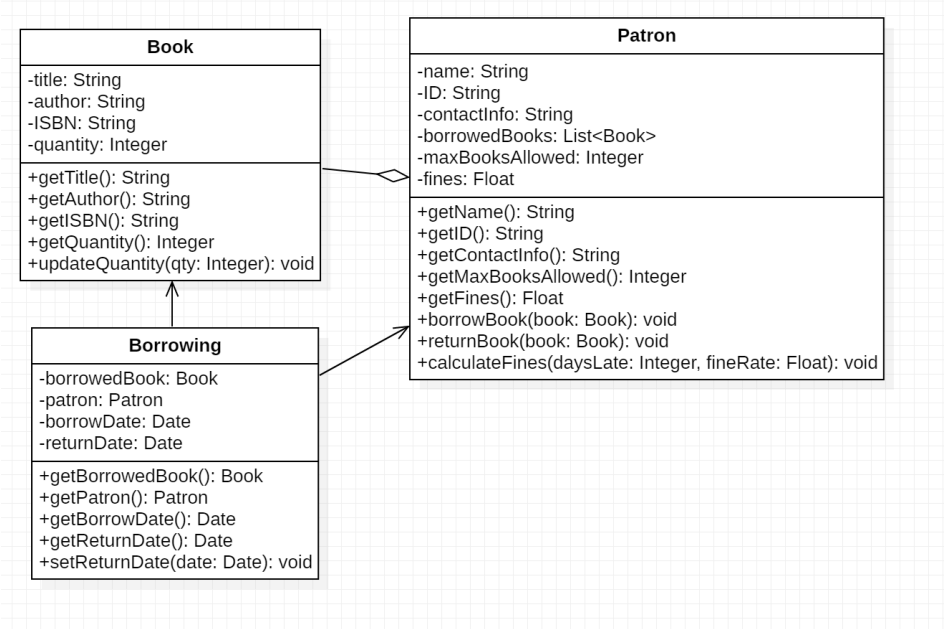
Problem 1: Library Management System

1. Class Diagram:



2. Database Diagram:

```sql
CREATE DATABASE library_management_sys;
GO

USE library_management_sys;
GO

CREATE TABLE [dbo].[Book](
        [bookID] [int] primary key NOT NULL,
        [title] [varchar](50) NULL,
        [author] [varchar](50) NULL);
GO

CREATE TABLE [dbo].[Patron](
        [patronID] [int] primary key NOT NULL,
        [name] [varchar](50) NULL,
        [contact] [varchar](50) NULL);
GO

CREATE TABLE [dbo].[Borrowing](
        [borrowID] [int] primary key NOT NULL,
        [book_id] [int] foreign key references Book(bookID),
        [patron_id] [int] foreign key references Patron(patronID) )
GO

INSERT [dbo].[Book] ([bookID], [title], [author]) VALUES (1, N'Irrfan Khan: A Life in Movies', N'Shubhra Gupta')
INSERT [dbo].[Book] ([bookID], [title], [author]) VALUES (2, N'The World: A Family History', N'Simon Sebag')
INSERT [dbo].[Book] ([bookID], [title], [author]) VALUES (3, N'Breaking Barriers', N'Kaki Madhava Rao')
INSERT [dbo].[Book] ([bookID], [title], [author]) VALUES (4, N'Ambedkar: A Life', N'Shashi Tharoor')
INSERT [dbo].[Book] ([bookID], [title], [author]) VALUES (5, N'Human Anatomy', N'Ashvini Kumar Dwivedi')
GO

INSERT [dbo].[Patron] ([patronID], [name], [contact]) VALUES (101, N'Petricescu Paul', N'0745234134')
INSERT [dbo].[Patron] ([patronID], [name], [contact]) VALUES (102, N'Antonesi Melisa', N'0749876234')
INSERT [dbo].[Patron] ([patronID], [name], [contact]) VALUES (103, N'Furcovici Dalia', N'0756123678')
INSERT [dbo].[Patron] ([patronID], [name], [contact]) VALUES (104, N'Cautis Luca', N'0756789967')
INSERT [dbo].[Patron] ([patronID], [name], [contact]) VALUES (105, N'Irimia Laurențiu', N'0756743212')

INSERT [dbo].[Borrowing] ([borrowID], [book_id], [patron_id]) VALUES (501, 1, 101)
INSERT [dbo].[Borrowing] ([borrowID], [book_id], [patron_id]) VALUES (502, 2, 102)
INSERT [dbo].[Borrowing] ([borrowID], [book_id], [patron_id]) VALUES (503, 3, 103)
INSERT [dbo].[Borrowing] ([borrowID], [book_id], [patron_id]) VALUES (504, 4, 104)
INSERT [dbo].[Borrowing] ([borrowID], [book_id], [patron_id]) VALUES (505, 5, 105)
```

```
SELECT* FROM Book
```

100 %

Results | Messages

	bookID	title	author
1	1	Irrfan Khan: A Life in Movies	Shubhra Gupta
2	2	The World: A Family History	Simon Sebag
3	3	Breaking Barriers	Kaki Madhava Rao
4	4	Ambedkar: A Life	Shashi Tharoor
5	5	Human Anatomy	Ashvini Kumar Dwivedi

```
SELECT* FROM Patron
```

100 %

Results | Messages

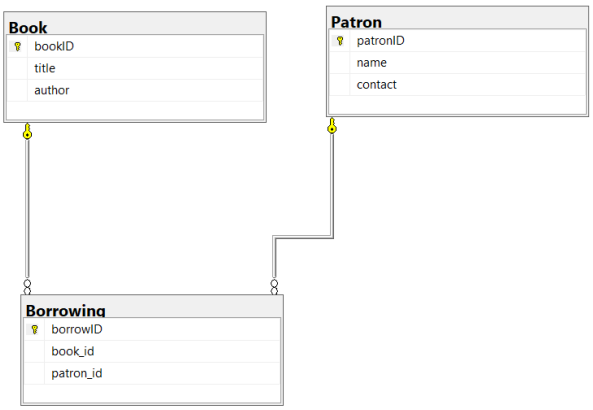
	patronID	name	contact
1	101	Petricescu Paul	0745234134
2	102	Antonesi Melisa	0749876234
3	103	Furcovici Dalia	0756123678
4	104	Cautis Luca	0756789967
5	105	Irimia Lauren?iu	0756743212

```
SELECT* FROM Book

SELECT* FROM Patron

SELECT* FROM Borrowing
```

100 %

Results | Messages

	borrowID	book_id	patron_id
1	501	1	101
2	502	2	102
3	503	3	103
4	504	4	104
5	505	5	105



The tables 'Book', 'Patron', and 'Borrowing' are linked using foreign keys ('book_id' and 'patron_id') in the 'Borrrowing' table, establishing relationships between books, patrons, and borrowings. The primary keys are: 'bookID' for the 'Book' table, 'patronID' for the 'Patron' table and 'borrowID' for the 'Borrowing' table.

This schema provides a foundation for the Library Management System's database, linking books to patrons through borrowings and allowing for efficient management of book transactions and patron details.

Problem 2: Online Quiz System

1. Logical Design:

Question Organization:

1. **Question Structure**: Each question should contain:
    - Question number or unique ID
    - Question text
    - Multiple-choice options (4)
    - Correct answer
2. **Question Pool:** Store the 50 questions in an array or object to easily access and track them during the quiz.

```
index.html   styles.css   question.js   app.js
C: > Users > nicol > Downloads > internshipSiemens > js > question.js > ...
// Array of Objects
const quiz = [
  {
    q:'What is the capital of France?',
    options:['London','Paris','Berlin','Madrid'],
    answer:1
  },
  {
    q:'What is 45 times 1 million?',
    options:['45,000,000','45','450','All of the above'],
    answer:0
  },
  {
```

User Progress:

1. **User Data:** Store user-related data such as:
    - User's selected answers
    - Current question number
    - User's score
2. **Handling Progress**:
    - When a user selects an answer, update the user's progress by storing their choice and moving to the next question.
    - Calculate and update the score based on correct answers.

2. Algorithm Implementation (Partial Code):

```
// set question number and question and options
function getNewQuestion(){
    // set question number
    questionNumber.innerHTML = "Question " + (questionCounter + 1) + " of " + quiz.length;

    // set question text
    // get random question
    const questionIndex = availableQuestions[Math.floor(Math.random() * availableQuestions.length)];
    currentQuestion = questionIndex;
    questionText.innerHTML = currentQuestion.q;
    // get the position of 'questionIndex' from the availableQuestion Array;
    const index1 = availableQuestions.indexOf(questionIndex);
    // remove the 'questionIndex' from the availableQuestion Array, so that the question does not repeat
    availableQuestions.splice(index1,1);

    // set options
    // get the length of options
    const optionLen = currentQuestion.options.length
    // push options into availableOptions Array
    for(let i=0; i<optionLen; i++){
        availableOptions.push(i)
    }
```

```
let animationDelay = 0.15;
// create options in html
for(let i=0; i<optionLen; i++){
    // random option
    const optonIndex = availableOptions[Math.floor(Math.random() * availableOptions.length)];
    // get the position of 'optonIndex' from the availableOptions Array
    const index2 = availableOptions. indexOf(optonIndex);
    // remove the 'optonIndex' from the availableOptions Array, so that the option does not repeat
    availableOptions.splice(index2,1);
    const option = document.createElement("div");
    option.innerHTML = currentQuestion.options[optonIndex];
    option.id = optonIndex;
    option.style.animationDelay = animationDelay + 's';
    animationDelay = animationDelay + 0.15;
    option.className = "option";
    optionContainer.appendChild(option)
    option.setAttribute("onclick","getResult(this)");
  }
  questionCounter++
```

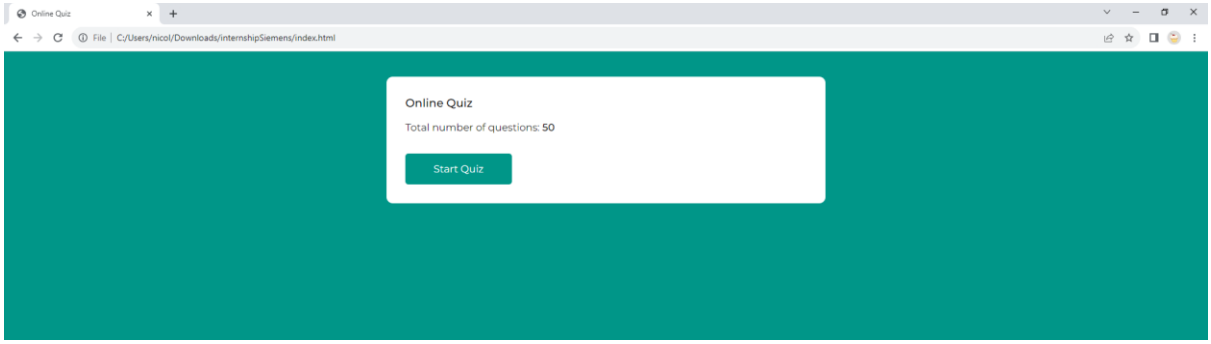
```
  <p>Total number of questions: <span class="total-question"></span></p>
  <button type="button" class="btn" onclick="startQuiz()">Start Quiz</button>
</div>
<div class="quiz-box custom-box hide">
  <div class="question-number">
  </div>
  <div class="question-text">
  </div>
  <div class="option-container">
  </div>
  <div class="next-question-btn">
    <button type="button" class="btn" onclick="next()">Next</button>
  </div>
  <div class="answers-indicator">
  </div>
```

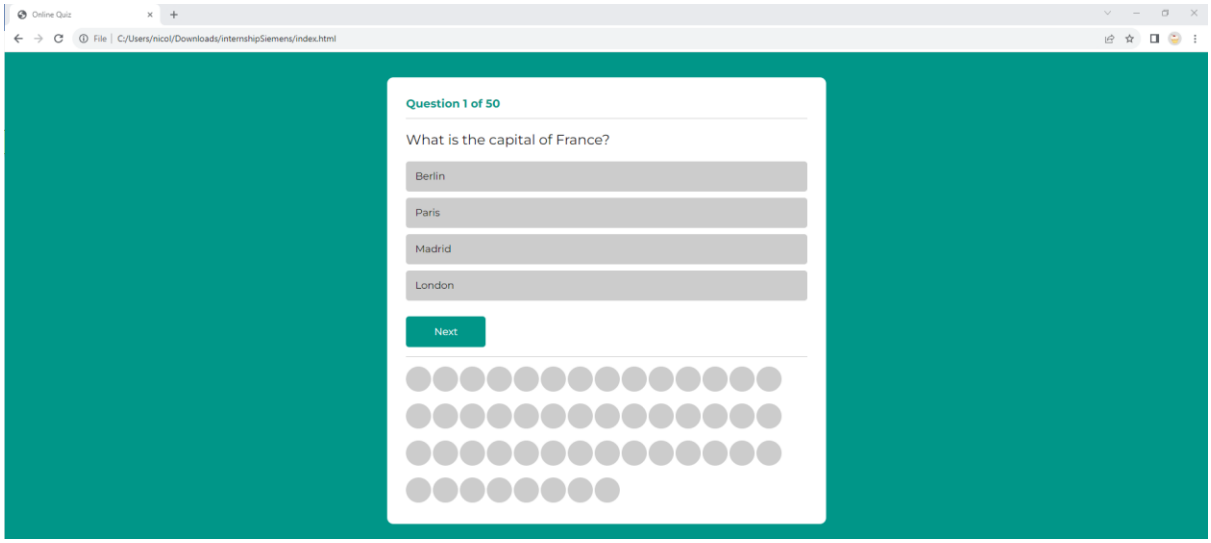
```
// get the result of current attempt question
function getResult(element){
  const id = parseInt(element.id);
  // get the answer by comparing the id of clicked option
  if(id === currentQuestion.answer){
    // set the green color to the correct option
    element.classList.add("correct");
    // add the indicator to correct mark
    updateAnswerIndicator("correct");
    correctAnswers++;
  }
  else{
    // set the red color to the incorrect option
    element.classList.add("wrong");
    // add the indicator to wrong mark
    updateAnswerIndicator("wrong");
```

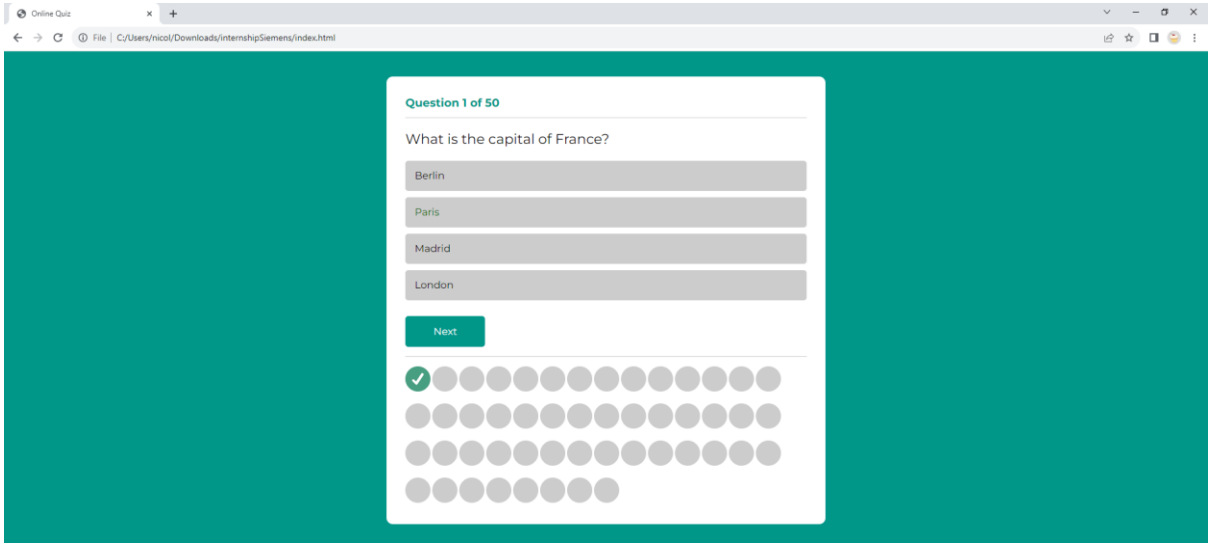
```
    else{
      // set the red color to the incorrect option
      element.classList.add("wrong");
      // add the indicator to wrong mark
      updateAnswerIndicator("wrong");

      // if the answer is incorrect the show the correct option by adding green color the correct option
      const optionLen = optionContainer.children.length;
      for(let i=0; i<optionLen; i++){
        if(parseInt(optionContainer.children[i].id) === currentQuestion.answer){
          optionContainer.children[i].classList.add("correct");
        }
      }
    }
    attempt++;
    unclickableOptions();
  }
```

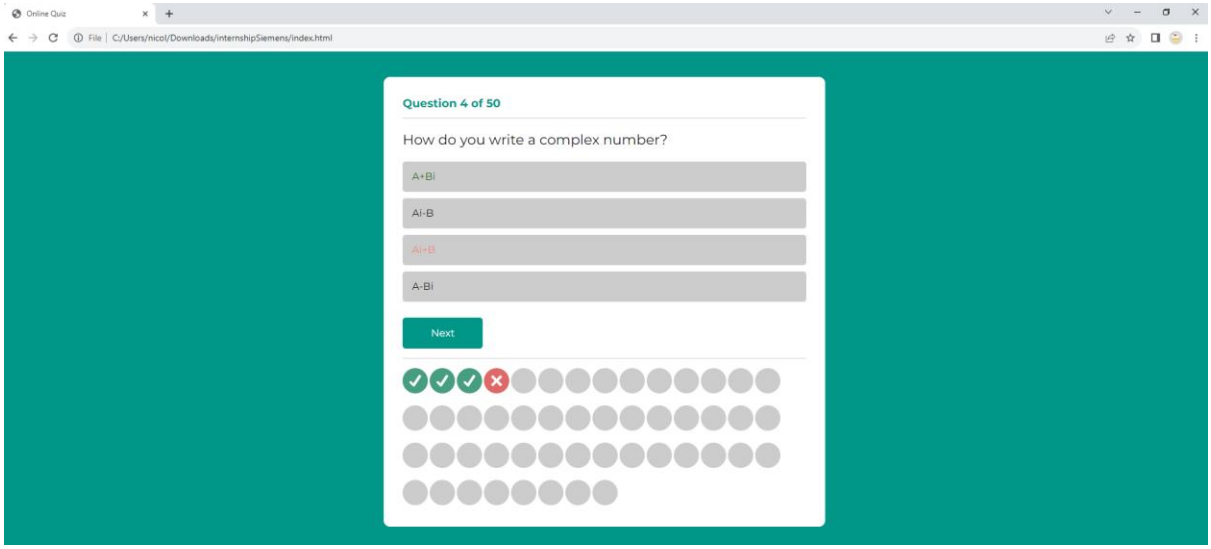
```
}
// make all the options unclickable once the user select a option (RESTRICT THE USER TO CHANGE THE OPTION AGAIN)
function unclickableOptions(){
  const optionLen = optionContainer.children.length;
  for(let i=0; i<optionLen; i++){
    optionContainer.children[i].classList.add("already-answered");
  }
}
```

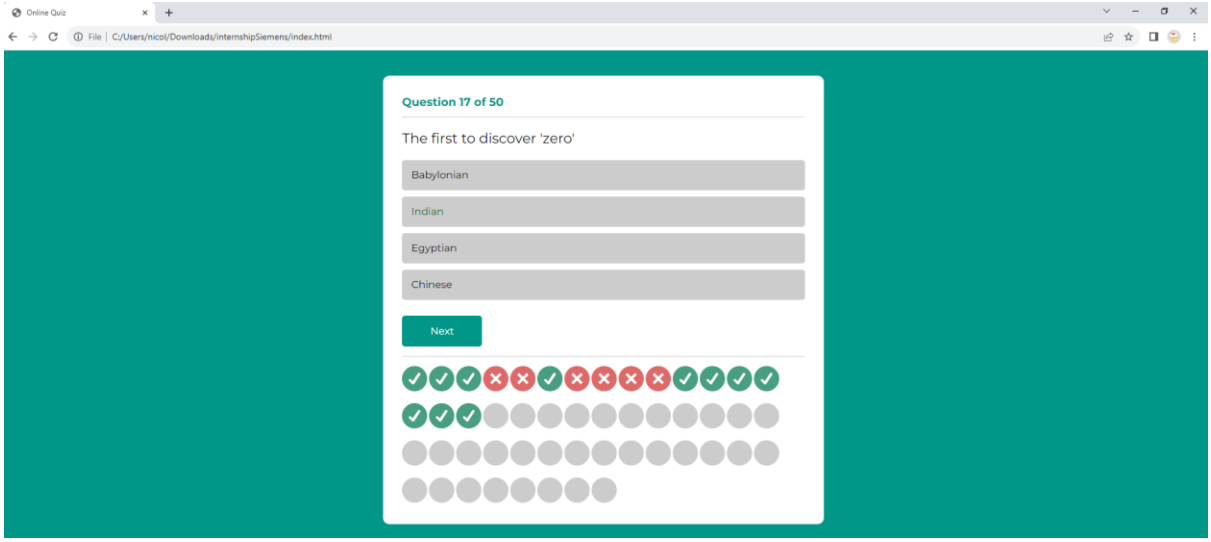
```
}
// get  the quiz Result
function quizResult(){
  resultBox.querySelector(".total-question").innerHTML = quiz.length;
  resultBox.querySelector(".total-attempt").innerHTML = attempt;
  resultBox.querySelector(".total-correct").innerHTML = correctAnswers;
  resultBox.querySelector(".total-wrong").innerHTML = attempt - correctAnswers;
  const percentage = (correctAnswers/quiz.length)*100;
  resultBox.querySelector(".percentage").innerHTML = percentage.toFixed(2) + "%";
  resultBox.querySelector(".total-score").innerHTML = correctAnswers +" / " + quiz.length;
}
```

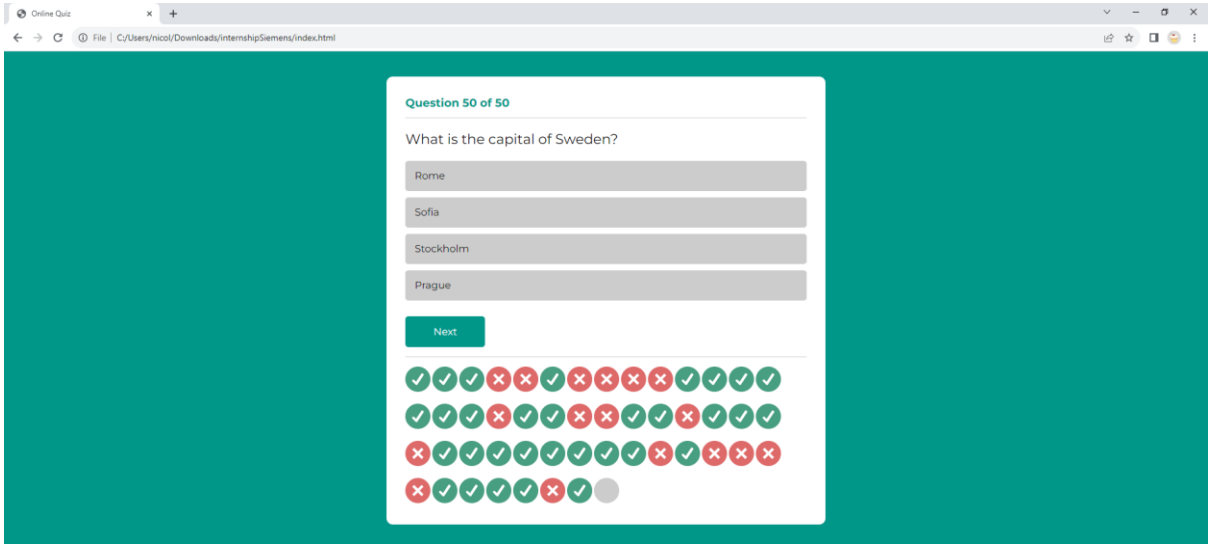


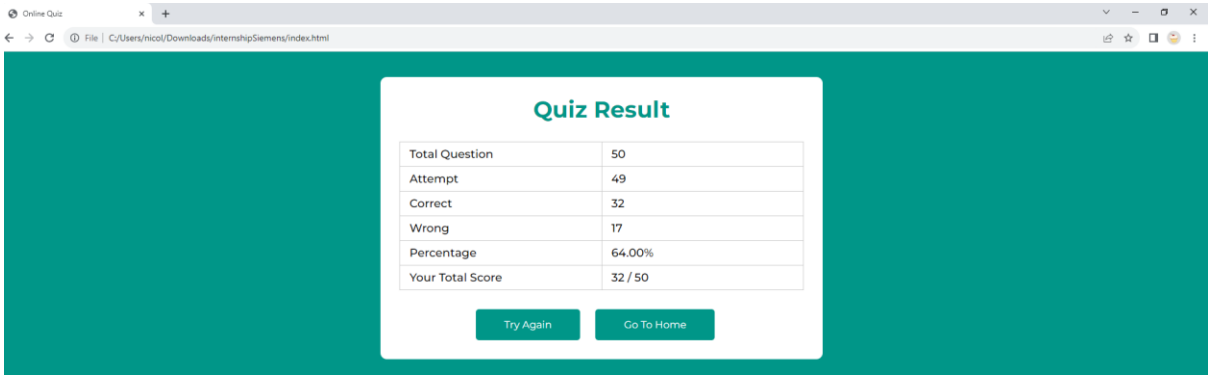

Online Quiz
File | C:/Users/nicol/Downloads/internshipSiemens/index.html
Question 1 of 50
What is the capital of France?
Berlin
Paris
Madrid
London
Next


Online Quiz
File | C:/Users/nicol/Downloads/internshipSiemens/index.html
Question 1 of 50
What is the capital of France?
Berlin
Paris
Madrid
London
Next

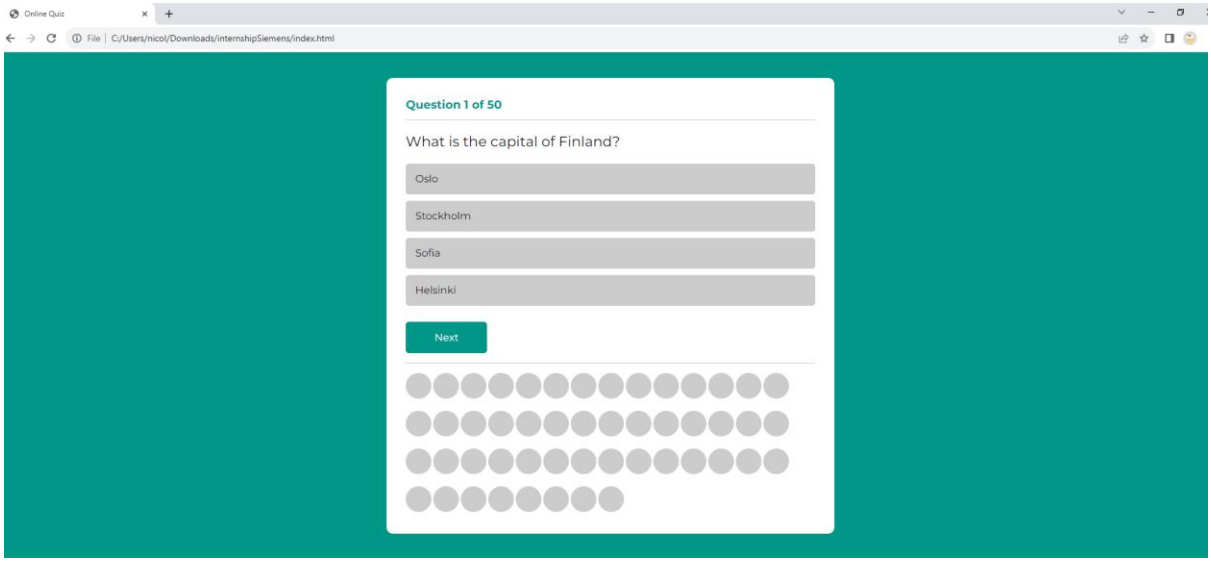

Online Quiz
File | C:/Users/nicol/Downloads/internshipSiemens/index.html
Question 4 of 50
How do you write a complex number?
A+Bi
Ai-B
Ai+B
A-Bi
Next


Online Quiz
File | C:/Users/nicol/Downloads/internshipSiemens/index.html
Question 17 of 50
The first to discover 'zero'
Babylonian
Indian
Egyptian
Chinese
Next

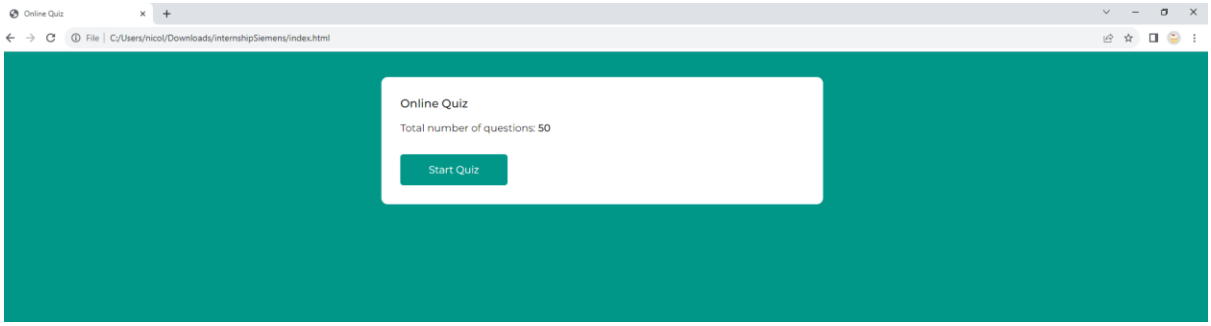

Online Quiz
File | C:/Users/nicol/Downloads/internshipSiemens/index.html
Question 50 of 50
What is the capital of Sweden?
Rome
Sofia
Stockholm
Prague
Next


Online Quiz
File | C:/Users/nicol/Downloads/internshipSiemens/index.html
Quiz Result
Total Question 50
Attempt 49
Correct 32
Wrong 17
Percentage 64.00%
Your Total Score 32 / 50
Try Again
Go To Home

Click ‘Try Again’:



Click ‘Go To Home’:



3. Class and Database Representation (Explanation Only):

Class Diagram:

1. **Question Class:**
    - Attributes: question number or ID, question text, options, answer
2. **User Class:**
    - Attributes: selected answers (attempt), score (correct answers/total questions), correct answers, wrong answers, progress (percentage), maybe a user ID

OR

1. **Question Class:**
    - Attributes:
        - questionID(or number): int
        - questionText: string
        - option: string[4]
        - correctAnswerIndex: int
2. **Quiz Class:**

- Attributes:
    - questionPool: Question []
    - currentQuestionIndex: int
    - userScore: int
    - userAnswers: {questionID: int, userChoice: int} []

3. **QuizInterface (UI) class:**
    - Responsibilities:
        - Displaying questions, options, and user interface elements
        - Handling user input and interactions
4. **QuizController**
    - Responsibilities:
        - Managing the flow of the quiz
        - Controlling the logic for question retrieval, score calculation, and user progress

Relationships:

- Quiz-Question (Composition): The quiz class contains a collection of Question instances representing the pool of questions available for the quiz.
- Quiz-QuizInterface(Association): The quiz class interacts with the QuizInterface class to display questions and handle user interactions.
- Quiz-QuizController(Association): The quiz class utilizes the QuizController to control the flow and logic of the quiz process.

Database Schema:

1. **Tables:**
    - Question tables with columns for ID, Question Text, Options, Correct Answer
    - Users table with columns for User ID, Selected Answers, Score, Progress
2. **Relationships:**
    - One-to-Many relationship between Users and Selected Answers (each user can have multiple selected answers)
    - No direct relationship between Questions and Users (questions are common for all users)

Flow of Data:

1. **During quiz:**
    - User interacts with the web interface.
    - User's selected answers and progress are stored and updated.
2. **After quiz:**
    - User's final score and answers can be stored in the database for future reference or analysis.