Qt Quiz Game - Project Documentation

Project Overview

This project is a Quiz Game Application built using Qt C++ and QML. It reads questions from a JSON file, displays them using a QML frontend, and handles user interaction with the quiz logic. The architecture separates UI from data handling using a custom QAbstractListModel.

Architecture Overview

C++ Backend:

- Handles loading and parsing JSON.
- Stores quiz questions and answers in a custom model.
- Exposes data to QML via context properties.

QML Frontend:

- Dynamically displays questions and options.
- Captures user input (selected answers).
- Shows correct/incorrect responses with basic UI transitions.

Key Components

1. JSON File Structure

Example content from Quiz.json:

```
{
  "quiz": [
    {
      "question": "What is the capital of France?",
      "options": ["London", "Berlin", "Paris", "Rome"],
      "answer": "Paris"
```

```
    },
    ...
  ]
}
```

2. QuizModel (C++)

- Inherits from QAbstractListModel.
- Roles defined: question, options, answer
- Provides: rowCount, data(), roleNames(), loadFromFile(QString)

3. Qmlsignalshandler (C++)

- Singleton object to expose QuizModel to QML.
- Function: add_DatatoModel() to load quiz data.
- Registered with QML using qmlRegisterType.

Data Flow

1. User launches app.
2. QuizModel is instantiated via Qmlsignalshandler.
3. Quiz data loaded using loadFromFile().
4. QML accesses model via ListView and roles.
5. User selects an answer - response is handled visually in QML.

QML Quiz Interface

Features:

- Displays one question at a time.
- Shows multiple-choice options.
- Highlights correct/incorrect answers.
- "Next" button to move to the next question.

Example Code Snippet:

```
ListView {
    model: quizhandler.m_quizmodel
    delegate: Column {
        Text { text: question }
        Repeater {
            model: options
            delegate: Button {
                text: modelData
                onClicked: {
                    if (modelData === answer)
                        result.text = "Correct!"
                    else
                        result.text = "Wrong!"
                }
            }
        }
        Text { id: result }
    }
}
```

Integration with Main Application

main.cpp:

```
qmlRegisterType<Qmlsignalshandler>("qmlsignalshandler", 1, 0, "Qmlsignalshandler");
```

Makes Qmlsignalshandler accessible from QML.

Loaded using QML engine to bind model and start interaction.

Key Learnings

- Custom QAbstractListModel usage.
- JSON parsing with Qt.
- QML data binding to C++ models.
- UI/UX in QML with dynamic data.
- Singleton pattern for global access.

Future Improvements

- Add timer and scoring system.
- Randomize questions.
- Add multiple categories or levels.
- Save progress and scores in a file.

Usage Instructions

- Place the Quiz.json file in the specified path.
- Run the Qt application.
- Select answers and move to the next question.
- View results instantly in the interface.