PyQt5 Intermediate-Level Exam: Advanced File Processor

Exercise: Multithreaded File Processor with Dynamic UI Updates

Create a PyQt5 application that processes text files in the background using ‘QThread‘. The application should allow the user to select multiple files, process them (e.g., count words), and display the results dynamically in a table. The UI should update in real-time as files are processed.

Requirements and Questions

Question 1: Create the Main Window (3 Marks)

- Create a ‘QMainWindow‘ with a fixed size of 800x600 pixels.
- Add a ‘QTableWidget‘ to display the file processing results (columns: File Name, Word Count, Status).
- Add a ‘QPushButton‘ to open a file dialog for selecting multiple text files.
- Add a ‘QProgressBar‘ to show the overall progress of file processing.

Question 2: Implement the File Processing Thread (5 Marks)

- Create a ‘QThread‘ subclass named ‘FileProcessorThread‘ to process files in the background.
- The thread should count the number of words in each file and emit signals to update the UI.
- Use a ‘QTimer‘ to simulate a delay in processing each file (e.g., 1 second per file).
- Emit signals to update the table and progress bar dynamically.

Question 3: Handle File Selection and Processing (4 Marks)

- When the user selects files, start processing them in the background using the ‘FileProcessorThread‘.
- Display the file names in the table immediately after selection.
- Update the table with the word count and status (e.g., ”Processing”, ”Completed”) as each file is processed.

Question 4: Display Real-Time Progress (3 Marks)

- Update the ‘QProgressBar‘ to reflect the overall progress of file processing.
- Display the number of files processed and the total number of files in a ‘QLabel‘.

Question 5: Handle Errors and Edge Cases (3 Marks)

- If a file cannot be processed (e.g., invalid format), update the table status to ”Error”.
- Ensure the application does not crash if the user closes the window while processing is ongoing.
- Display a ‘QMessageBox‘ when all files are processed, showing the total word count across all files.

Question 6: Add a Custom Widget for File Progress (2 Marks)

- Create a custom widget (e.g., a ‘QWidget‘ with a ‘QProgressBar‘ and ‘QLabel‘) to show individual file progress.
- Add this custom widget to the table for each file being processed.

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‘, ‘QTableWidget‘, ‘QProgressBar‘, and custom widgets. It also evaluates your understanding of file handling, error handling, and dynamic UI updates in PyQt5. Good luck!