Technical Report

The application consists in a CLI program where the user can buy an arcade machine, choosing the material and color, search a videogame and any videogame can be added if the user wants to. The administrator can see all the purchases. This can be resumed in the following user stories:

- As a client, I want to choose the type of materials of the machine, so what I can be sure of the machine's quality.
- As a client, I want to see the list of available games, so what I know the games of the offered machine.
- As a client, I want to add games to the machine, so what I can play more games.
- As a client, I want to add my information for delivery, so what I can receive the machine soon as possible.
- As a client, I want to customize the design of the machine like the color, so what It can be unique.
- As a client, I want to search a videogame in the catalog by name or genre, so what I can see if there is any videogame I like.
- As an admin, I want to see all the purchases in a file, so what I can deliver them.

Despite of workshop only specifies only four functions, I wanted to add more like the purchases that are included in a text file or search certain videogame.

The class design of the arcade videogame machine purchasing system reflects a clear separation of responsibilities, following the principles of object-oriented design. The `Videogame` class represents individual game objects with attributes like name, genre, and release year, encapsulating all information relevant to games. The `ArcadeMachine` class focuses on the physical attributes of arcade machines and their associated catalog of games, using composition to include a `Catalog` object. The `Catalog` class manages a collection of `Videogame` objects and provides functionality for searching and displaying games. The `User` class captures user information and actions, including adding games to the machine’s catalog. The `Purchase` class ties together a `User` and an `ArcadeMachine` to represent a transaction, while the `Manager` class oversees all purchases and writes them to a file. This separation allows each class to focus on its core functionality, making the system modular, maintainable, and extendable.