# CIS045-3 Distributed Service Architectures – AY 24/25

# Student Name: Rashmi Proboda Denagamage Dona Student Number: 2425007

# Student Signature: Date: November 2024 (Week 8)

# Software Architecture (Low Coupling / High Cohesion)

*Low Coupling: My code is designed in a way that the game components don’t depend too much on each other. Where they are independent which improves flexibility and makes the game easier to maintain.*

* ***Session Class****: Manages login/logout functionality*
* ***GameEngine Class****: Handles the game logic (such as game levels, score checking, etc.) without being directly connected to the user interface.*
* ***Game UI****: The UI is separate from the game logic, displaying the game’s image or puzzles without affecting the game mechanics. It interacts with the GameEngine class for game data but doesn't directly influence the game rules or score.*
* ***Leaderboard****: The leaderboard (via the UserScore class) manages score data separately and doesn’t directly influence the game logic or UI.*

*High Cohesion: Each class or component is responsible for a singular task.*

* ***Session Class****: managing the logged-in user’s information only.*
* ***GameEngine Class****: This class is responsible for managing the game's flow, levels, scoring, and checking answers. It doesn't mix with other concerns (like UI or leaderboard), making it cohesive.*
* ***Game UI****: The UI class is focused on presenting the game’s images and puzzles to the player. It does not handle game logic or scoring, which helps it maintain cohesion.*
* ***UserScore Class****: The UserScore class is solely focused on storing and managing score and time-related information. It doesn’t manage the game flow or UI, keeping its responsibilities clear and cohesive.*

*More over the code segment is divided into 03 packages as for code which holds the GUIs of the game,* *Code.banana.engine which stores the games logic and Code. database which stores the database for the implementation of the Game.*

# Event Driven Architectures

# *Login Event: The Session class is triggered to save the user's information and start the game when they log in.*

# *Level Change: When the player completes a level or starts a new one, the game triggers events that switch to the next level and adjusts the difficulty.*

# *Game UI Interaction: The game reacts to the player’s actions, like selecting matching pair of images, which updates the score and checks whether it’s correct.*

# *Equation Game: Once the player has finished the puzzle on the game interface, an equation timer begins to run and waits for them to finish it in the allotted 60 seconds which is the Banana Api integrated part.*

# *Leaderboard Event: The score is updated and shown on the leaderboard once the player completes a game.*

# Interoperability

*Do you use the API* [*https://marcconrad.com/uob/banana/api.php*](https://marcconrad.com/uob/banana/api.php)*?*

# *Yes, my code uses the API https://marcconrad.com/uob/banana/api.php to get game data. It fetches a Base64-encoded image and solution, which is then decoded and used in the game*

# Virtual Identity

# *In my code Virtual Identity is handled through the Session class. When a user logs in, their username is saved, and it’s checked to see if they are logged in or not. This way, the app knows who the player is. Letting the player to log in and log out from the Game.*

# Any other interesting features:

***Login System****: The Session class manages login/logout, ensuring the app knows which player is currently playing.*

***Game Levels****: The game includes different levels of difficulty (beginner, intermediate, advanced), each affecting the puzzle and equation difficulty with different time allocations based on the level of difficulty the player has selected.*

***Puzzle Game****: The players are supposed to match the similar pairs of images within the time duration varied by the Game Difficulty and the Time.*

***Equation Game****: After solving the puzzle, players must solve a math equation within a time limit, to win the game.*

***Leaderboard****: The User Score class tracks players’ scores, times, and when they played, allowing a leaderboard to be displayed based on performance.*