**Question No.1**

**Use Case Analysis:**

**Actors:**

User

**Actor Actions:**

A user will be able to communicate with the system and it either provide state of the game or start playing from the new screen. The user can either win or defeat or else the game will be drawn

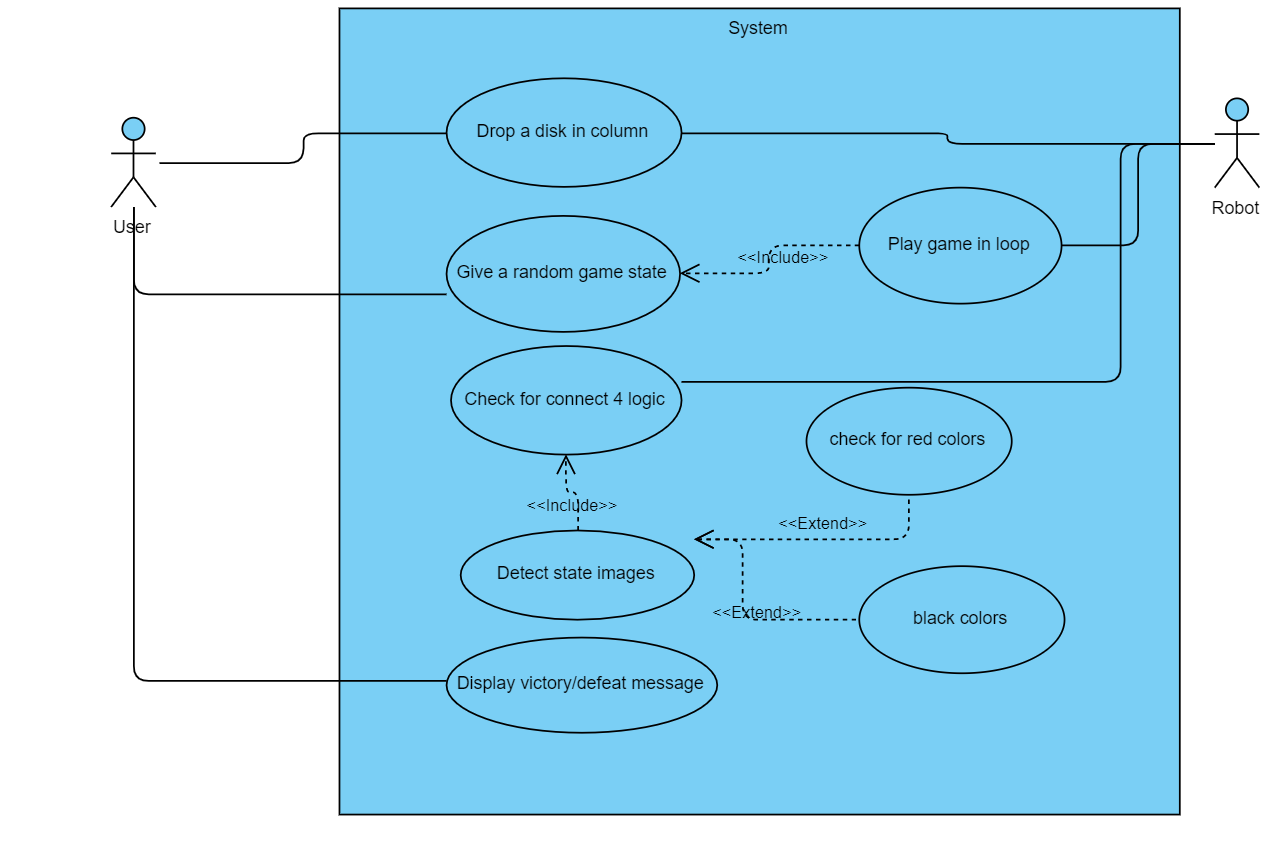
Robot

**Actor Actions:**

A robot will react to the moves of the user. It will try to block the moves played by the user by dropping a disk in particular column where there might be a possibility to create four consecutive disks in any alignment (row, column and diagonals).

The robot will detect the states of the game by analyzing the images provided by the user.

**Use case diagram:**



**Use case 1:**

Drop a disk in column

**Primary actors:** User, Robot

**Details:** The user will click on the column in order to drop the disk at bottom place if it is occupied by any other disk, it will be stacked on the top of it. The robot will also stack a disk on top of a random column in its turn

**Preconditions:** The game must be running and its an actor turn to play the game

**Triggers:** The user have clicked on the column

**Main success scenario:** The user have clicked on the column and a disk is dropped on the bottom of the column and stacked on the top of other discs if it is already present there, regardless of the color.

**Alternative Flow 1:** The user have missed the click on the exact column and nothing happened from the system

**Alternative Flow 2:** The user have clicked on the column but it is already filled to the top and nothing happens

**Use case 2:**

Give random game state

**Primary actors:** User

**Details:** The user will provide an image to the board and system will detect numbers and places of the disks present in the picture.

**Preconditions:** The game must be running and user provides input image

**Triggers:** The user loaded the state of the image

**Main success scenario:** The user has provided the image and the system and successfully detected and mapped its board from it.

**Alternative Flow 1:** The user have provided a wrong path and system displays an error message.

**Alternative Flow 2:** The user have clicked on the column but it is already filled to the top and nothing happens

**Use case 3:**

Check for connect four logic

**Primary actors:** Robot

**Details:** The robot will play its turn and system will check if any four consecutive discs appear in any column.

**Preconditions:** The game must be running and user have made a move

**Triggers:** Any disc is placed in its new location

**Main success scenario:** The system has successfully detected combination for four in rows, columns or diagonal places. And a result is generated

**Alternative Flow 1:** The system was unable to detect combination of 4 discs and no result is generated.

**Use case 4:**

Check for connect four logic

**Primary actors:** Robot

**Details:** The robot will play its turn and system will check if any four consecutive discs appear in any column.

**Preconditions:** The game must be running and user have made a move

**Triggers:** Any disc is placed in its new location

**Main success scenario:** The system has successfully detected combination for four in rows, columns or diagonal places. And a result is generated

**Alternative Flow 1:** The system was unable to detect combination of 4 discs and no result is generated.

**Use case 5:**

Display victory/ defeat message

**Primary actors:** User, Robot,System

**Details:** The robot will play its turn and system will check if any four consecutive discs appear in any column. If this move was done by a user then victory message for user will be displayed else if robot have places four in a row then victory message for robot will be displayed.

**Preconditions:** The game must be running and user have made a move which makes 4 consecutive pair

**Triggers:** Any 4th disc is placed in the column

**Main success scenario:** The system has successfully detected combination for four in rows, columns or diagonal places. Winner is announced

**Alternative Flow 1:** The system was unable to detect combination of 4 discs and game is drawn without any result.

**Question No.2 Design documentation**

The goal is to create a connect four game where a user interacts with the system and clicks on the columns to place the discs on the bottom of the column to stack them on one another until a four consecutive pair is formed which makes the game decisive.

**Implemented and tested functional Requirements:**

1. **FR1-**  The user shall be able to give the states of the image and system should detect the states and should continue game after those states.
2. **FR2-** The user should be able to interact with the game to drop the discs in the particular columns at the bottom places
3. **FR3-** The system should be able to interact on the move of the user by placing a disc to counter the move of user.
4. **FR4-** The system should be able to detect any four consecutive discs appearing in any row, column or diagonal position
5. **FR5-** The system shall be able to display a result either victory or defeat depending on the user who finished first in connecting.

**Design Architecture:**

The overall code is designed in terms of reusability and performance. OOP concepts are highly used in order to create low coupling and high cohesion.

There are three main classes:

1. Circle\_Node

Circle node is system circle or disc which is storing basic information about the disc and later it becomes the part of a board like Nodes are associated with linked lists.

1. Board

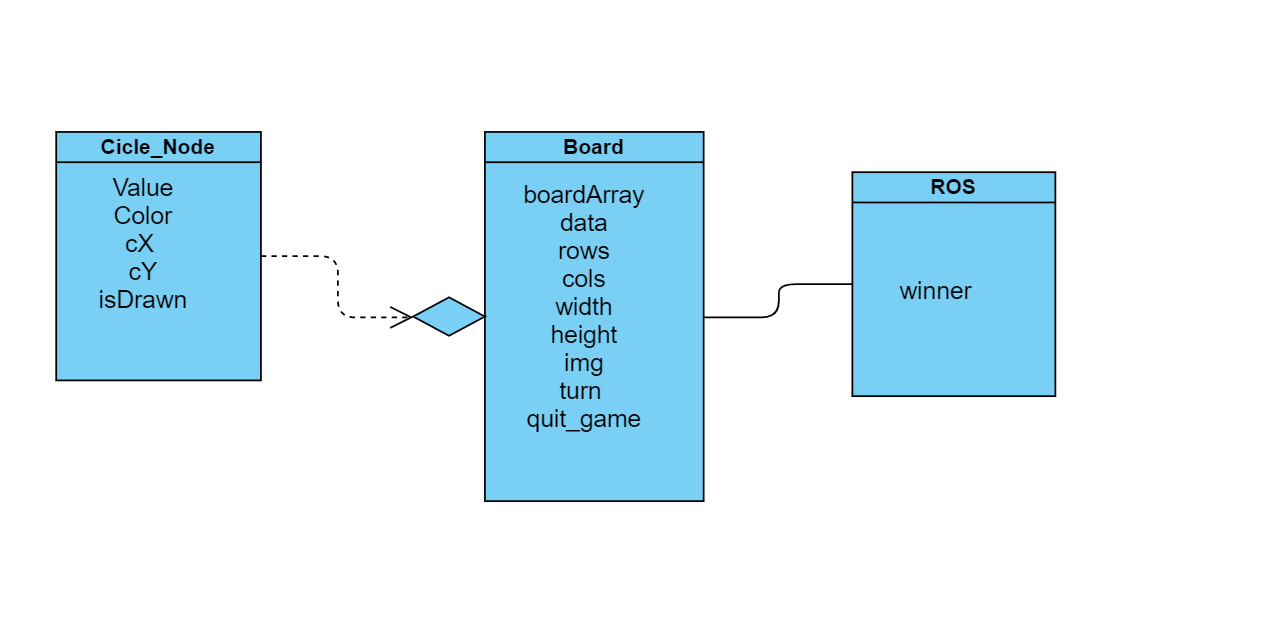
This class is the main class which populates the board and manages board functions like printing the board, placing a new disc, detecting circles from the images and there are many other functions related to board

1. ROS (Robot Operating System)

This class is mainly concerned with checking 4 connect four logic by the robot and and user. This class is made so that all the code is not stuffed in board class and is associated with Board class by ROS object.

These classes interact which each other.

**Class diagram**

****

The code is highly commented and all the test cases are implemented. The file is connect\_four.ipynb Run this file on jupyter notebook.