<u>**Approach C prompt for requirements engineer (written by project manager AI instance and requirements engineer (ChatGPT))**</u>

Project Outline

We are developing a classic Tetris game using Python. The project involves multiple stakeholders, each playing a crucial role in the development process. As the Software Architect, your role is to design the overall architecture of the game, including major components and their interactions. You will be working closely with the Developer, Code Reviewer, and Technical Writer to ensure that the design is effectively implemented and documented.

Role of the Software Architect

As the Software Architect, your primary responsibilities include:

- Architectural Design: Create a high-level design for the Tetris game, detailing the major components and their interactions.
- Component Interaction: Define how different components of the game will interact with each other.
- Design Documentation: Provide detailed documentation of the architecture for the development team.
- Collaboration: Work closely with other stakeholders to ensure the architecture aligns with the overall project requirements and objectives.

Project Requirements

The Requirements Engineer has outlined the following comprehensive list of functional and non-functional requirements for the Tetris game:

Functional Requirements

Game Mechanics

- Start/Pause/Reset
  - The game should allow players to start a new game.
  - The game should allow players to pause and resume the game.
  - The game should allow players to reset the game to its initial state.
- Classic Tetris Gameplay
  - The game should feature tetrominoes (I, O, T, S, Z, J, L shapes) falling from the top of the screen.
  - Players should be able to move tetrominoes left and right.
  - Players should be able to rotate tetrominoes clockwise and counterclockwise.
  - Players should be able to make tetrominoes fall faster by pressing a key.
- Row Detection and Removal
  - The game should detect when a row is completely filled with blocks.

    - The game should remove the filled row and shift the above rows downward.
    - The game should award points for each cleared row.
- Difficulty Progression
    - The game should increase the speed of falling tetrominoes as the player progresses through levels.
    - The game should display the current level.

User Interface

- Score and Level Display
    - The game should display the current score and level on the screen.
    - Next Tetromino Display
    - The game should show the next tetromino that will fall in a designated area.
- Visual Feedback
    - The game should provide visual feedback when rows are cleared.
    - The game should highlight the falling tetromino.
- Game Over Screen
    - The game should display a game over screen when the player loses.
    - The game over screen should show the final score and an option to restart or exit the game.

Controls

- Keyboard Controls
    - The game should support keyboard controls for moving left (default: left arrow key), moving right (default: right arrow key), rotating (default: up arrow key), and dropping (default: down arrow key).
    - The game should support keyboard control for pausing the game (default: P key).
- Customizable Controls
    - The game should allow players to customize the control keys via a settings menu.

Sound and Music

- Sound Effects
    - The game should include sound effects for moving tetrominoes, rotating tetrominoes, clearing rows, and game over.
- Background Music
    - The game should have background music that can be toggled on and off by the player.
    - The game should allow players to adjust the volume of sound effects and background music.

Non-Functional Requirements

Performance

Consistent Frame Rate

- The game should run smoothly at a consistent frame rate (e.g., 60 frames per second) on a typical computer with minimum hardware specifications:
    - CPU: 2 GHz

    - RAM: 2 GB
    - GPU: Integrated graphics

Performance Handling

- The game should handle increasing game speed without performance degradation.

Usability

- Intuitive Interface
    - The game should have an intuitive and easy-to-use interface with clear labels and instructions.
- Help Menu
    - The game should provide instructions or a help menu for new players.

Compatibility

- Python Compatibility
    - The game should be compatible with the latest version of Python.
- Operating System Compatibility
    - The game should run on major operating systems, including Windows, macOS, and Linux.

Reliability

- Error Handling
    - The game should handle unexpected inputs gracefully without crashing.
    - The game should display error messages for invalid inputs.
- Progress and High Scores
    - The game should save the player's progress and high scores securely.

Maintainability

- Code Documentation
    - The game code should be well-documented with comments explaining the functionality of key components.
- Modular Design
    - The game should have a modular design to allow easy addition of new features or modifications.

Examples of potential extensions include:

- Additional game modes (e.g., time attack, endless mode)
- Different visual themes or skins for tetrominoes
- Multiplayer support

Next Steps

Please generate a high-level architectural design for the Tetris game. Your design should include:

Major Components: Identify and describe the major components of the game (e.g., game engine, user interface, input handler, score manager).

Component Interactions: Define how these components will interact with each other.

Design Documentation: Create a detailed document outlining your architectural design.

Once you have completed the architectural design, we will review it with the other stakeholders to ensure alignment and completeness.