RĪGAS TEHNISKĀ UNIVERSITĀTE

Connect-4 vs AI

Riga Technical University

Applied System Software

PRESENTATION

MEET GROUP

Meet our team

Kohei Miyamoto

Prototyping/Integration

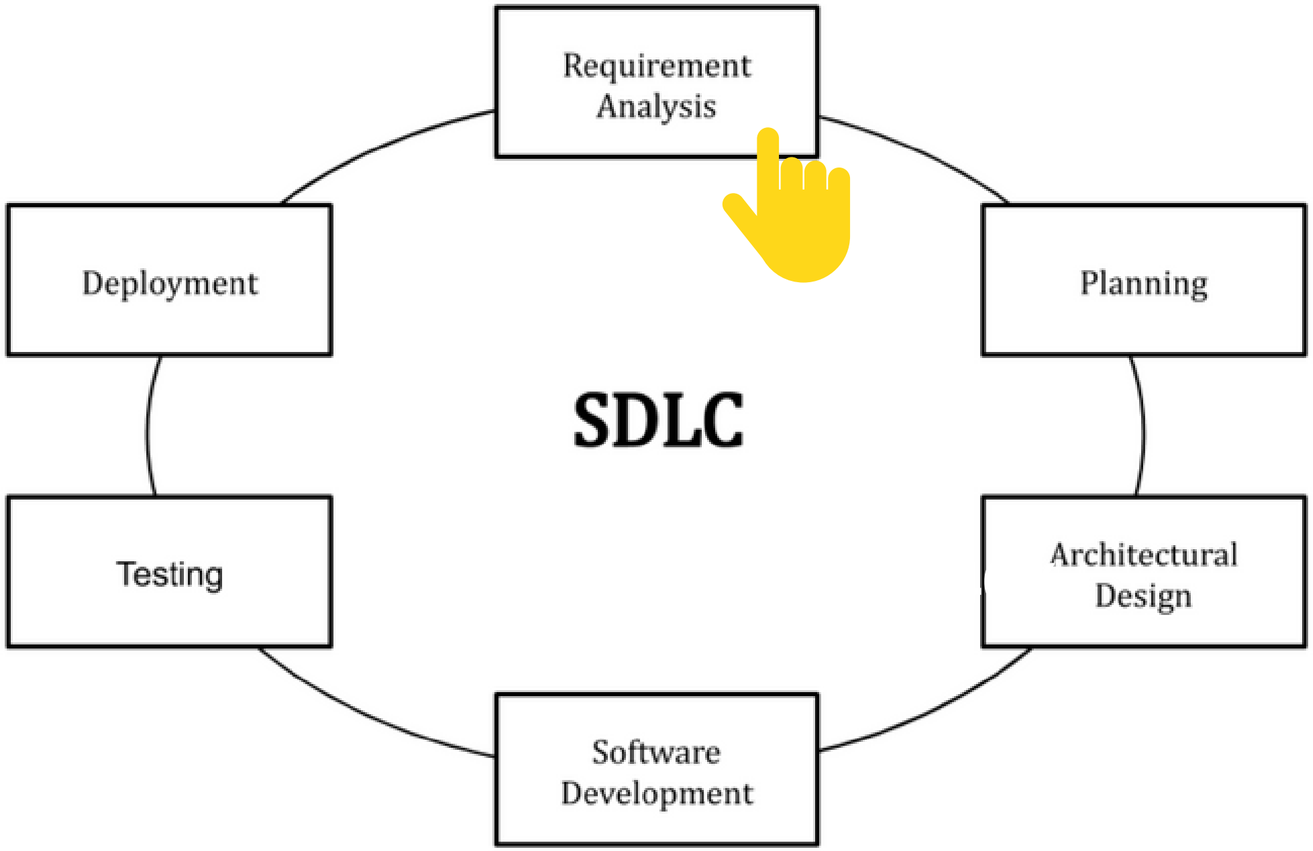
Mio Tabayashi

AI development

B. Aritha Kumarasinghe

UI/UX develpment

Requirement Analysis



Requirement Analysis

S R S

Functional

- Receive input
- Fill in the slot
- Check the winning condition
- Display the winner

Nonfunctional

- English
- can be executed in Windows, MacOS and Linux
- can be executed without the Internet connection

U R S

User can choose several play modes

- vs AI
- vs Other user
- User can play using GUI (button)
- User can watch the state of the game after each choice

S y s R S

- Python 3.10.6
- GUI: PyQt5

Requirements

1.1 The software will be a computerized version of the board game Connect 4.

1.2 The software must be installable.

1.3

1.3.1: The software must have a Graphical User Interface.

1.3.2: The GUI within the software must allow each player to fill one of the slots within the grid for each of their turns.

1.3.3: The GUI within the software must announce the winner of the game based on the results produced by the GRI.

1.4 The game within the software must be able to be played between two users, or between a user and the computer.

Requirements

1.5

1.5.1: The game within the software can only have one player or the computer start by making a move then follow a sequential manner of where one player after the other makes moves.

1.5.2: The game will only allow each of the players to make a maximum of 21 moves.

1.5.3: The moves will consist of the players filling the slots within the grid with 21 tokens of a specific color(either blue or orange)given to them at the start of the game

1.6

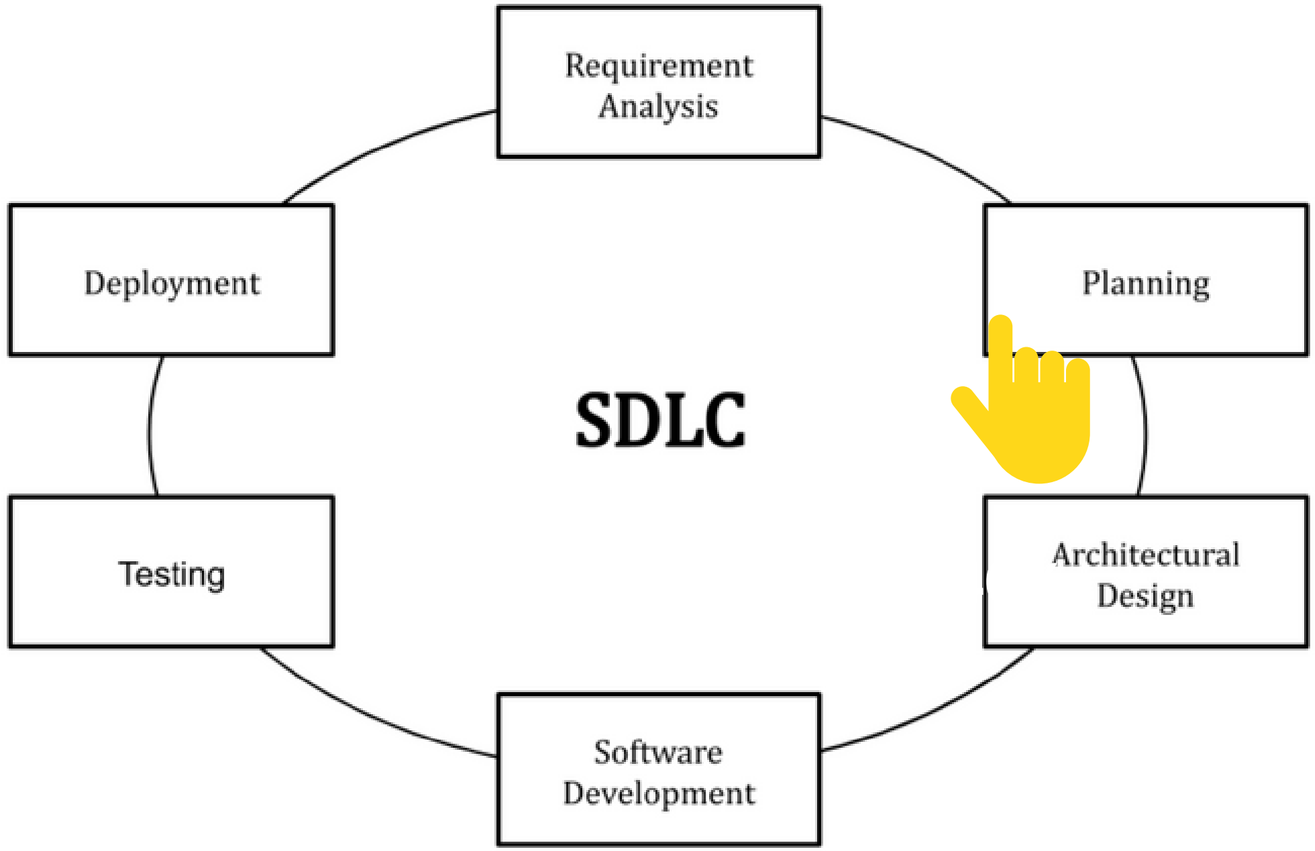
1.6.1: The software will have a game rule implementation(GRI) mechanism.

1.6.2: The GRI will use a game state matrix in order to keep track of the moves made.

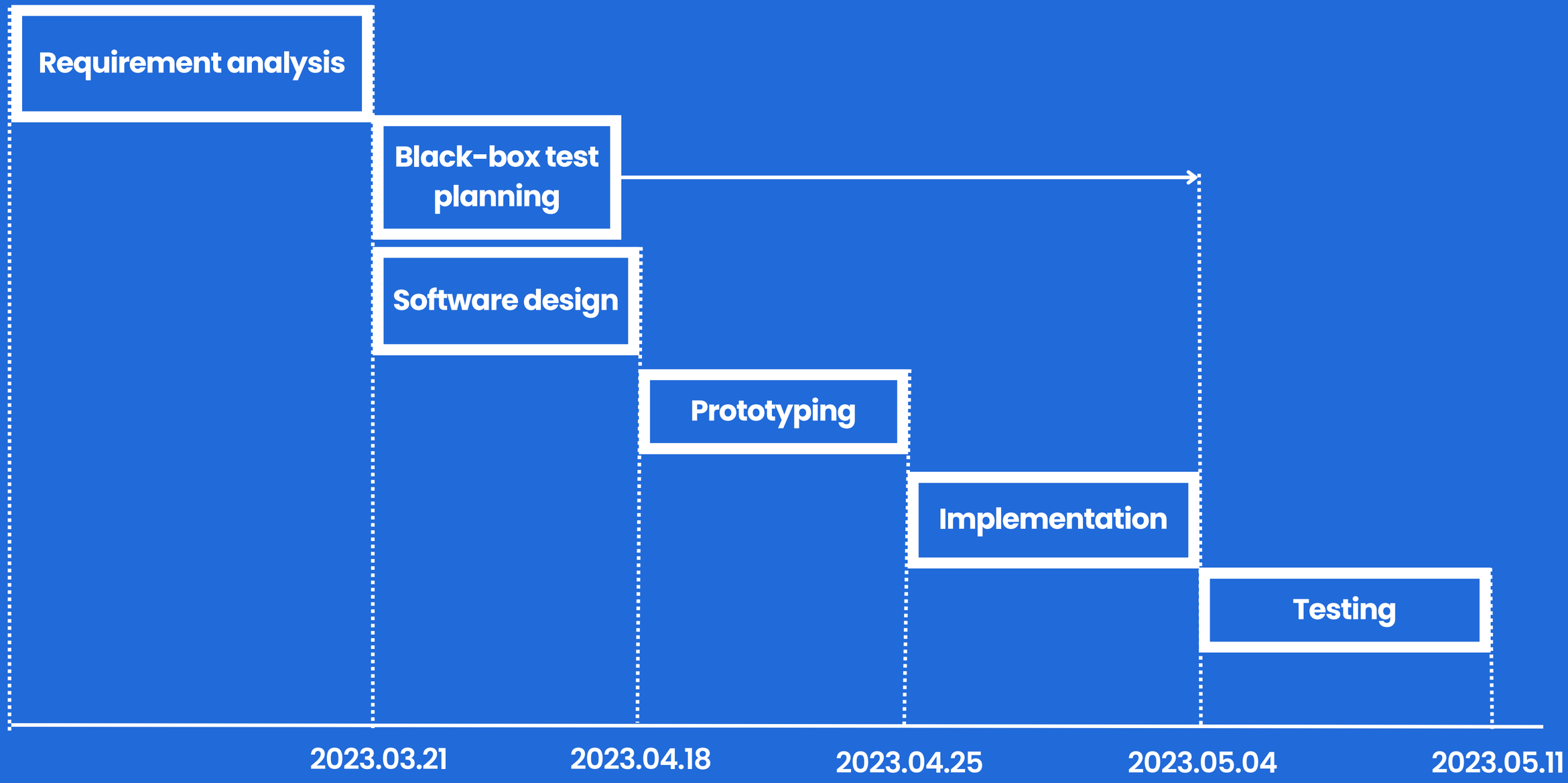
1.6.3: The GRI will analyze the game state matrix for 4 slots filled in the same color and decide the winner based on the color.

24/7
Canva

Planning


Requirement Analysis
Planning
Architectural Design
Software Development
Testing
Deployment
SDLC

Development plan



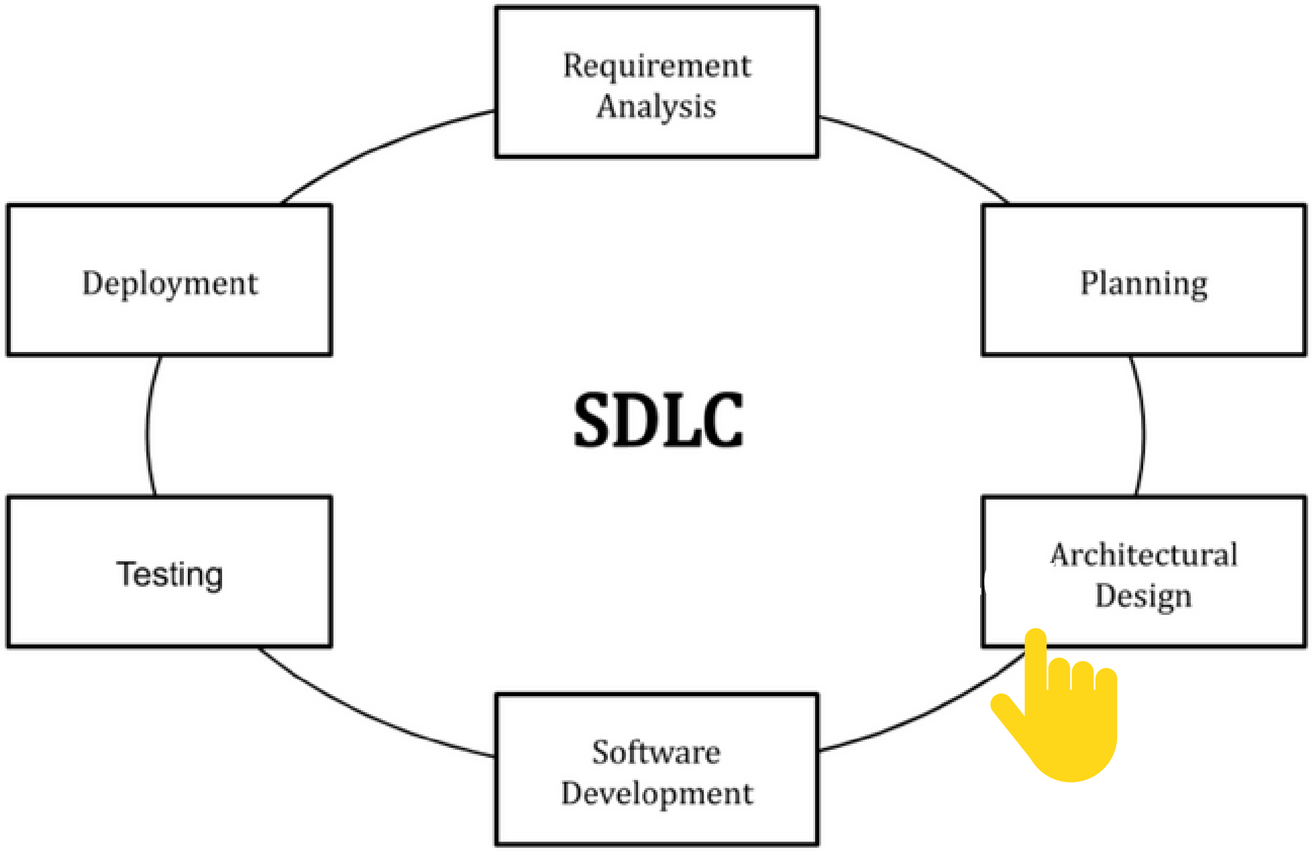
Test plan

Test ID	Description	Expected Result
1.1	(vs person mode) If the user inputs the column to put the pack, it should insert to the bottom of the row based on the gravity factor, then the other player adds another pack in the same manner.	The visualization of the game board with 2 newly added packs.
1.2	(vs AI mode) If the user inputs the column to put the pack, it should insert to the bottom of the row based on the gravity factor, then the AI adds another pack in the same manner.	The visualization of the game board with 2 newly added packs.
1.3	The player turn should come once in every two turns	User adds the pack onto an odd number of packs or even number of packs depending on the turn.
1.3.1	(vs player mode) The player1 inputs in odd turn	Input happens on the 1st, 3rd, 5th ... 2N-1th time
1.3.2	(vs player mode) The player2 inputs in even turn	Input happens on the 2nd, 4th, 6th ... 2Nth time

Test plan

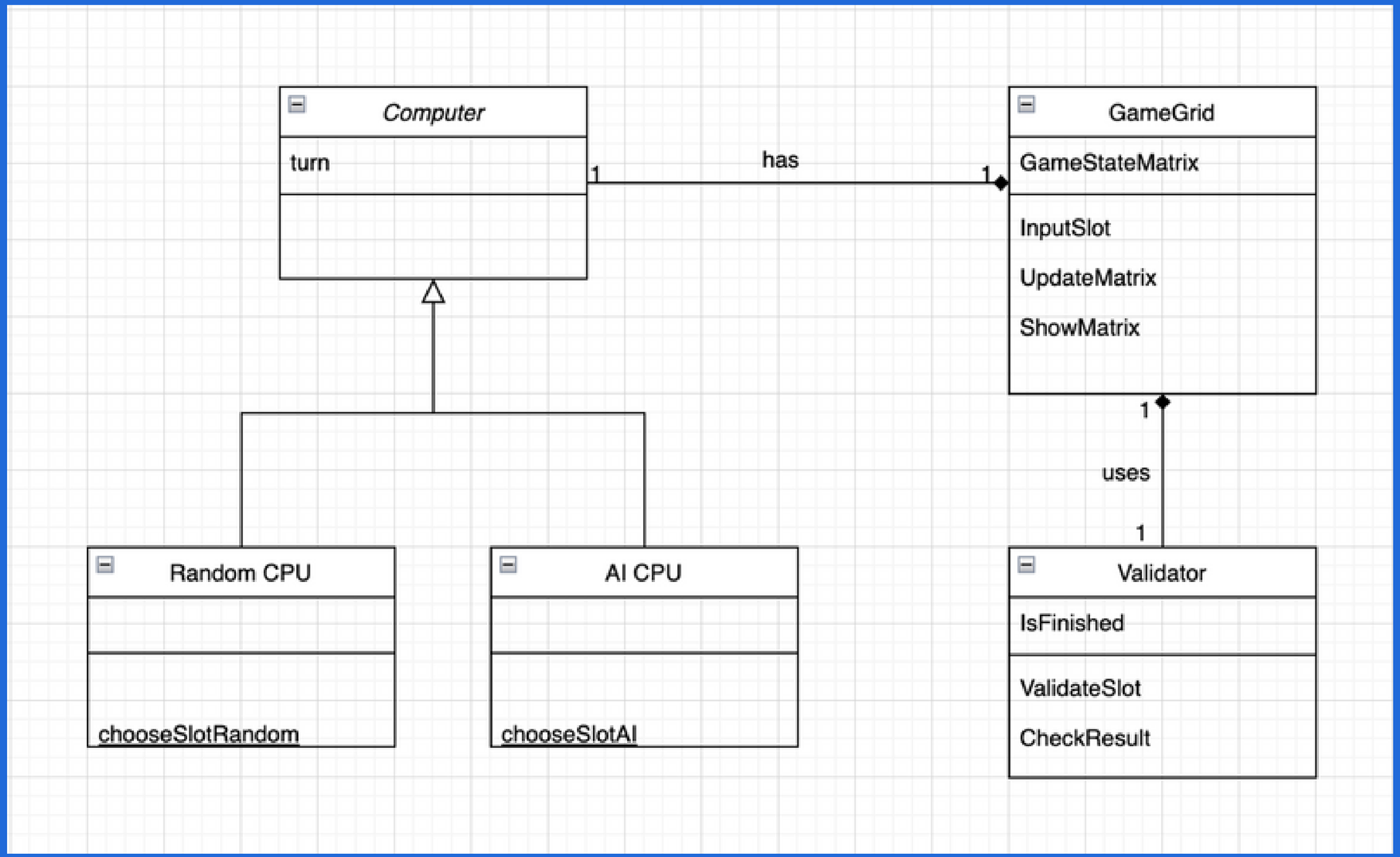
Test ID	**Description**	**Expected Result**
2	If there are 4 sequential packs of the same color in horizontal, vertical or diagonal gamespace, it should stop the game and display who wins the game.	Message saying who is the winner along with the final result of the game board.
3.1	If the user inputs the invalid data, it should reject the input and let the user do it again	Error message and another input try.
3.2	If users inputs letters other than column name (A, B, C, D, E, F and G), rejects the input.	Error message and another input try
3.3	If the user inputs the column which is already full, rejects the input.	Error message and another input try

Architectural Design



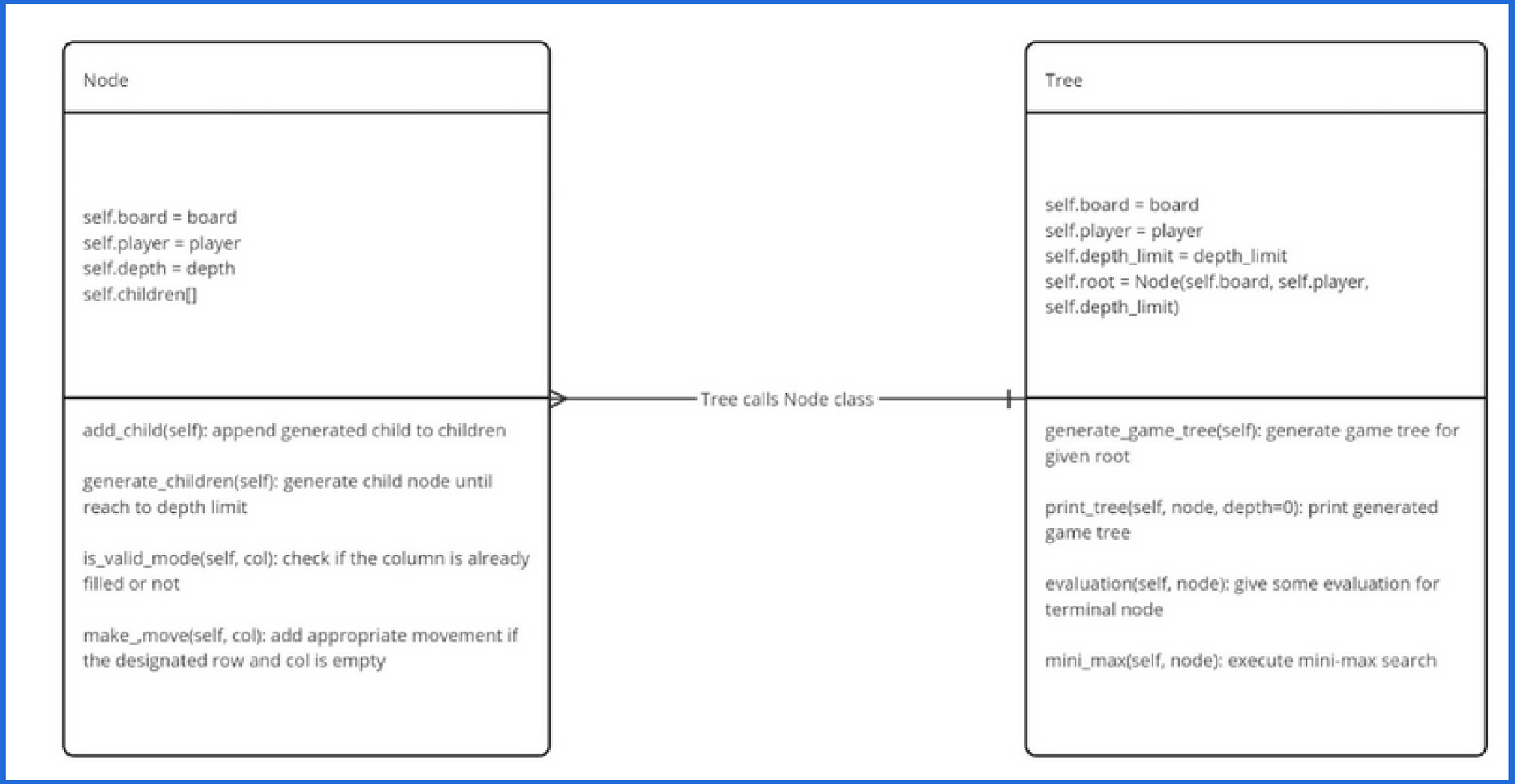
SOFTWARE ARCHITECTURE

Class diagram

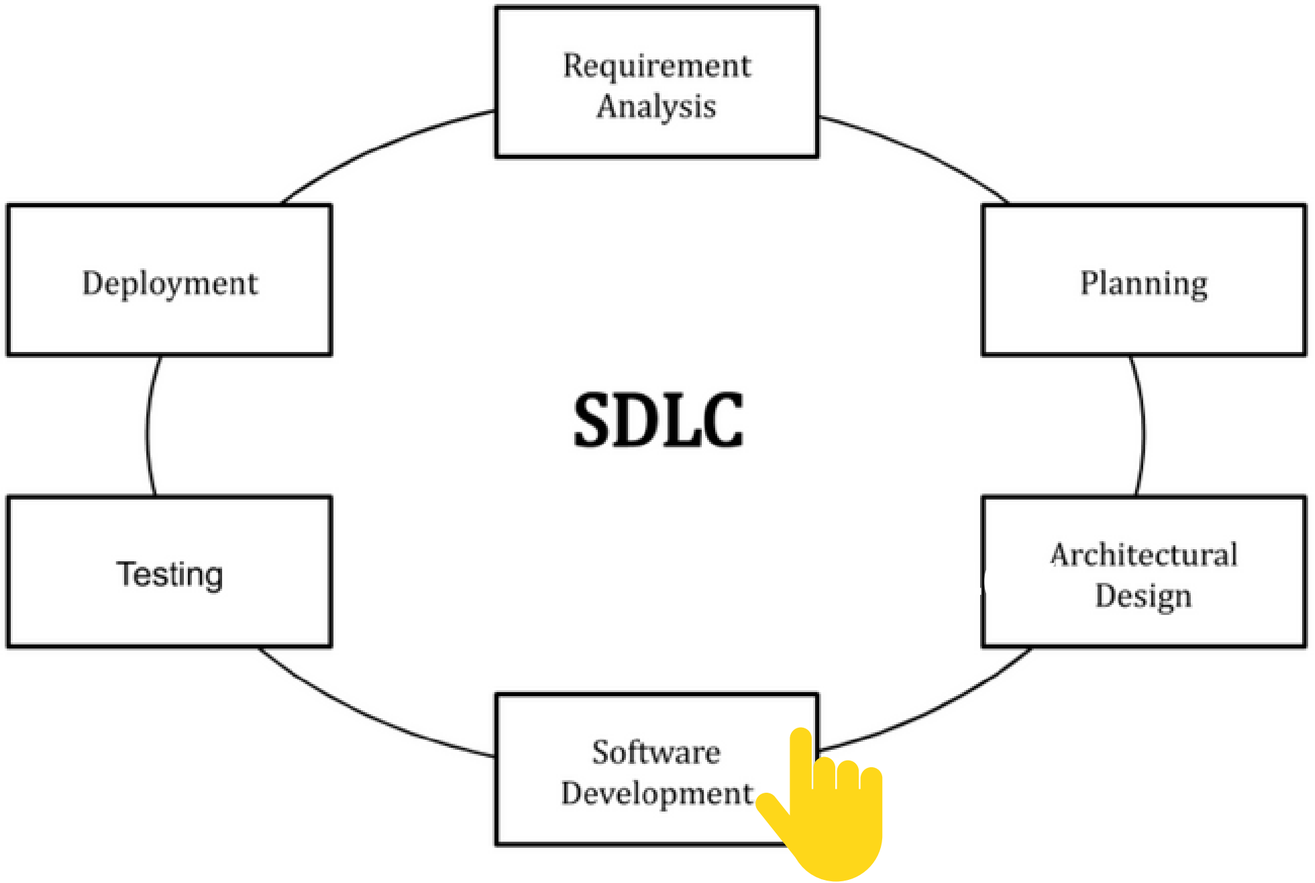


SOFTWARE ARCHITECTURE

AI class

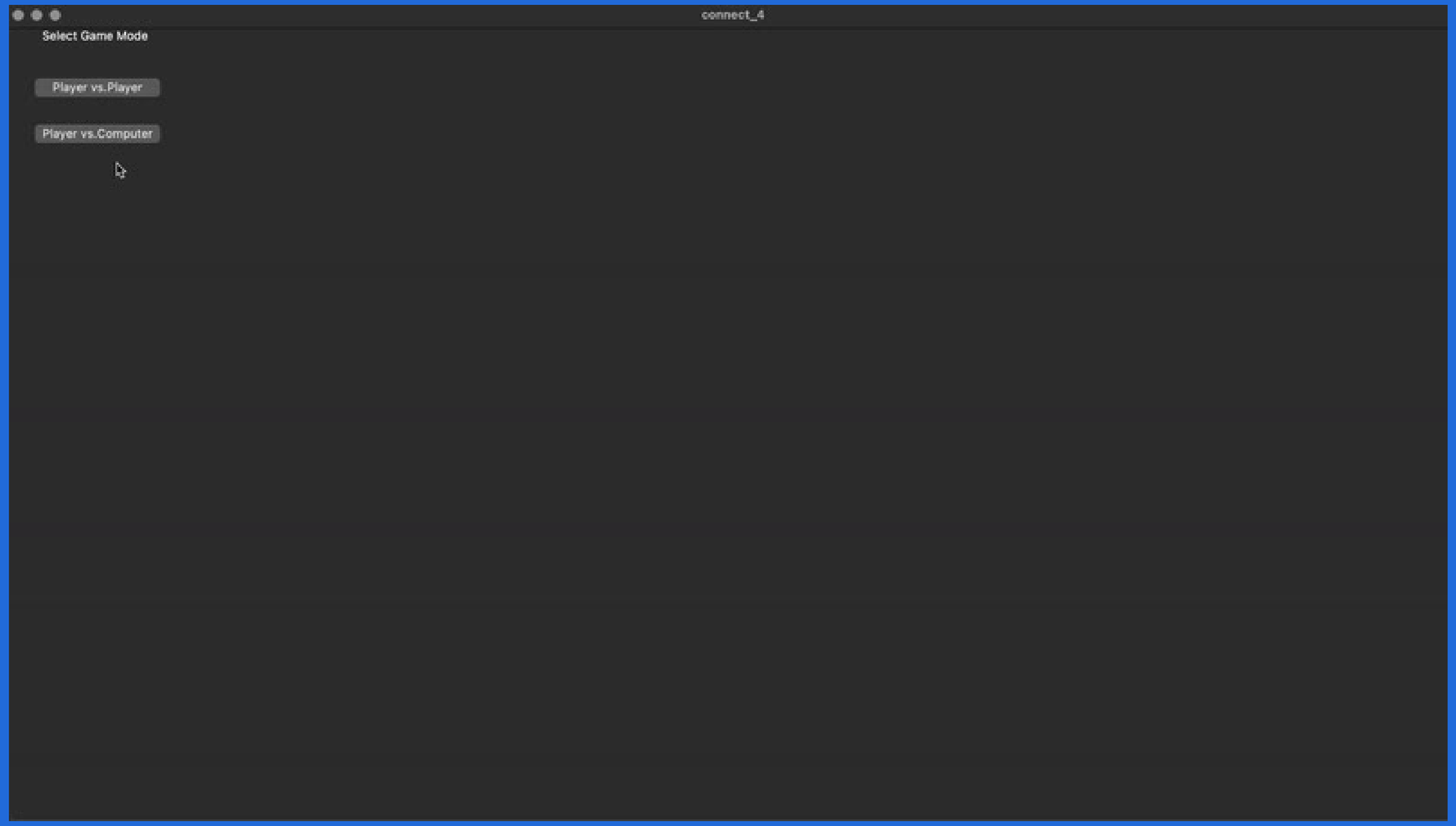


Software Development

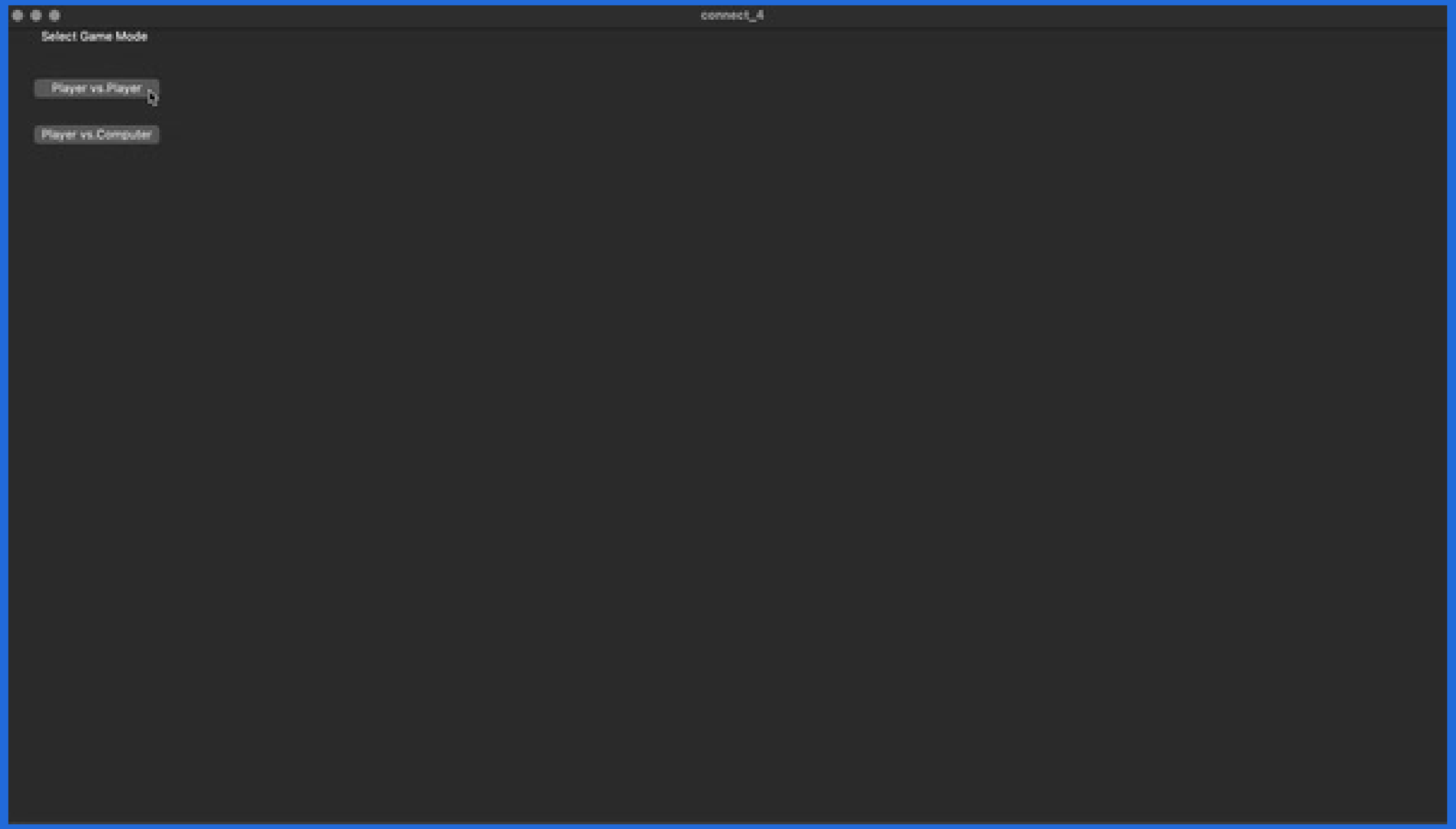


SOFTWARE DEVELOPMENT

Demo: Choose game

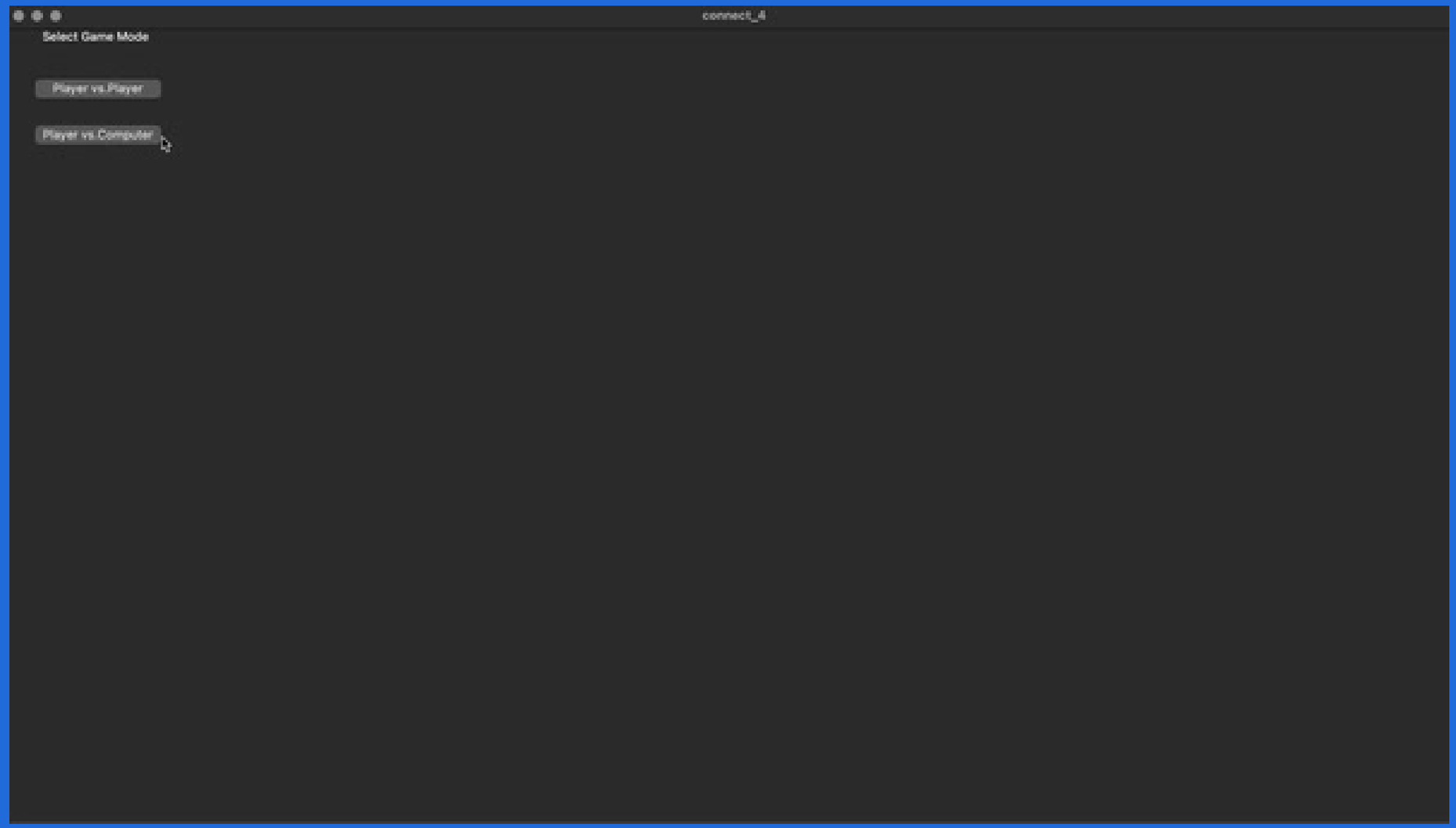


Demo: player-vs-player

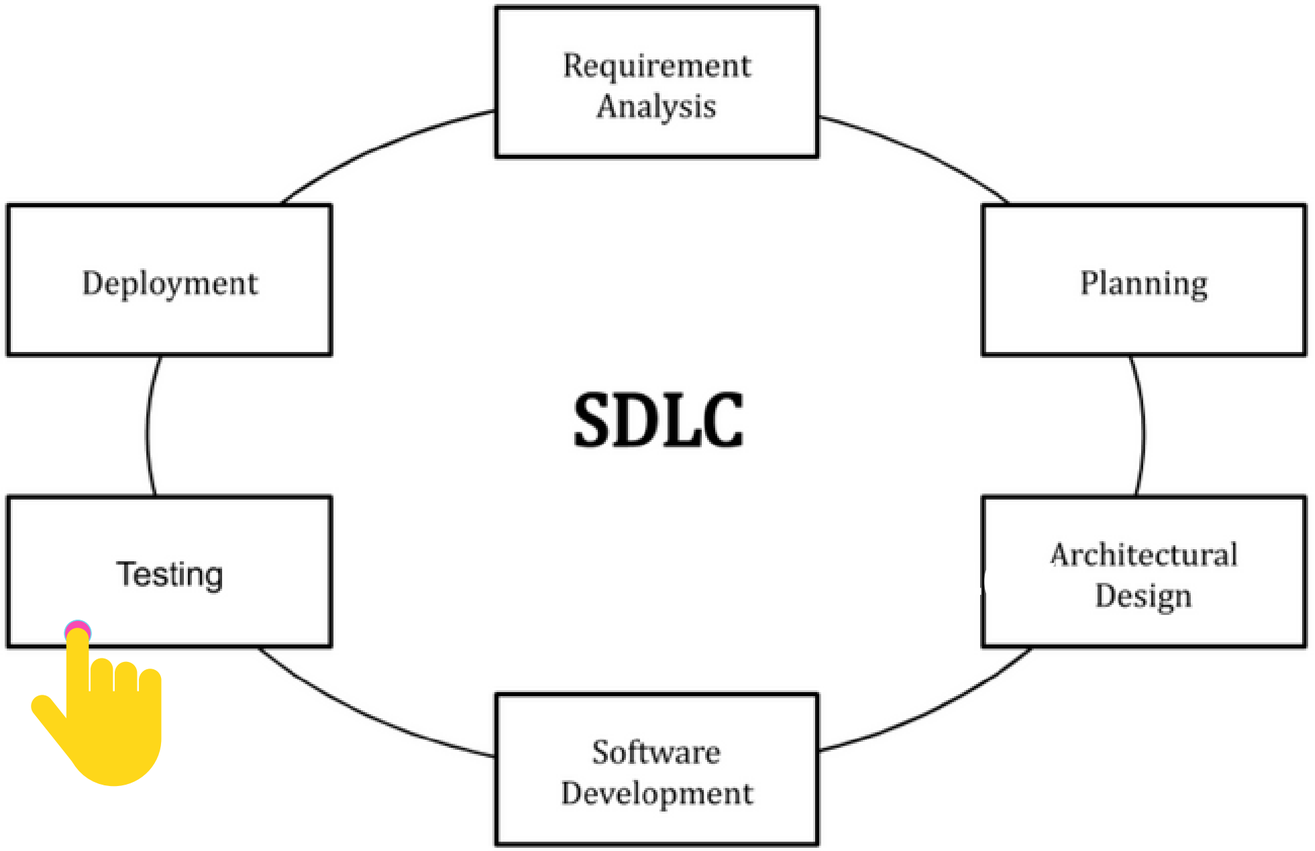


SOFTWARE DEVELOPMENT

Demo: player vs AI



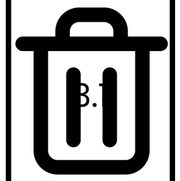
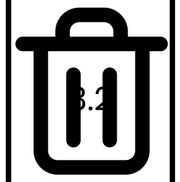
Testing


Requirement Analysis
Planning
Architectural Design
Software Development
Testing
Deployment
SDLC

Black-Box Test Result

Test ID	Description	Expected Result
1.1	(vs person mode) If the user inputs the column to put the pack, it should insert to the bottom of the row based on the gravity factor, then the other player adds another pack in the same manner.	The visualization of the game board with 2 newly added packs.
1.2	(vs AI mode) If the user inputs the column to put the pack, it should insert to the bottom of the row based on the gravity factor, then the AI adds another pack in the same manner.	The visualization of the game board with 2 newly added packs.
1.3	The player turn should come once in every two turns	User adds the pack onto an odd number of packs or even number of packs depending on the turn.
1.3.1	(vs player mode) The player1 inputs in odd turn	Input happens on the 1st, 3rd, 5th ... 2N-1th time
1.3.2	(vs player mode) The player2 inputs in even turn	Input happens on the 2nd, 4th, 6th ... 2Nth time

Black-Box Test Result

Test ID	Description	Expected Result
	If there are 4 sequential packs of the same color in horizontal, vertical or diagonal gamespace, it should stop the game and display who wins the game.	Message saying who is the winner along with the final result of the game board.
	If the user inputs the invalid data, it should reject the input and let the user do it again	Error message and another input try.
	If users inputs letters other than column name (A, B, C, D, E, F and G), rejects the input.	Error message and another input try
	If the user inputs the column which is already full, rejects the input.	Error message and another input try

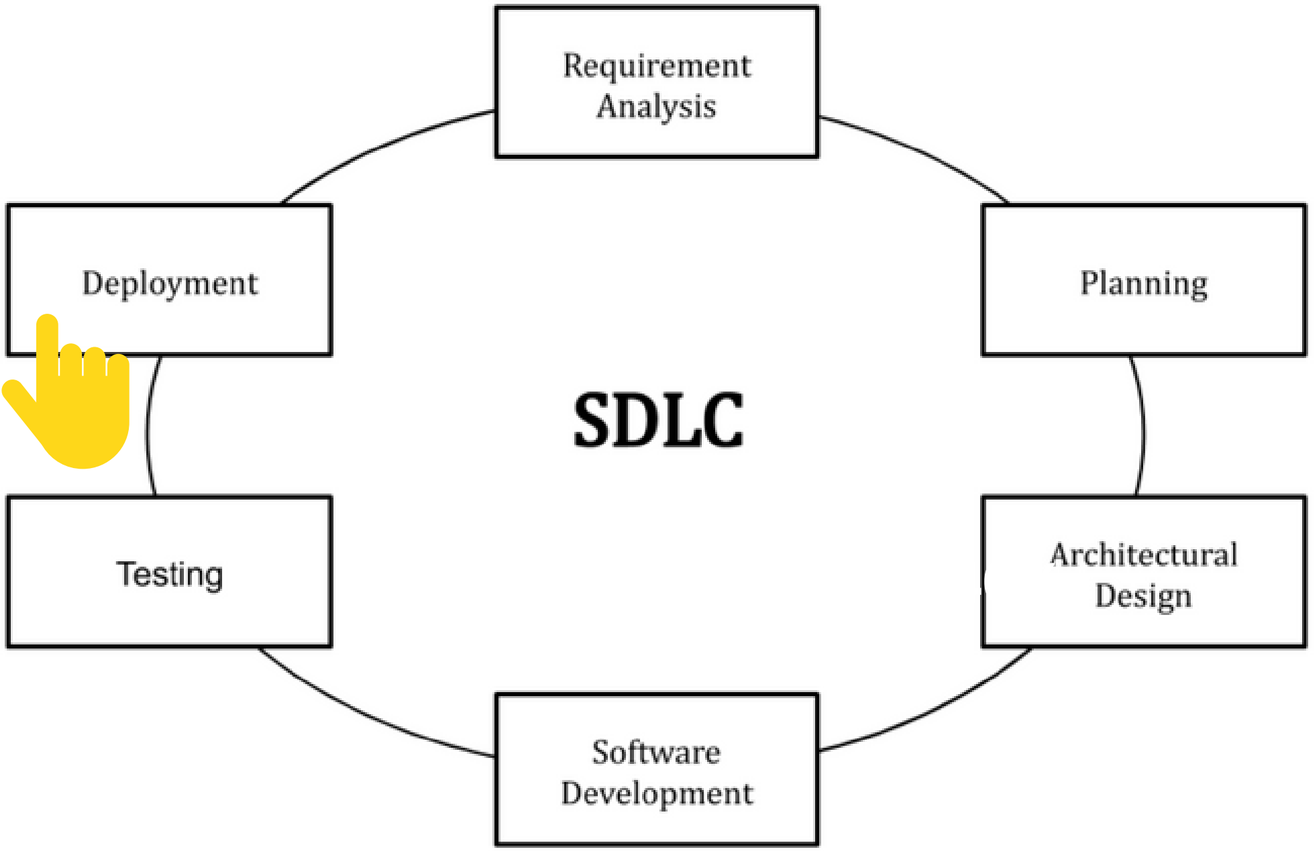
AI Module Unit Test Result

```
state1 = [
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [1,0,0,0,0,0,0],
    [1,2,0,0,0,0,0],
    [1,2,0,0,0,0,0]]

state2 = [
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [2,2,0,0,0,0,0],
    [1,1,1,0,0,0,0]]

state3 = [
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [1,1,0,0,0,0,0],
    [2,1,1,0,0,0,0],
    [2,2,2,1,0,0,0]]

state4 = [
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,1,1],
    [0,0,0,0,1,1,2],
    [0,0,0,1,2,2,2]]

state5 = [
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [0,0,0,2,0,0,0],
    [0,0,0,1,0,0,1],
    [0,0,0,1,0,0,2],
    [0,0,0,1,0,0,2]]

state6 = [
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [0,0,0,0,0,0,0],
    [1,0,0,0,0,0,0],
    [2,0,0,0,0,0,0],
    [2,0,0,2,1,1,1]]
```

```
for i, state in enumerate(state_list):
    tree = Tree(state, False, 5)
    tree.generate_game_tree()
    row, col = tree.make_best_move(tree.root)
    print(f"{i}th promising row:{row} and col:{col}")
```

```
0th promising row:2 and col:0
1th promising row:5 and col:3
2th promising row:5 and col:6
3th promising row:2 and col:5
4th promising row:5 and col:2
5th promising row:4 and col:3
```

Deployment


Requirement Analysis
Planning
Architectural Design
Software Development
Testing
Deployment
SDLC

Scan ME

Our app is on github!

Takeaways

01 **Communication**
Our daily chats can reduce the miscommunication throughout the project.

02 **Architecture for Integration**
The successful integration depends on the successful architecture of the software.

03 **Test it right, sleep well at night**
The more elaborate test is expected for the product of better quality.

Scan ME

Thank you for your attention!!