MILESTONE 3 - Application Design and Data Specification (System Architecture and Assets + Class Diagrams + Data Structures and Data Flow)

Name(s): Shubham Mohanty **|** Devarsh Limbachiya **|** Aryan Jajadiya **|** Naqi Kadri

Group: Group 6

IDs: 100898560 **|** 100893942 **|** 100894608 **|** 100903721

Date: March 8th, 2024

Course: COSC-2200-07

System Architecture and Assets

Design Architecture:

Application Type: The Durak game will be organized into different parts. One part will show the game to players, another part will control how the game works, and a third part will store important information like player scores. This setup helps keep things neat and easy to manage.

MVC (Model-View-Controller):We'll use the MVC design pattern to organize our code well. This pattern breaks down the application into three connected parts:

Model: This part will handle all the important stuff like how the game works, what cards players have, and the rules of the game.

View: Here, we'll take care of what players see and how they interact with the game. We'll use Pygame to show buttons, cards, and other things on the screen.

Controller: This part is like the middleman. It takes the player's actions, like clicking buttons or moving cards, and makes sure the game reacts properly.

Libraries Utilized:

Pygame: We'll use Pygame to create the game's visual interface, manage player interactions, and display graphics.

JSON for Data Persistence: JSON format will store game progress, user's personalized data, user choices, and system configurations, ensuring easy access and modification.

Custom Libraries:

Custom classes will be developed to manage different aspects of the game:

Card: Represents each playing card, including its suit, rank, and methods for comparing cards.

Deck: Handles the game deck, like shuffling, dealing cards, and keeping track of remaining cards.

- **Player and AIPlayer**: Represent human and computer players, with methods for playing cards and managing hands.

Game: Controls the main game rules, turn order, and determining the winner.

GameWindow:Manages UI elements, like displaying cards, buttons, and game messages.

For organization:

We'll use a modular design to keep different parts of the code separate and reusable.

Python packages and modules will help organize the code into logical sections, like game logic, AI behavior, UI management, and data storage.

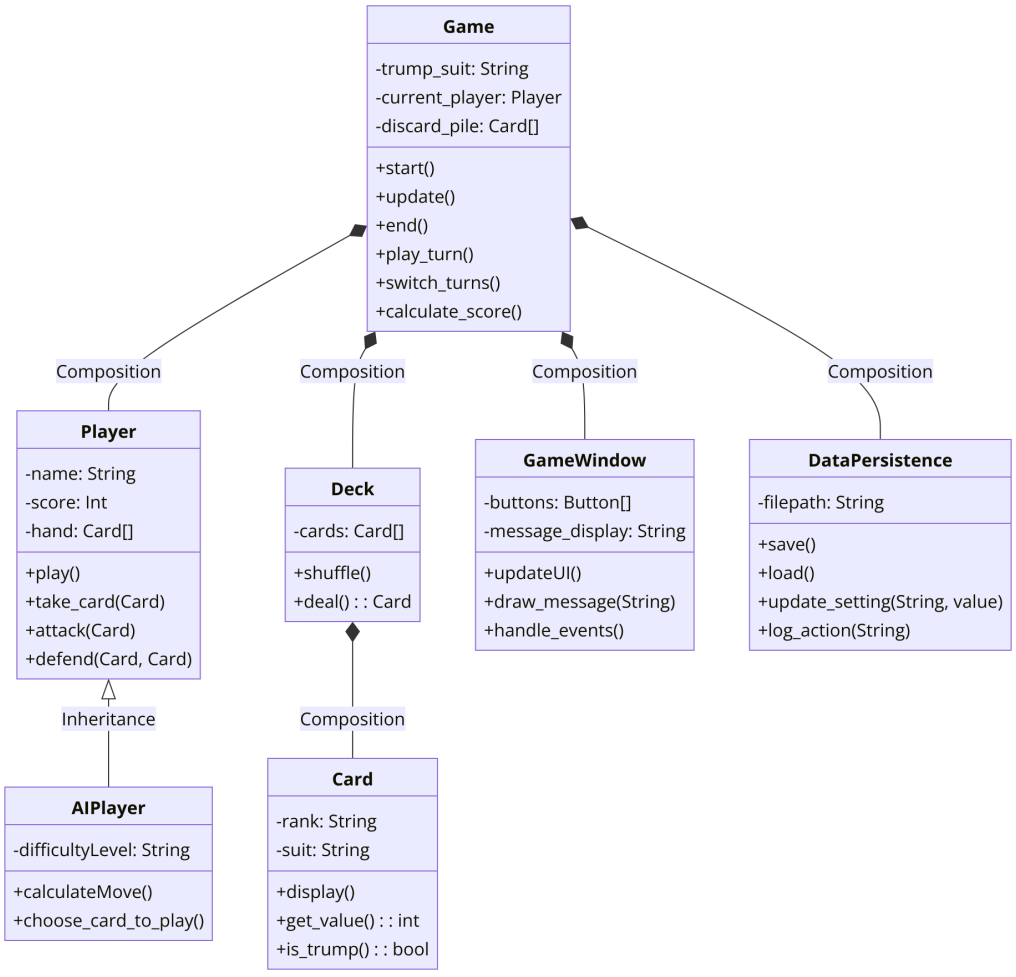
External Assets:Code Organization Strategy

Graphical Images: Gcard faces and UI elements, will come from free sources or be made for the game. We'll give proper credit to the original creators.

External Libraries: Pygame will be the main library for creating the game's UI. We may add other libraries for extra features, with citations to show where we got them from.

Class Diagrams

> This is the 3rd version of the class diagram:



Data Structures and Data Flow

Data Structures:

Data Source Type:We'll use JSON format to save our game information because it's easy to understand, can be read by humans, and works well with Python.

Data Fields/Objects: Things like how the game is going, what the player likes, and how the computer players act will be saved in JSON files. This makes it easy to find and change them.

Security Considerations:We'll make sure only the right people can read and change the game data to keep it safe and correct. If there's a problem, we'll handle it properly.

Data Access: We'll use a special part of Python called the json module to read and write JSON files. We'll also create a special file, like data_persistence.py, to handle all the data stuff, making it easier to use.

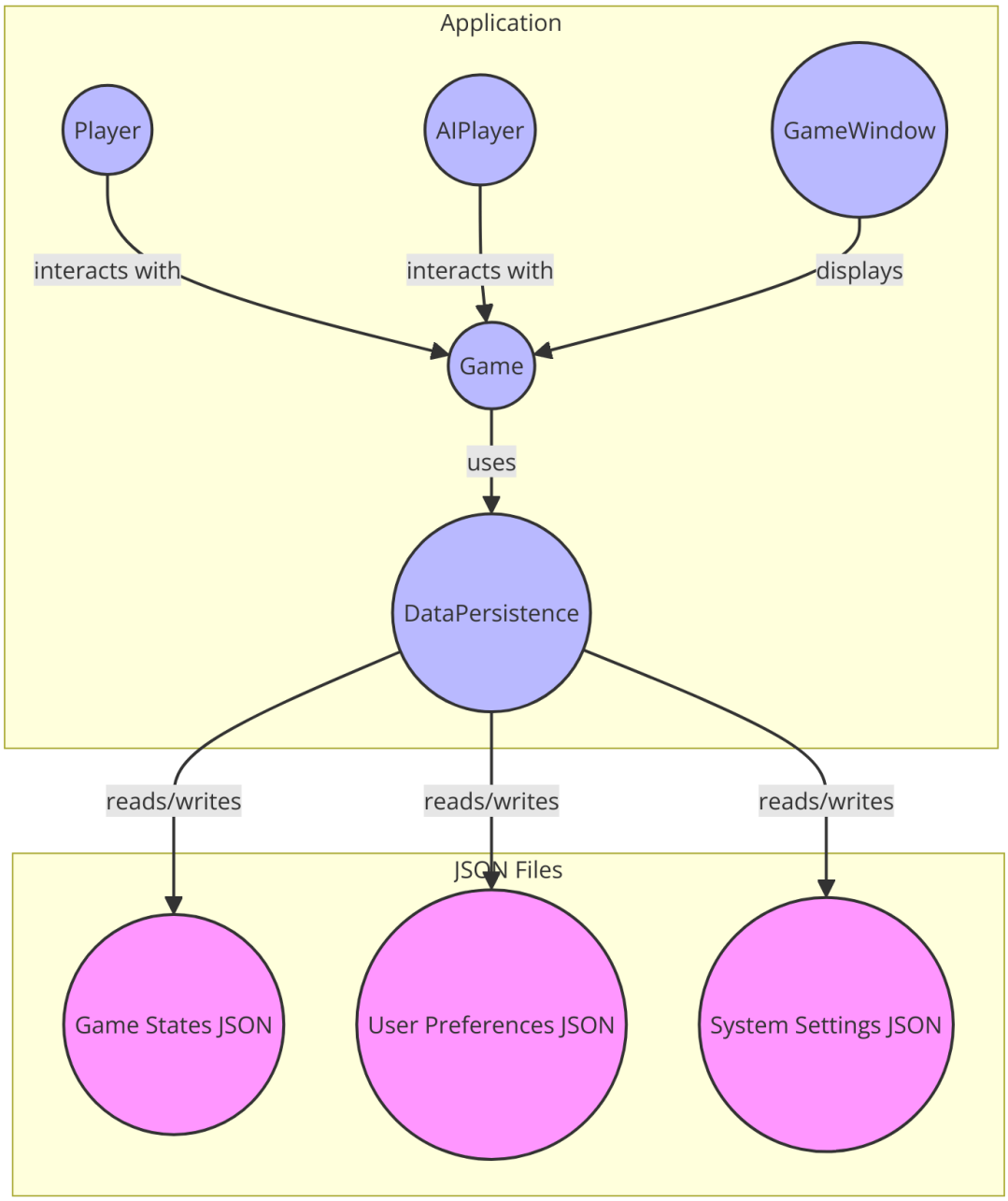
Linking Data Sources to Classes: Our data part will talk to the game rules part to save and get the information we need. This helps us keep things organized and easy to use.

Data Storage in the Project: We'll use special parts of Python to keep our data organized and working smoothly in the game, so it runs well and can handle a lot of information.

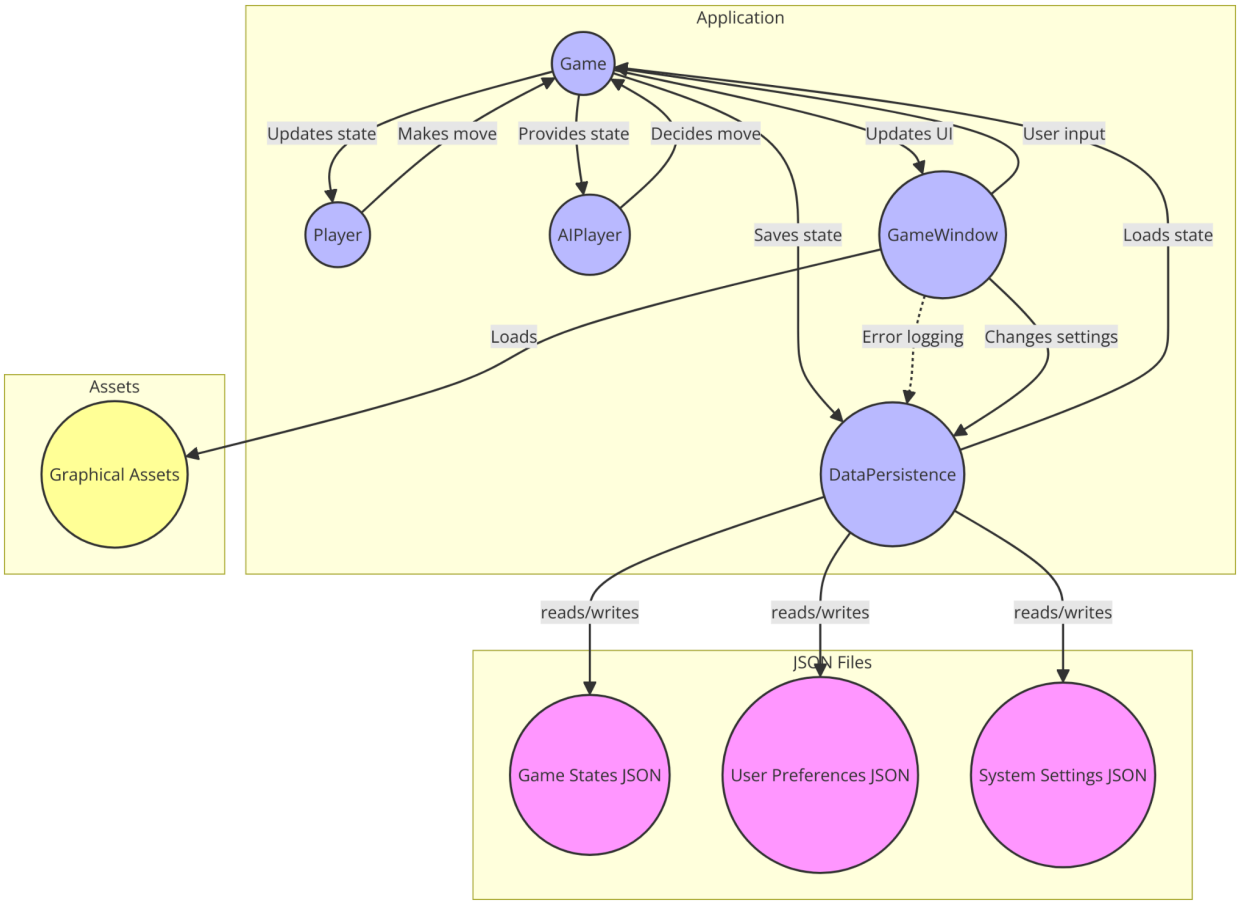
Data Flow:

DFD - V1 (Level 1)

- Player: This is where you play the game, make your moves, and see what's happening.
- AI Player: Just like you, but it's the computer making moves based on the game's rules.
- Game Window: This is your game screen, where you can see and do everything in the game.
- Game: Think of it as the brain of the game; it knows all the rules and keeps the game going. It also remembers and stores game info.
- Data Persistence: This is like a memory box; it keeps all the game data safe and sound.
- JSON Files: These are like notebooks where:
    - Game States JSON keeps track of what's currently happening in the game.
    - User Preferences JSON remembers how you like your game settings.
    - System Settings JSON holds onto the general game settings for everyone


Application
Player
AIPlayer
GameWindow
interacts with
interacts with
displays
Game
uses
DataPersistence
reads/writes
reads/writes
reads/writes
JSON Files
Game States JSON
User Preferences JSON
System Settings JSON

DFD - V2 (Level 2)



- Player: You tell the game what you want to do, like attacking or blocking, and the game keeps you posted on what's happening.
- AI Player: It gets the latest game news, thinks of a smart move, and tells the game what it wants to do.
- Game: It's a bit like a referee, keeping track of both your and the AI's moves, thinking about what happens next, and remembering everything by writing it down.
- Game Window: It's like your game's hands and eyes; it takes your commands and shows you the game's world. It also helps you change settings and might keep an eye out for any hiccups.

- Data Persistence: Think of it as the game's diary; it writes down your game's story and remembers how you've set things up.
- Assets: These are all the pretty things you see in the game, like the art and designs.