**COURSEWORK SUBMISSION FORM**

| **STUDENT USE** | | **STAFF USE** | |
| --- | --- | --- | --- |
| Module Name | Database systems development | First Marker’s  (acts as signature) |  |
| Module Code |  | Second Marker’s  (acts as signature) |  |
| Lecturer Name | Dmitriy Pochitayev | Agreed Mark |  |
| UoW Student IDs |  | **For Registrar’s office use only (hard copy submission)** | |
| WIUT Student IDs | 12387, 12460, 11064 |
| Deadline date |  |
| Assignment Type | 🗌Group🗌Individual |

**SUBMISSION INSTRUCTIONS**

**COURSEWORKS *must* be submitted in *both* HARD COPY (to the Registrar’s Office) *and* ELECTRONIC unless instructed otherwise.**

For hardcopy submission instructions refer to: <http://intranet.wiut.uz/Shared%20Documents/Forms/AllItems.aspx> - Coursework hard copy submission instructions.doc

For online submission instructions refer to: <http://intranet.wiut.uz/Shared%20Documents/Forms/AllItems.aspx> - Coursework online submission instructions.doc

| **MARKERS FEEDBACK (Continued on the next page)** |
| --- |
|  |

## Table Of Contents

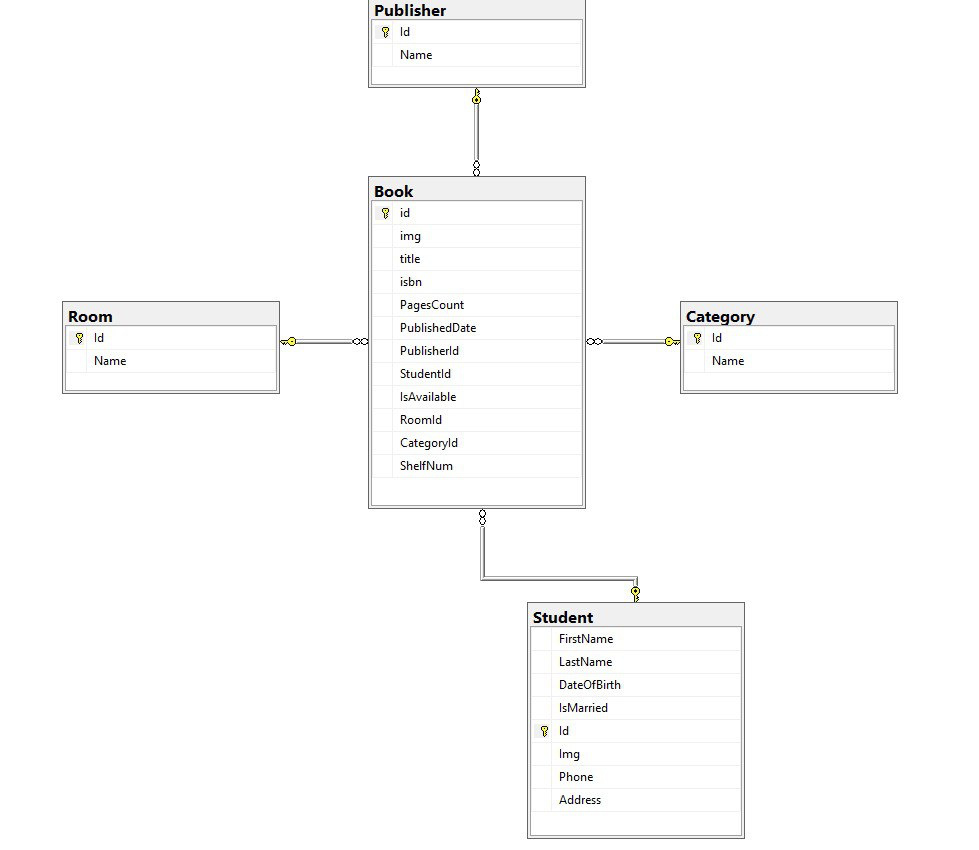
[Relational Database Diagram 2](#_p69c8d1xjt5r)

[Evaluation 2](#_qhlk0un2gu0x)

[Backbox Testing 3](#_hxthdgcnphz6)

## 

## Relational Database Diagram



## Evaluation

A library management system was created with the functionality of borrowing books from the library for students. As a Minimum Viable Product (MVP) version of the application, as a beginning, systems allow users to manage to Create, Read, Update, and Delete operations above the Books and Students tables. As it was shown in the above diagram, Student consists of Id, FirstName, LastName, Img, Phone, Address, DateOfBirth, and IsMarried properties and linked to the Book table. At the same time, the Book table has Id, Img, Title, ISBN, PagesCount, PublishedDate, PublisherId, StudentId, IsAvailable status, RoomId, CategoryId and ShelfNum columns with relation to Publisher, Room, Category, and Student tables with respective Foreign Key attributes which could be null.

The system was developed using the MVC project of .NET Framework 3.1 with a Dapper package to manage database control over the application. Data Access Layer Interfaces and Repositories were created to give the full abstract functionality of database usage. To manage students plain SQL commands were implemented and to get the full functionality of SQL Server stored procedures were applied to control Books.

The application has several extra functionalities, for instance, unlinking students to books when they are deleted; limits for bonding at most five books to one student, and denying delete operations after 17:00. Furthermore, select queries with multiple where conditions were optimized using SQL indexes, related comments were written inside procedures.sql file.

## Backbox Testing

| Test Description | Expected results | Actual results | Comments |
| --- | --- | --- | --- |
| Student’s CRUD functionality | When a user goes to the Student page the Student List should be shown. A new student can be added to the list and it is saved in the database. Basic CRUD operations can be done. 1) Create button should direct to another page where a user can insert the values of a new Student and it should save the value into a database. All inputs should be validated. 2) Edit button should direct the user to the same page as Create button but with the date of the specific chosen Student and it should save the changes. 3) Delete button should remove the data from the database using the Student Id. | The student page shows the List of students. A new student can be added to the list and it actually saves it in the database. Basic CRUD operations work properly. 1) Create button shows relevant inputs and they are validated properly. 2) Edit button shows exact Student data and can be modified and saved. 3) Delete button works, it takes the Student's information using its ID and deletes it from the database. | Expected actions are done. Moreover, student deletion unlinks students from the book |
| A user should be able to do CRUD operations with book page | There are 3 buttons for creating a book Create, Edit, and Delete. 1) Create button should direct to another page where a user can insert values of a new book and it should save the value into a database. All inputs should be validated. 2) Edit button should direct the user to the same page as Create button but with the date of the specific book clicked and it should save the changes. 3) Delete button should remove the data from the database using the Book Id. | There are 3 buttons with proper names and colors. 1) Create button shows relevant inputs and they are validated properly. 2) Edit button shows exact book data and can be modified and saved. 3) Delete button works, it takes the book information using its ID and deletes it from the database. | The expected results match the Actual results.  Furthermore, operations which are done after 17:00 are denied |
| A user can filter the books with relevant inputs | Filtering books in the books page should be by Title, PublisherId, CategoryId, and RoomId and the filter select the relevant book out of all books | There is a Filter on the book page. The Title, PublisherId, CategoryId, and RoomId inputs work, and they extract the relevant book from the list | The expected results match the Actual results. |
| A user can import and export the filtered books into a CSV file | The application should have in import and export into a CSV file, and it should import only the information of filtered books | There is a link that can import and export the data of filtered books into a CSV file | The expected results match the Actual results. |
| A user can import and export the filtered books into an XML file | The application should have in import and export into an XML file, and it should import only the information of filtered books | There is a link that can import and export the data of filtered books into an XML file | The expected results match the Actual results. |
| A user can import and export the filtered books into an JSON file | The application should have in import and export into an JSON file, and it should import only the information of filtered books | There is a link that can import and export the data of filtered books into an JSON file | The expected results match the Actual results. |