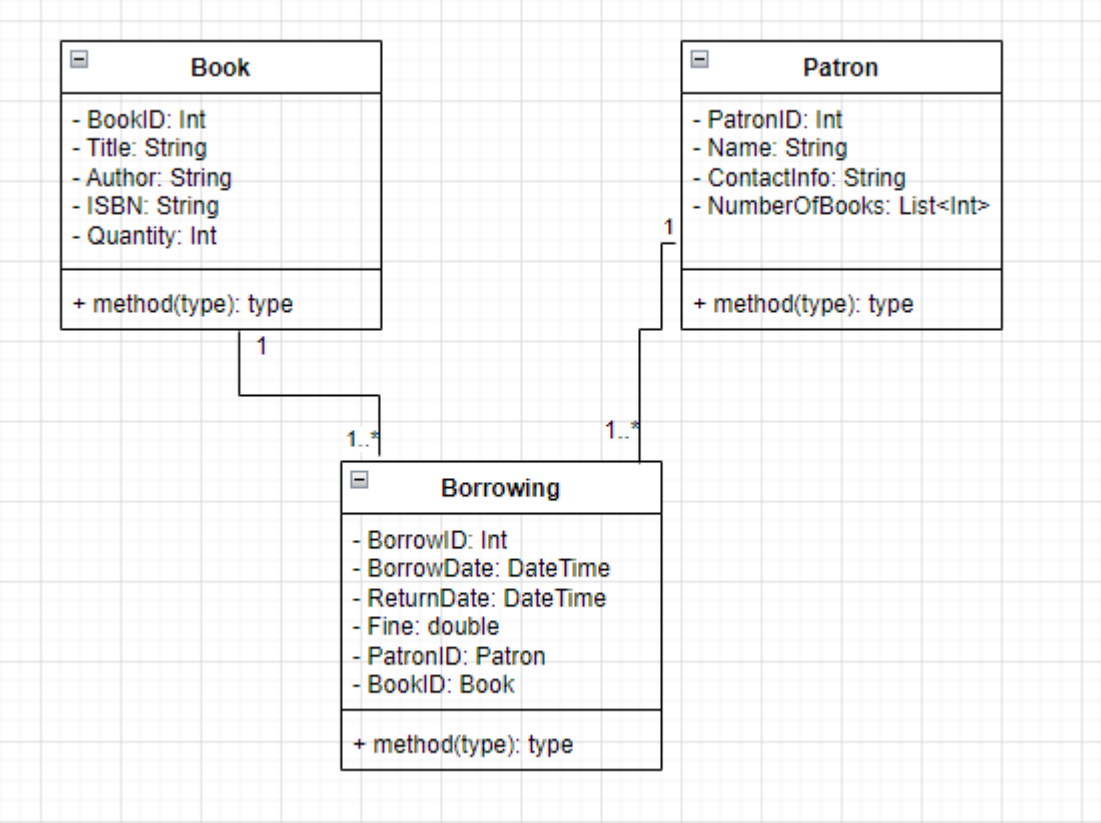
Test internship SIEMENS

Nume participant: Ghicajanu Oana-Cristina

Problem 1: Library Management System:

1) Class Diagram:



Atributele:

In tabela **Book,** am adaugat BookID-ul, acesta fiind unic, not null si reprezentand Primary Key, Title, Author, ISBN si Quantity.

In tabela **Patron,** am adaugat PatronID-ul, acesta fiind unic, not null si reprezentand Primary Key, Name, ContactInfo, si am mai adaugat List<Int> NumberOfBooks pentru a putea tine evidenta cate carti a imprumutat clientul.

In tabela **Borrowing,** am adaugat BorrowID-ul, acesta fiind unic, not null si reprezentand Primary Key, PatronID(reprezentand Foreign key), BookID(reprezentand Foreign key), BorrowDate si ReturnDate pentru a tine evidenta de perioada in care a fost imprumutata o carte, si Fine pentru amenda

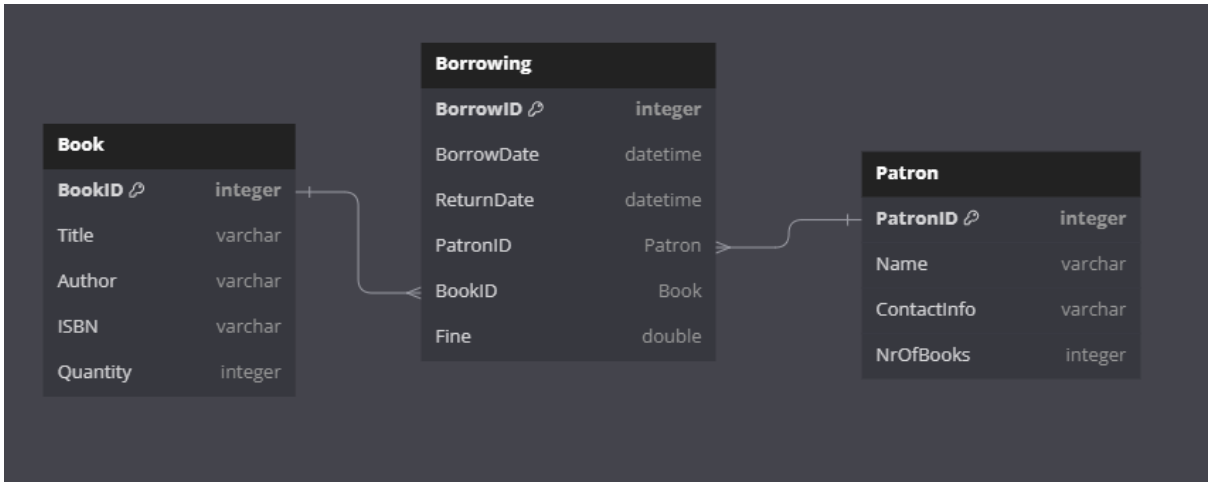
in caz ca Patron a depasit limita de perioada. M-am gandit ca BorrowDate este introdus in momentul in care clientul vine si imprumuta o carte, apoi ReturnDate este introdus atunci cand clientul returneaza cartea, facandu-se update la tabela pentru campurile ReturnDate si Fine.

Relatiile:

Intre tabelele **Book** si **Patron**, exista o relatie de many-to-many. Astfel, tabela **Borrowing** reprezinta tabela care face join intre celelalte 2 tabele.

Avand o tabela de join, vom avea o relatie de one-to-many intre tabelele Book si Borrowing, si inca o relatie de one-to-many intre tabelele Patron si Borrowing.

2) Database Diagram:



Problem 2: Online Quiz System:

1)Logical Design:

Plan:

-Vom avea un array care va stoca intrebarile, variantele multiple și raspunsul corect.
-Testul va incepe direct. Id-ul intrebarilor va fi de la 0 la 49; Aici, prima intrebare va aparea pe interfata in mod aleatoriu (ex: prima intrebare ar putea avea id=10) și optiunile multiple.
-Utilizatorul trebuie sa aleaga un raspuns, altfel nu va putea trece la urmatoarea intrebare
-Dupa ce va raspunde la intrebare, va aparea urmatoarea intrebare random
-Fiecare intrebare va fi eliminata din array dupa raspunsul utilizatorului
-Vom verifica daca lungimea grupului de intrebari este egala cu 0, atunci inseamna ca utilizatorul a raspuns la toate intrebarile.
-Dupa terminarea testului, scorul utilizatorului va aparea cu numarul total de raspunsuri corecte.

Pseudocode:

```
function randomizeQuestions() {…}

function nextQuestion() {…}

Press button Start
Quiz starts
if(currentIdQuestion >= 0 && currentIdQuestion < 50)
{ Select answer
        if (answer == correct) then userScore++;
  currentIdQuestion++;


        if(currentQuestionId < length(questions)) then nextQuestion();
       else
        Quiz finished; Show user score;
}
```

2)Code Implementation:
Am atasat fisierele de index.html, script.js, questions.js si style.css.

3) Class and Database Representation:

Class structure:

Vom avea **4** clase: Quiz, Question, Answers si User.
Clasa **Quiz** va avea atributele private:
-quizID: int(primary key si unic)
-User user (atributul user de tip clasa User)
-List<Question> questionObject
Metodele vor fi:
-startQuiz()

Clasa **Question** va avea atributele private:
-questionId (primary key si unic)
-questionText: string
-List<Answers>choices (lista cu raspunsuri pt intrebarea respectiva)
Metodele vor fi:
-getRandomQuestion()
-showNextQuestion()

Clasa **Answers** va avea atributele private:
-answerID (primary key si unic)
-answerText: string (textul raspunsului corect)
-isCorrect: bool (raspunsul corect la intrebarea respectiva)
-questionID: Question (carei intrebari ii vor corespunde raspunsurile)
Metodele vor fi:
-getAnswers()
-setCorrectAnswer(int answerID)

Clasa **User** va avea atributele private:
-userId (primary key si unic)
-username
-userScore (scorul final)

Metodele vor fi:
-submitAnswer()
-updateUserScore(int points)
-showUserScore() (arata scorul final)

Database structure:
Avem **4** tabele:
Tabela **Quiz**:
-quizID (primary-key)
-userID (foreign-key)
In tabela Quiz adaugam userID ca si foreign-key deoarece avem o relatie de many-to-one astfel incat un user poate da mai multe quiz-uri.

Tabela **User**:
-userID (primary-key)

Tabela **Question**:
-questionID (primary-key)
-quizID (foreign-key)
In tabela Question adaugam quizID ca si foreign-key deoarece avem o relatie de many-to-one astfel incat mai multe intrebari se afla intr-un singur quiz.

Tabela **Answer**:
-choiceID (primary-key)
-questionID (foreign-key)
In tabela Answer adaugam questionID ca si foreign-key deoarece avem o relatie de many-to-one astfel incat sunt mai multe optiuni de raspuns la o singura intrebare.

Flow of Data:
User-ul care vrea sa dea quiz-ul, are un id unic, un username, iar scorul este initial 0. De asemenea, va fi creat un quiz cu un id unic si cu referinta catre id-ul user-ului care va da testul. Se va actualiza tabela User, fiind adaugat un camp nou cu id-ul user-ului si numele user-ului, scorul fiind inca 0.
Dupa ce va incepe testul, va aparea cate o intrebare pe rand, in mod random. Sa presupunem ca prima intrebare aparuta are id-ul 12. Tabela Question va fi

actualizata, fiind adaugat un rand nou cu id-ul, textul intrebarii si optiunile de raspuns.
Se va face update la tabela Answer cu cele 4 optiuni de raspuns, fiecare camp avand id unic si coloana „isCorrect” actualizata cu true sau false.
Dupa ce user-ul va finaliza quiz-ul, se face update la tabela User la campul „userScore” unde se va completa cu scorul final.

Multumesc pentru timpul si atentia acordata rezolvarilor prezentate!