Assignment
Stage One Submission
2805ICT/3815ICT/7805ICT

Student name: Ler Theng Loo Student ID: s5212872 Enrolled Course Code: 2805 ICT

Student name: Luke Fisher Student ID: s5220734 Enrolled Course Code: 2805 ICT

Student name: Malachi Klar Student ID: s2937839 Enrolled Course Code: 2805 ICT

Student name: Richard Budden Student ID: s5097875 Enrolled Course Code: 2805 ICT

Table of Contents

1.0 Project Planning and Documentation

1.1 Time Schedule

The table below shows the tasks done by each team member, and also the expected and actual hours taken to complete the tasks.

Task		Plan				Actual		
#	Task Name	Student	Planed Time	Cumulative Time	Finished Date	Time	Cumulative Time	Finished Date
1.0	**Project plan**							
1.1	Time schedule	Richard	1hr	1hr	30/08/21	**1hr**	**1hr**	**06/08/21**
1.2	Working hours	Richard	10min	1h10m	30/08/21	**1h 10m**	**1h 10m**	**06/08/21**
1.3	E & C table	Richard	10min	1h20m	30/08/21	**10min**	**1h 20m**	**06/09/21**
		Luke	10min	1h30m	30/08/21	**10min**	**1h 30m**	**06/09/21**
		LT Loo	10min	1h40m	30/08/21	**10min**	**1h 40m**	**06/09/21**
		Malachi	10min	1h50m	30/08/21	**10min**	**1h 50m**	**06/09/21**
		Task Total	40min	N/A	30/08/21	**40min**	**1h 50m**	**06/09/21**
1.4	VCS	Richard	10min	2hrs	30/08/21	**20min**	**2h 10m**	**06/09/21**
2.0	**Requirements Analysis**							
2.1	Identify: Functional Requirement	Luke	2hr	4 hr	15/08/21	**1h 30m**	**2h 40m**	**23/08/21**
2.2	Identify: Non-Functional Requirements	Luke	2hr	6 hr	15/08/21	**1h 30m**	**4h 10m**	**23/08/21**
2.3	Use case diagram	Richard	5hr	12 hr	16/08/21	**4hr**	**8h 10m**	**24/08/21**
2.4	Full-case description	Richard	1hr	14 hr	17/08/21	**1hr**	**9h 10m**	**25/08/21**
		LT Loo	1hr	15 hr	18/08/21	**1hr**	**10h 10m**	**25/08/21**
2.5	Use case traceability matrix	Richard	2hr	17 hr	18/08/21	**2hr**	**11h 10m**	**26/08/21**
3.0	**Design and software architecture**							
3.1	Class diagram	LT Loo	3 hr	20 hr	20/08/21	**4 hr**	**15h 10m**	**01/09/21**
		Malachi	2 hr	22 hr	20/08/21	**6 hr**	**21h 10m**	**02/09/21**
3.2	Sequence Diagram	Malachi	3 hr	25 hr	25/08/21	**4 hr**	**25h 10m**	**06/09/21**
3.3	Activity diagram	Luke	1 hr	26 hr	25/08/21	**1 hr**	**26h 10m**	**06/09/21**
3.4	C&C View	Luke	1 hr	27 hr	28/08/21	**1 hr**	**27h 10m**	**06/09/21**
3.5	Implementation style view	LT Loo	3 hr	30 hr	29/08/21	**4 hr**	**30h 10m**	**06/09/21**
3.6	Deployment style view	Malachi	1 hr	31 hr	30/08/21	**3 hr**	**33h 10m**	**06/09/21**

4.0	Video development							
4.1	Video	Malachi	1 hr	32 hours	01/09/21	**2 hr**	**35h 10m**	**06/09/21**
4.2	Cross platform	Malachi	10 min	32h 10m	01/09/21	**10 min**	**35h 20m**	**04/09/21**
4.3	Start up page & Help page	LT Loo	3hr	35h 10m	22/08/21	**3hr**	**38h 20m**	**24/08/21**
4.4	Pacman and ghost	LT Loo	4hr	39h 10m	25/08/21	**5hr**	**43h 20m**	**26/08/21**
4.5	Game program - Game Page - Predefined map	LT Loo	7hr	46h 10m	28/08/21	**8hr**	**51h 20m**	**30/08/21**

1.2 Total working hours

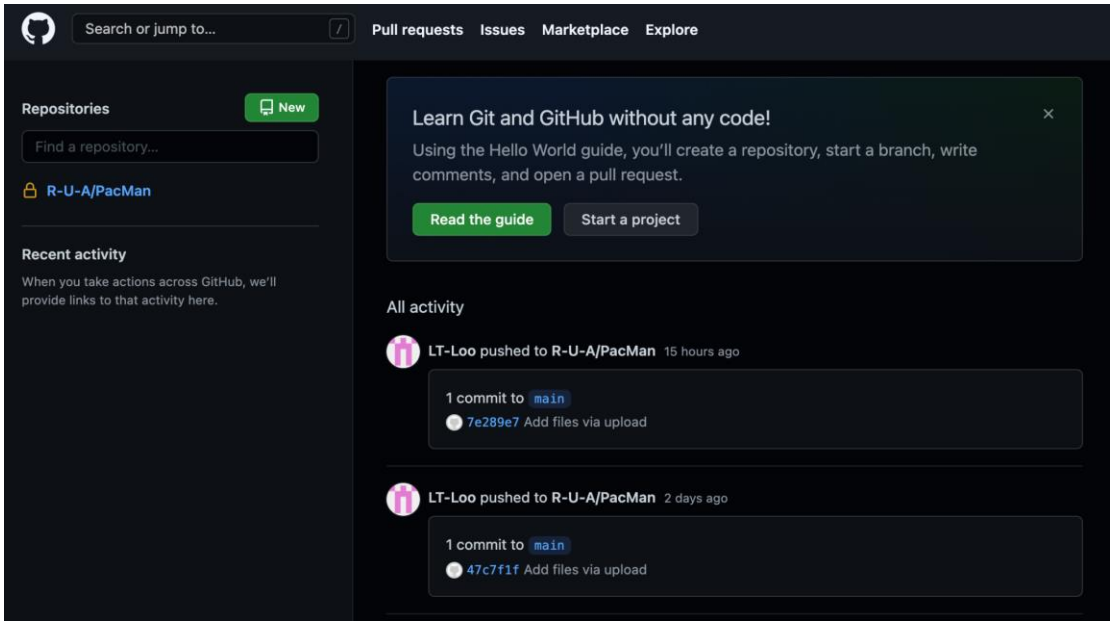
Student Name (#ID)	Plan (hours)	Actual (hours)
Ler Theng Loo (s5212872)	21.17	**25.17**
Luke Fisher (s5220734)	6.17	**5.17**
Malachi Klar (s2937839)	6.33	**12.33**
Richard Budden (s5097875)	9.5	**8.5**
Total working hours	43.17	**51.17**
Average working hours per person	10.79	**12.79**

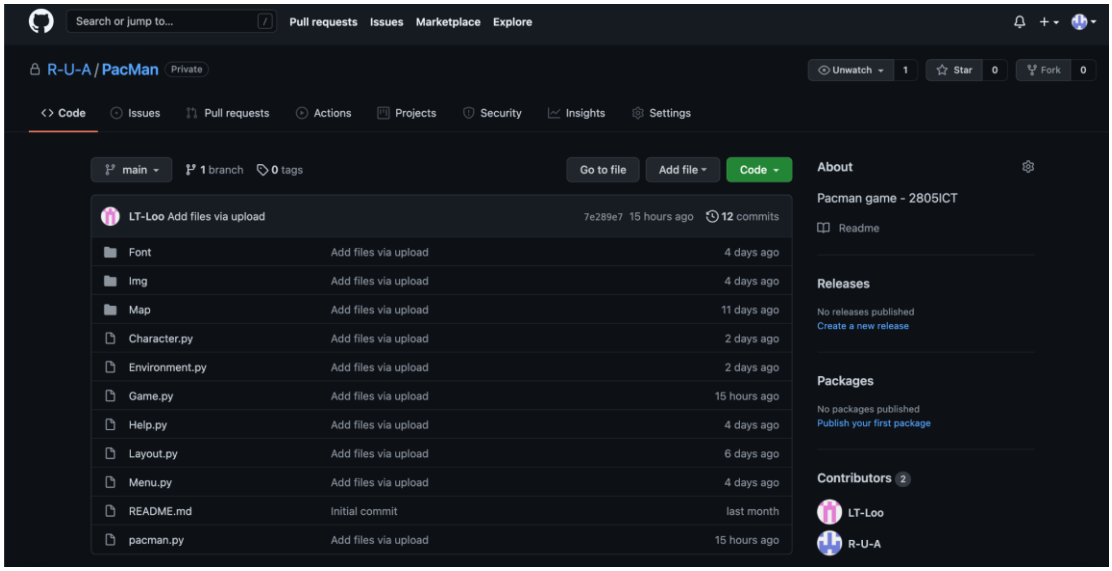
1.3 Effort and contribution table

Student	Effort Level (Rating from 0 – 5)	Contribution Level (Rating from 0 – 5)	Justification
Richard	5	5	Great communication and contribution, produced an exceptional level of work for the report and helped others in the team.
Ler Theng	5	5	Strongest with code, developed prototype, greatest contributor to dev, efficient worker, excellent team member and communicator.
Luke	5	5	Excellent team member, great communication, worked effectively and efficiently with significant contribution.
Malachi	5	5	Great team member, efficient and effective worker, excellent contribution to the project.
Total	20	20	

1.4 Version Control System

In this project, we are using Git as a version control system (VSC) to manage the source code development of the Pac-man game.





2.0
Requirements Analysis

In order to sufficiently analyse the requirements necessary for the project, both functional and non-functional requirements are identified and justified to define the functionality, design constraints and quality of the product. After making an established set of requirements, use cases and a traceability matrix are constructed to display and validate the requirements.

2.1 Functional requirements

The table below shows the identified requirements for the Pac-man system.

Requirement Identifier	Functional Requirement
RQ1	Upon execution, the program shall automatically display the title screen with “PAC-MAN” and the logo when started, followed by the year and course code.
RQ2	The user shall control the game with the arrow keys on the keyboard to navigate Pac-man.
RQ3	The system should allow the user to exit the game via a displayed ‘QUIT’ button (in the Game Page). If button is pressed at the Start Up Page, terminates the program.
RQ4	The system should allow the user to pause in-game via a ‘PAUSE’ button, then resume the game via ‘RESUME’ button.
RQ5	The system shall provide a configuration setting when the ‘Map’ button is pressed, where the user can select the normal predefined maze and randomly generated mazes, pressing ‘OK’ to confirm and return to the title screen.
RQ6	The game shall be started when the user presses a dedicated ‘START’ button on the Start Up Page.
RQ7	The system shall function like the original pac-man game by Namco in 1980.
RQ8	The system shall display an end board on the Game Page when the game is over, two varying on the result of the game. A ‘victory’ board when the conditions are met, otherwise a ‘game over’ board. Each board shall include two functions, ‘PLAY AGAIN’ button to start another game and ‘BACK TO MENU’’ to return the user to the title screen.
RQ9	The system shall display student ids’ and names within the game title screen.
RQ10	The system shall allow the user access to an instruction manual when a dedicated ‘HELP’ button is pressed. A ‘BACK’ button to get the user back to the title screen.
RQ11	The system shall allow the user to return to the previous state within the menu interface with a dedicated ‘BACK/OK’ button.

The table below shows the identified requirements for the Pac-man game. The requirements focus more on the execution of the game.

Requirement Identifier	Pac-man Functional Requirement
RQ7.1	Some ghosts shall move in a random direction within the maze system. While some ghost sprites shall move in a dedicated path towards the pac-man sprite.
RQ7.2	The system shall stop when pac-man touches a ghost sprite, this will decrease the lives and cause another pacman sprite to reappear, all dots remain the same. All ghost sprites shall return to the original start location.
RQ7.3	The score shall increase as pac-man collects the dots through the maze system.
RQ7.4	The dots shall disappear after pac-man comes in contact with them.
RQ7.5	When pac-man contacts a ‘power pellet’ all ghost sprites touched by pac-man will disappear without causing the ‘killed by a monster function' and reappear in the ‘spawn’ location.
RQ7.6	The dedicated Pacman music shall play while the system is on, additionally the sound effects shall play when any sprite interacts.
RQ7.7	When initiated the system shall countdown for the game to start from 3 seconds.
RQ7.8	The game shall end when all ‘dots’ are collected by the users’ avatar (pac-man).
RQ7.9	The game shall end when all ‘lives’ are lost.
RQ13	The system shall display animations for all characters when moving, and still images when stationary.

2.2 Non-functional requirements

The non-functional requirements are identified based on the FURPS+ format, allowing them to be developed further through the design process.

Functionality - The overall functionality of the system is to display the complete game system onto the visible computer monitor with all the required functions included while appropriately representing the theme and rules of the game, Pac-man.

Usability - The usability of the system should avoid complexity within the program and user interface with visually spaced out inputs with concise language clearly explaining the function of the feature, the overall system shall be displayed in a windowed view within the console allowing the user to exit or minimise the game system. The ease of use within the interface and game controls is required as it allows the user control the game as intended, therefore is required to use a mouse and keyboard as the input devices.

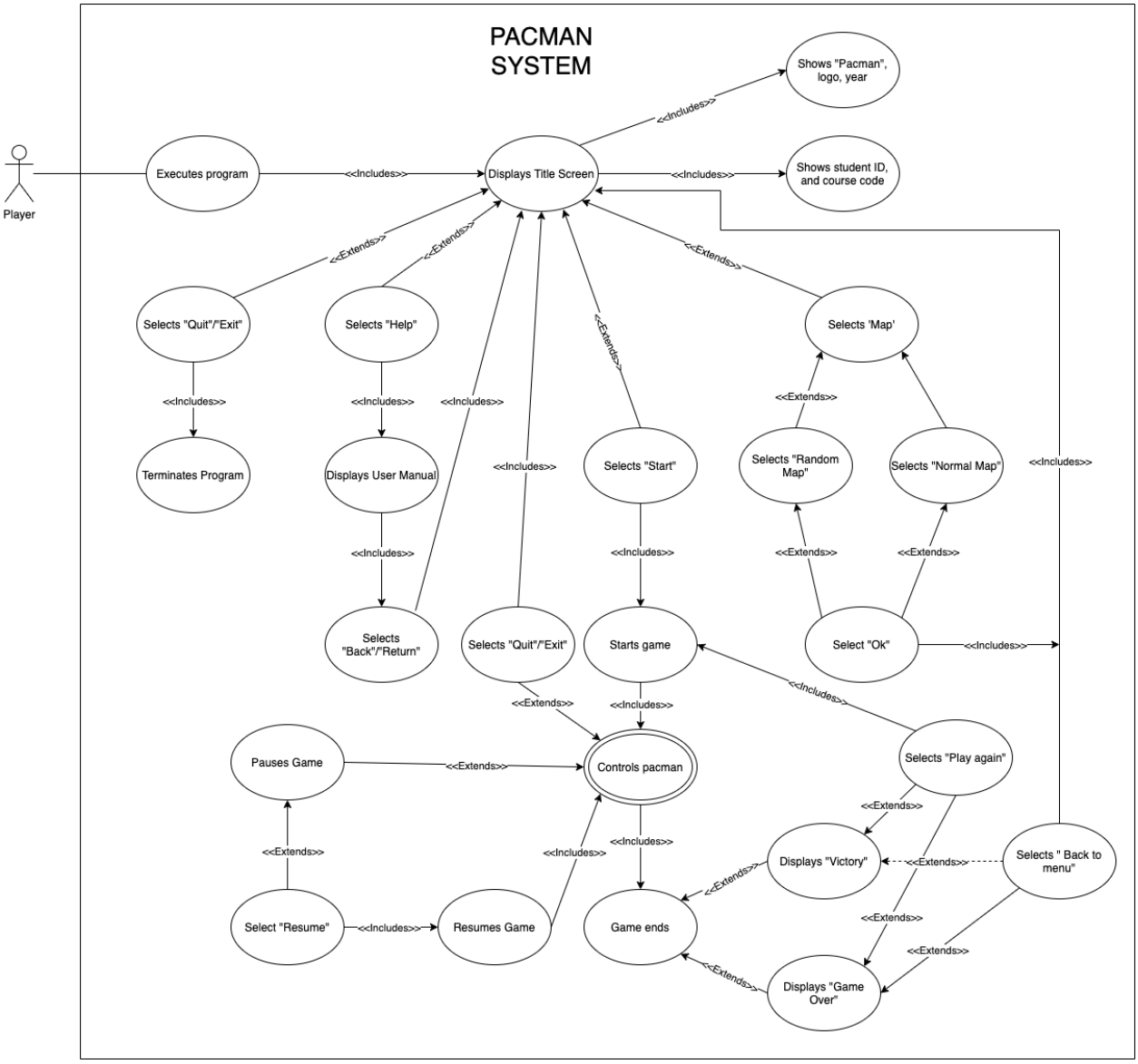
Reliability - The reliability of the system is vital, the software should function with no errors and have rectifiable errors if present. The program must consistently load properly with all features and functions operating as intended. Additionally, the program must consistently exit properly, stopping all functions and features within the game and displaying the appropriate console for further inputs from the user.

Performance - The performance of the system should allow the user to operate the game with no ‘lag’ present and overall response time. This includes the movement of Pac-man, it should appear to move with little to no delay to the user’s keystrokes. Additionally, the movement of the ghosts should display fluid movement within the developed maze system.

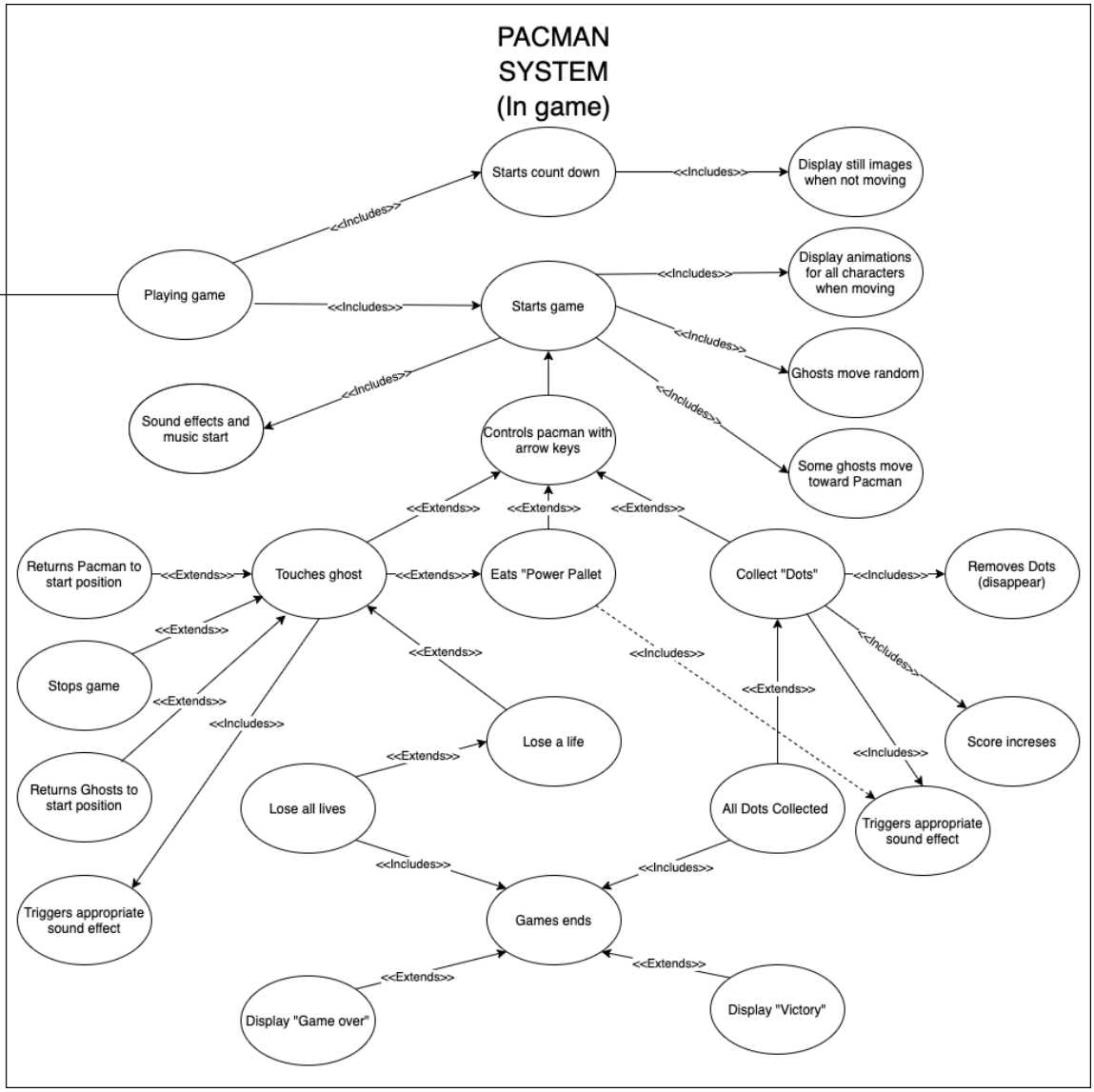
Supportability - The game should be compatible in multiple environments with no altercation. The system must be maintainable with a structured system allowing any change if necessary, additionally, the system must have the ability to adapt to any change made within the system with little to no error.

2.3 Use case diagram

A use case diagram is displayed below shows the **overall** interaction between the user and the Pac-man system, and the relationship between the use cases.



The use case diagram below focuses more on the interaction between the user and the Pac-man system **during the execution of the game**.



2.4 Full use case description

<table>
<tr><td>Use case name:</td><td colspan="2">Selects “Normal map”</td></tr>
<tr><td>Scenario:</td><td colspan="2">Player wants to select the default predefined map to play.</td></tr>
<tr><td>Triggering event:</td><td colspan="2">Clicking on “MAP” button from Start Up Page to navigate to Configure Page.</td></tr>
<tr><td>Brief description:</td><td colspan="2">This sets the map that the player then plays pacman on, if selected the program will initialize the default map to play.</td></tr>
<tr><td>Actors:</td><td colspan="2">Player</td></tr>
<tr><td>Related use cases:</td><td colspan="2">Can be selected from the Select map use case.</td></tr>
<tr><td>Stakeholders:</td><td colspan="2">N/A</td></tr>
<tr><td>Preconditions:</td><td colspan="2">The “MAP” button must be functioning.
Player must have selected “MAP” from the Start Up Page and the program must be able to navigate to the Configure Page.
The map option buttons must be functioning.</td></tr>
<tr><td>Postconditions:</td><td colspan="2">Normal map will be generated and displayed on the Game Page for the game to play on.
Game characters must be able to move on the path generated on the map.
The white dots and power pellets should be shown along the path of the maze.</td></tr>
<tr><td rowspan="5">Flow of Activities:</td><td>Actor</td><td>System</td></tr>
<tr><td>1. Player clicks on the “MAP” button (configure button)</td><td>1.1 System navigates to the Configure Page
1.2 System displays the Configure Page on the screen
1.3 System displays map options</td></tr>
<tr><td>2. Player selects “Default/Predefined Map” from map options</td><td>2.1 System shows that “Default/Predefined Map” has been selected by player</td></tr>
<tr><td>3. Player clicks on “OK” button to confirm selection</td><td>3.1 System saves player’s map selection
3.2 System navigates back to Start Up Page</td></tr>
<tr><td>4. Player clicks on “START” button to play game</td><td>4.1 System creates game
4.1.1 System creates game characters (Pac-man and ghosts)</td></tr>
</table>

		4.1.2 System extracts selected map from database 4.1.3 System interprets map and collect info of each path on the map 4.1.4 System creates and draws dots and map 4.2 System displays predefined map on Game Page
Exception Conditions:	1.1 Player does not select a map option. 3.1 Player does not confirm selection by clicking “OK” button. 4.1 Player does not choose map before selecting “START” button.	

2.5 Requirement - use case traceability matrix

The use case traceability matrix allows us to monitor requirements, checking that all are included within the program design. The table below shows the use case traceability matrix of the **first use case diagram (system diagram)**. Texts represent the use cases, whereas X’s indicate that the requirement is present/evident for the attached use cases.

Use case

RQ – ID	Display title screen	Shows course & students	Display logo, pacman, year	Select quit/exit (in menu)	Terminate program	Selects Help	Display user manual	Select return/back	Selects play	Starts game	Pause game	Selects resume	Resumes game	Selects map	Selects random map	Selects default map	Selects ok	Game ends	Display Victory	Display Game over	Selects Return to menu	Selects play again	Selects quit (in-game)	Control pac-man
RQ1	X	X																						
RQ2																								X
RQ3				X	X																		X	
RQ4											X	X	X											
RQ5														X	X	X	X							
RQ6									X	X														
RQ7																								X
RQ8																		X	X	X	X	X		
RQ9			X																					
RQ10						X	X																	
RQ11								X																

The table below shows the use case traceability matrix of the **second use case diagram (in-game requirements)**.

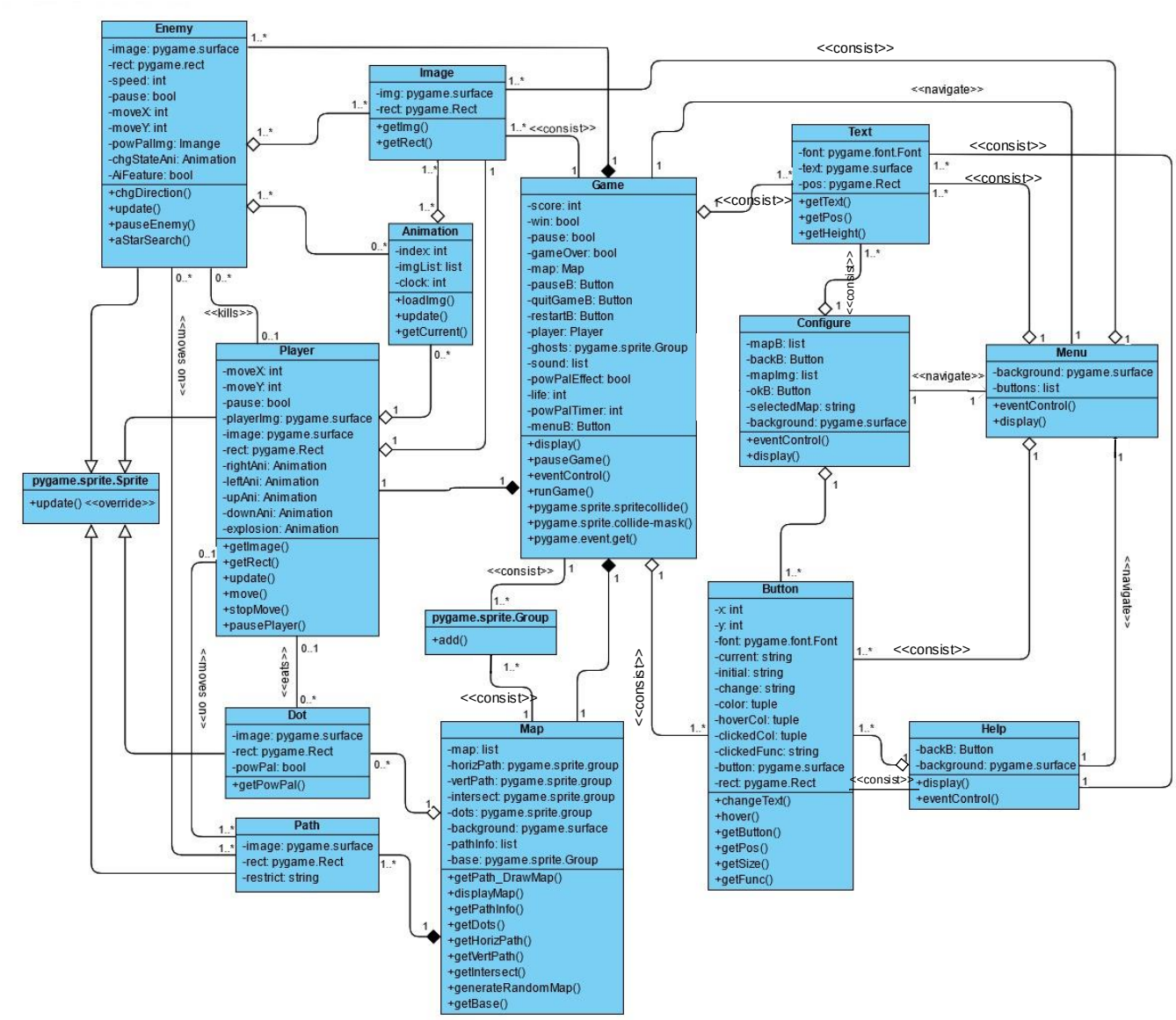
Use case

RQ – ID	Display countdown	Display still images when still game	Plays appropriate sound effect	Display animations when moving	Sound effects and music start	Ghosts move at random	Touches ghost	Stops game	Returns pac-man to start position	Returns ghost to start position	Loses life	Lose all lives	Game ends	Displays game over	Collects dots	Increases score	Dots disappear	Collects all dots	Displays victory	Eats power pallet
RQ7.1						X														
RQ7.2							X	X	X	X	X									
RQ7.3															X	X				
RQ7.4																	X			
RQ7.5										X										X
RQ7.6			X		X															
RQ7.7	X																			
RQ7.8													X					X	X	
RQ7.9												X	X	X						
RQ13		X		X																

3.0 Design and software architecture

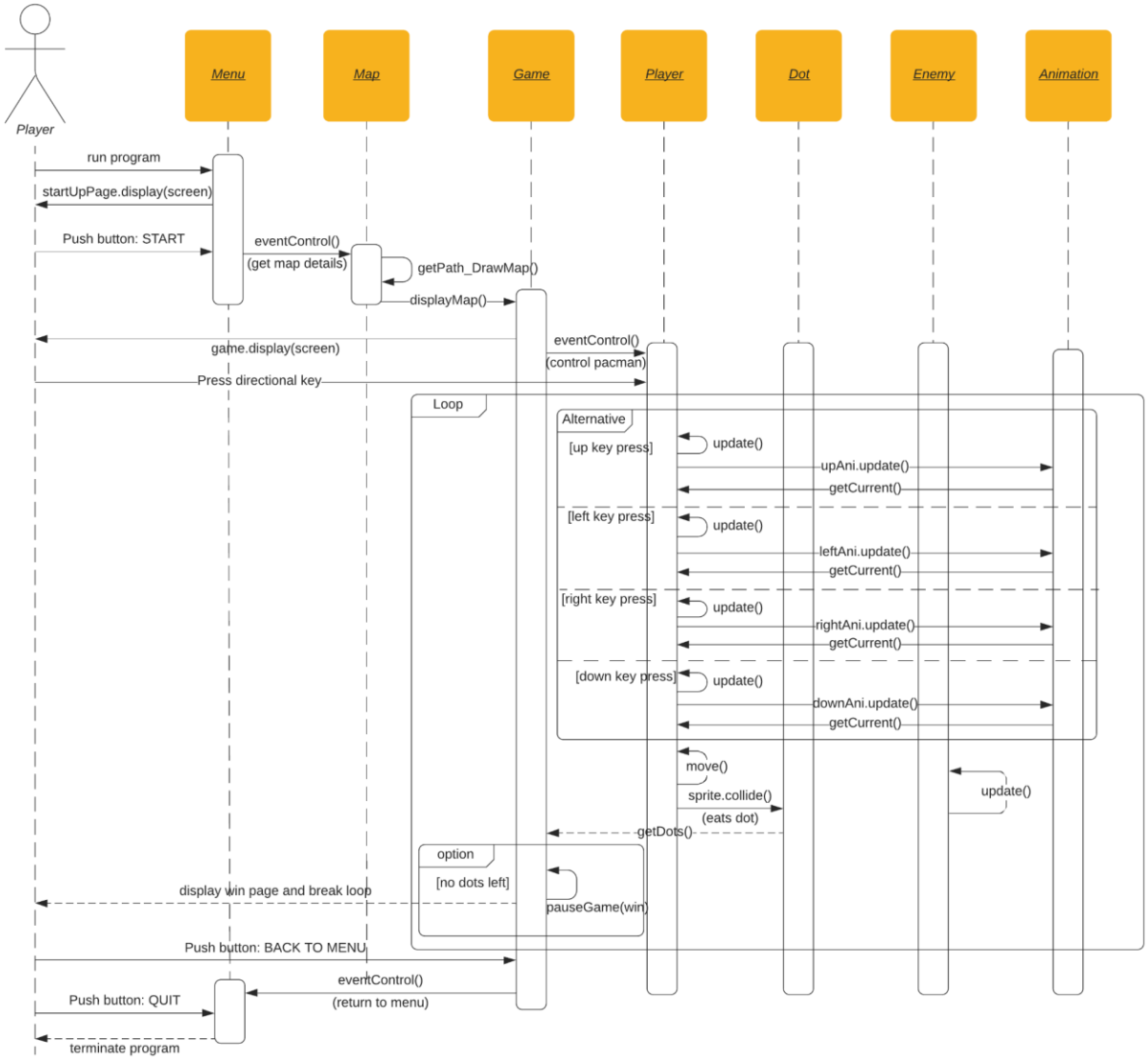
3.1 Class diagram

The class diagram shows a direct breakdown of the class structure implemented in the program. It shows the relationship between each class as well as the attributes and methods used.



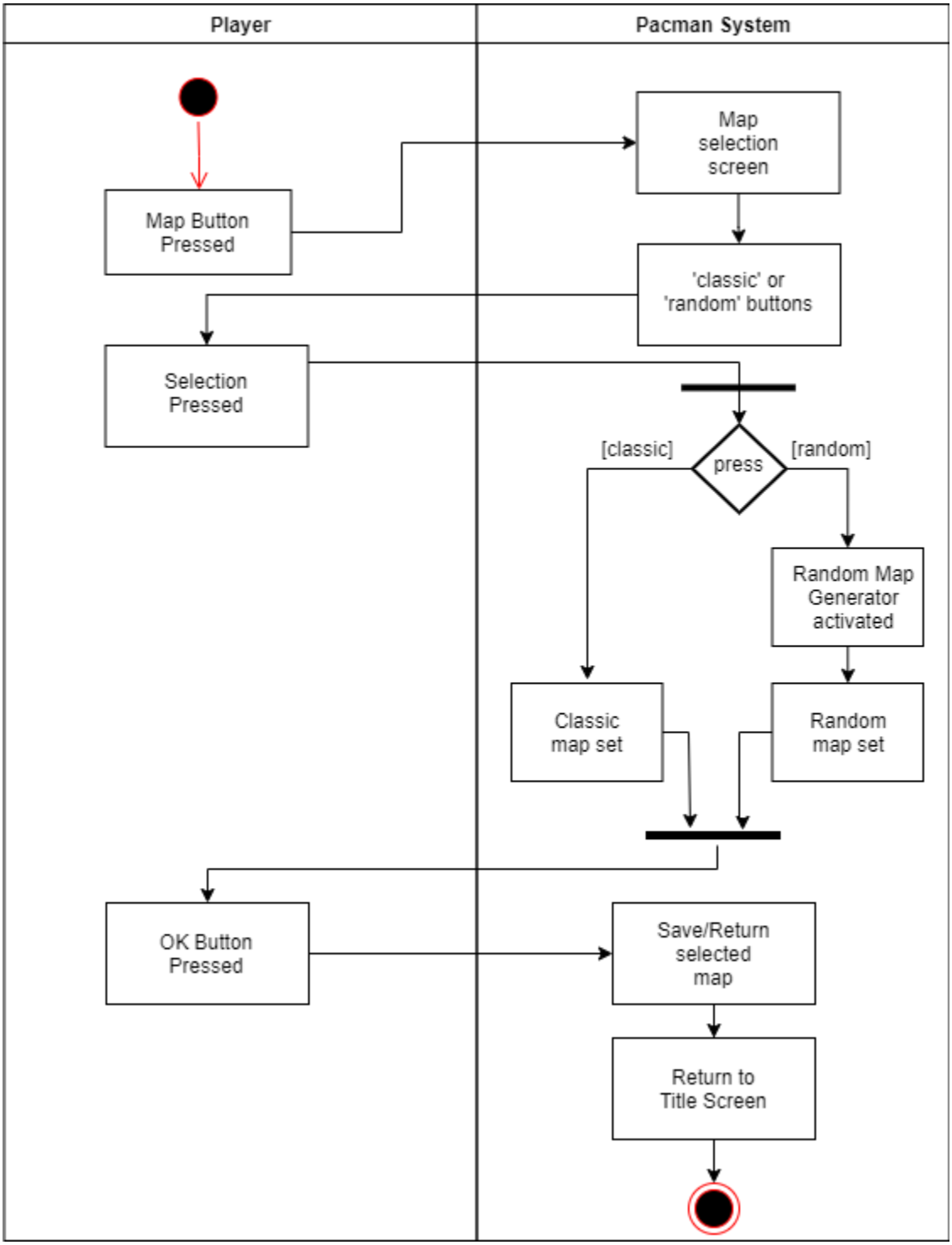
3.2 Sequence diagram

The following sequence diagram describes an instance where the player starts the game from the menu. The program will then create the desired map and execute the game. The player then successfully collects all the dots by controlling the movement of the Pac-man character while avoiding the ghosts, thereby winning the game. The player then exits the game via the menu, terminating the program.



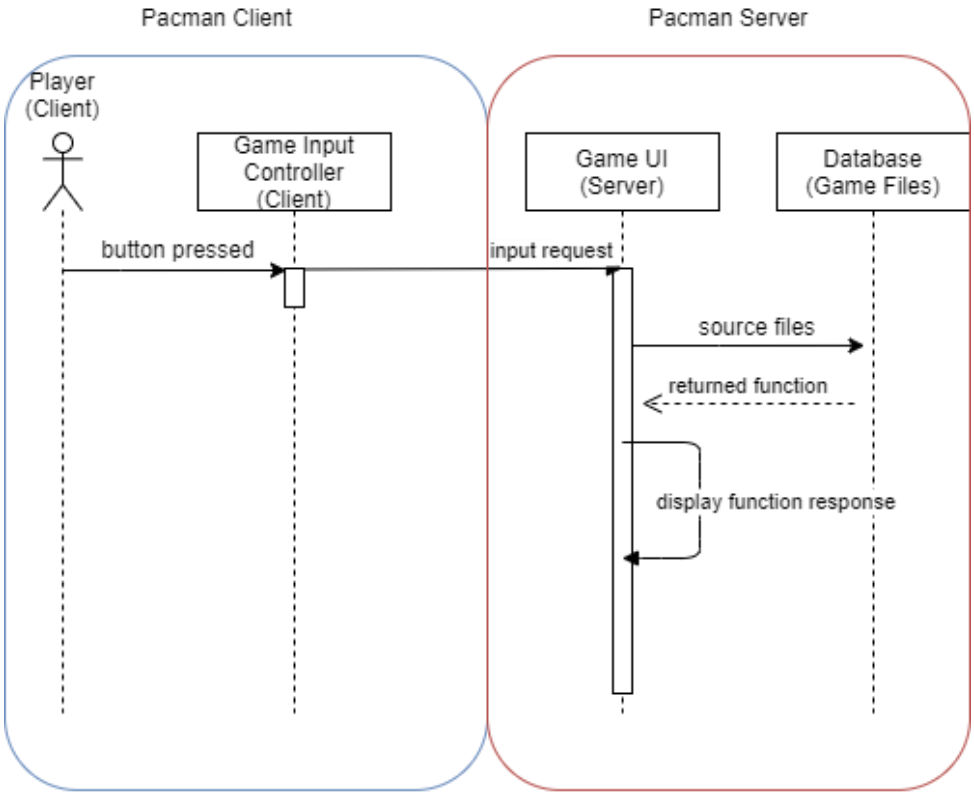
3.3 Activity diagram

Displaying an activity diagram of an active use case within the system will display the program flow plan and the individual functionality of the functions. The diagram below is a representation of the 'Map' settings button with the map selection functions. It shows how the player interacts with the system to select the type of map to be generated for the game.

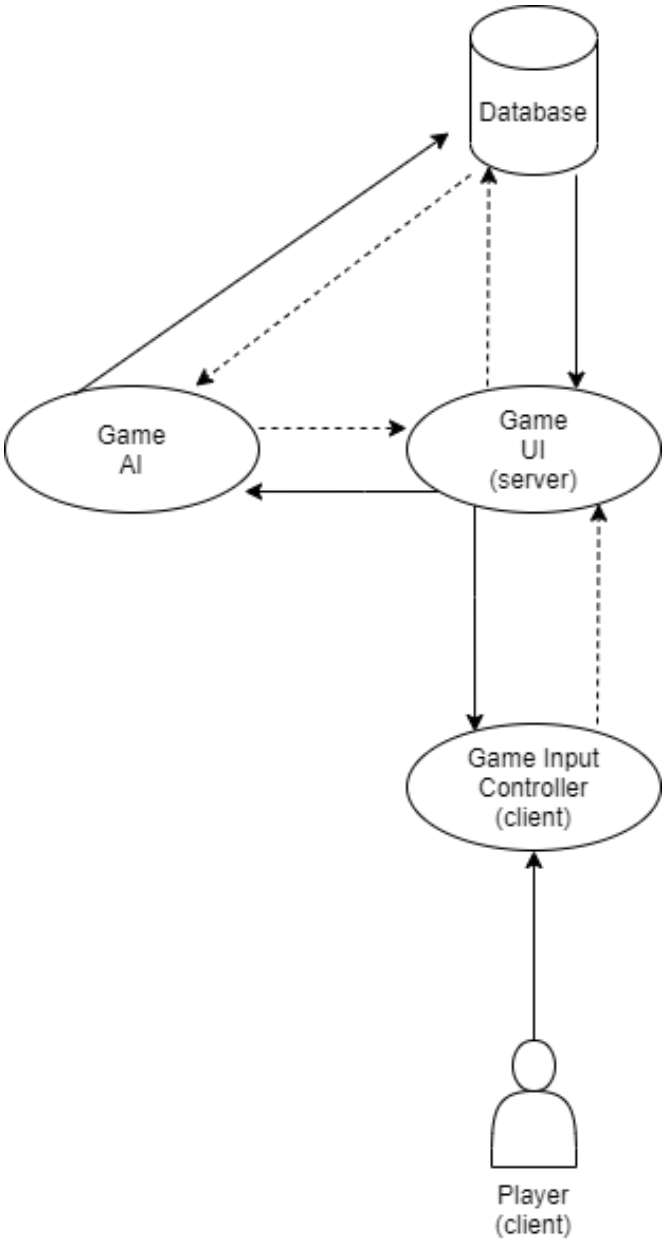


3.4 C & C View

The implemented style of the C & C view of the game system is a client-server style as it can appropriately display the system architecture and relationships between the providers of the system servers and clients. The diagram below displays the communications present within the systems used to develop and run the game, it clearly displays the communication between the game controller and user interface. Another diagram has been provided to display a simplistic view on the client server links between the different components of the software architecture.

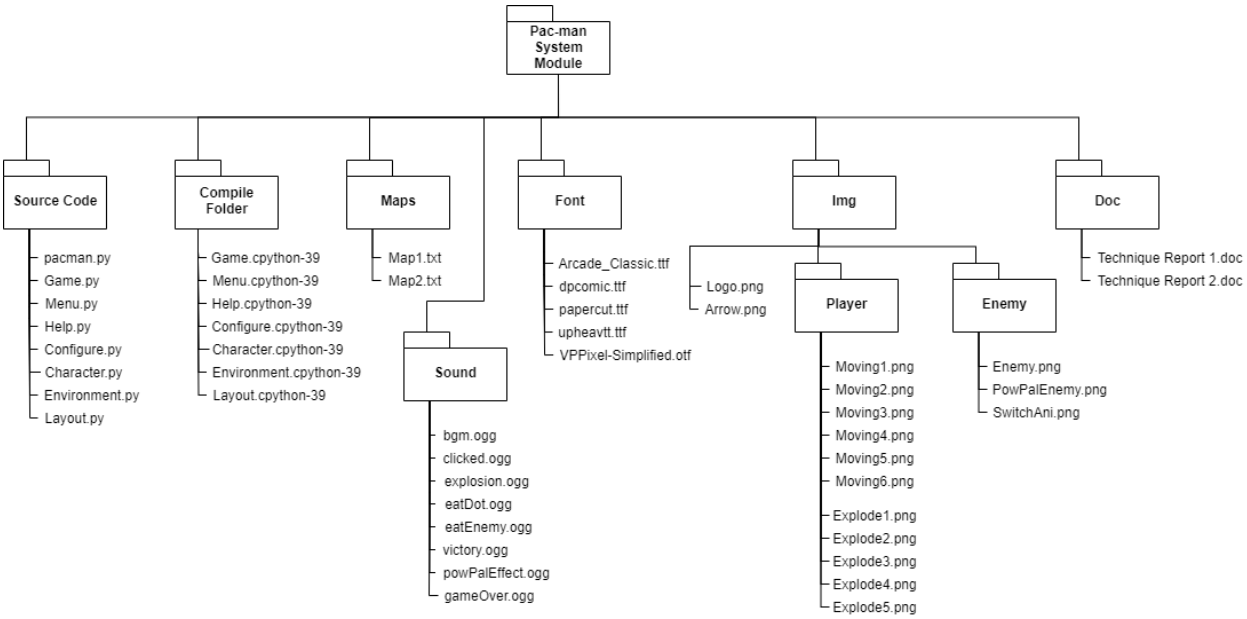


Another diagram has been provided to display a simplistic view on the client server links between the different components of the software architecture.



3.5 Implementation style view

Implementation style view provides a view of the development infrastructure of the project. The following diagram shows the hierarchy of directories in a file system.



A table is also constructed below to provide brief explanations of the functions and key contents of each folder and file.

Folders/Files	Key Contents and Functions
Pac-man System Module	Consists of all important folders and files of the Pac-man game.
Source Code	All source codes of the Pac-man game are stored in this folder.
. pacman.py	- Consists of the main running code of the Pac-man game. - To create pages for the game such as Start Up Page, Game Page etc. - To determine which page to display on the screen according to the user's choice.

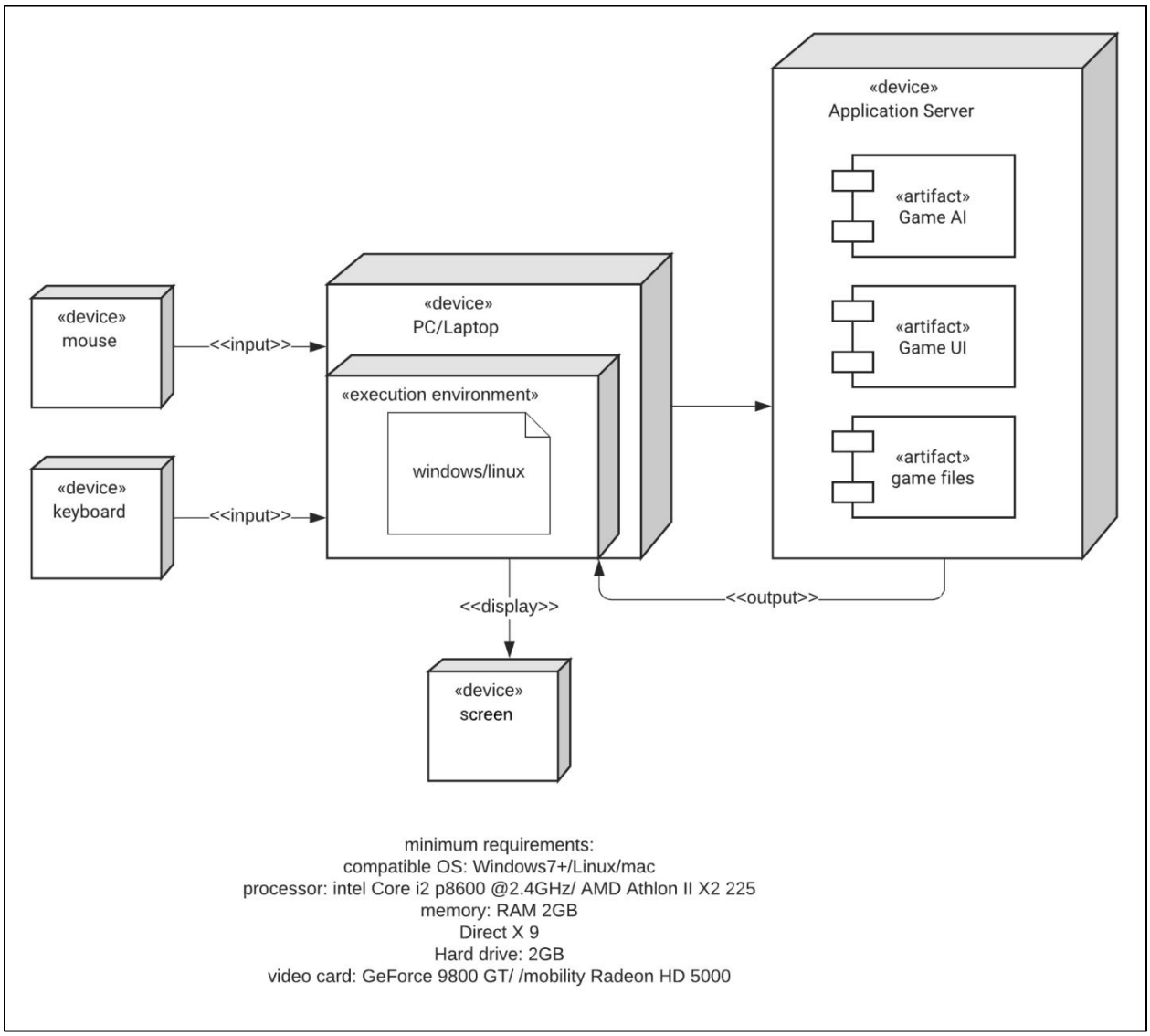
. Game.py	- The code of Game class is written in this python file. Game Page is created from here. Runs and updates the game. - To create and initialize the attributes (player, ghosts, score etc.) and components (buttons) of the actual game. - Accepts inputs from external devices (keyboard, mouse) and shows responses accordingly. For example, the Pac-man character moves left when the left arrow key is pressed and stops moving when the key is released. - The ability to pause or resume the game. - The ability to carry out appropriate actions after detecting collisions between Pac-man and dots, and between Pac-man and ghosts.
. Menu.py	- Consists of the code of Menu class that creates the Start Up Page. - To display the title and logo of Pac-man, the year and course code, and the list of students in the group. - Consists of “START” button that brings the user to the Game Page if clicked, “MAP” button to allow user to select desired map in the Configure Page, “HELP” button that shows the instructions of the game in the Help Page, and “QUIT” button that terminates the entire program if clicked.
. Help.py	- Consists of the code of Help class that creates the Help Page. - To display information about the Pac-man game. - Consists of “BACK” button to get the user back to the Start Up Page.
. Configure.py	- Consists of the code of Configure class that creates the Configure Page. - The ability to let the user choose either predefined maps or randomly generated maps for the game. - Consists of “OK” button to confirm the user's selected type of map and return to the Start Up Page.
. Character.py	- Consists of the code of Enemy class, Player class and Animation class. - Enemy class is responsible for creating ghosts and updating the movement of the ghosts. - Player class creates the Pac-man character and updates its movement that is controlled by the user. - Animation class produces animation for both the Pac-man character and the ghosts.

. Environment.py	- Consists of the code of Path class, Dot class and Map class. - Is mainly used to create and draw the map of the game. - Path class creates sprite for every block of the path in the maze. These sprites make sure that the game characters stay on lane and do not move out of path. - Dot class creates the sprites for white dots and power pellets. - Both Path class and Dot class are used in Map class to generate and draw map.
. Layout.py	- Consists of the code of the basic components in the game, which are the Text class, Image class and Button class. - Text class and Image class produces text message and images that will be displayed on various pages. - Button class is used to create buttons with different functions. Most of the buttons in this program allow the user to navigate between pages.
Compile Folder . Game.cpython-39 . Menu.cpython-39 . Help.cpython-39 . Configure.cpython-39 . Character.cpython-39 . Environment.cpython-39 . Layout.cpython-39	- Consists of all .cpython-39 files that are created by the Python interpreter when the .py files (modules) are imported. The .py files are compiled into bytecode the first time they are executed. This substantially improves the execution of the codes next time the modules are imported or executed, resulting in a better game performance.
Maps . Map1.txt . Map2.txt	- Consists of the predefined maps for the Pac-man game. To form the maze of the game.
Sound	- Consists of the music and the sound effects files of the Pac-man game.
. bgm.ogg	- To make the game more interesting and entertaining.
. clicked.ogg	- Is used when a button is clicked.
. explosion.ogg	- Is emitted when the Pac-man character explodes after colliding with one of the ghosts.
. eatDot.ogg	- Is emitted whenever the Pac-man character eats a white dot.

. eatEnemy.ogg	- Is used when the Pac-man character eats a ghosts under the power pellet effect.
. powPalEffect.ogg	- Sound is emitted when the Pac-man character eats a power pellet and activates the power pellet effect.
. victory.ogg	- Sound is emitted when the Pac-man character eats all the dots and power pellets, winning the game.
. gameOver.ogg	- Is emitted when the user has exhausted all three lives of the Pac-man character and loses the game.
Font . Arcade_Classic.ttf . dpcomic.ttf . papercut.ttf . upheavtt.ttf . VPPixel-Simplified.otf	- Consists of all .ttf and .otf font files that are used in the program. - Different fonts are used in the game to improve the appearance of the game displayed on the screen.
Img	- Consists of .png image files that are used in the program. - Also consists of folders that keep the sprite images for the game characters.
. Logo.png	- The logo image is displayed on the Start Up Page as required.
. Arrow.png	- The arrow image is shown in the Help Page. It is rotated or flipped accordingly to represent different arrow keys.
. Player (folder) .. Moving(n).png .. Explode(n).png	- Consists of images that are used in the moving and explosion animation of the Pac-man character.
. Enemy (folder) .. Enemy.png .. PowPalEnemy.png .. SwitchAni.png	- Consists of images that are used to represent ghosts that are either in the normal state or in the influence of the power pellet effect. - These images are also used to create animation when the ghost character changes its state.
Doc . Technique Report 1.doc . Technique Report 2.doc	- Consists of the documentations of the Pac-man program. - Show clear information and explanation of the program.

3.6 Deployment style view

The deployment view provides a basic overview of how the software interacts with a computer system's hardware, along with the deployment requirements for the Pac-man game.



4.0 Video link

The video demonstrates the execution of the prototype of the game.

https://youtu.be/otjn7rz4Cq0