Include documentation of your application with the following sections.

Chosen Application Theme: Game Level Management

Rationale for your choice:

This theme was chosen to demonstrate the practical application of data structures in a gaming context. By integrating stacks and queues, the project highlights how these structures can be used to manage undo actions and player turns—key functionalities in many games. It also serves as an educational tool for understanding these data structures in an interactive manner.

Explanation of the Implemented Application and its features:

Stack Operations (Undo Actions):

- **Push Action**: Adds an action to the stack to simulate an event that can be undone.
- **Pop Action**: Removes the most recent action from the stack, simulating an undo.
- **Peek Stack**: Displays the most recent action (shown in the output).

Queue Operations (Player Turn Management):

- **Enqueue Player**: Adds a new player to the queue to represent a turn.
- **Dequeue Player**: Removes the player at the front of the queue, simulating the end of their turn.
- **Peek Queue**: Shows the next player in line (shown in the output).

GUI Features:

- Interactive buttons for each operation with distinct colors for better user experience.
- A display box showing the current state of the stack and queue.
- A dark-themed background for aesthetics and readability.

MainWindow Class: Handles the GUI and connects user input to the GameLevelManager.

Users can:

Enter actions and players into input fields.

Click buttons to perform stack and queue operations.

View stack (actions) and queue (players) updates in real time.

Three Test Cases for the chosen application:

1 **Test Case 1**: **Push and Pop Action**

- **Input**: Push "Move Forward," "Attack," and "Defend."
- **Expected Outcome**: Stack should display "Move Forward" at the bottom, with "Defend" on top.
- **Action**: Click "Pop Action."
- **Expected Outcome**: "Defend" is removed; "Attack" becomes the top action.

2 **Test Case 2**: **Enqueue and Dequeue Player**

- **Input**: Enqueue players "Player1" and "Player2."
- **Expected Outcome**: Queue should show "Player1" first and "Player2" next.
- **Action**: Click "Dequeue Player."
- **Expected Outcome**: "Player1" is removed; "Player2" becomes the first player.

3 **Test Case 3**: **Mixed Stack and Queue Operations**

- **Input**: Push "Collect Item," Enqueue "Player3," and push "Jump."
- **Action**: Pop action and dequeue player.
- **Expected Outcome**: "Jump" is removed from the stack, and "Player3" is removed from the queue.

Challenges Faced during development:

Qt Setup: Initial setup of the Qt environment and managing dependencies for QApplication were challenging, especially configuring it for cross-platform compatibility.
Installing Qt: The installation process for Qt took significant time and occasionally failed, which added delays.
Finding GUI Libraries: Locating appropriate libraries and resources for GUI development was sometimes difficult and required extensive research.

Real-time Display Updates: Implementing accurate and responsive updates to the stack and queue displays required creating temporary copies of the data structures to avoid altering the actual game state.

Testing: Ensuring consistent behavior across various combinations of stack and queue operations required extensive testing.

Roles of Each Member and their contributions

- Developer – Raiden Villapando
- Designer – Mharon Irvin Vergara
- Tester – Karl Vincent Regio