MCQ BASED ONLINE EXAM APPLICATION

Overview:

The MCQ Based Online Exam Application is the application that helps to handle and conduct multiple-choice question-type exams with greater efficiency. The application comprises two main modules: a question management system, question_master.py, where questions are administered and an exam client, exam_client.py, responsible for the conducting of the exams for students.

Objectives:

- To read and manage questions stored in a CSV file.
- To provide an interface for adding, searching, deleting, modifying, and displaying questions.
- To conduct an online exam and evaluate the student's performance.
- To implement logging, exception handling, and modular programming practices.

Functionality

1.Question Management

This module will provide a menu-driven interface for managing questions. The following functionalities will be available:

- **Add a Question:** Automatically assign a question number and allow the user to enter a new question along with its options and the correct answer.
- **Search for a Question:** Retrieve and display a question based on its number.
- **Delete a Question:** Remove a question from the dataset by its number.
- **Modify a Question:** Edit an existing question based on its number.
- **Display All Questions:** Show all questions in the current dataset.
- **Exit:** Terminate the program.

2. Exam Client (exam_client.py)

This module will handle the exam-taking process:

- Display the current date and time.
- Prompt the user for their name and university.
- Present each question along with its options, allowing the user to select an answer.
- After all questions have been answered, calculate and display the student's score.

Error Handling

- Ensure that user inputs are validated.
- Use try-except blocks for handling potential exceptions (e.g., file not found, invalid inputs).

Logging

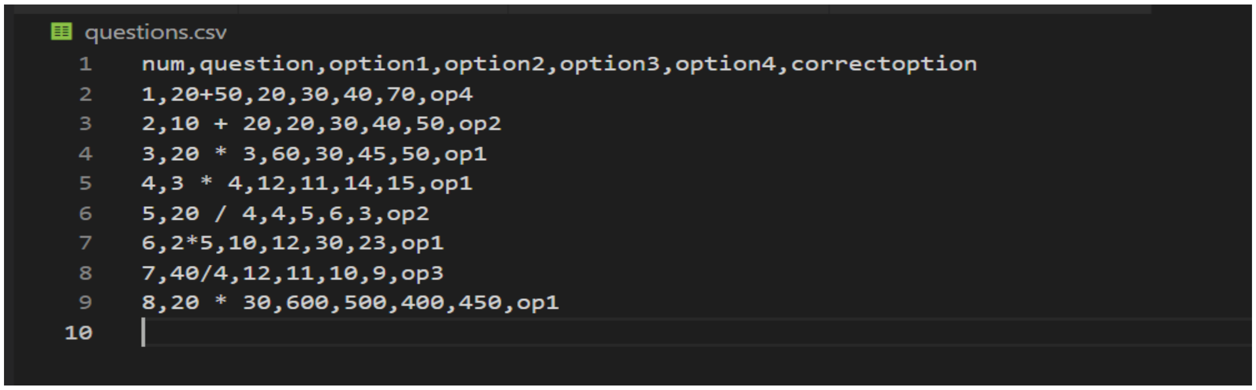
Implement logging to track actions performed in the application, such as adding or deleting questions, and recording user activity during the exam.

Requirements

- Python 3.x
- Libraries: csv, datetime, logging, os, random

Sample CSV Format

- The input CSV file (question.csv) should be structured as follows:



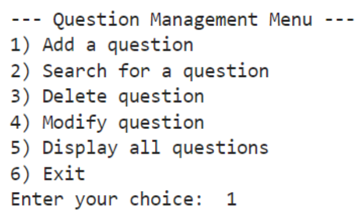
questions.csv

```
num,question,option1,option2,option3,option4,correctoption
1,20+50,20,30,40,70,op4
2,10 + 20,20,30,40,50,op2
3,20 * 3,60,30,45,50,op1
4,3 * 4,12,11,14,15,op1
5,20 / 4,4,5,6,3,op2
6,2*5,10,12,30,23,op1
7,40/4,12,11,10,9,op3
8,20 * 30,600,500,400,450,op1
```

OPERATIONS PERFORMED:

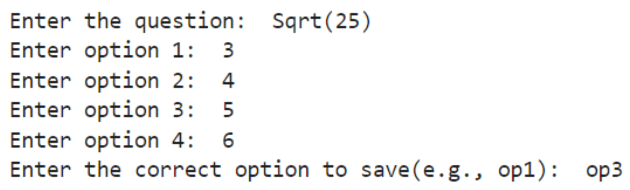
1.Adding a Question

Input



```
--- Question Management Menu ---
1) Add a question
2) Search for a question
3) Delete question
4) Modify question
5) Display all questions
6) Exit
Enter your choice:  1
```

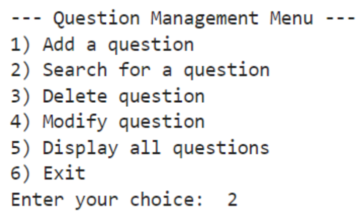
Output



```
Enter the question:  Sqrt(25)
Enter option 1:  3
Enter option 2:  4
Enter option 3:  5
Enter option 4:  6
Enter the correct option to save(e.g., op1):  op3
```

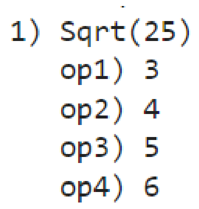
2.Search Question

Input



```
--- Question Management Menu ---
1) Add a question
2) Search for a question
3) Delete question
4) Modify question
5) Display all questions
6) Exit
Enter your choice:  2
```

Output



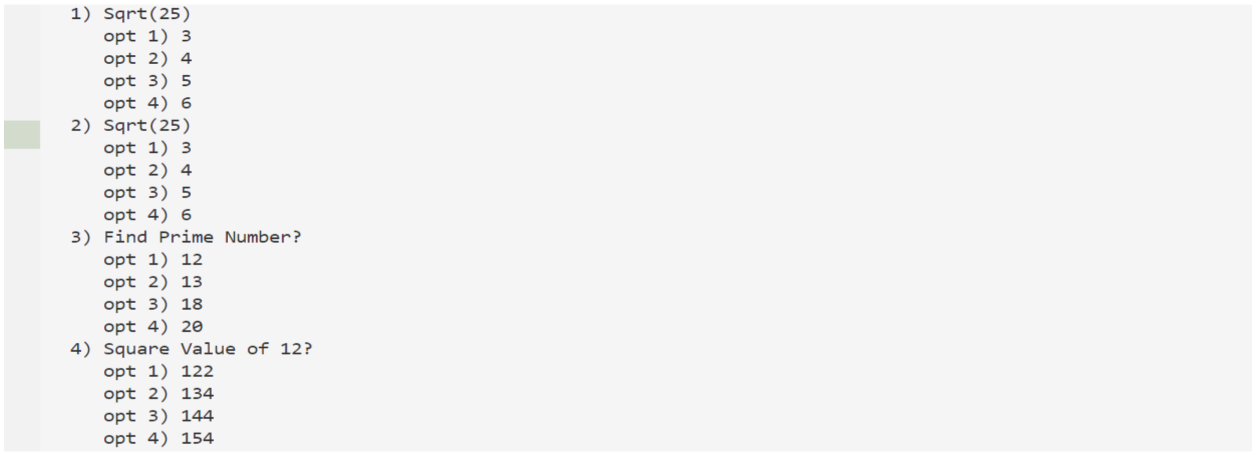
```
1) Sqrt(25)
   op1) 3
   op2) 4
   op3) 5
   op4) 6
```

3.Display All Questions

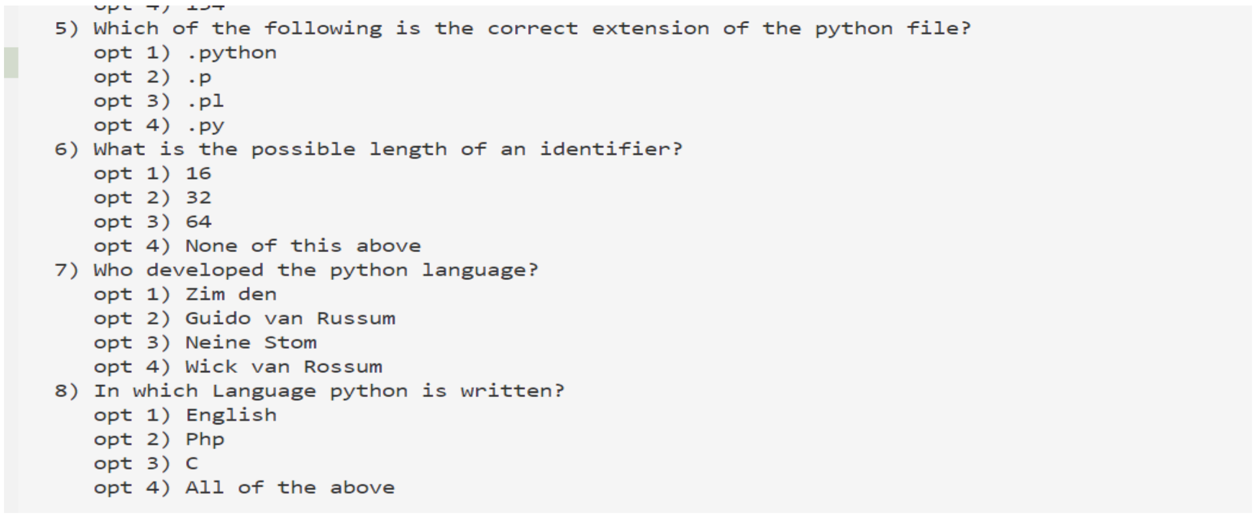
Input

```
--- Question Management Menu ---
1) Add a question
2) Search for a question
3) Delete question
4) Modify question
5) Display all questions
6) Exit
Enter your choice:  5
```

Output

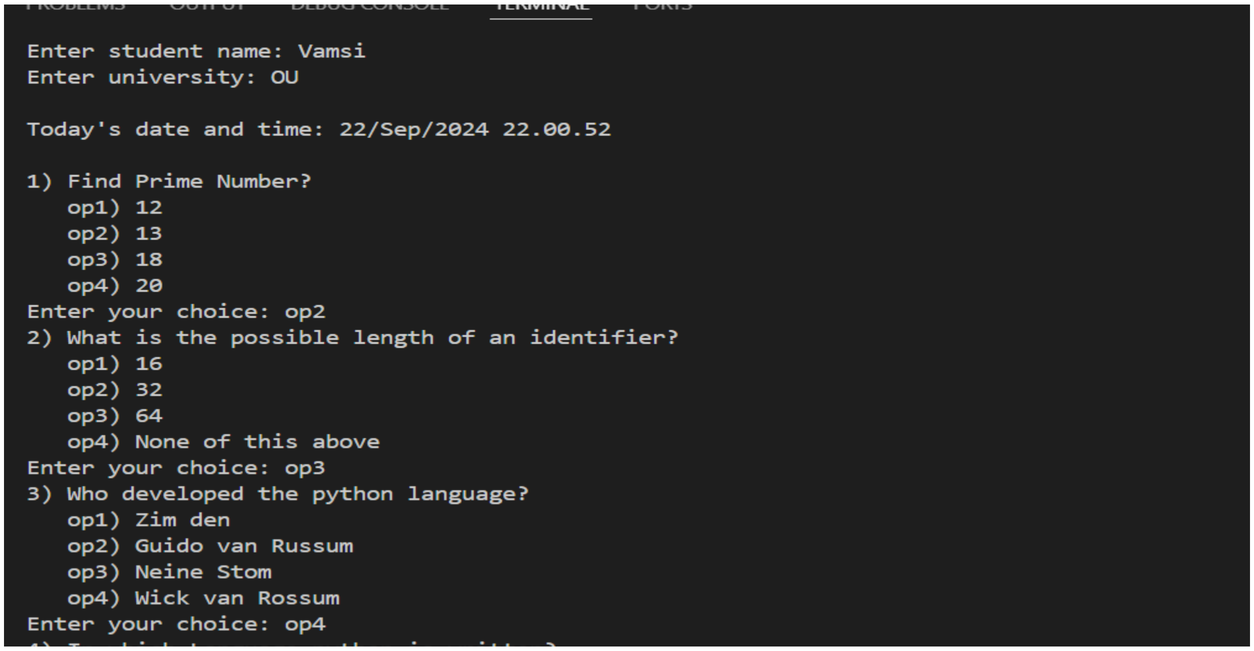


```
1) Sqrt(25)
   opt 1) 3
   opt 2) 4
   opt 3) 5
   opt 4) 6
2) Sqrt(25)
   opt 1) 3
   opt 2) 4
   opt 3) 5
   opt 4) 6
3) Find Prime Number?
   opt 1) 12
   opt 2) 13
   opt 3) 18
   opt 4) 20
4) Square Value of 12?
   opt 1) 122
   opt 2) 134
   opt 3) 144
   opt 4) 154
```

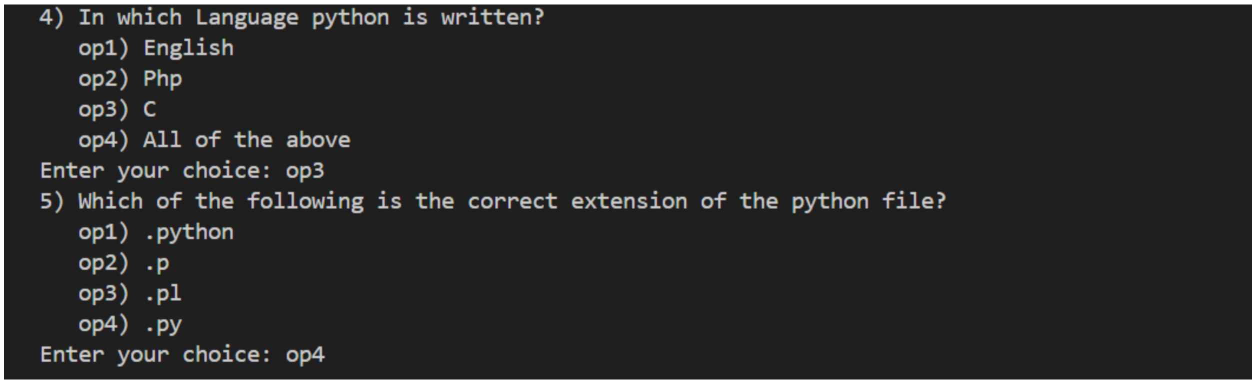


```
5) Which of the following is the correct extension of the python file?
   opt 1) .python
   opt 2) .p
   opt 3) .pl
   opt 4) .py
6) What is the possible length of an identifier?
   opt 1) 16
   opt 2) 32
   opt 3) 64
   opt 4) None of this above
7) Who developed the python language?
   opt 1) Zim den
   opt 2) Guido van Russum
   opt 3) Neine Stom
   opt 4) Wick van Rossum
8) In which Language python is written?
   opt 1) English
   opt 2) Php
   opt 3) C
   opt 4) All of the above
```

Student Exam:

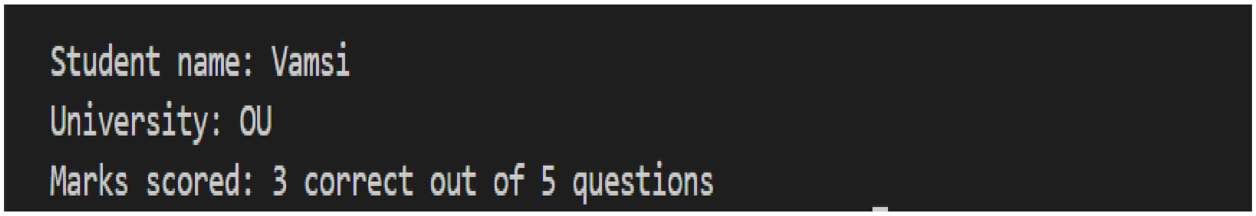


```
Enter student name: Vamsi
Enter university: OU

Today's date and time: 22/Sep/2024 22.00.52

1) Find Prime Number?
   op1) 12
   op2) 13
   op3) 18
   op4) 20
Enter your choice: op2
2) What is the possible length of an identifier?
   op1) 16
   op2) 32
   op3) 64
   op4) None of this above
Enter your choice: op3
3) Who developed the python language?
   op1) Zim den
   op2) Guido van Russum
   op3) Neine Stom
   op4) Wick van Rossum
Enter your choice: op4
```



```
4) In which Language python is written?
   op1) English
   op2) Php
   op3) C
   op4) All of the above
Enter your choice: op3
5) Which of the following is the correct extension of the python file?
   op1) .python
   op2) .p
   op3) .pl
   op4) .py
Enter your choice: op4
```

Student Result



```
Student name: Vamsi
University: OU
Marks scored: 3 correct out of 5 questions
```

Conclusion

The MCQ-Based Online Exam Application is a completely clear mechanism for managing and conducting exams. It comprises two main scripts:

- question_master.py: Manages questions by allowing users to add, search, modify, delete, and display them.
- exam_client.py: Allows learners to take quizzes, read questions, and view the results.

Special Characteristics:

- Modular Structure: Functions are clearly separated for easy maintenance and efficient performance.
- Logging and Error Handling: The system tracks actions and resolves errors effectively.
- User-Friendly: It has a simple command-line interface (CLI) for the user's convenience.

Future Enhancements:

- Graphical User Interface (GUI): For a more visual experience.
- Database Support: To improve performance and scalability.
- Enhanced Reporting: To provide detailed insights into exam performance.

In short, this software is like a strong foundation for online exams and it will be better in the next versions.

Code

Question_master.py:

```
import csv

import os

import logging


# Setup logging

logging.basicConfig(filename='question_master.log', level=logging.INFO)


class QuestionManager:

    def __init__(self, filename='questions.csv'):

        self.questions = []

        self.filename = filename

        self.load_questions()


    #Adding Question to the csv File

    def add_question(self, question_text, options, correct_option):

        num = len(self.questions) + 1

        question = Question(num, question_text, options, correct_option)

        self.questions.append(question)

        self.save_questions()

        logging.info(f"Question added: {question_text}")


    def search_question(self, num):

        for question in self.questions:

            if question.num == num:

                return question

        return None


    def delete_question(self, num):

        self.questions = [q for q in self.questions if q.num != num]

        self.save_questions()

        logging.info(f"Deleted question number: {num}")
```

```
def modify_question(self, num, new_question_text, new_options, new_correct_option):
# Ensure new_options is a non-empty list with valid options
    if not new_options or not isinstance(new_options, list) or len(new_options) == 0:
        logging.error("New options must be a non-empty list.")
        raise ValueError("Options must not be null or empty."
    # Check that all options are non-null and non-empty strings
    for option in new_options:
        if option is None or not isinstance(option, str) or option.strip() == "":
            logging.error("All options must be valid (non-null and non-empty strings).")
            raise ValueError("All options must be valid (non-null and non-empty strings).")

    # Check if the question number exists
    for question in self.questions:
        if question.num == num:
            question.question = new_question_text
            question.options = new_options
            question.correct_option = new_correct_option
            self.save_questions()
            logging.info(f"Modified question number: {num} to '{new_question_text}' with options {new_options}.")
            return

    logging.warning(f"Question number {num} not found for modification.")
    raise ValueError(f"Question number {num} does not exist.")



def display_questions(self):
    for question in self.questions:
        print(f"{question.num}) {question.question}")
        for idx, option in enumerate(question.options, start=1):
            print(f"   opt {idx}) {option}")

def load_questions(self):
        if os.path.exists(self.filename):
            with open(self.filename, mode='r') as file:
                reader = csv.DictReader(file)
                for row in reader:
                    num = int(row['num'])
                    question = row['question']
                    options = [row['option1'], row['option2'], row['option3'], row['option4']]
                    correct_option = row['correctoption']
                    self.questions.append(Question(num, question, options, correct_option))
            logging.info("Loaded questions from CSV.")
        else:
            logging.warning("CSV file not found. Starting with an empty question list.")

    def save_questions(self):
        with open(self.filename, mode='w', newline='') as file:
            fieldnames = ['num', 'question', 'option1', 'option2', 'option3', 'option4', 'correctoption']
            writer = csv.DictWriter(file, fieldnames=fieldnames)
            writer.writeheader()
            for q in self.questions:
                writer.writerow({
                    'num': q.num,
                    'question': q.question,
                    'option1': q.options[0],
                    'option2': q.options[1],
                    'option3': q.options[2],
                    'option4': q.options[3],
                    'correctoption': q.correct_option
                })
            logging.info("Saved questions to CSV.")


class Question:
    def __init__(self, num, question, options, correct_option):
        self.num = num
        self.question = question
        self.options = options
        self.correct_option = correct_option

def menu():
    manager = QuestionManager()
    while True:
        print("\n--- Question Management Menu ---")
        print("1) Add a question")
        print("2) Search for a question")
        print("3) Delete question")
        print("4) Modify question")
        print("5) Display all questions")
        print("6) Exit")
        choice = input("Enter your choice: ")

        if choice == '1':
            question_text = input("Enter the question: ")
            options = [input(f"Enter option {i+1}: ") for i in range(4)]
            correct_option = input("Enter the correct option to save(e.g., op1): ")
            manager.add_question(question_text, options, correct_option)

        elif choice == '2':
            num = int(input("Enter question number to search: "))
            question = manager.search_question(num)
            if question:
                print(f"{question.num}) {question.question}")
                for idx, option in enumerate(question.options, start=1):
                    print(f"   op{idx}) {option}")
            else:
                print("Question not found.")
        elif choice == '3':
            num = int(input("Enter question number to delete: "))
            manager.delete_question(num)

        elif choice == '4':
            num = int(input("Enter question number to modify: "))
            new_question_text = input("Enter the new question: ")
            new_options = [input(f"Enter new option {i+1}: ") for i in range(4)]
            new_correct_option = input("Enter the new correct option (e.g., op1): ")
            manager.modify_question(num, new_question_text, new_options, new_correct_option)

        elif choice == '5':
            manager.display_questions()

        elif choice == '6':
            print("Exiting...")
            break

        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    menu()
```

Exam_Student.py:

```
import datetime
import logging
import random
from question_master import QuestionManager

# Setup logging
logging.basicConfig(filename='exam_client.log', level=logging.INFO)
```

```
def take_exam(manager, num_questions=5):
    name = input("Enter student name: ")
    university = input("Enter university: ")
    score = 0

    print(f"\nToday's date and time: {datetime.datetime.now().strftime('%d/%b/%Y %H.%M.%S')}\n")

    # Select a fixed number of random questions
    if len(manager.questions) < num_questions:
        logging.error("Not enough questions available.")
        print(f"Not enough questions available. Available: {len(manager.questions)}, Required: {num_questions}")
        return
    selected_questions = random.sample(manager.questions, num_questions)
    for idx, question in enumerate(selected_questions, start=1):
        print(f"{idx}) {question.question}")
        for option_idx, option in enumerate(question.options, start=1):
            print(f"   op{option_idx}) {option}")
        answer = input("Enter your choice: ")
        if answer == question.correct_option:
            score += 1
    print(f"\nStudent name: {name}")
    print(f"University: {university}")
    print(f"Marks scored: {score} correct out of {num_questions} questions")
    logging.info(f"Exam completed for {name}, Score: {score}")
if __name__ == "__main__":
    manager = QuestionManager()
    take_exam(manager, num_questions=5)
```

Submitted By:

Bhaskar Jamma(bjamma)

Vinukonda Sai Vamsi(savinuko)