PyQt5 Intermediate-Level Exam

Exercise: Multithreaded File Downloader with Progress Visualization

Create a PyQt5 application that simulates a file downloader. The application should use ‘QThread‘ to handle the download process in the background, ‘QTimer‘ to simulate download progress, and ‘QProgressBar‘ to visualize the progress. Additionally, the application should allow the user to pause/resume the download and display a summary when the download is complete.

Requirements and Questions

Question 1: Create the Main Window (3 Marks)

- Create a ‘QMainWindow‘ with a fixed size of 600x400 pixels.
- Add a ‘QProgressBar‘ to display the download progress.
- Add three buttons: "Start Download", "Pause/Resume", and "Cancel".
- Add a ‘QLabel‘ to display the download status (e.g., "Downloading...", "Paused", "Completed").

Question 2: Implement the Download Thread (5 Marks)

- Create a ‘QThread‘ subclass named ‘DownloadThread‘ to simulate the download process.
- The thread should emit signals to update the progress bar and status label.
- Use a ‘QTimer‘ inside the thread to simulate progress updates every 100 milliseconds.
- The download should take 10 seconds to complete, and the progress bar should update accordingly.

Question 3: Handle Pause/Resume Functionality (4 Marks)

- Implement the "Pause/Resume" button to pause and resume the download.
- When paused, the ‘QTimer‘ in the ‘DownloadThread‘ should stop, and the status label should update to "Paused".
- When resumed, the ‘QTimer‘ should restart, and the status label should update to "Downloading...".

Question 4: Handle Cancel Functionality (3 Marks)

- Implement the "Cancel" button to stop the download and reset the progress bar.
- When canceled, the ‘DownloadThread‘ should stop, and the status label should update to "Canceled".
- Ensure the thread is properly terminated when canceled.

Question 5: Display Download Summary (3 Marks)

- When the download completes, display a ‘QMessageBox‘ with a summary of the download (e.g., "Download completed in 10 seconds").
- If the download is canceled, display a ‘QMessageBox‘ with the message "Download canceled".

Question 6: Add a Graphical Visualization (2 Marks)

- Add a ‘QGraphicsScene‘ and ‘QGraphicsView‘ to the main window.
- Display a graphical representation of the download progress (e.g., a growing rectangle or circle) in the ‘QGraphicsScene‘.
- The graphical element should update in sync with the progress bar.

Total Marks: 20

- Question 1: 3 Marks
- Question 2: 5 Marks
- Question 3: 4 Marks
- Question 4: 3 Marks
- Question 5: 3 Marks
- Question 6: 2 Marks

Estimated Duration: 3 Hours

This exam tests your ability to work with ‘QThread‘, ‘QTimer‘, ‘QProgressBar‘, ‘QMessageBox‘, and ‘QGraphicsScene‘. It also evaluates your understanding of event handling, signal-slot communication, and multithreading in PyQt5. Good luck!