DEPARTMENT OF COMPUTER TECHNOLOGY

MADRAS INSTITUTE OF TECHNOLOGY

CS6110 OBJECT ORIENTED DESIGN AND ANALYSIS

MINI PROJECT

PAC-MAN GAME

SOFTWARE REQUIREMENT SPECIFICATION DOCUMENTATION

MENTOR: Dr. S. Neelavathy Pari

BY:

Kartikeyan TR - 2020503520

Suriyaa V - 2020503550

Guru Raman C - 2020503510

INDEX

1.Problem

1.1 Problem Statement:

Designing a game Pac-man to be played by one player where a player can sit and get engaged physically and mentally. Making various modifications to the game UI such as game theme and player avatar and the game maze

1.2 Objectives:

- To create a game using OOPS concepts
- Analyze various game algorithms
- To understand OOAD concepts better
- To create an interactive GUI
- To implement proper software design methodologies [70% design and 30% implementation]

2.SOFTWARE REQUIREMENT SPECIFICATION

2.1 Introduction

2.1.1 Purpose

The main purpose of this project is to create a PAC-MAN game using the OOAD concepts in JAVA (a OOPs language).

2.1.2 Scope

The software is application called Pac- man. It can used to play the retro-famous PAC-MAN game.

The application will also allow users to modify their avatar in the game and also change the theme of the game.

The application is meant for users who would like to play the PAC-MAN game but in a different style.

2.1.3 Intended Audience

The intended audience of this document would be pre teenage children. The SRS document can be used in any case regarding the requirements of the project and the solutions that have been followed. The document would finally provide a clear idea about the system that is being built.

2.1.4 Overview of Game

PAC-MAN is an action maze chase video game; the player controls the eponymous character through an enclosed

maze. The objective of the game is to eat all of the dots within the given timer that are placed in the maze while avoiding four colored ghosts. The player with the maximum points tops the leaderboard.

2.1.5 References

IEEE. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications. IEEE Computer Society, 1998.

2.2. Overall Description

2.2.1 Product Functions

This is the game so its most important function is that it is a source of entertainment.

2.2.2 Operating Environment

- This application (Game) is operated as an windows application.
- The application can be run as an application or also in browser.

2.2.3 User Documentation

The control scheme of the PAC-MAN game same as in the original PAC-MAN game. The controls are simple and user will be able to learn it within minutes.

- There will be a PAC MAN named sprite which will move in according to directions given by us.

- PAC-MAN will eat dots and there will be a score board which will show the number of dots eaten.

- PAC-MAN will have to survive against the enemies who will move randomly in the tunnels.
- When level will be completed next level would come.
- There would be 3-4 levels of the game which will have different type of mazes. To make game more excited we shall make 3 lives of PAC-MAN.
- Score, levels and lives will be visible on the screen.
- There will be buttons on screen to control some actions of game.
- There will be background sound of PAC-MAN Game.
- There will be all of the features of an arcade game.
- There will be a High Score board which will be shown when game will end.
- There will be tunnel through which PAC-MAN can pass walls and can appear from the other side.

2.2.4 Assumptions and Dependencies

We assume that the user has an windows operating system installed in his/her PC.

2.3 Specific Requirements

2.3.1 Functional Requirements

The following were the functional requirements identified:

- The system offers a graphical user interface (GUI) that allows a player to attempt to complete a challenge as a game, the challenge is to locate all hidden objects to be eaten in the visual representation of the environment
- The system should enable a menu-item option for the player to select different difficulty levels. The levels consist of larger environments of the same type. The density of the game levels in an environment must be roughly the same across difficulty levels.
- The system should enable a menu-item option for the player to select a different environment There should be 3 environments: a square-grid, hexagonal grid, and a graph layout.
- The system should display a timer in seconds. Starting the game starts the countdown of the timer. The game does not end if the timer expires
- The system should display the total number of food available requiring localization in the environment output.

2.3.2 Non-Functional Requirements

- **Performance** – Since game is light-weight the performance will be good even in a low spec PC.
- **Safety** – The game occupies only about 30mb of RAM. So the game will no overheat and wont cause damage to the hardware in which it is being played.
- **Security** – The game doesn't collect any personal data. All the data are actually being locally stored in the computer so no data leak is possible.
- **Software Quality –** The game runs without any frame drop even in low-end laptop. Since the application is light-weight,

the performance will be solid. Also the game refreshes at a rate equal to the screen refresh-rate of the computer so there will not be any response delay in the software side although the response time mainly depends on the hardware.

2.3.3 External Interface

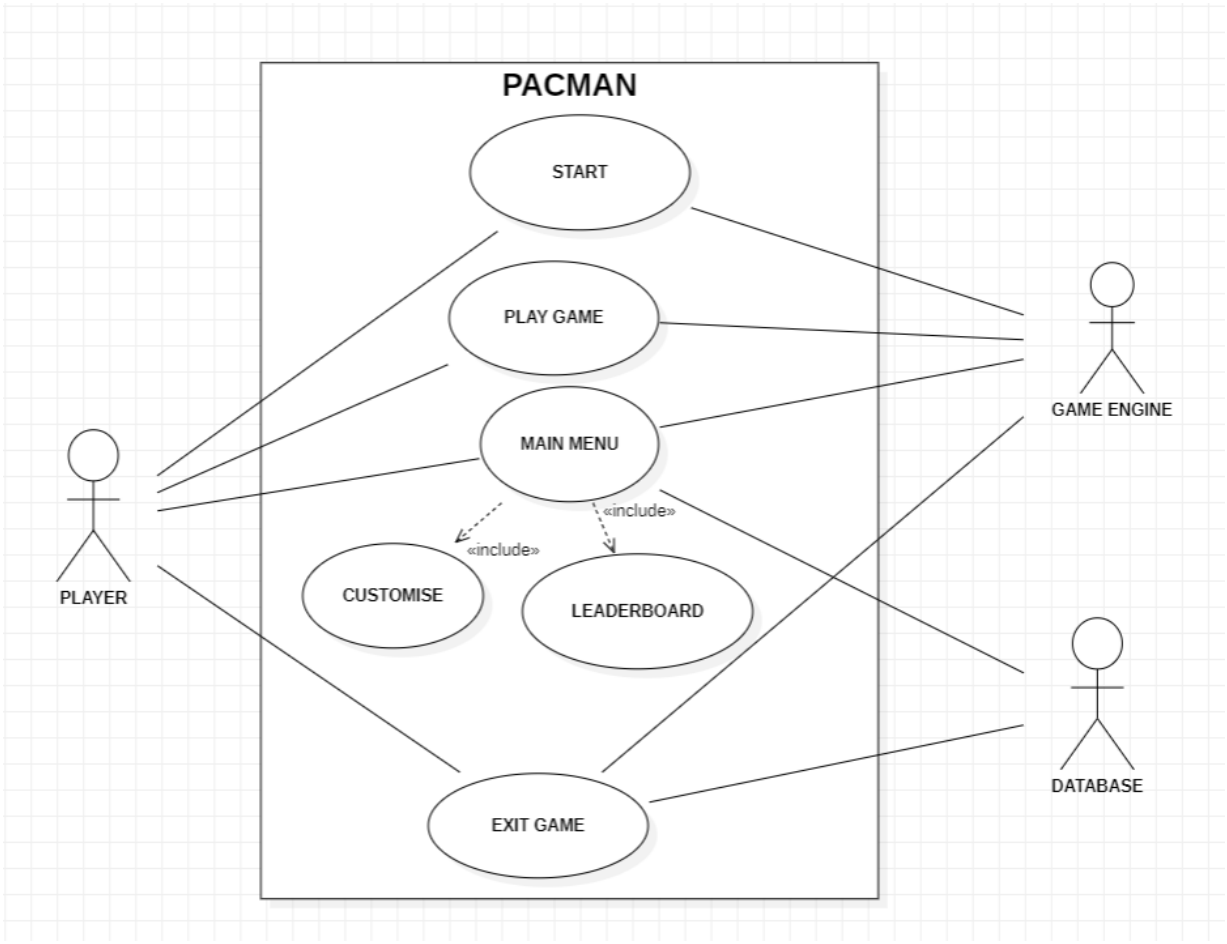
3.3.1 Hardware Interfaces

Hardware Environment	Any OS
System Configuration	8 GB RAM and 256 GB storage
Operating System	Windows 10/11

3.3.2 Software Interfaces

Front End	Java
Back End	Java, SQL

5.USE CASE DIAGRAM



```
UC 1: Start
Scope: Game System
Level: Player(User)
Goal in context: To start the game
Actors:
Primary: Player - Starts the game
Secondary:  Game Developer
Basic Flow:
1. Player arrives to play the game.
2. He starts the game first.
```

```
UC 2: Play Game
Scope: Game System
Level: Player (User)
Goal in context: To Play the game
Actors:
Primary: Player - Plays the game
Secondary:  Game Developer
Basic Flow:
1. Player plays the game.
```

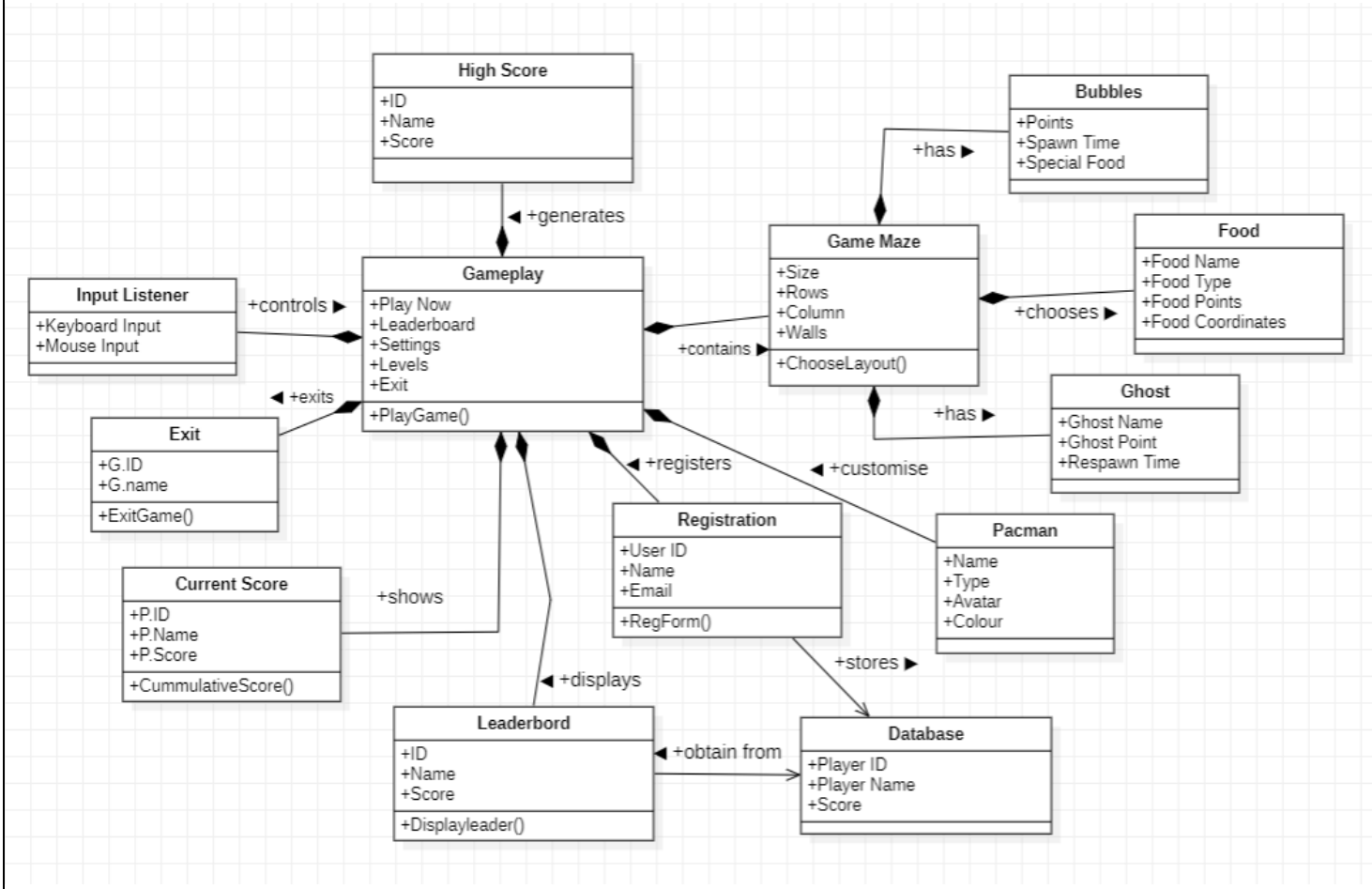
```
UC 3:  Main Menu
Scope: Game System
Level: Player (User)
Goal in context: Contains all the features of the game.
Actors:
Primary: Player - Chooses the options and customizes
the Agame to his convenience
Secondary:  Game Developer , Database - All the game
settings/customizations are stored in this
Basic Flow:
1. Player clicks the menu.
2. He can choose grid layouts.
3. He can change the player avatar.
4. Leaderboard can also be viewed.
```

```
UC 4: Customise
Scope: Game System
Level: Player (User)
Goal in context: Customize the game options and settings
Actors:
Primary: Player - Chooses the options and
customizes the game to his convenience
Secondary:  Game Developer , Database - All
the game settings/customizations are stored in this
Basic Flow:
1. He can choose grid layouts.
2. He can change the player avatar.
```

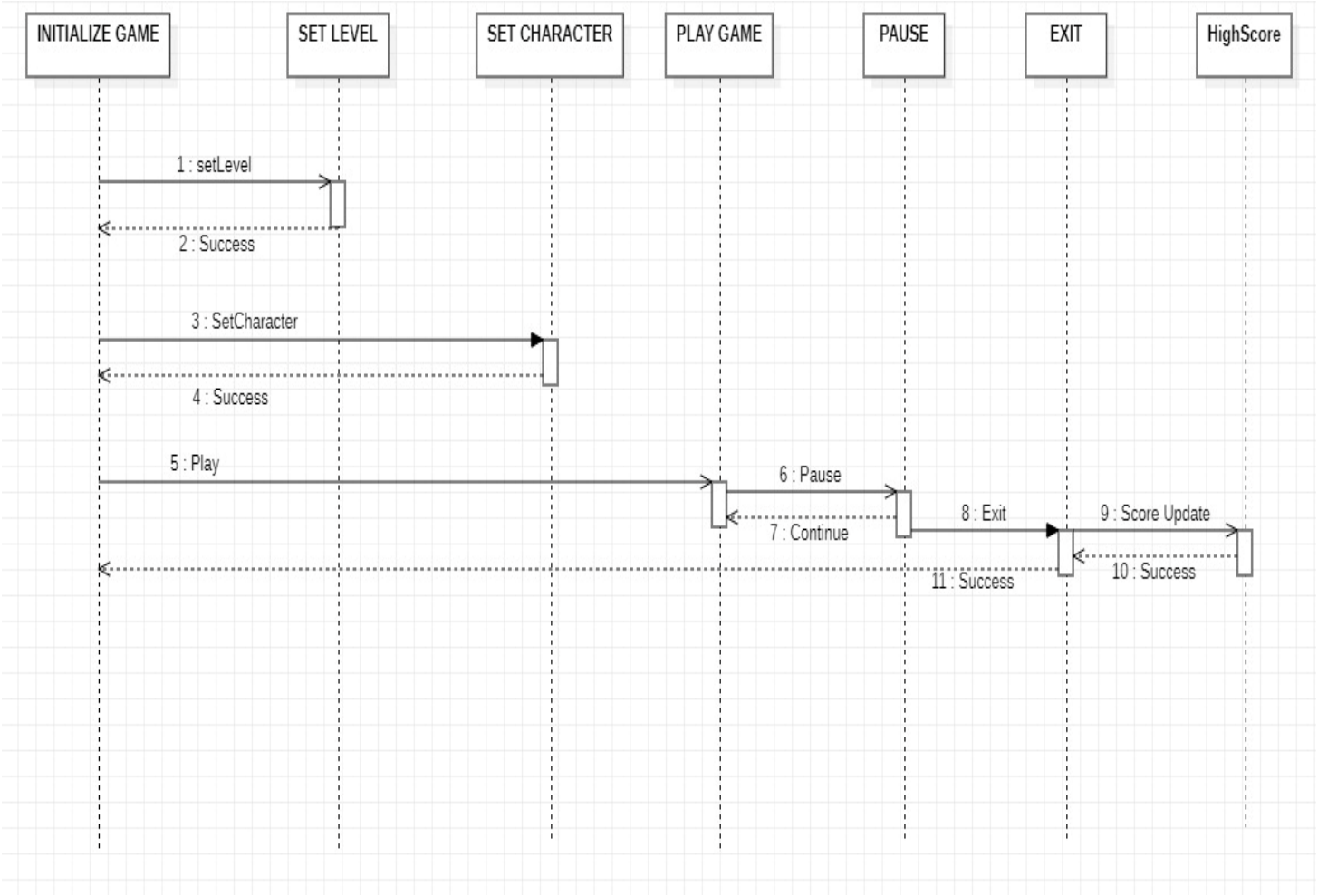
```
UC 5: Leaderboard
Scope: Game System
Level: Game Developer
Goal in context: Scores of all the
players who have played the game
Actors:
Primary: Game Developer , Database - All the
games high scores are stored in this database.
Secondary: Player - The score of the current
player is stored in the leaderboard.

Basic Flow:
1. Scores of different games are stored and
retrieved from the database.
2. Players can view who is the best player
in this game.
```

```
UC 6: Exit Game
Scope: Game System
Level: Player (User)
Goal in context: Exiting the game
Actors:
Primary: Player - Exits the game whenever needed by him
Secondary: Game Developer , Database
Basic Flow:
1. Player exits the game after it ends.
2. Players can also leave the game midway.
```

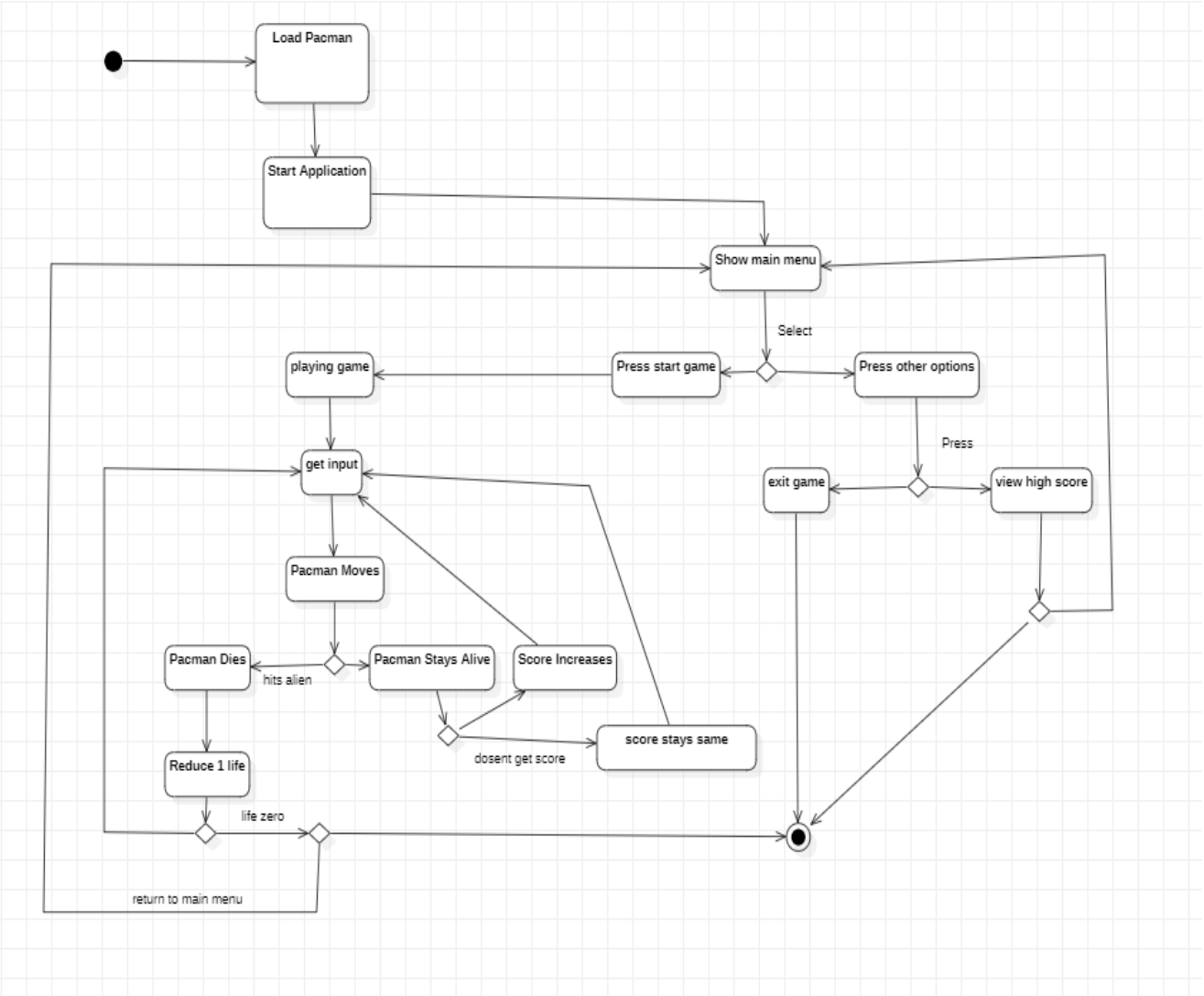
6. CLASS DIAGRAM



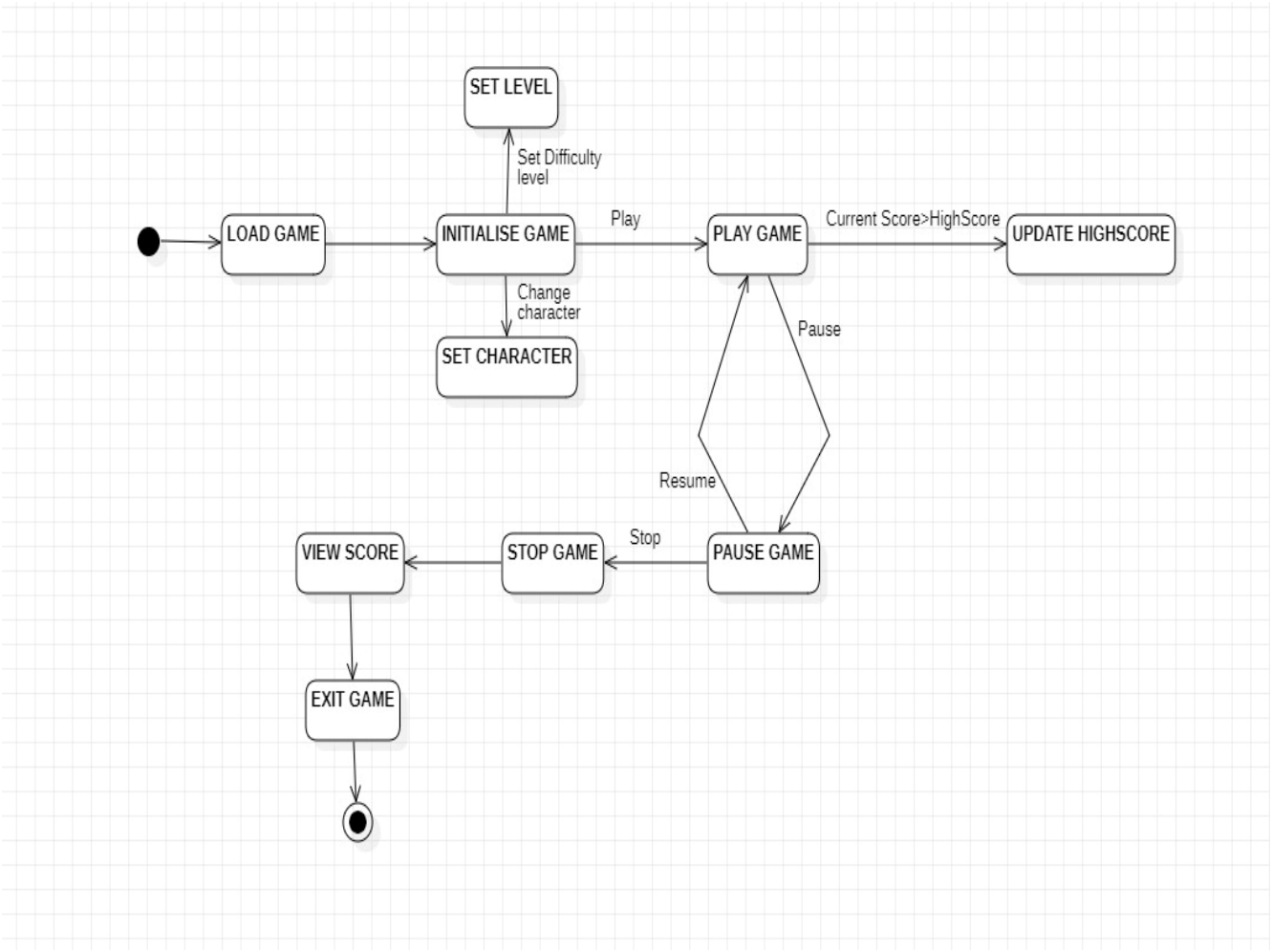
7.SEQUENCE DIAGRAM



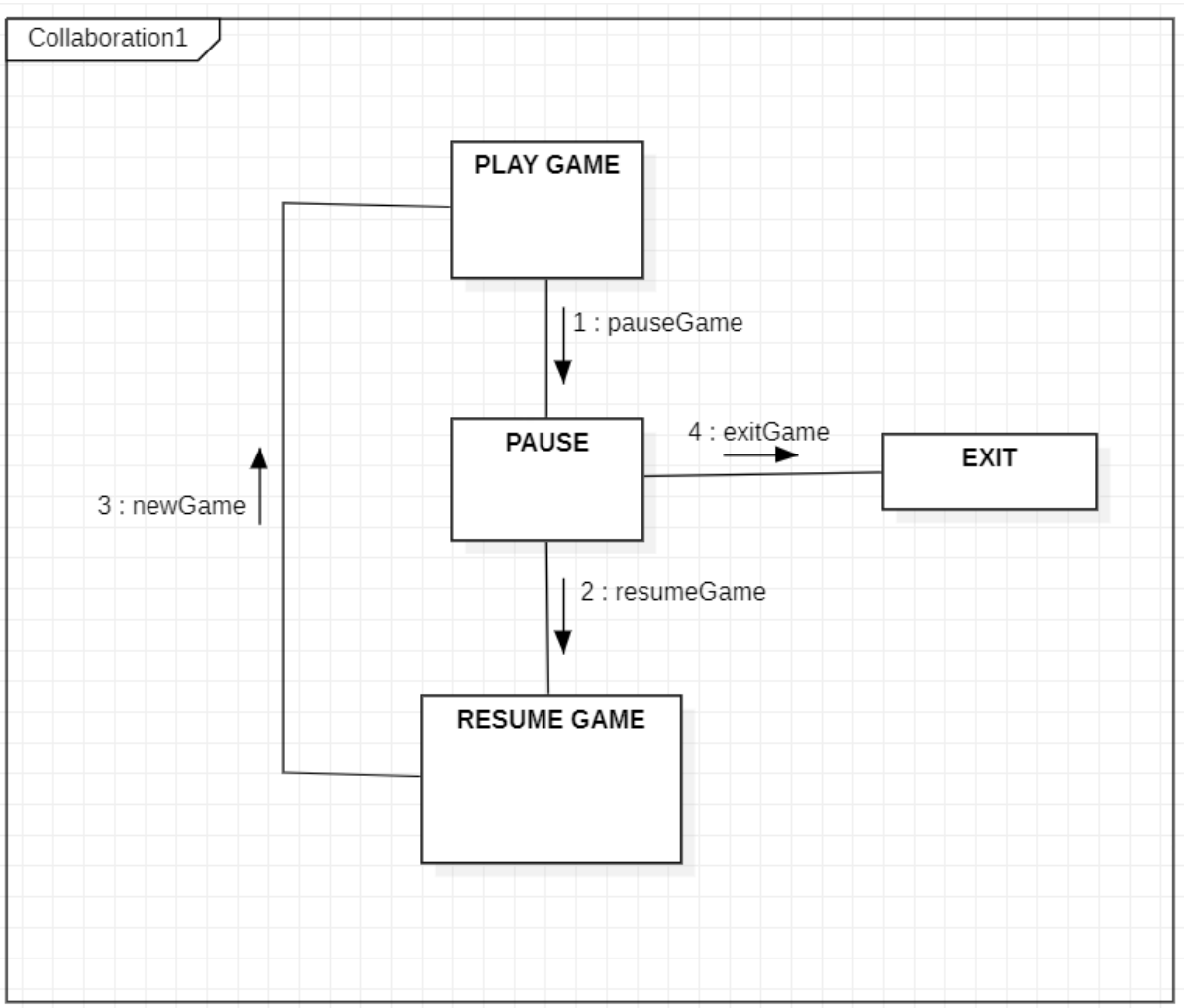
8.ACTIVITY DIAGRAM

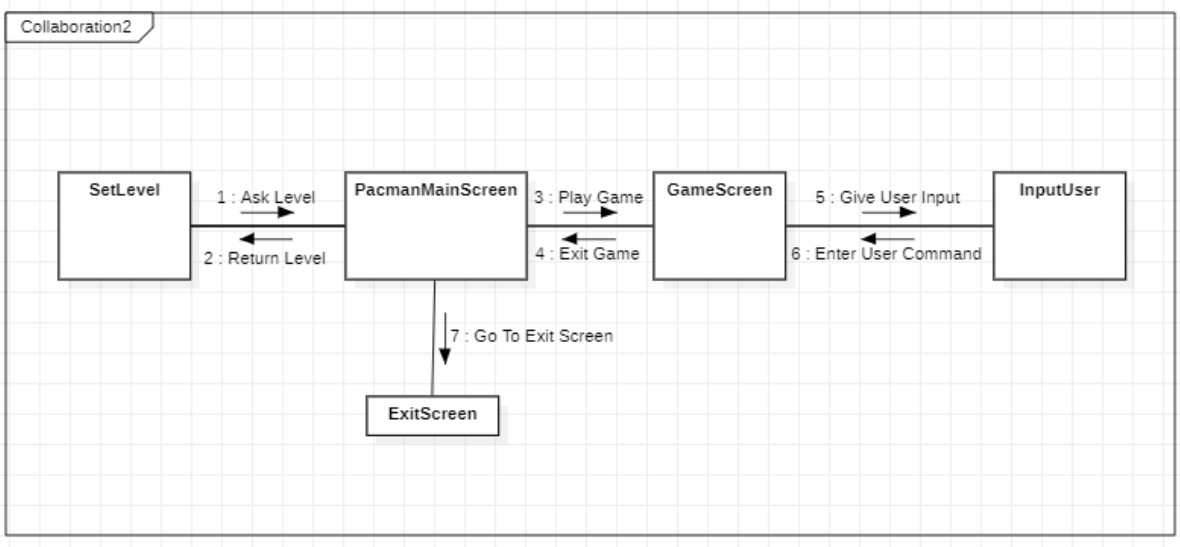


9.STATE DIAGRAM

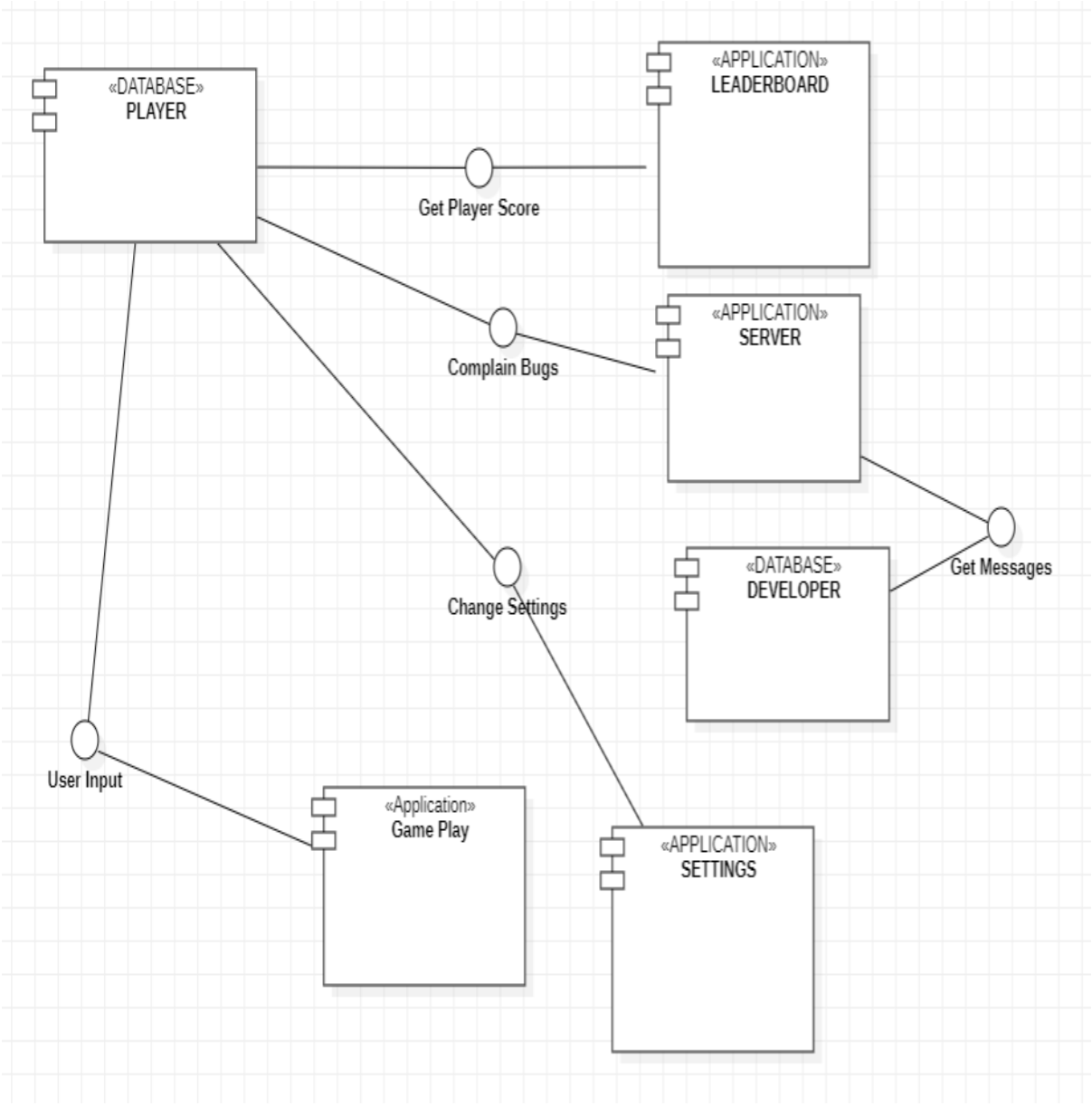


10.COMMUNICATION DIAGRAM

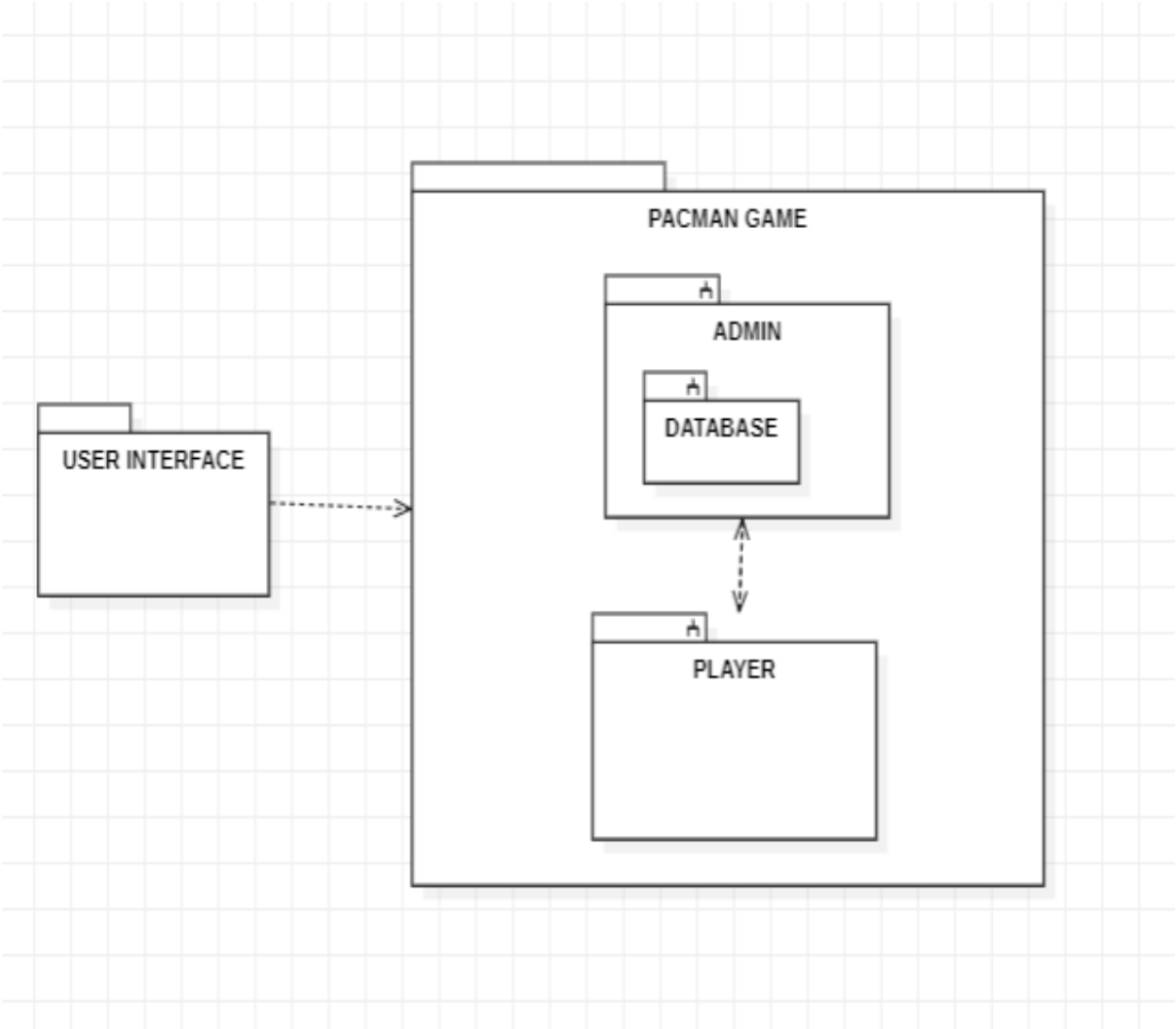




11.COMPONENT DIAGRAM



12.PACKAGE DIAGRAM



13.DEPLOYMENT DIAGARM