Product Requirements Document (PRD) — Pong Game (Python)

1. Overview

This document defines the functional, technical, and operational requirements for developing a modern, arcade-style version of **Pong**, written in **Python**, supporting **1 or 2 players**, with increasing AI difficulty and clean graphics. The goal is to create a playable, well-structured, fully tested application suitable for local desktop use.

The project will use modern graphics (via the **Arcade** library), include configurable settings, support sound effects, and maintain a modular file structure supportive of development, testing, and maintenance.

---

2. Objectives

- Build a modernized Pong game with slight 3D visual characteristics using the Python **arcade** library.
- Support **single-player mode** (with adaptive AI difficulty) and **two-player mode**.
- Maintain high-quality code with **unit tests covering ≥80%**.
- Package dependencies in a **requirements.txt** file.
- Deliver full documentation for installation, configuration, gameplay, and removal.
- Maintain a clean project directory structure excluding the user's personal `workflow/` directory and the `pong_venv/`environment.

---

3. Features & Functional Requirements

3.1 Game Modes

Single Player Mode

- Player uses keyboard controls (W/S).
- AI controls opposing paddle.
- AI difficulty increases over time:
    - AI paddle speed gradually increases.
    - Ball speed gradually increases.
    - AI prediction accuracy increases.

    - Combined scaling behavior.

Two Player Mode

- Player 1: W/S
- Player 2: Up/Down arrow keys
- Game starts immediately with both paddles under player control.

3.2 Visual Style

- Modern *neon arcade aesthetic* with subtle **3D lighting or shadow effects** on paddles and ball.
- Smooth animations (Arcade library supports high FPS rendering).
- Optional depth shading on walls, ball, and paddles.
- Clean, minimal UI layout.

3.3 Audio

- Include sound effects:
    - Ball bounce
    - Scoring
    - Game start/end sounds
- Must be toggleable via a **Settings menu**.

3.4 Game Mechanics

- Ball movement uses realistic rebound physics.
- Paddles move with smooth acceleration/deceleration.
- Score increases when ball passes a paddle.
- Game ends when a configurable score limit is reached (default: 10).
- Ability to restart the match from pause menu or after game completion.

3.5 Adaptive Difficulty System

- Difficulty increases at timed intervals or based on player performance.
- Scaling attributes:

    - AI paddle movement speed
    - Ball speed multiplier
    - Accuracy of AI tracking algorithm

3.6 Screen Resolution

- Use **custom resolution**, defined in configuration (default recommended: 1280×720).
- Support window resizing.
- Fullscreen toggle included in **settings menu**.

3.7 Optional Features (Selected)

Pause/Resume (B)

- Player can pause at any time (press ESC).
- Paused overlay screen with options:
    - Resume
    - Settings
    - Quit to main menu

Fullscreen Toggle (C)

- Switch fullscreen on/off via menu or hotkey (F11).

Settings Menu (D)

- Contains:
    - Difficulty preset (Easy/Normal/Hard for single-player)
    - Audio toggle
    - Screen mode (Window/Fullscreen)
    - Resolution selection
    - Control display screen

---

4. Non-Functional Requirements

4.1 Performance

- Maintain ≥ 120 FPS if supported by hardware.
- Input response must have < 50ms delay.

4.2 Reliability

- Game must not crash during long sessions (continuous testing in single-player loop recommended).
- Handle all key interrupts gracefully.

4.3 Portability

- Must run on:
  - macOS
  - Windows
  - Linux
    (Arcade library supports all three.)

---

5. Technology Stack

5.1 Programming Language

- Python 3.10+ recommended.

5.2 Primary Libraries

- **arcade** – graphics, animations, game loop
- **pytest** – unit testing
- **pytest-cov** – code coverage reporting
- **pydantic or dataclasses** – for settings/config models
- **typing** – type hints
- **pathlib** – file handling
- **toml or yaml** – configuration storage (config file)

---

6. Controls

Player 1

- Move Up → W

- Move Down → S

Player 2

- Move Up → Up Arrow
- Move Down → Down Arrow

Other Controls

- Pause/Resume → ESC
- Fullscreen Toggle → F11
- Menu Navigation → Arrow keys + Enter

---

7. Testing Requirements

Unit Tests

- Use **pytest**.
- Target **≥ 80% code coverage**.
- Tests required for:
    - Paddle movement
    - Ball physics
    - Collision detection
    - Game state transitions
    - AI difficulty logic
    - Settings loader
    - Sound toggle logic
    - Scorekeeping

Continuous Integration (optional)

If later desired, can integrate GitHub Actions for:

- Linting
- Testing
- Coverage

---

8. Documentation Requirements

Must include:

1. **Installation Guide**
    - How to install Python
    - How to install `requirements.txt`
    - How to run the game (`python main.py`)
2. **Configuration Guide**
    - Editing config file (resolution, difficulty, audio)
    - Saving and loading settings
3. **Gameplay Manual**
    - Controls
    - Game modes
    - Scoring rules
    - Difficulty behavior
4. **Uninstallation Guide**
    - How to remove the project directory
    - Removing virtual environment (`pong_venv` remains untouched)

---

9. Installation Requirements

requirements.txt must include:

- arcade
- pytest
- pytest-cov
- pydantic or pyyaml (depending on configuration choice)
- Any additional sound or utility libraries used

---

10. Project Structure

The PRD requires the following structure to be created:

```
project_root/
│
├── src/
│   ├── main.py
│   ├── game/
│   │   ├── __init__.py
│   │   ├── pong_window.py
│   │   ├── paddle.py
│   │   ├── ball.py
│   │   ├── ai_controller.py
│   │   ├── settings.py
│   │   ├── audio_manager.py
```

```
│   │   └── ui/
│   │       ├── main_menu.py
│   │       ├── pause_menu.py
│   │       ├── settings_menu.py
│   │       └── components/
│   └── assets/
│       ├── sounds/
│       └── images/
│
├── tests/
│   ├── test_paddle.py
│   ├── test_ball.py
│   ├── test_ai.py
│   ├── test_settings.py
│   └── test_game_state.py
│
├── requirements.txt
├── README.md
└── .gitignore
```

Must NOT overwrite or modify:

- `workflow/`
- `pong_venv/`

---

11. Launch & Execution

User runs the game via:

```
python src/main.py
```

No packaging into standalone executables is required.

---

12. Future Expansion (Not Required Now)

- Local multiplayer over LAN
- Online multiplayer
- Skin/theme packs
- Power-ups
- Enhanced physics modes