of the user.*	*Table*	*round*	*Rahu_pos, clicked_pos,correct_pos*

All the sweep times.	*date_start time, react_time*	*To see the average times that users use in each round and make the user know their developme nt.*	*Line graph*	*date_start time*	*react_time*
Total num of the position of correctly clicked card.	*result, clicked_po s*	*To see how much the user can click the correct card and what's the position of the card.*	*Table Lens*	*Count clicked_po s*	*Clicked_po s*

	What can this data be used for? Why is it good to have this data?	*How will you obtain 50 values of this feature data?*	*Which variable and which class will you collect this from?*	*How will you display this feature data (via summarization statistics or via graph)?*
Date - Start time	*The date - start time will be stored to be the primary key so we can know what's happening in the round since we collect the data as soon as the user clicks the card and we also have a lot of rounds until the user or the computer can click 18 cards.*	*By playing the game until the data is enough to calculate.*	*Collect from self.start_dateti me in Game class*	*It will display along with the average time that users use to click the card and the time that users use to memorize the card.*
Time users use to click the card.	*The time that the player uses to find the right card and click it can be used to analyse if they are faster or slower or if the user can't remember the poem or the position so they use a lot more time than the other card.*	*By playing the game until the data is enough to calculate.*	*Collect from self.time_round in User class*	*The data will be represented in a line graph to compare all the time that the user uses.*
Index of the correct card.	*The index of the correct card to compare with the index of the clicked card to*	*By playing the game until the data is enough to calculate.*	*Collect from self.piece_mana ger.sound.correc t_index in PieceManager*	*The index will be used to find the poem and compare it to the poem of the*

	see if the user clicks the incorrect card, does it have the similar first words? Or if it has a similar sound?		*class*	*clicked card.*
Position of the correct card.	*Position of the correct card to compare to the position of the clicked card to see that the user may have clicked the wrong card because it's located nearby.*	*By playing the game until the data is enough to calculate.*	*Collect from 'self.card_data' in Board class (self.cor_pos)*	*The position of the correct data will be used to represent in the table to check with the position of the RAHU.*
Index of the clicked card	*The index can tell which card the user tends to remember the poem the most and which card that user tends to forget the poem.*	*By playing the game until the data is enough to calculate.*	*Collect from self.player_selected_card in Game clas*	*The cards will be displayed as a pie chart showing which card that the user tends to click the most (which means that the user can remember that card well) and also used to find the poem and compare it to the correct poem index.*
Position of the clicked card	*To know which part of the board that user pays attention to or which part that user often overlooks.*	*By playing the game until the data is enough to calculate.*	*Collect from 'self.card_data' in Board class (self.card_pos)*	*The card's position data will display as a pie chart with 'top left', 'top right', 'bottom left', and 'bottom right' with the percentage that the user clicked the card on that side. Along with*

				the suggestion such as "Maybe pay more attention to the left side's card!"
Position of Rahu	*To know the position that the user can't see, to check if the user can remember the position even though they can't see the card.*	*By playing the game until the data is enough to calculate.*	*Collect from self.get_rahu_pos() in Rahu Class (self.pos_rahu)*	*The position of rahu can also be used to check if the user played the normal mode or hard mode and the data will also be displayed in the table.*
Result	*To know if the user clicks the card correctly or not.*	*By playing the game until the data is enough to calculate.*	*Collect from 'result'*	*This data feature will help the other data to display such as pie chart of the incorrect position. The data will need to filter out the correct position since we only want the incorrect one. This is where result data will be used*

4. Project Timeline

Week	**Task**
1 (10 March)	Proposal submission / Project initiation
2 (17 March)	Full proposal submission
3 (24 March)	Initial game function work properly
4 (31 March)	Finish user and RAHU

5 (7 April)	Overall, the game function correctly without bug
6 (14 April)	Submission week (Draft)

26 March - 2 April	Finish all the main game part
3 April - 9 April	Finish read poems part
10 April - 16 April	Collect data and start tkinter
17 April - 23 April	tkinter
24 April - 11 May	Finish tkinter part.

5. Document version

Version: *4.0*

Date: *31 March 2025*

Date	Name	Description of Revision, Feedback, Comments
14/3	Rattapoom	Very detailed proposal 👍
16/3	Parima	Overall, the proposal is great in detail and well formatted. Good Work 🌟
28/3	Rattapoom	• Please also put the variable that you intended to use for collecting data in the UML class diagram. • Is Date - Start time collected every time that you collect the data for other events? Or is it collected every time a new game session starts? If it's the latter, then you'd have to play the game 50 times, which is quite tedious. Also I don't think a primary key is necessary, since for this project you can analyse different data features independently. And you aren't using a SQL database. You can also store different data on different SQL files. • Also for "Time user use to memorize the card". If I understand the rule of Traditional Karuta correctly, that would be the time after the "Torifuda" cards are laid down and before the reciter starts reciting. Which happens once every session. So you may also need to play the game 50 times to get the data, which is tedious to do so. • Other data features does not require the user to play 50 rounds, which is great 👍
30/3	Parima	I think the UML diagram needs some revision, but other parts are great.