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

[PHẦN 1 MỤC LỤC 2](#_Toc169040725)

[PHẦN 1 TÓM TẮT – ABSTRACT 3](#_Toc169040726)

[PHẦN 2 GIỚI THIỆU – INTRODUCTION 3](#_Toc169040727)

[2.1. Thông tin nền – Background Information 3](#_Toc169040728)

[2.2. Mục tiêu – Objectives 3](#_Toc169040729)

[2.3. Phạm vi – Scope 3](#_Toc169040730)

[PHẦN 3 PHƯƠNG PHÁP – METHODOLOGY 4](#_Toc169040731)

[3.1. Phát Biểu Vấn Đề - Problem Statement 4](#_Toc169040732)

[3.2. Tiếp Cận – Approach 4](#_Toc169040733)

[3.3. Thiết Kế - Desgin 4](#_Toc169040734)

[3.4. Công Cụ và Công Nghệ – Tools and Technologies 5](#_Toc169040735)

[PHẦN 4 TRIỂN KHAI – IMPLEMENTATION 5](#_Toc169040736)

[4.1. Tổng Quan Cấu Trúc Chương Trình 5](#_Toc169040737)

[4.2. Chi Tiết Các Module Của Chương Trình 5](#_Toc169040738)

[Chi Tiết Module ‘questionBank\_src.py’: 5](#_Toc169040739)

[Chi Tiết Module ‘quesAdmin\_src.py’ 8](#_Toc169040740)

# 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

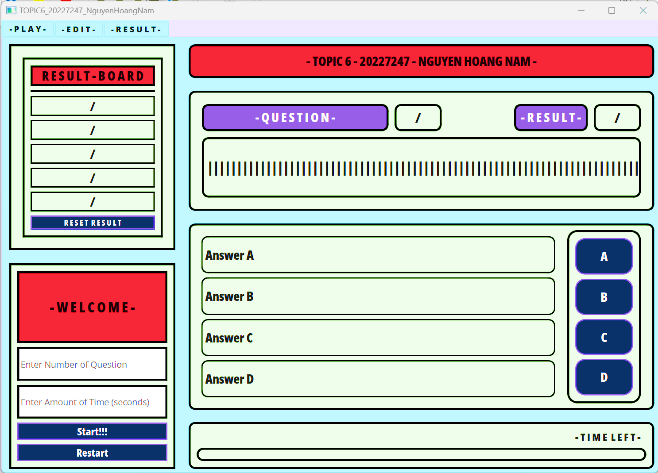
Các công cụ được áp dụng, liên kết với nhau giúp chương trình được thiết kế và hoạt động hiệu quả.

# 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

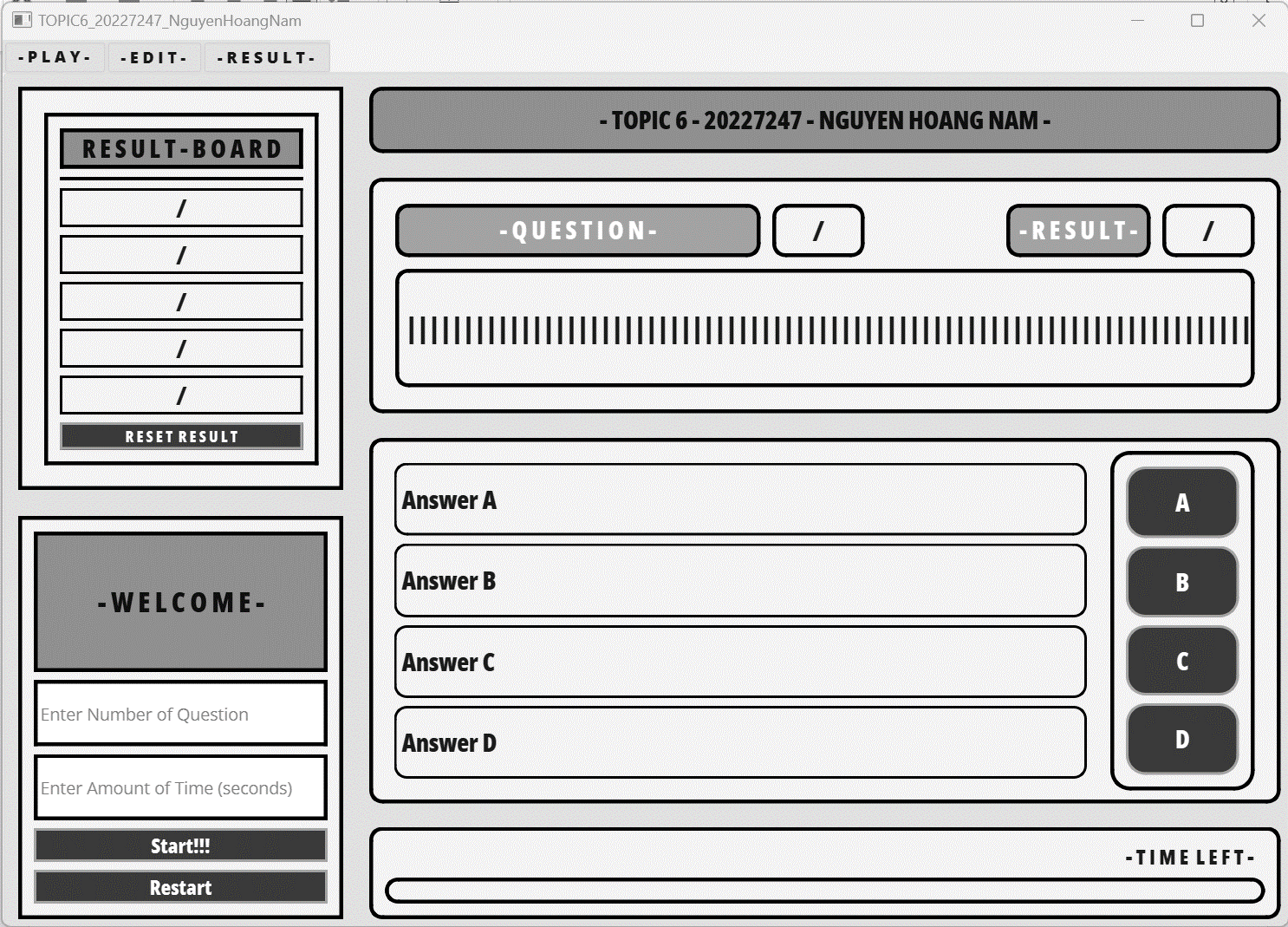


* Tab ‘PLAY’: Giao diện làm bài kiểm tra cho phép người dùng nhập số câu hỏi kiểm tra, nhập thời gian kiểm tra, bắt đầu bài kiểm tra. Thêm vào đó còn hiển thị kết quả các lần kiểm tra gần đây
* Tab ‘EDIT’: Giao diện quản lý câu hỏi cho phép người dùng thấy được ngân hàng câu hỏi, chỉnh sửa các thành 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

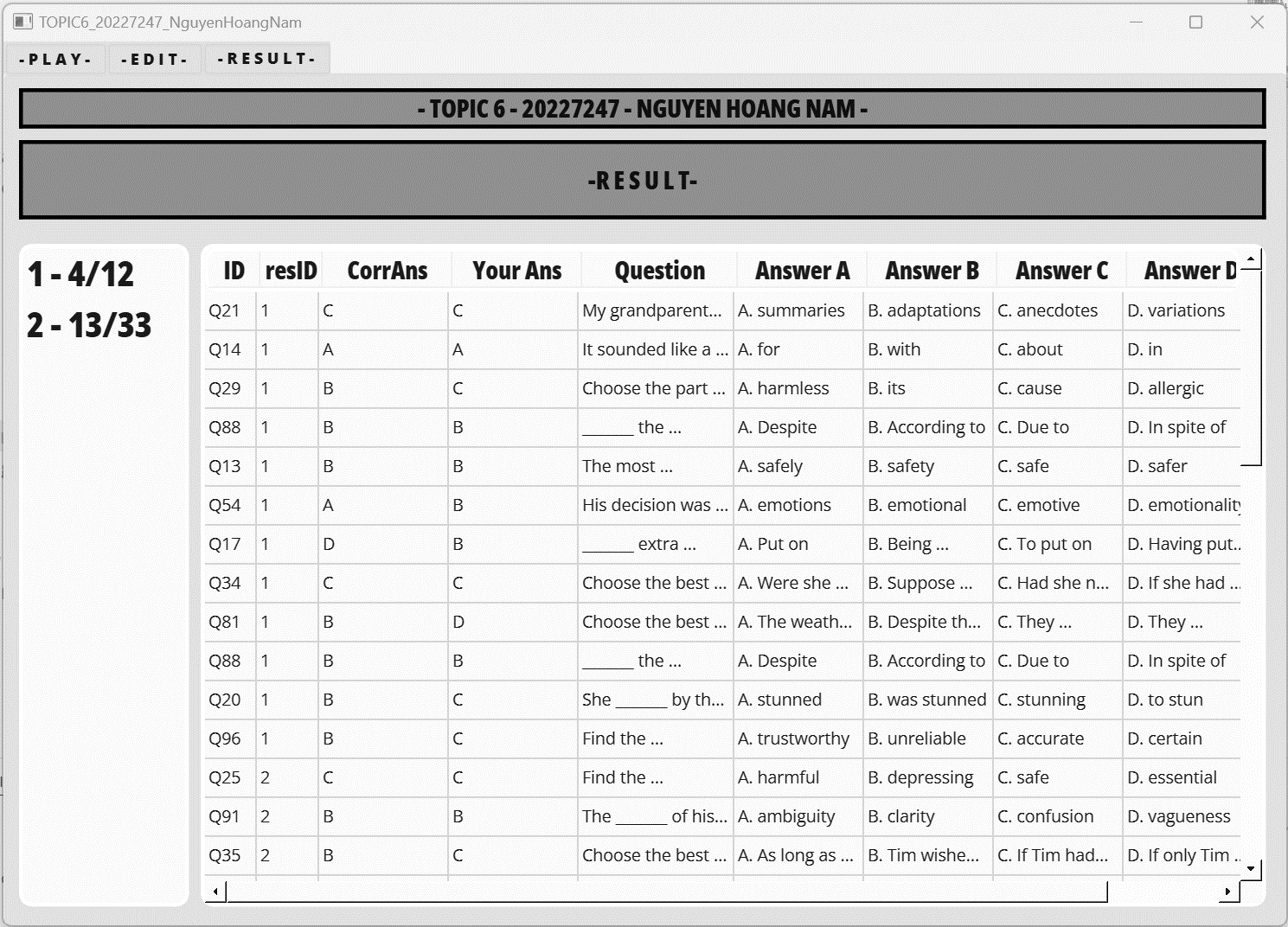
(4) Bảng hiển thị câu hỏi: Hiển thị thông tin từng câu hỏi

(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**(question, question\_label, ansA\_label, ansB\_label, ansC\_label, ansD\_label):
  + 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’:

Nhiệm vụ chính của module là tạo lập cơ sở dữ liệu, các bảng ban đầu cho các câu hỏi và kết quả bài kiểm tra của người dùng. Module sử dụng sqlite cho việc tạo cơ sở dữ liệu và các bảng trong cơ sở dữ liệu

#### Tạo Lập Cơ Sở Dữ Liệu:

##### Chi Tiết Chức Năng:

Thiết Lập Kết Nối: đoạn mã kết nối với SQL Server đang chạy trên máy người dùng qua thư viện ‘**pyodbc**’. Cụ thể ở đây là ‘HOANGNAM\\SQLEXPRESS’.

Kiểm Tra Tồn Tại: đoạn mã truy vấn ‘master.sys.databases’ để kiểm tra sự tồn tại của database có tên ‘QuestionBank’.

Điều Kiện Khởi Tạo Database: nếu ‘QuestionBank’ chưa tồn tại (điều này được quyết đinh do ‘cursor.**fetchone**()’ trả về giá trị ‘None’), đoạn mã đóng kết nối hiện tại và tái kết nối lại và khởi tạo database ‘QuestionBank’.

#### Tọa Lập Bảng Trong Cơ Sở Dữ Liệu:

###### def createTables():

    conn = **pyodbc**.**connect**('Driver={SQL Server};'

                          'Server=HOANGNAM\\SQLEXPRESS;'

                          'Database=QuestionBank;'

                          'Trusted\_Connection=yes;')

    cursor = conn.**cursor**()

*# Check for table 'questions' existent*

    cursor.**execute**(

        "SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'questions'")

    if not cursor.**fetchone**():

        cursor.**execute**('''

        CREATE TABLE questions (

            quesID VARCHAR(255) UNIQUE,

            Question VARCHAR(255),

            AnswerA VARCHAR(255),

            AnswerB VARCHAR(255),

            AnswerC VARCHAR(255),

            AnswerD VARCHAR(255),

            CorrectAnswer CHAR(1),

            PRIMARY KEY (quesID, Question)

        )

        ''')

*# Check for table 'userResult' existent*

    cursor.**execute**(

        "SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_NAME = 'userResult'")

    if not cursor.**fetchone**():

        cursor.**execute**('''

        CREATE TABLE userResult (

            resID INT IDENTITY(1,1) PRIMARY KEY,

            result VARCHAR(255)

        )

        ''')

    conn.**commit**()

    conn.**close**()

##### Chi Tiết Chức Năng:

Thiết Lập Kết Nối: đoạn mã kết nối với SQL Server đang chạy trên máy người dùng, sử dụng database ‘QuestionBank’ qua ‘**pyodbc**’. Cụ thể ở đây là ‘HOANGNAM\\SQLEXPRESS’.

Kiểm Tra Tồn Tại: đoạn mã truy vấn trong ‘QuestionBank’ để kiểm tra sự tồn tại của hai bảng ‘questions’ và ‘userResult’.

Điều Kiện Khởi Tạo Table: nếu ‘questions’ và ‘userResult’ chưa tồn tại (điều này được quyết đinh do ‘cursor.**fetchone**()’ trả về giá trị ‘None’), đoạn mã thực hiện khởi tạo database ‘questions’ và ‘userResult’.

##### 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’

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.

#### Hàm Tạo ‘.Bin’ File Từ Dữ Liệu Trong Database

*# Serialize data from database to .bin file*

###### def dbToBinFIle():

    conn = **pyodbc**.**connect**('Driver={SQL Server};'

                          'Server=HOANGNAM\\SQLEXPRESS;'

                          'Database=QuestionBank;'

                          'Trusted\_Connection=yes;')

    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**()

##### Chi Tiết Chức Năng:

**Thiết Lập Kết Nối:** đoạn mã kết nối với SQL Server Database ‘QuestionBank’ đang chạy trên máy người dùng qua thư viện ‘**pyodbc**’.

**Truy vấn:** hàm thực hiện ‘SELECT’ để lấy về tất cả các hàng trong bảng ‘questions’

**Phục hội dữ liệu:** đoạn mã lấy các hàng được trả về bởi truy vấn trên thông qua ‘cursor.**fetchall**()’

**Chuyển đổi data:**

* Các hàng được lấy về sau đó được chuyển về dạng nhị phân thông qua module từ thư viện ‘pickle’
* Dữ liệu dạng nhị phân được lưu vào file tên ‘questions.bin’

#### Hàm Chuyển Dữ Liệu Từ ‘Questions.Bin’ Vào Database:

*# Deserialize .bin file*

###### def BinFileTodb():

    conn = **pyodbc**.**connect**('Driver={SQL Server};'

                          'Server=HOANGNAM\\SQLEXPRESS;'

                          'Database=QuestionBank;'

                          'Trusted\_Connection=yes;')

    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**()

##### Chi Tiết Chức Năng:

**Thiết Lập Kết Nối:** đoạn mã kết nối với SQL Server đang chạy trên máy người dùng qua thư viện ‘**pyodbc**’. Cụ thể ở đây là ‘HOANGNAM\\SQLEXPRESS’.

**Tải dữ liệu:** chương trình mở file nhị phân tên ‘questions.bin’ ở chế độ read-binary (‘rb’) và dùng module trong thư viện ‘**pickle**’ để deserialize(giải mã hóa) file nhị phân đó thành một danh sách các hàng.

**Chèn dữ liệu:**

* Với mỗi hàng trong danh sách, đoạn mã kiểm tra sự tồn tại của dữ liệu thuộc cột ‘quesID’(mã câu hỏi) qua lệnh ‘SELECT’
* Nếu ‘quesID’ không tồn tại (điều này được quyết đinh do ‘cursor.**fetchone**()’ trả về giá trị ‘None’), đoạn mã tiến hành ‘INSERT’ các hàng vào bảng.

#### Hàm hiển Thị Dữ Liệu Từ Database Vào Bảng Trên Màn Hình Giao Diện

*# Load data from database into QTable*

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

    conn = **pyodbc**.**connect**('Driver={SQL Server};'

                          'Server=HOANGNAM\\SQLEXPRESS;'

                          'Database=QuestionBank;'

                          'Trusted\_Connection=yes;')

    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**()

##### Chi Tiết Chức Năng:

**Thiết Lập Kết Nối:** đoạn mã kết nối với SQL Server đang chạy trên máy người dùng, sử dụng database ‘QuestionBank’ qua ‘**pyodbc**’. Cụ thể ở đây là ‘HOANGNAM\\SQLEXPRESS’.

**Truy vấn:** hàm thực hiện ‘SELECT’ để lấy về tất cả các hàng trong bảng ‘questions’

**Phục hội dữ liệu:** đoạn mã lấy các hàng được trả về bởi truy vấn trên thông qua ‘cursor.**fetchall**()’

**Cài đặt bảng:**

* Số lượng hàng của ‘tiện ích’ bảng (table widget) được đặt bằng với số hàng phục hồi từ databse
* Tạo một ‘tiện ích’ bảng để hiện thị dữ liệu

**Nhập dữ liệu:**

* Hàm tạo vòng lặp duyệt qua từng hàng của dữ liệu
* Với mỗi hàng, hàm kiểm tra từng giá trị trị của hàng đó
* Mỗi giá trị được chuyển sang dạng ‘string’ và được thêm vào ‘tiện ích’ bảng

#### Hiển Thị Giá Trị Của Bảng Vào Thanh Công thức

*# Display question and its answer in QEditLine*

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

    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())

##### Chi Tiết Chức Năng

**Phục hội dữ liệu:** hàm nhận hai tham số: ui là đại diện cho lớp giao diện người dùng và hàng là một số nguyên đại diện cho mã hàng trong bảng

**Nhập dữ liệu:**

* Hàm đọc các chữ ở mỗi ô trong một hàng nhất định trong ‘tiện ích’ bảng
* Các chữ trong ô được hiện thị tương ứng cho từng hộp chữ (text box)

#### Hàm Thêm Câu Hỏi Vào Database

*# Create a QMessageBox for confirmation*

###### def insert\_question(ui):

    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 = ui.IDEditWidget.text()

        question = ui.quesBoxEditWidget.text()

        ansA = ui.ansAEditWidget.text()

        ansB = ui.ansBEditWidget.text()

        ansC = ui.ansCEditWidget.text()

        ansD = ui.ansDEditWidget.text()

*# Get the current text from the QComboBox*

        corrAns = 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 = **pyodbc**.**connect**('Driver={SQL Server};'

                              'Server=HOANGNAM\\SQLEXPRESS;'

                              'Database=QuestionBank;'

                              'Trusted\_Connection=yes;')

        cursor = conn.**cursor**()

*# 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**(ui.tableWidget)

        ui.quesBoxEditWidget.setText("")

        ui.ansAEditWidget.setText("")

        ui.ansBEditWidget.setText("")

        ui.ansCEditWidget.setText("")

        ui.ansDEditWidget.setText("")

        ui.corrAnscomboBox.setCurrentIndex(0)

        ui.addQuesWidget\_2.hide()

**dbToBinFIle**()

##### Chi Tiết Chức Năng:

**Xác nhận người dùng:**

* Một hộp thoại sẽ được hiện thị để xác nhận yêu cầu thêm câu hỏi
* Nếu người dùng nhấn ‘Yes’ để xác nhận, hàm được thực hiện

**Phục hồi dữ liệu:** Hàm lấy các dữ liệu của câu hỏi từ nhiều thành phần ‘QLineEdit’

**Kiểm tra điều kiện:**

* + Hàm kiểm tra các hộp chữ đã được điển đầy đủ chưa
  + Nếu có hộp nào rỗng, một thông báo sẽ được hiện thị

**Thiết lập kết nối:** Hàm tạo một kết nối tới database có tên ‘QuestionBank’

**Nhập liệu:** dữ liệu chi tiết của câu hỏi được thêm vào bảng ‘questions’ sử dụng lệnh ‘INSERT’ trong SQL Server

**Thông báo thành công:** một hộp thoại thông báo thành công được hiện thị cho thấy dữ liệu đã được nhập vào database

**Cập nhật giao diện:**

* + Bảng QtableWidget được làm mới, hiển thị lại thông tin mới
  + Các hộp chữ QlineEdit được làm trống
  + Ẩn cửa sổ nhập thông tin câu hỏi
  + Viết đè lên file nhị phân

#### Hàm Sửa Câu Hỏi Trong Database

*# Update question function*

###### def updateQues(ui):

*# 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 = **pyodbc**.**connect**('Driver={SQL Server};'

                              'Server=HOANGNAM\\SQLEXPRESS;'

                              'Database=QuestionBank;'

                              'Trusted\_Connection=yes;')

        cursor = conn.**cursor**()

*# Get the updated values from QLineEdit widgets*

        quesID = ui.IDtxtbox.text()

        question = ui.questiontxtbox.text()

        ansA = ui.ansAtxtbox.text()

        ansB = ui.ansBtxtbox.text()

        ansC = ui.ansCtxtbox.text()

        ansD = ui.ansDtxtbox.text()

        corrAns = ui.corrAnstxtbox.text()

*# 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**(ui.tableWidget)

##### Chi Tiết Chức Năng:

**User Confirmation:**

A message box is displayed to confirm if the user wants to update the question.

If the user clicks ‘Yes’, the function proceeds; otherwise, it exits.

**Database Connection:**

A connection to the ‘QuestionBank’ database is established using pyodbc.

**Data Retrieval:**

The function retrieves updated text from various QLineEdit widgets for the question details and correct answer.

**Data Update:**

The updated question details are used to execute an UPDATE statement that modifies the existing question in the database.

**Commit and Close Connection:**

The transaction is committed to save changes to the database.

The database connection is closed.

**UI Update:**

The binary file containing the database data is updated with dbToBinFIle.

The QTableWidget is reloaded with loadData to reflect the updated question.

# 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 |