TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

**KHOA TOÁN – TIN**

A red and white logo

Description automatically generated

**BÁO CÁO DỰ ÁN CUỐI KÌ**

**KỸ THUẬT LẬP TRÌNH**

**CHỦ ĐỀ 6: CHƯƠNG TRÌNH LÀM BÀI TRẮC NGHIỆM**

**Giảng viên hướng dẫn:** TS. Vũ Thành Nam

**Sinh viên thưc hiện:** Nguyễn Hoàng Nam

**MSSV:** 20227247 – K67

**Lớp:** Hệ thống thông tin quản lý – 02

Hà Nội, 06-2024

# MỤC LỤC

[MỤC LỤC 2](#_Toc169209917)

[PHẦN 1 TÓM TẮT – ABSTRACT 3](#_Toc169209918)

[PHẦN 2 GIỚI THIỆU – INTRODUCTION 3](#_Toc169209919)

[1.1. Thông tin nền – Background Information 3](#_Toc169209920)

[1.2. Mục tiêu – Objectives 3](#_Toc169209921)

[1.3. Phạm vi – Scope 3](#_Toc169209922)

[PHẦN 3 PHƯƠNG PHÁP – METHODOLOGY 3](#_Toc169209923)

[1.4. Phát Biểu Vấn Đề - Problem Statement 3](#_Toc169209924)

[1.5. Tiếp Cận – Approach 3](#_Toc169209925)

[1.6. Thiết Kế - Desgin 3](#_Toc169209926)

[1.7. Công Cụ và Công Nghệ – Tools and Technologies 3](#_Toc169209927)

[PHẦN 4 TRIỂN KHAI – IMPLEMENTATION 3](#_Toc169209928)

[1.8. Tổng Quan Giao Diện Chương Trình 3](#_Toc169209929)

[1.9. Chi Tiết Giao Diện Chương Trình 3](#_Toc169209930)

[Tab ‘PLAY’ 3](#_Toc169209931)

[Tab ‘EDIT’ 3](#_Toc169209932)

[Tab ‘RESULT’ 3](#_Toc169209933)

[1.10. Tổng Quan Cấu Trúc Chương Trình 3](#_Toc169209934)

[1.11. Chi Tiết Các Module Của Chương Trình 3](#_Toc169209935)

[Chi Tiết Module ‘questionBank\_src.py’: 3](#_Toc169209936)

[Chi Tiết Module ‘quesAdmin\_src.py’ 3](#_Toc169209937)

[PHẦN 5 Phụ lục – Appendices 3](#_Toc169209938)

[Phụ lục 1: Chi Tiết Các Thành Phần Tab ‘PLAY’ 3](#_Toc169209939)

[Phụ lục 2: Chi Tiết Các Thành Phần Của Tab ‘EDIT’ 3](#_Toc169209940)

[Phụ lục 3: Chi Tiết Các Thành Phần Tab ‘RESULT’ 3](#_Toc169209941)

[Phụ lục 4: Chi tiết các hàm trong ‘questionBank\_src.py’ 3](#_Toc169209942)

[Phụ lục 5: Chi tiết các hàm trong ‘quesAdmin\_src.py’ 3](#_Toc169209943)

[Phụ lục 6: Chi tiết các hàm trong ‘examAdmin\_src.py’ 3](#_Toc169209944)

# TÓM TẮT – ABSTRACT

Chương trình được thực hiện là một ứng dụng làm trắc nghiệm môn tiếng Anh được thiết kế đầy đủ các chức năng để quản lý các câu hỏi, thực hiện các lần kiểm tra. Chương trình sự dụng một ngân hàng lưu trữ câu hỏi với các thành phần của từng câu hỏi: mã ID câu hỏi, nội dung câu hỏi, các đáp án, đáp án đúng. Các chức năng quản lý cơ bản đã được tích hợp đầy đủ trong chương trình bao gồm thêm, sửa, xóa câu hỏi, đáp án. Thêm vào đó chương trình có thể ngẫu nhiên tạo ra các bộ câu hỏi thi trắc nghiệm, hiện thị và lưu kết quả thi của người dùng. Giao diện chương trình được thiết kế bắt mắt dễ sử dụng đảm bảo tăng cường trải nghiệm người dùng. Báo cáo này thể hiện chi tiết thiết kế, cách triển khai và đi sâu vào các chức năng của chương trình.

# GIỚI THIỆU – INTRODUCTION

## Thông tin nền – Background Information

Chương trình luyện tập trả lời trắc nghiệm được sử dụng rộng rãi trong các trường học và các cơ sở giáo dục nhắm đánh giá kiển thức, hiểu biết của học sinh về các chủ đề khác nhau. Chương trình này đòi hỏi hệ thống quản lý câu hỏi mạnh mẽ đảm bảo tính toàn vẹn của câu hỏi. Nhu cầu về một chương trình hiệu quả, thân thiện với người dùng là rất cần thiết với người dùng công cụ giáo dục điện tử trong thời đại công nghệ phát triển hiện nay.

## Mục tiêu – Objectives

Mục tiêu chính của dự án là xây dựng một chương trình làm trắc nghiệm có thể quản lý các câu hỏi kiểm tra một cách hiệu quả, mang lại trải nghiệm kiểm tra trắc nghiệm tiện dụng cho người dùng. Chương trình hướng tới:

* Quản lý một ngân hàng câu hỏi qua cơ sở dữ liệu, bao gồm các chức năng thêm sửa xóa thông tin câu hỏi, đảm bảo tính đơn trị cho các bộ dữ liệu.
* Tạo ra được bộ câu hỏi ngẫu nhiên theo số lượng câu hỏi người dùng yêu cầu.
* Cho phép lưu lại kết quả làm bài, quản lý kết quả làm bài.

## Phạm vi – Scope

Báo cáo bao hàm việc phát triển và các chức năng của chương trình bao gồm:

Cách thức xử lý làm việc với ngân hàng câu hỏi qua file “question.bin”

* Triển khai các tính năng để quản lý câu hỏi
* Phương thức quản lý lưu trữ câu hỏi
* Phương thức quản lý lưu trữ kết quả bài làm
* Kiểm tra tính đúng đắn của các chức năng trong chương trình

Qua việc xây dựng phát triển chương trình trả lời câu hỏi trắc nghiệm, có thể mở rộng phát triển để cung cấp cho các cơ sở giáo dục, nhà giáo dục công cụ hiệu quả tạo dựng môi trường rèn luyện cho các sinh viên, học sinh.

# PHƯƠNG PHÁP – METHODOLOGY

## Phát Biểu Vấn Đề - Problem Statement

Vấn đề chính mà chương trình cần giải quyết là một hệ thống lưu trữ hiệu quả, quản lý toàn diện các câu hỏi kiểm tra; tạo ra các bộ câu hỏi theo số lượng yêu cầu cho người dùng, quản lý, lưu trữ, hiển thị kết quả cho người dùng. Để giải quyết cần thiết một giao diện tương tác dễ sử dụng, có thể dễ dàng thực hiện các chức năng cần thiết trong quản lý dữ liệu và các thao tác cơ bản cần thiết với dữ liệu.

## Tiếp Cận – Approach

Giải quyết các vấn đề trên cần các bước chính:

* **Quản trị câu hỏi:** xây dựng cơ sở dữ liệu lưu trữ câu hỏi triển khải các chức năng để thêm, sửa, xóa câu hỏi, kiểm tra tính đơn trị của câu hỏi.
* **Tạo bộ câu hỏi ngẫu nhiên:** tạo lập đoạn chương trình chọn ngẫu nhiên các câu hỏi từ ngân hàng câu hỏi có sẵn.
* **Quán trị làm trắc nghiệm:** thiết kế chức năng giúp người dùng có thể bắt đầu một bài kiểm tra, hiện thị các câu hỏi trên màn hình và chọn đáp án đúng
* **Thiết kế giao diện người dùng:** đảm bảo giao diện được thiết kế thân thiện dễ dùng, không gây khó khăn trong trải nghiệm của người dùng

## Thiết Kế - Desgin

Từ phân tích vấn đề và đưa ra cách tiếp cận, chương trình được thiết kế với các thành phần sau:

* **Database lưu trữ câu hỏi và kết quả:** Với các bảng lưu trữ câu hỏi kiểm tra và kết quả làm bài của người dùng, dữ liệu của các câu hỏi được lưu trữ trong file mang tên ‘questions.bin’
* **Module quán trị câu hỏi:** Chịu trách nhiệm thực hiện các chức năng thêm sửa xóa, quản lý các câu hỏi.
* **Module quản trị bài kiểm tra:** Chịu trách chọn lựa bộ câu hỏi từ ngân hàng câu hỏi với số lượng theo yêu cầu người dùng. Quản lý kết quả kiểm tra của người dùng
* **Giao diện làm kiểm tra:** Cung cấp giao diện giúp người dùng thực hiện kiểm tra, chịu trách nhiệm hiển thị câu hỏi, kết quả làm bài của người dùng.
* **Giao diện chỉnh sửa câu hỏi:** Hiển thị đầy đủ, chi tiết các câu hỏi, các đáp án, giúp người dùng dễ theo dõi, thêm, sửa, xóa các câu hỏi.

## Công Cụ và Công Nghệ – Tools and Technologies

Chương trình được xây dựng và thiết kế bằng các công cụ và công nghệ sau:

**Language:** Python 3.12.1 64-bit

**Libraries:** sqlite3(database),pickle (serialization) , random (randomization)

**Frameworks:** Qt6 v6.7.0 (GUI)

**IDE:** Visual Studio Code (code editor) , Qt Creator (GUI desgin)

**Version Control:** GitHub

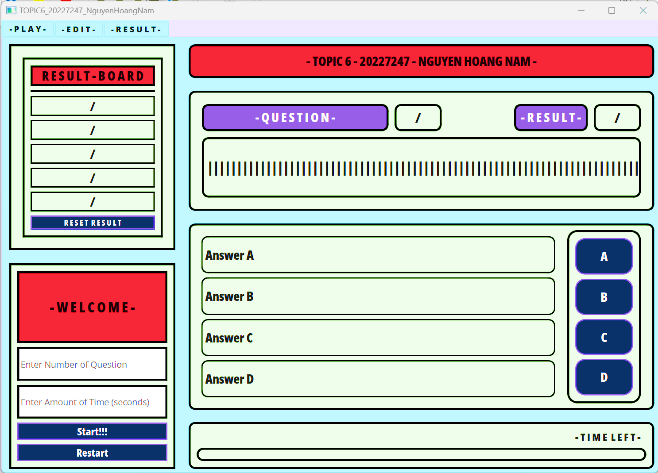
Các công cụ được áp dụng, liên kết với nhau giúp chương trình được thiết kế và hoạt động hiệu quả.

# TRIỂN KHAI – IMPLEMENTATION

## Tổng Quan Giao Diện Chương Trình

Giao diện được thiết kế sử dụng Qt Creator, được lưu trữ dưới dạng file ‘.ui’. Được kết nối qua PyQt6. Giao diện gồm 3 phần chính, gọi là các ‘tab’ được đặt ở góc trên bên trái của của sổ ứng dụng:

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

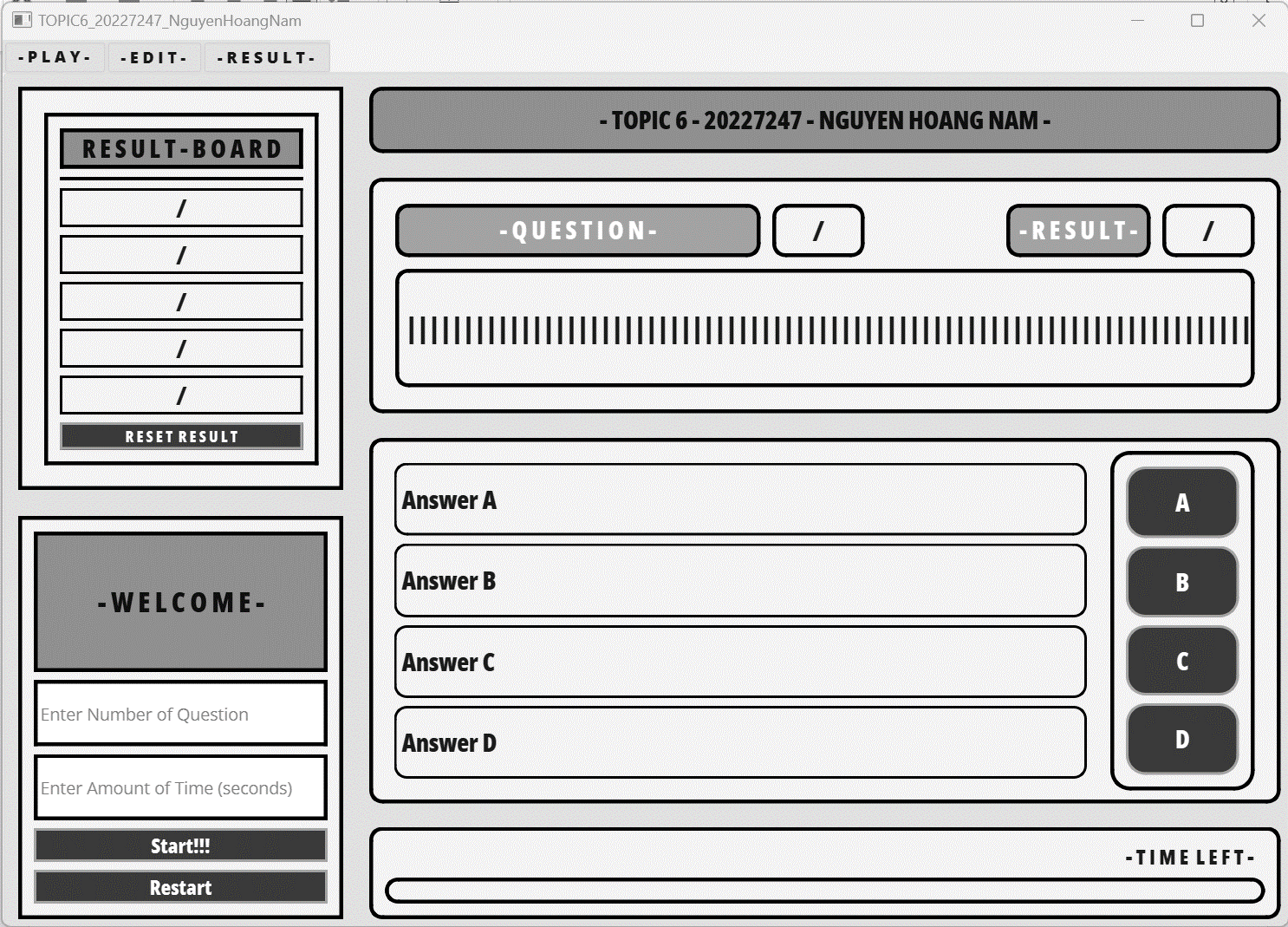


* Tab ‘PLAY’: Giao diện làm bài kiểm tra cho phép người dùng nhập số câu hỏi kiểm tra, nhập thời gian kiểm tra, bắt đầu bài kiểm tra. Thêm vào đó còn hiển thị kết quả các lần kiểm tra gần đây
* Tab ‘EDIT’: Giao diện quản lý câu hỏi cho phép người dùng thấy được ngân hàng câu hỏi, chỉnh sửa các thành phần của câu hỏi, thêm câu hỏi mới, xóa câu hỏi được chọn.
* Tab ‘RESULT’: Giao diện hiển thị lịch sử làm bài, hiển thị thông tin chi tiết của các lần kiểm tra

Mỗi tab sẽ đảm nhận một chức năng hiển thị khác nhau giúp người dùng sử dụng các chức năng của chương trình một cách thuận tiện và hiệu quả.

## Chi Tiết Giao Diện Chương Trình

### Tab ‘PLAY’

Các thành phần chính của Tab ‘PLAY’ bao gồm:

**(6)**

)

**(7)**

)

**(9)**

**(10)**

**(11)**

)

**(12)**

**(15)**

**(14)**

**(13)**

**(8)**

**(5)**

**(4)**

**(3)**

**(2)**

**(1)**

(1) Ô nhập đầu vào số câu hỏi: Cho phép người dùng nhập số câu hỏi

(2) Ô nhập thời gian: Cho phép người dùng nhập thời gian kiểm tra

(3) Nút bắt đầu kiểm: Nhấn để bắt đầu thực hiền làm bài kiểm

(4) Nút bắt đầu lại: Nhấn đề bắt đầu lại bài kiểm tra

(5) (6) (7) (8) Các nút chọn đáp án

(9) (10) (11) (12) Nơi hiển thị các đáp án

(13) Nơi hiển thị câu hỏi làm bài

(14) Nơi hiển thị kết quả làm

(15) Nút xóa kết quả kiểm tra

Chi tiết từng thành phần xem [Phụ lục 1](#_Phụ_lục_1:).

### Tab ‘EDIT’

Các thành phần chính của Tab ‘EDIT’ bao gồm:

**(4)**

**(11)**

)

**(10)**

**(9)**

**(8)**

**(7)**

)

**(6)**

)

**(5)**

**(3)**

**(2)**

**(1)**

(1) NútA screenshot of a computer

Description automatically generated sửa câu hỏi: Nhấn để xác nhận thay đổi câu hỏi

(2) Nút thêm câu hỏi: Nhấn để hiện cửa sổ thêm câu hỏi

(3) Nút xóa câu hỏi: Nhấn để xóa câu hỏi được chọn

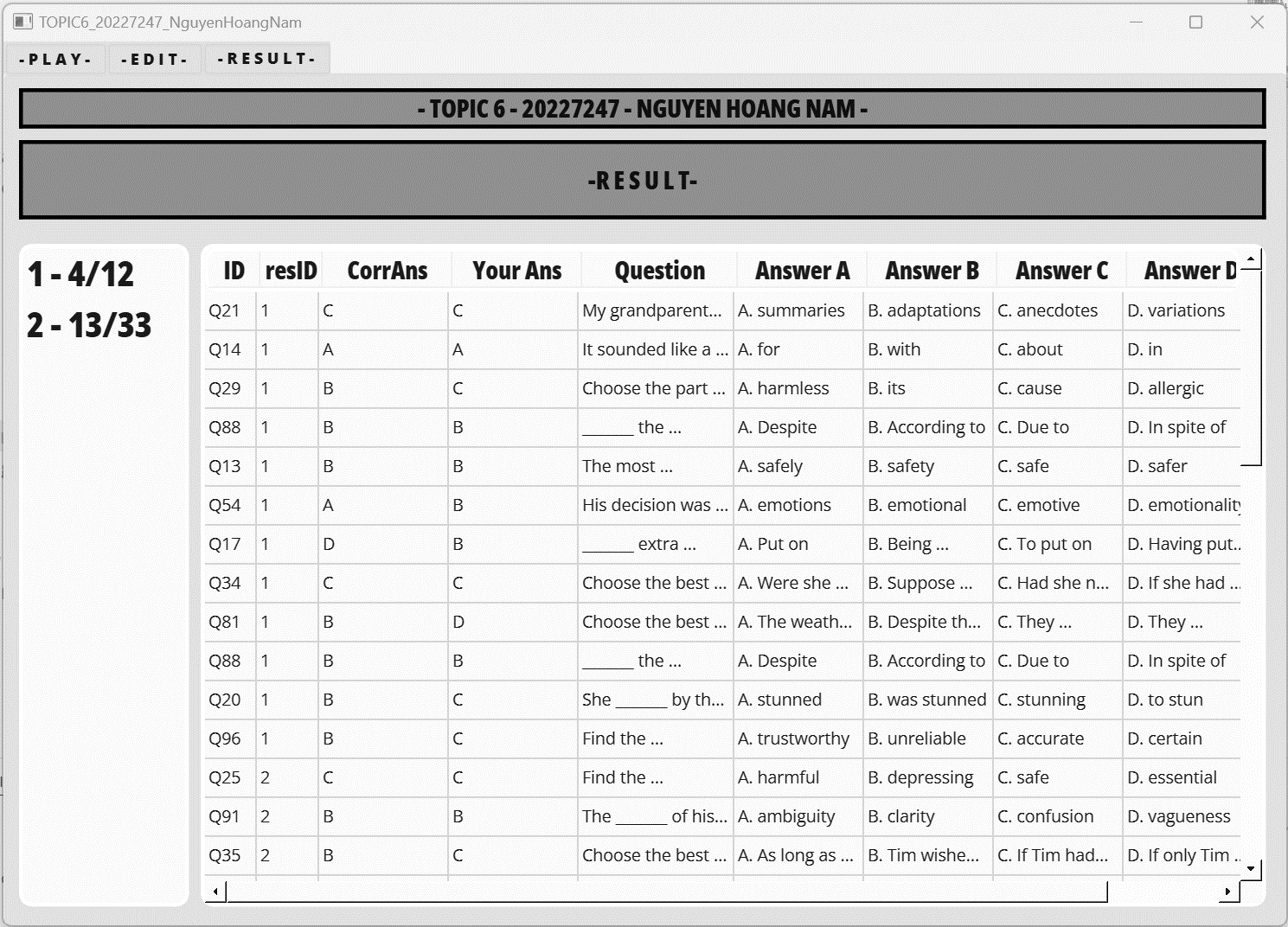
(4) Bảng hiển thị câu hỏi: Hiển thị thông tin từng câu hỏi

(5) (6) (7) (8) (9) (10) (11) Các hộp hiển thị câu hỏi (Formula Bar)

Chi tiết từng thành phần xem [Phụ lục 2](#_Phụ_lục_2:).

### Tab ‘RESULT’

Các thành phần chính của Tab ‘RESULT’ bao gồm:



**(2)**

**(1)**

(1) Danh sách kết quả làm bài

(2) Bảng chi tiết kết quả làm bài

Chi tiết từng thành phần xem [Phụ lục 3](#_Phụ_lục_3:).

## Tổng Quan Cấu Trúc Chương Trình

Chương trình được được chia thành nhiều file nguồn, mỗi file có một chức năng riêng thực hiện cho một mục đích nhất định:

* **‘mainWindow\_src.py’:** file chính kết nối các module. Chịu trách nhiệm khởi động giao diện chương trình, khởi tạo các thiết lập ban đầu, liên kết các chức năng với giao diện hiển thị.
* **‘questionBank\_src.py’:** module chịu trách nhiệm khởi tạo cơ sở dữ liệu bao gồm: khởi tạo cơ sở dữ liệu chính, khởi tạo các các bảng trong cơ sở dữ liệu
  + def **createDatabase**():
* **‘examAdmin\_src.py”:** module chịu trách nhiệm quản trị bài kiểm tra. Bao gồm việc khởi tạo bộ câu hỏi ngẫu nhiên, tiến hành các bài kiểm tra, kiểm tra đáp án đúng và hiện thị, tổng hợp các kết quả làm bài, quản lý chi tiết kết quả làm bài
  + class **ExamAdmin**:  
     def **\_\_init\_\_**(self, ui):  
     self.ui = ui  
     self.scoreTrack = 0  
     self.quesCount = 1  
     self.timer = **QTimer**()  
     self.timer.timeout.**connect**(self.**checkTimeLeft**)
  + def **startQuiz**(self):
  + def **checkCorrAns**(self, answer):
  + def **checkTimeLeft**(self):
  + def **resetToInitialState**(self):
  + def **resetResultBoard**(self):
  + def **displayResult**(self):
  + def **saveResultToDB**(self):
  + def **saveDetailResultToDB**(self):
  + def **insertUserAnswerToDB**(self, answer):
  + def **insertQuesIDToDB**(self, quesID):
  + def **displayAllResults**(self):
  + def **displaySelectedResultFromList**(self):
  + def **displayResultsInList**(self):
  + def **cleanUpResultsDetail**():
  + def **chooseRandomQues**():
  + def **displayQues**(…):
  + def **startTimer**(amouOfTime, progress\_bar):
  + def **updateProgressBar**(progress\_bar):
  + def **enableAnsButton**(self):
* **‘quesAdmin\_src.py’:** module chịu trách nhiệm quản trị các câu hỏi. Bao gồm việc thêm, sửa và xóa các câu hỏi, tổng hợp hiển thị các câu hỏi.
  + class **QuestionAdmin**:  
     def **\_\_init\_\_**(self, ui):  
     self.ui = ui
  + def **insertQues**(self):
  + def **delQues**(self):
  + def **updateQues**(self):
  + def **displayInFormulaBar**(ui, row):
  + def **resetFormulaBar**(ui):
  + def **loadData**(table\_widget):
  + def **dbToBinFIle**():
  + def **BinFileTodb**():
  + def **ReloadQTableWidget**(ui):

## Chi Tiết Các Module Của Chương Trình

### Chi Tiết Module ‘questionBank\_src.py’:

Chương trình sử dụng SQLite để quản lý dữ liệu. Chương trình được đã thiết kế ba bảng: **questions**, **userResult**, và **resultsDetail**.

def **createDatabase**(): [Phụ lục 4](#_Phụ_lục_4:)

Bảng **questions** lưu trữ các câu hỏi của chương trình. Mỗi câu hỏi có một ID duy nhất (quesID), nội dung câu hỏi (Question), bốn câu trả lời có thể có (AnswerA, AnswerB, AnswerC, AnswerD), và câu trả lời đúng (CorrectAnswer).

Bảng **userResult** lưu trữ kết quả của mỗi lần kiểm tra của người dùng. Mỗi kết quả có một ID duy nhất (resID) và thông tin kết quả (result).

Bảng **resultsDetail** lưu trữ chi tiết của mỗi kết quả. Mỗi bản ghi chi tiết có ID kết quả (resID), ID câu hỏi (quesID), và câu trả lời của người dùng (urAns). Các trường resID và quesID là khóa ngoại tham chiếu đến các bảng userResult và questions, tương ứng.

Chương trình sử dụng module **sqlite3** trong Python để tương tác với cơ sở dữ liệu SQLite. Đầu tiên, chương trình kết nối với cơ sở dữ liệu (hoặc tạo nó nếu chưa tồn tại) bằng cách sử dụng **sqlite3.connect().** Sau đó,một đối tượng con trỏ được tạo ra và được sử dụng để thực thi các lệnh SQL.

#### Đặc Tả Cấu Trúc Bảng

**questions**(quesID, Question, AnswerA, AnswerB, AnswerC, AnswerD, CorrectAnswer):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Rằng buộc** | **Mô tả** |
| 1 | quesID | TEXT | PRIMARY KEY | Mã số câu hỏi |
| 2 | Question | TEXT |  | Nội dung câu hỏi |
| 3 | AnswerA | TEXT |  | Đáp án A |
| 4 | AnswerB | TEXT |  | Đáp án B |
| 5 | AnswerC | TEXT |  | Đáp án C |
| 6 | AnswerD | TEXT |  | Đáp án D |
| 7 | CorrectAnswer | CHAR(1) |  | Đáp án đúng |

**userResult**(resID, result):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Rằng buộc** | **Mô tả** |
| 1 | resID | INTEGER | PRIMARY KEY  AUTOINCREMENT | Mã số kết quả |
| 2 | Question | TEXT |  | Kết quả kiểm tra |

**resultsDetail**(resID, quesID, urAns):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Kiểu dữ liệu** | **Rằng buộc** | **Mô tả** |
| 1 | resID | INTEGER | FOREIGN KEY userResult.resID | Mã số kết quả |
| 2 | quesID | TEXT | FOREIGN KEY questions.quesID | Mã số câu hỏi |
| 3 | urAns | CHAR(1) |  | Đáp án được chọn |

### Chi Tiết Module ‘quesAdmin\_src.py’

# PHỤ LỤC – APPENDICES

### Phụ lục 1: Chi Tiết Các Thành Phần Tab ‘PLAY’

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Class** | **Object** | **Class** |
| playTab | QWidget | questionDisplaybox | QLabel |
| verticalFrame\_5 | QFrame | quesCountlabel | QLabel |
| welcomelabel | QLabel | scoretlabel | QLabel |
| numofQuesInputbox | QLineEdit | resuslttiltlelabel\_2 | QLabel |
| amountofTimeInputbox | QlineEdit | displayAnsframe | QFrame |
| startButton | QPushButton | verticalFrame\_10 | QFrame |
| reStartButton | QPushButton | ansAlabel | QLabel |
| scoreGroupbox | QGroupBox | ansBlabel | QLabel |
| verticalFrame\_8 | QFrame | ansClabel | QLabel |
| resuslttiltlelabel | QLabel | ansDlabel | QLabel |
| line | Line | frame | QFrame |
| result1displaylabel | QLabel | ansApushButton | QPushButton |
| result2displaylabel | QLabel | ansBpushButton | QPushButton |
| result3displaylabel | QLabel | ansCpushButton | QPushButton |
| result4displaylabel | QLabel | ansDpushButton | QPushButton |
| result5displaylabel | QLabel | verticalFrame\_6 | QFrame |
| resetBoard | QPushButton | label\_4 | QLabel |
| displayQuestionframe | QFrame | progressBar | QProgressBar |
| questiontitlelabel | QLabel | welcomelabel\_2 | QLabel |

### Phụ lục 2: Chi Tiết Các Thành Phần Của Tab ‘EDIT’

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Class** | **Object** | **Class** |
| editTab | QWidget | IDtxtbox | QLineEdit |
| tableWidget | QTableWidget | welcomelabelTeacher | QLabel |
| label | QLabel | addQuesWidget\_2 | QWidget |
| frame\_2 | QFrame | verticalFrame\_4 | QFrame |
| gridFrame | QFrame | addqueslabelstack2\_2 | QLabel |
| confirmEditButton | QPushButton | quesBoxEditWidget | QLineEdit |
| addQuesbutton | QPushButton | ansAEditWidget | QLineEdit |
| delQuesbutton | QPushButton | ansBEditWidget | QLineEdit |
| corrAnstxtbox | QLineEdit | ansCEditWidget | QLineEdit |
| ansAtxtbox | QLineEdit | ansDEditWidget | QLineEdit |
| ansBtxtbox | QLineEdit | corrAnscomboBox | QComboBox |
| ansCtxtbox | QLineEdit | IDEditWidget | QLineEdit |
| ansDtxtbox | QLineEdit | submitButton | QPushButton |
| questiontxtbox | QLineEdit | cancelButton | QPushButton |

### Phụ lục 3: Chi Tiết Các Thành Phần Tab ‘RESULT’

|  |  |  |  |
| --- | --- | --- | --- |
| **Object** | **Class** | **Object** | **Class** |
| editTab | QWidget | welcomelabelTeacher | QLabel |
| tableWidget | QTableWidget | addQuesWidget\_2 | QWidget |
| label | QLabel | verticalFrame\_4 | QFrame |
| frame\_2 | QFrame | addqueslabelstack2\_2 | QLabel |
| gridFrame | QFrame | quesBoxEditWidget | QLineEdit |
| confirmEditButton | QPushButton | ansAEditWidget | QLineEdit |
| addQuesbutton | QPushButton | ansBEditWidget | QLineEdit |
| delQuesbutton | QPushButton | ansCEditWidget | QLineEdit |
| corrAnstxtbox | QLineEdit | ansDEditWidget | QLineEdit |
| ansAtxtbox | QLineEdit | corrAnscomboBox | QComboBox |
| ansBtxtbox | QLineEdit | IDEditWidget | QLineEdit |
| ansCtxtbox | QLineEdit | submitButton | QPushButton |
| ansDtxtbox | QLineEdit | cancelButton | QPushButton |
| questiontxtbox | QLineEdit | listExamWidget | QListWidget |
| IDtxtbox | QLineEdit | resultTableWidget | QTableWidget |

### Phụ lục 4: Chi tiết các hàm trong ‘questionBank\_src.py’

def **createDatabase**():

*# Connect to the SQLite database (or create it if it doesn't exist)*

    conn = **sqlite3**.**connect**('QuestionBank.db')

    cursor = conn.**cursor**()

*# Create tables*

    cursor.**execute**('''

        CREATE TABLE IF NOT EXISTS questions (

            quesID TEXT PRIMARY KEY,

            Question TEXT,

            AnswerA TEXT,

            AnswerB TEXT,

            AnswerC TEXT,

            AnswerD TEXT,

            CorrectAnswer CHAR(1)

        )

    ''')

    cursor.**execute**('''

        CREATE TABLE IF NOT EXISTS userResult (

            resID INTEGER PRIMARY KEY AUTOINCREMENT,

            result TEXT

        )

    ''')

    cursor.**execute**('''

        CREATE TABLE IF NOT EXISTS resultsDetail (

            resID INTEGER,

            quesID TEXT,

            urAns CHAR(1),

            FOREIGN KEY(resID) REFERENCES userResult(resID),

            FOREIGN KEY(quesID) REFERENCES questions(quesID)

        )

    ''')

*# Save (commit) the changes*

    conn.**commit**()

*# Close the connection*

    conn.**close**()

### Phụ lục 5: Chi tiết các hàm trong ‘quesAdmin\_src.py’

class **QuestionAdmin**:

    def **\_\_init\_\_**(self, ui):

        self.ui = ui

    def **insertQues**(self):

*# Create a QMessageBox for confirmation*

        msgBox = **QMessageBox**()

        msgBox.**setIcon**(**QMessageBox**.**Icon**.Question)

        msgBox.**setText**("Are you sure you want to add this question?")

        msgBox.**setWindowTitle**("Confirmation")

        msgBox.**setStandardButtons**(

**QMessageBox**.**StandardButton**.Yes **|** **QMessageBox**.**StandardButton**.No)

*# Show the message box and get the user's response*

        returnValue = msgBox.**exec**()

        if returnValue == **QMessageBox**.**StandardButton**.Yes:

*# Get the text from the QLineEdit widgets*

            quesID = self.ui.IDEditWidget.text()

            question = self.ui.quesBoxEditWidget.text()

            ansA = self.ui.ansAEditWidget.text()

            ansB = self.ui.ansBEditWidget.text()

            ansC = self.ui.ansCEditWidget.text()

            ansD = self.ui.ansDEditWidget.text()

*# Get the current text from the QComboBox*

            corrAns = self.ui.corrAnscomboBox.currentText()

*# Check if all QLineEdit widgets have been filled out*

            if not **all**([quesID, question, ansA, ansB, ansC, ansD, corrAns]):

                msgBox = **QMessageBox**()

                msgBox.**setIcon**(**QMessageBox**.**Icon**.Warning)

                msgBox.**setText**("Please fill out all fields before submitting.")

                msgBox.**setWindowTitle**("Warning")

                msgBox.**exec**()

                return

            conn = **create\_connection**()

            cursor = conn.**cursor**()

*# Check if quesID already exists in the database*

            cursor.**execute**('SELECT quesID FROM questions WHERE quesID = ?', (quesID,))

            if cursor.**fetchone**():

                msgBox = **QMessageBox**()

                msgBox.**setIcon**(**QMessageBox**.**Icon**.Warning)

                msgBox.**setText**("Question ID already exists.")

                msgBox.**setWindowTitle**("Warning")

                msgBox.**exec**()

                return

*# Insert the new question into the database*

            cursor.**execute**('''

                INSERT INTO questions (quesID, Question, AnswerA, AnswerB, AnswerC, AnswerD, CorrectAnswer)

                VALUES (?, ?, ?, ?, ?, ?, ?)

            ''', (quesID, question, ansA, ansB, ansC, ansD, corrAns))

            conn.**commit**()

            conn.**close**()

*# Success message box*

            msgBox = **QMessageBox**()

            msgBox.**setIcon**(**QMessageBox**.**Icon**.Information)

            msgBox.**setText**("Question added successfully.")

            msgBox.**setWindowTitle**("Success")

            msgBox.**exec**()

*# Reload the QTableWidget*

**loadData**(self.ui.tableWidget)

            self.ui.quesBoxEditWidget.setText("")

            self.ui.ansAEditWidget.setText("")

            self.ui.ansBEditWidget.setText("")

            self.ui.ansCEditWidget.setText("")

            self.ui.ansDEditWidget.setText("")

            self.ui.IDEditWidget.setText("")

            self.ui.corrAnscomboBox.setCurrentIndex(0)

            self.ui.addQuesWidget\_2.hide()

**dbToBinFIle**()

    def **delQues**(self):

*# Create a QMessageBox*

        msgBox = **QMessageBox**()

        msgBox.**setIcon**(**QMessageBox**.**Icon**.Question)

        msgBox.**setText**("Are you sure you want to delete this question?")

        msgBox.**setWindowTitle**("Confirmation")

        msgBox.**setStandardButtons**(

**QMessageBox**.**StandardButton**.Yes **|** **QMessageBox**.**StandardButton**.No)

        msgBox.**setDefaultButton**(**QMessageBox**.**StandardButton**.No)

*# Show the message box and get the user's response*

        returnValue = msgBox.**exec**()

        if returnValue == **QMessageBox**.**StandardButton**.Yes:

            row = self.ui.tableWidget.currentRow()

            quesID = self.ui.tableWidget.item(row, 0).text()

            conn = **create\_connection**()

            cursor = conn.**cursor**()

*# Delete the question from the 'resultsDetail' table*

            cursor.**execute**(

                'DELETE FROM resultsDetail WHERE quesID = ?', (quesID,))

*# Delete the question from the 'questions' table*

            cursor.**execute**('DELETE FROM questions WHERE quesID = ?', (quesID,))

            conn.**commit**()

            conn.**close**()

*# Reload the QTableWidget*

**ReloadQTableWidget**(self.ui)

**loadData**(self.ui.tableWidget)

**dbToBinFIle**()

    def **updateQues**(self):

*# Create a QMessageBox*

        msgBox = **QMessageBox**()

        msgBox.**setIcon**(**QMessageBox**.**Icon**.Question)

        msgBox.**setText**("Are you sure you want to update this question?")

        msgBox.**setWindowTitle**("Confirmation")

        msgBox.**setStandardButtons**(

**QMessageBox**.**StandardButton**.Yes **|** **QMessageBox**.**StandardButton**.No)

        msgBox.**setDefaultButton**(**QMessageBox**.**StandardButton**.No)

*# Show the message box and get the user's response*

        returnValue = msgBox.**exec**()

        if returnValue == **QMessageBox**.**StandardButton**.Yes:

            conn = **create\_connection**()

            cursor = conn.**cursor**()

*# Get the updated values from QLineEdit widgets*

            quesID = self.ui.IDtxtbox.text()

            question = self.ui.questiontxtbox.text()

            ansA = self.ui.ansAtxtbox.text()

            ansB = self.ui.ansBtxtbox.text()

            ansC = self.ui.ansCtxtbox.text()

            ansD = self.ui.ansDtxtbox.text()

            corrAns = self.ui.corrAnstxtbox.text()

*# Check if all fields are filled out*

            if not **all**([quesID, question, ansA, ansB, ansC, ansD, corrAns]):

**QMessageBox**.**warning**(

                    self.ui, 'Warning', 'Please fill out all fields before submitting.')

                return

*# Check if the correct answer is one of 'A', 'B', 'C', or 'D'*

            if corrAns not in {'A', 'B', 'C', 'D'}:

**QMessageBox**.**warning**(

                    self.ui, 'Warning', 'The correct answer must be one of A, B, C, or D.')

                return

*# Update the question in the database*

            cursor.**execute**('''

                UPDATE questions

                SET Question = ?, AnswerA = ?, AnswerB = ?, AnswerC = ?, AnswerD = ?, CorrectAnswer = ?

                WHERE quesID = ?

            ''', (question, ansA, ansB, ansC, ansD, corrAns, quesID))

            conn.**commit**()

            conn.**close**()

**dbToBinFIle**()

**loadData**(self.ui.tableWidget)

def **dbToBinFIle**():

    conn = **create\_connection**()

    cursor = conn.**cursor**()

*# Execute the query*

    cursor.**execute**("SELECT \* FROM questions")

*# Get all rows*

    rows = cursor.**fetchall**()

*# Convert the data to a binary file*

    with **open**('questions.bin', 'wb') as binFile:

**pickle**.**dump**(rows, binFile)

    conn.**close**()

def **BinFileTodb**():

    conn = **create\_connection**()

    cursor = conn.**cursor**()

*# Load the data from the binary file*

    with **open**('questions.bin', 'rb') as binFile:

        rows = **pickle**.**load**(binFile)

*# Insert the data back into the database*

    for row in rows:

        quesID = row[0]

        cursor.**execute**(

            'SELECT quesID FROM questions WHERE quesID = ?', (quesID,))

        if not cursor.**fetchone**():

            cursor.**execute**('''

            INSERT INTO questions (quesID, Question, AnswerA, AnswerB, AnswerC, AnswerD, CorrectAnswer)

            VALUES (?, ?, ?, ?, ?, ?, ?)

            ''', row)

    conn.**commit**()

    conn.**close**()

def **loadData**(table\_widget):

    conn = **create\_connection**()

    cursor = conn.**cursor**()

*# Execute the query*

    cursor.**execute**("SELECT \* FROM questions ORDER BY quesID")

*# Get all rows*

    rows = cursor.**fetchall**()

*# Set the number of rows in the table*

    table\_widget.setRowCount(**len**(rows))

*# Loop through each row*

    for i, row in **enumerate**(rows):

*# Loop through each column*

        for j, value in **enumerate**(row):

*# Add the item to the table*

            table\_widget.setItem(i, j, **QTableWidgetItem**(**str**(value)))

    conn.**close**()

def **displayInFormulaBar**(ui, row):

    ui.delQuesbutton.setDisabled(False)

    ui.IDtxtbox.setText(ui.tableWidget.item(row, 0).text())

    ui.questiontxtbox.setText(ui.tableWidget.item(row, 1).text())

    ui.ansAtxtbox.setText(ui.tableWidget.item(row, 2).text())

    ui.ansBtxtbox.setText(ui.tableWidget.item(row, 3).text())

    ui.ansCtxtbox.setText(ui.tableWidget.item(row, 4).text())

    ui.ansDtxtbox.setText(ui.tableWidget.item(row, 5).text())

    ui.corrAnstxtbox.setText(ui.tableWidget.item(row, 6).text())

    ui.delQuesbutton.setDisabled(False)

def **resetFormulaBar**(ui):

    ui.IDtxtbox.setText("")

    ui.questiontxtbox.setText("")

    ui.ansAtxtbox.setText("")

    ui.ansBtxtbox.setText("")

    ui.ansCtxtbox.setText("")

    ui.ansDtxtbox.setText("")

    ui.corrAnstxtbox.setText("")

def **ReloadQTableWidget**(ui):

    ui.IDtxtbox.setText("")

    ui.questiontxtbox.setText("")

    ui.ansAtxtbox.setText("")

    ui.ansBtxtbox.setText("")

    ui.ansCtxtbox.setText("")

    ui.ansDtxtbox.setText("")

    ui.corrAnstxtbox.setText("")

### Phụ lục 6: Chi tiết các hàm trong ‘examAdmin\_src.py’

def **create\_connection**():

    return **sqlite3**.**connect**('QuestionBank.db')

class **ExamAdmin**:

    def **\_\_init\_\_**(self, ui):

        self.ui = ui

        self.scoreTrack = 0

        self.quesCount = 1

        self.timer = **QTimer**()

        self.timer.timeout.**connect**(self.**checkTimeLeft**)

    def **startQuiz**(self):

**cleanUpResultsDetail**()

        self.scoreTrack = 0

        self.quesCount = 1

        numOfQues = self.ui.numofQuesInputbox.text().strip()

        amouOfTime = self.ui.amountofTimeInputbox.text().strip()

*# Check condition for user input*

        if not numOfQues.isdigit() or not amouOfTime.isdigit():

**QMessageBox**.**warning**(self.ui, "Input Error",

                                "Please enter integer values.")

            self.ui.numofQuesInputbox.setText("")

            self.ui.amountofTimeInputbox.setText("")

            return

        if **int**(numOfQues) < 1 or **int**(numOfQues) > self.ui.tableWidget.rowCount():

**QMessageBox**.**warning**(self.ui, "Input Error",

                                "Invalid question number.")

            self.ui.numofQuesInputbox.setText("")

            self.ui.amountofTimeInputbox.setText("")

            return

        if not numOfQues or not amouOfTime:

**QMessageBox**.**warning**(

                self.ui, 'Warning', 'Please enter the number of questions and time.')

            return

*# Get a random question and display it*

        self.curQues = **chooseRandomQues**()

        self.**insertQuesIDToDB**(self.curQues[0])

**displayQues**(self.curQues, self.ui.questionDisplaybox,

                    self.ui.ansAlabel, self.ui.ansBlabel, self.ui.ansClabel, self.ui.ansDlabel)

*# Enable the answer buttons*

**enableAnsButton**(self)

        self.ui.quesCountlabel.setText(f"{self.quesCount}/{numOfQues}")

        amouOfTime = **int**(self.ui.amountofTimeInputbox.text().strip())

        self.ui.progressBar.setMaximum(amouOfTime)

        self.ui.progressBar.setValue(0)

        self.timer.**start**(1000)

        self.ui.quesCountlabel.setText(

            f"{self.quesCount}/{self.ui.numofQuesInputbox.text().strip()}")

    def **checkTimeLeft**(self):

        curValue = self.ui.progressBar.value()

        if curValue < self.ui.progressBar.maximum():

            self.ui.progressBar.setValue(curValue + 1)

        else:

            self.timer.**stop**()

            score\_text = f"{

                self.scoreTrack}/{self.ui.numofQuesInputbox.text().strip()}"

            self.ui.scoretlabel.setText(score\_text)

            self.**saveResultToDB**()

            self.**saveDetailResultToDB**()

            self.**displayResult**()

            self.**resetToInitialState**()

            self.**displayResultsInList**()

            self.**displayAllResults**()

**QMessageBox**.**information**(self.ui, "Time's up!", "Time has ended.")

    def **checkCorrAns**(self, answer):

*# Insert user's answer into 'urAns'*

        self.**insertUserAnswerToDB**(answer)

*# Check if the selected answer is correct*

        if answer == self.curQues[6]:

            self.scoreTrack += 1

        self.quesCount += 1

        self.ui.quesCountlabel.setText(

            f"{self.quesCount}/{self.ui.numofQuesInputbox.text().strip()}")

*# Check if there are more questions*

        if self.quesCount <= **int**(self.ui.numofQuesInputbox.text().strip()):

*# Proceed to the next question*

            self.curQues = **chooseRandomQues**()

*# Insert 'quesID' into 'resultsDetail'*

            self.**insertQuesIDToDB**(self.curQues[0])

**displayQues**(self.curQues, self.ui.questionDisplaybox,

                        self.ui.ansAlabel, self.ui.ansBlabel, self.ui.ansClabel, self.ui.ansDlabel)

        else:

*# Display final score*

            self.timer.**stop**()

            score\_text = f"{

                self.scoreTrack}/{self.ui.numofQuesInputbox.text().strip()}"

            self.ui.scoretlabel.setText(score\_text)

            self.**saveResultToDB**()

            self.**saveDetailResultToDB**()

            self.**displayResult**()

            self.**resetToInitialState**()

            self.**displayResultsInList**()

            self.**displayAllResults**()

**QMessageBox**.**information**(

                self.ui, "Quiz Finished", "You have answered all questions.")

    def **insertUserAnswerToDB**(self, answer):

        conn = **create\_connection**()

        cursor = conn.**cursor**()

        cursor.**execute**(

            "UPDATE resultsDetail SET urAns = ? WHERE quesID = ?", (answer, self.curQues[0]))

        conn.**commit**()

        conn.**close**()

    def **insertQuesIDToDB**(self, quesID):

        conn = **create\_connection**()

        cursor = conn.**cursor**()

        cursor.**execute**(

            "INSERT INTO resultsDetail (quesID) VALUES (?)", (quesID,))

        conn.**commit**()

        conn.**close**()

    def **displayResult**(self):

        conn = **create\_connection**()

        cursor = conn.**cursor**()

*# Retrieve the last 5 scores from the 'userResult' table*

        cursor.**execute**(

            "SELECT result FROM userResult ORDER BY resID DESC LIMIT 5")

        results = cursor.**fetchall**()

*# Display the results in the QLabel widgets*

        labels = [self.ui.result1displaylabel, self.ui.result2displaylabel,

                  self.ui.result3displaylabel, self.ui.result4displaylabel, self.ui.result5displaylabel]

        for i, result in **enumerate**(results):

            labels[i].setText(result[0])

        conn.**close**()

    def **saveResultToDB**(self):

        conn = **create\_connection**()

        cursor = conn.**cursor**()

        score\_text = f"{

            self.scoreTrack}/{self.ui.numofQuesInputbox.text().strip()}"

        cursor.**execute**(

            "INSERT INTO userResult (result) VALUES (?)", (score\_text,))

        conn.**commit**()

        conn.**close**()

    def **saveDetailResultToDB**(self):

        conn = **create\_connection**()

        cursor = conn.**cursor**()

        cursor.**execute**(

            "SELECT resID FROM userResult ORDER BY resID DESC LIMIT 1")

        resID = cursor.**fetchone**()[0]

        cursor.**execute**(

            "UPDATE resultsDetail SET resID = ? WHERE resID IS NULL", (resID,))

        conn.**commit**()

        conn.**close**()

    def **displayAllResults**(self):

*# Connect to the database*

        conn = **create\_connection**()

        cursor = conn.**cursor**()

*# Execute a query to select all data*

        cursor.**execute**("""

            SELECT q.quesID, rd.resID, q.CorrectAnswer, rd.urAns, q.Question, q.AnswerA, q.AnswerB, q.AnswerC, q.AnswerD

            FROM resultsDetail rd

            INNER JOIN questions q ON rd.quesID = q.quesID

        """)

*# Fetch all rows from the last executed statement*

        results = cursor.**fetchall**()

*# Clear the QTableWidget*

        self.ui.resultTableWidget.setRowCount(0)

*# Add items to the QTableWidget*

        for row\_number, row\_data in **enumerate**(results):

            self.ui.resultTableWidget.insertRow(row\_number)

            for column\_number, data in **enumerate**(row\_data):

                self.ui.resultTableWidget.setItem(

                    row\_number, column\_number, **QTableWidgetItem**(**str**(data)))

*# Close the connection*

        conn.**close**()

    def **displayResultsInList**(self):

*# Connect to the database*

        conn = **create\_connection**()

        cursor = conn.**cursor**()

*# Execute the query*

        cursor.**execute**("SELECT resID, result FROM userResult")

*# Fetch all rows from the last executed statement*

        results = cursor.**fetchall**()

*# Clear the QListWidget*

        self.ui.listExamWidget.clear()

*# Add items to the QListWidget*

        for result in results:

            item\_text = f"{result[0]} - {result[1]}"

            self.ui.listExamWidget.addItem(item\_text)

*# Close the connection*

        conn.**close**()

    def **displaySelectedResultFromList**(self):

*# Get the selected item from the QListWidget*

        selected\_item = self.ui.listExamWidget.currentItem()

        if selected\_item is not None:

*# Connect to the database*

            conn = **create\_connection**()

            cursor = conn.**cursor**()

*# Get the resID from the selected item's text*

            resID = selected\_item.text().split(' - ')[0]

*# Execute the query*

            cursor.**execute**("""

                SELECT q.quesID, rd.resID, q.CorrectAnswer, rd.urAns, q.Question, q.AnswerA, q.AnswerB, q.AnswerC, q.AnswerD

                FROM resultsDetail rd

                INNER JOIN questions q ON rd.quesID = q.quesID

                WHERE rd.resID = ?

            """, (resID,))

*# Fetch all rows from the last executed statement*

            results = cursor.**fetchall**()

*# Clear the QTableWidget*

            self.ui.resultTableWidget.setRowCount(0)

*# Add items to the QTableWidget*

            for row\_number, row\_data in **enumerate**(results):

                self.ui.resultTableWidget.insertRow(row\_number)

                for column\_number, data in **enumerate**(row\_data):

                    self.ui.resultTableWidget.setItem(

                        row\_number, column\_number, **QTableWidgetItem**(**str**(data)))

*# Close the connection*

            conn.**close**()

    def **resetToInitialState**(self):

        self.timer.**stop**()

        self.quesCount = 1

        self.scoreTrack = 0

        self.ui.ansApushButton.setDisabled(True)

        self.ui.ansBpushButton.setDisabled(True)

        self.ui.ansCpushButton.setDisabled(True)

        self.ui.ansDpushButton.setDisabled(True)

        self.ui.numofQuesInputbox.setDisabled(False)

        self.ui.amountofTimeInputbox.setDisabled(False)

        self.ui.quesCountlabel.setText("/")

        self.ui.ansAlabel.setText("Answer A")

        self.ui.ansBlabel.setText("Answer B")

        self.ui.ansClabel.setText("Answer C")

        self.ui.ansDlabel.setText("Answer D")

        self.ui.numofQuesInputbox.setText("")

        self.ui.amountofTimeInputbox.setText("")

        self.ui.questionDisplaybox.setText(

            "| | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |"

        )

        self.ui.progressBar.setValue(0)

    def **resetResultBoard**(self):

        msgBox = **QMessageBox**()

        msgBox.**setIcon**(**QMessageBox**.**Icon**.Question)

        msgBox.**setText**("Are you sure you want to reset the board?")

        msgBox.**setWindowTitle**("Confirmation")

        msgBox.**setStandardButtons**(

**QMessageBox**.**StandardButton**.Yes **|** **QMessageBox**.**StandardButton**.No)

        returnValue = msgBox.**exec**()

        if returnValue == **QMessageBox**.**StandardButton**.Yes:

            conn = **create\_connection**()

            cursor = conn.**cursor**()

*# Clear 'resultsDetail' table*

            cursor.**execute**('DELETE FROM resultsDetail')

*# Clear 'userResult' table*

            cursor.**execute**('DELETE FROM userResult')

            conn.**commit**()

            conn.**close**()

            self.ui.result1displaylabel.setText("/")

            self.ui.result2displaylabel.setText("/")

            self.ui.result3displaylabel.setText("/")

            self.ui.result4displaylabel.setText("/")

            self.ui.result5displaylabel.setText("/")

            self.ui.listExamWidget.clear()

        self.**displayAllResults**()

        self.**displayResult**()

def **startTimer**(amouOfTime, progress\_bar):

    timer = **QTimer**()

    timer.timeout.**connect**(lambda: **updateProgressBar**(progress\_bar))

    timer.**start**(1000)

*# Initialize progress bar*

    progress\_bar.setMaximum(amouOfTime)

    progress\_bar.setValue(amouOfTime)

def **updateProgressBar**(progress\_bar):

    curValue = progress\_bar.value()

    if curValue > 0:

        progress\_bar.setValue(curValue - 1)

    else:

        progress\_bar.setValue(0)

def **displayQues**(question, question\_label, ansA\_label, ansB\_label, ansC\_label, ansD\_label):

*# Display the question and its answers*

    question\_label.setText(question[1])

    ansA\_label.setText(question[2])

    ansB\_label.setText(question[3])

    ansC\_label.setText(question[4])

    ansD\_label.setText(question[5])

def **enableAnsButton**(self):

    self.ui.ansApushButton.setDisabled(False)

    self.ui.ansBpushButton.setDisabled(False)

    self.ui.ansCpushButton.setDisabled(False)

    self.ui.ansDpushButton.setDisabled(False)

def **cleanUpResultsDetail**():

    conn = **create\_connection**()

    cursor = conn.**cursor**()

*# Delete rows where 'resID' is null*

    cursor.**execute**("DELETE FROM resultsDetail WHERE resID IS NULL")

    conn.**commit**()

    conn.**close**()