# Project Title

# Object-Oriented Quiz Game in C++


**Team Members:**

• Samiya Khan – 24K-0768
• Iqra Ashraf – 24K-0952
• Aseelah Masood Khan – 24K-0881

## Introduction

**Background:**
This project is aimed at developing educational software using Object-Oriented Programming (OOP) principles in C++. With the increasing popularity of digital learning tools, quiz-based applications provide an interactive and engaging way for learners to reinforce their knowledge.

**Problem Statement:**
Traditional learning methods can often be static and unengaging. Students lack access to interactive tools that provide instant feedback and personalized learning. This project addresses the gap by developing a modular and extensible quiz game using OOP, with multiple subjects, a scoreboard system, and real-time feedback through a graphical user interface (GUI).

**Objectives:**

- To implement a modular quiz game in C++ using OOP
- To demonstrate file handling through external question and score files
- To apply OOP concepts including encapsulation, inheritance, and polymorphism
- To build an interactive user experience using GUI
- To support subject-wise quiz selection and score tracking

## Scope of the Project

**Inclusions:**

- Subject selection functionality
- Multiple Choice Questions (MCQs) with four options
- Randomized question order
- Score tracking with feedback
- GUI-based interaction for all major game components

**Exclusions:**

- Online or multiplayer features
- Persistent user login or session saving

## Project Description

**Overview:**
The quiz game is built using C++ and GUI libraries (Raylib/RayGUI). It includes classes such as QuizQuestion, Subject, Player, QuizGame, and ScoreManager. The game reads subject-wise questions from files, records user scores, and displays performance at the end.

**Technical Requirements:**

- Visual Studio / Visual Studio Code
- C++ with Standard Library
- Raylib and RayGUI for graphical interface
- File system access for reading/writing questions and score files

**Project Phases:**

1. **Research:** Study quiz apps, file formats, and GUI frameworks
2. **Planning:** Design class responsibilities and GUI screen flow
3. **Design:** Implement core and supporting classes
4. **Implementation:** Code the backend and GUI modules
5. **Integration:** Merge individual modules into a complete system
6. **Testing:** Validate logic, debug, and polish UI

## Methodology

**Approach:**
An iterative development approach was followed. Basic console logic was developed first, and then GUI integration was done as a team. Code reviews and collaborative debugging sessions ensured cohesive functionality.

**TeamResponsibilities:**
While **initial main () setup** and overall integration were a joint effort by all members, actual GUI and class implementations were divided as follows:

**Samiya Khan**

- Backend: Player and QuizGame classes (handles player data and game flow)
- GUI: Name Entry Screen, Main Menu, Quiz Display Screen, Result Screen

- Code Logic: case NAMEENTRY, case MENU, case QUIZ, case RESULT, case EXIT

**Iqra Ashraf**

- Backend: QuizQuestion and Subject classes (handles questions, subjects, file reading)
- GUI: Subject Selection Screen, Final Score Screen
- Code Logic: case SUBJECTSELECT, case FINALSCORE in main flow

**Aseelah Masood Khan**

- Backend: ScoreManager class (handles high scores and scoreboard saving/displaying)
- GUI: High Score Screen, Scoreboard Screen
- Code Logic: case HIGHSCORE, case SCOREBOARD

**Note:** Although individual components were assigned, all members collaborated heavily, especially during integration and debugging of the GUI components.

## Expected Outcomes

**Deliverables:**

- A fully working C++ GUI-based quiz game
- Well-commented and organized source code
- A detailed project report explaining design and implementation
- A user manual or guide for gameplay

**Relevance to ICT:**

- Demonstrates OOP concepts (e.g., abstraction, encapsulation)
- Shows practical use of file handling
- Highlights GUI development in C++
- Encourages interactive and self-paced learning

## Resources Needed

**Software:**

- Visual Studio or VS Code

- C++ compiler supporting C++11 or later
- Raylib and RayGUI libraries

**Other:**

- Sample MCQ datasets in text files
- Instructor feedback and peer testing
- Online C++ and Raylib documentation