**Project Title**

**Tetris Reimagined: An Object-Oriented Approach**

**Group Members**

1. Amail Uddin – 24K-2575

**Submission Date**

April 24 2025

**1. Executive Summary**

**• Overview:**  
This project aimed to develop a classic Tetris game using object-oriented programming in C++, enhanced with graphical output using the Raylib library. The tasks included designing class structures, implementing game logic, integrating graphics, and building a user interface.

**• Key Findings:**  
The project successfully demonstrated core OOP principles such as encapsulation, inheritance, and polymorphism in a real-time game setting. The use of Raylib made it possible to create an engaging visual experience while maintaining a structured, modular codebase.

**2. Introduction**

**• Background:**  
Tetris is a timeless game with simple mechanics, making it ideal for demonstrating the capabilities of OOP in a C++ environment. This project allowed us to explore how game development can be managed through clean and reusable code design.

**• Project Objectives:**

* Apply OOP concepts in structuring game logic.
* Use Raylib for graphical rendering.
* Deliver a playable and user-friendly version of Tetris.

**3. Project Description**

**• Scope:**  
**Inclusions:**

* Block generation and movement
* Rotation and collision detection
* Line clearing and scoring system
* Game UI (Start, Pause, Game Over screens)

**Exclusions:**

* Multiplayer mode
* AI-controlled players
* Advanced graphics or sound integration

**• Technical Overview:**

* **Language:** C++
* **IDE/Editor:** Visual Studio Code
* **Graphics Library:** Raylib
* **Version Control:** Git and GitHub

**5. Project Implementation**

**• Design and Structure:**  
The project uses a modular class structure:

* Game – Manages game states and flow
* Block – Represents block types and behavior
* Grid – Handles collision and placement logic
* UI – Manages rendering of scores and menus

**• Functionalities Developed:**

* Tetromino spawning and rotation
* Grid-bound movement and collision detection
* Line clearing and scoring
* Pause, resume, and restart features

**• Challenges Faced:**

* **Rotation logic:** Ensuring rotated blocks did not collide with existing ones or boundaries. Solved through rotation matrix and wall kick logic.
* **Grid management:** Designing an efficient grid for real-time updates. Solved by using a 2D array and color mapping.

**6. Results**

**• Project Outcomes:**

* A working Tetris game playable on desktop
* Modular, readable, and documented source code
* Basic UI for interaction with the game

**• Testing and Validation:**

* Manual testing for all features
* Multiple rounds of gameplay to ensure functionality
* Code review for logic accuracy and memory handling

**7. Conclusion**

**• Summary of Findings:**  
This project meets all intended objectives and successfully showcased OOP principles in a functional game. The integration of Raylib helps bridge the gap between logic and user interaction.

**• Final Remarks:**  
This project helped improve both my technical and collaborative skills.