Object‑Oriented Pac‑Man Game

## Group Members

1. Shahira khan, 24k-0639
2. Mehak Zainab, 24K-0508
3. Bashair Yaqoob, 24k-0810

# 1. Introduction

**Background**  
Classic arcade games like Pac‑Man demonstrate fundamental programming concepts. By recreating Pac‑Man, we explore object‑oriented design patterns—classes, objects, and encapsulation—to structure game entities (player, ghosts, map) and game logic.

**Problem Statement**  
Design and implement a playable Pac‑Man clone using OOP principles, addressing how to manage multiple interacting objects (characters, pellets, walls) and their behaviors without tangled procedural code.

**Objectives**

* Build a functioning Pac‑Man game in C++
* Demonstrate abstraction for game entities
* Showcase encapsulation by isolating maze and collision logic

# 2. Scope of the Project

**Inclusions**

* Player movement and input handling using SFML
* Ghost with simple chase logic
* Maze rendering and collision detection
* Pellet collection and scoring

**Exclusions**

* Advanced pathfinding (e.g., A\* search) for ghosts
* Multiplayer or networked play
* Graphical effects beyond basic sprites

# 3. Project Description

**Overview**  
The project implements Pac‑Man’s core mechanics: a player navigates a maze/ map, collects pellets, and avoids ghosts. It collides with the walls and makes sure not to cross them. We also have a green and a red booster that changes the speed of the player and the ghosts respectively. We’ll create classes for game, Player, Ghost, Booster and Maps demonstrating inheritance and encapsulation.

**Technical Requirements**

* C++ compiler (e.g., g++), Mingw
* Code editor or IDE: Visual Studio Code or Microsoft Visual Studio
* Simple graphics library (e.g., SFML or SDL)

**Project Phases**

1. **Research & Planning**: Review OOP basics and choose graphics library
2. **Design**: Introduction to SFML and Graphical user interface. Defining UML diagrams for classes and interactions
3. **Implementation**: Code classes
4. **Testing & Debugging**: Playtest scenarios and fix bugs

# 4. Methodology

**Approach**  
We will work in short iterative cycles, implementing one feature at a time (e.g., player movement → map → ghost behavior → booster placement →dots →collision), then reviewing as a group.

**Team Responsibilities**

* **Shahira Khan:** Research and set up of SFML; implement Player and map class
* **Mehak Zainab**: Design and code Player class, Booster and collision detection
* **Bashair Yaqoob**: Develop Ghost class, walls and, dots placement.

# 5. Expected Outcomes

**Deliverables**

* Complete code for the Pac‑Man game
* UML class diagram
* A prompt to tell user their score when the game stops
* A message that displays “You win” if player wins and “You lose” if it doesn’t.

**Relevance**  
This project reinforces ICT topics: OOP design (classes, inheritance, encapsulation), basic graphics programming, and event‑driven input handling.

# 6. Resources Needed

**Software**

* Visual Studio Code (or Visual Studio)
* SFML graphics library
* GitHub

**Other Resources**

* Online tutorials for SFML/SDL setup
* Instructor guidance for for setting up SFML
* YouTube guides to understand sf library.