Databases

Project

Group 3 - Altay Qarayev, Martti Tarro

The aim of the project is to construct and use a database to automate the services of the library of a university.

We approached the project with the following tools:

- The initial conceptual model in ER Assistant
- The improved conceptual model in PowerDesigner
- The physical model in PowerDesigner
- DDL from the physical model to MySQL
- Triggers, functions, views, user profiles in MySQL
- Sample data composed in Excel (CSV files)
- Sample data population in Talend
- Graphic interface in Windows Forms in C#

We initially scrapped our approach of doing the conceptual model in ER Assistant, since PowerDesigner proved much more appropriate for the task at hand.

The conceptual model was then assembled in PowerDesigner (schema in Appendix 1, and in the PowerDesigner file Project_conceptual.cdm). The empty tables seen in the conceptual model will have foreign keys in the physical model. This was done with the goal of normalization.

The physical model was then composed (schema in Appendix 2, and in the PowerDesigner file Project_physical_model.pdm).

From this, we extracted a table creation script (create_tables.sql), which was slightly cleaned up in MySQL. For example, PowerDesigner did not include the checks for emails, years, etc. in the script, so these were added manually.

The test data (in folder test_data as CSV's) was generated manually in Excel (through randomization and some sample input values online). This was then imported into MySQL through Talend subjobs, as depicted in Appendix 3. For some reason, the subjobs book_publisher and book_copy did not work as intended when running the whole job but did work when other jobs were deactivated.

The limits to how many books a university or non-university student can borrow was achieved through implementing a trigger. Triggers were also used to populate some columns in the table borrow when a librarian would enter the book copy's copy_id and the card's card_id. These triggers are in file triggers.sql.

Viewing who has checked out a particular book was achieved through a function which takes the book ISBN as input and gives the student_id's who have checked out that book.

Viewing which books have been checked out by a particular student was achieved through a function which takes the student_id as input and gives the list of ISBNs borrowed by them.

Viewing the copies of books whose due dates have passed was achieved through a view which displays the ISBN's and copy_id's of the books whose due dates are before the day when it is viewed.

Viewing the number of remaining copies of a given book was achieved through a function which takes an ISBN as input and gives the number of books as output.

Three user profiles were created in script user_profiles.sql, where different profiles were given different privileges. The student profile could have been further developed, since right now, the privileges do not consider that the student could only view their own borrowed books and cards, but they could view all of them.

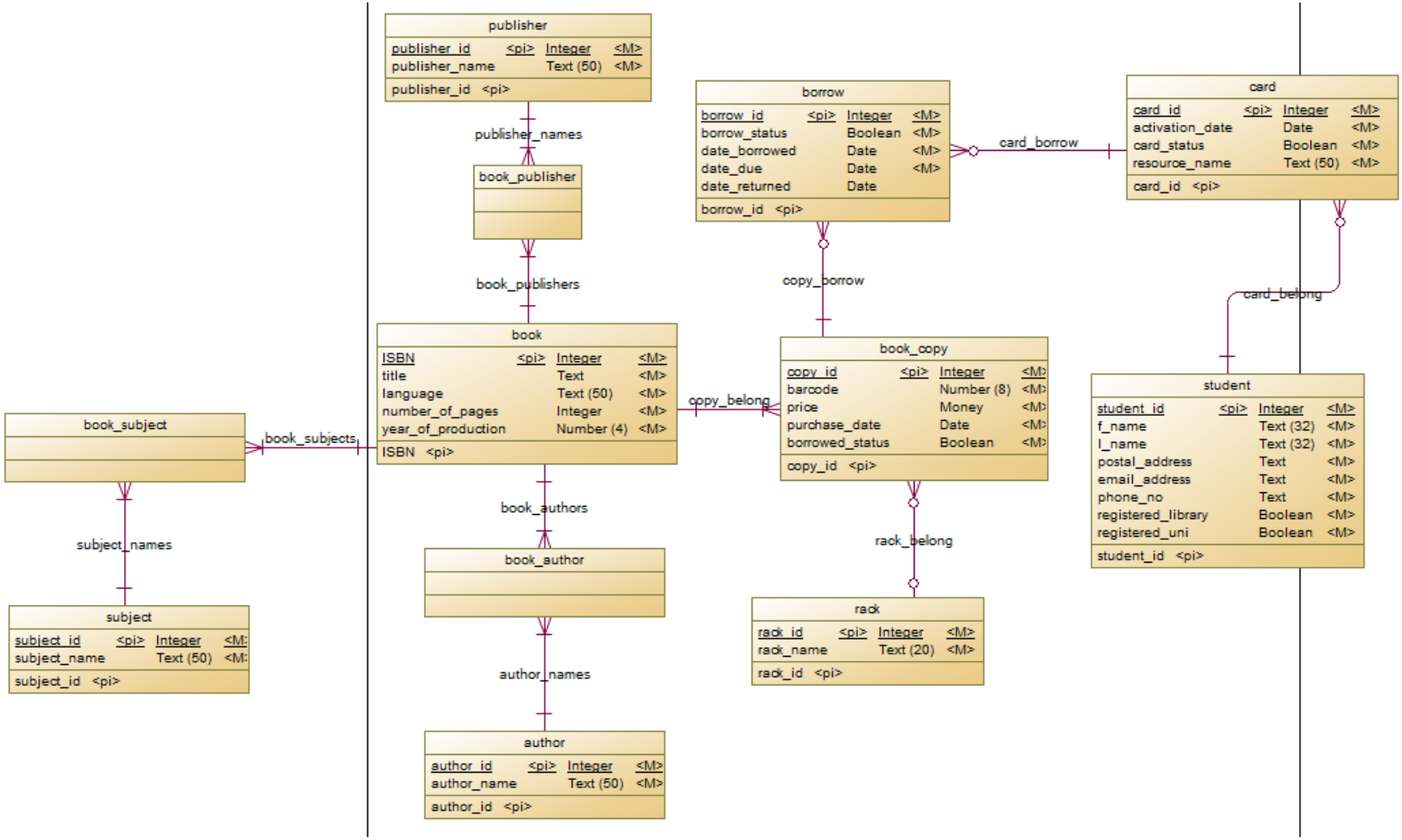
The graphic interface was done in Windows Forms in C#. Our interface has 3 main pages (administrator page, librarian page, and student page). Each page has its own design and permission depending on their access level. For example, administrators can add students and books, as well as update students' status (block or unblock their access). Library agents can see all books that are due, and all books borrowed by a specific student. Students can search for available books, see all books borrowed by them and see all their overdued books. The screenshots of the interface are in Appendix 4.

We approached the project so that we both talked through the conceptual model, then Martti developed the MySQL scripts (discussing the parts through with Altay), and Altay developed the graphic interface.

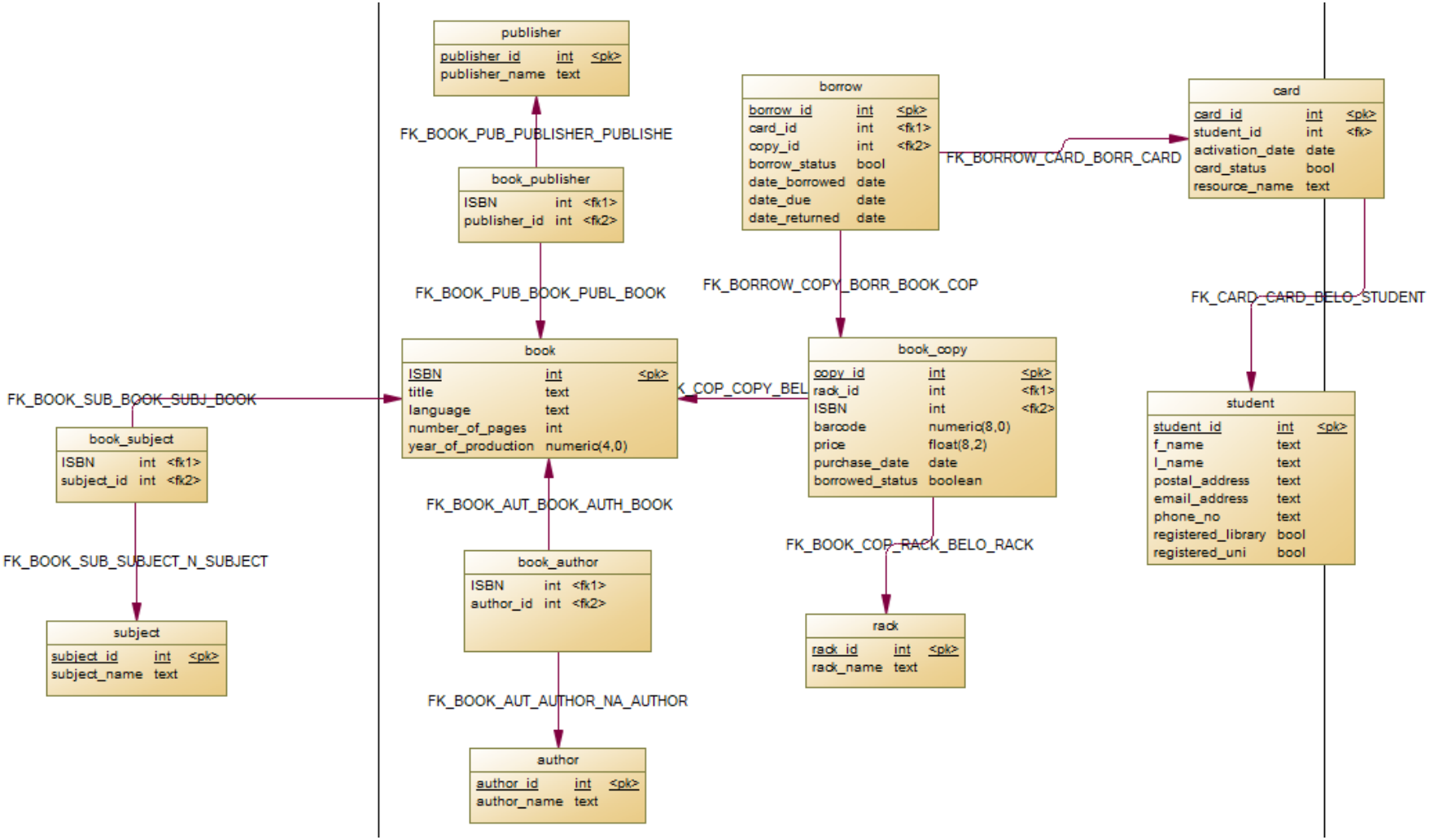
Structural constraints:

- Student one-to-many Card
    - Each student may have one or more cards (0, 1, or more)
    - Each card must have one and only one student (only 1)
- Card one-to-many Borrow
    - Each card may borrow once or more (0, 1, or more)
    - Each time one book is borrowed, it can have one and only one card (only 1)
- Borrow many-to-one Book_copy
    - Each book copy may be borrowed once or more times (0, 1, or more)
    - Each time one book is borrowed, it is reflected once and only once in Borrow (only 1)
- Book_copy many-to-one Rack
    - Each rack may have one or more book copy (0, 1, or more)
    - Each book copy may have at most one rack (0 or 1, since some books might not be on a rack, but one book cannot be on more than one rack)
- Book_copy many-to-one Book
    - Each book must have one or more book copy (1 or more)
    - Each book copy must have one and only one book (only 1)
- Book one-to-many Book_author
    - Each book must have one or more authors (1 or more)
    - Each author of a certain book must have one and only one book (only 1)
- Book_author many-to-one Author
    - Each author must have one or more author of a certain book (1 or more)
    - Each author of a certain book must have one and only one author (only 1)
- Book one-to-many Book_subject
    - Each book must have one or more subjects (1 or more)
    - Each subject of a certain book must have one and only one book (only 1)
- Book_subject many-to-one Subject
    - Each subject must have one or more subject of a certain book (1 or more)
    - Each subject of a certain book must have one and only one subject (only 1)
- Book one-to-many Book_publisher
    - Each book must have one or more publisher (1 or more)
    - Each publisher of a certain book must have one and only one book (only 1)
- Book_publisher many-to-one Publisher
    - Each publisher must have one or more publisher of a certain book (1 or more)
    - Each publisher of a certain book must have one and only one publisher (only 1)

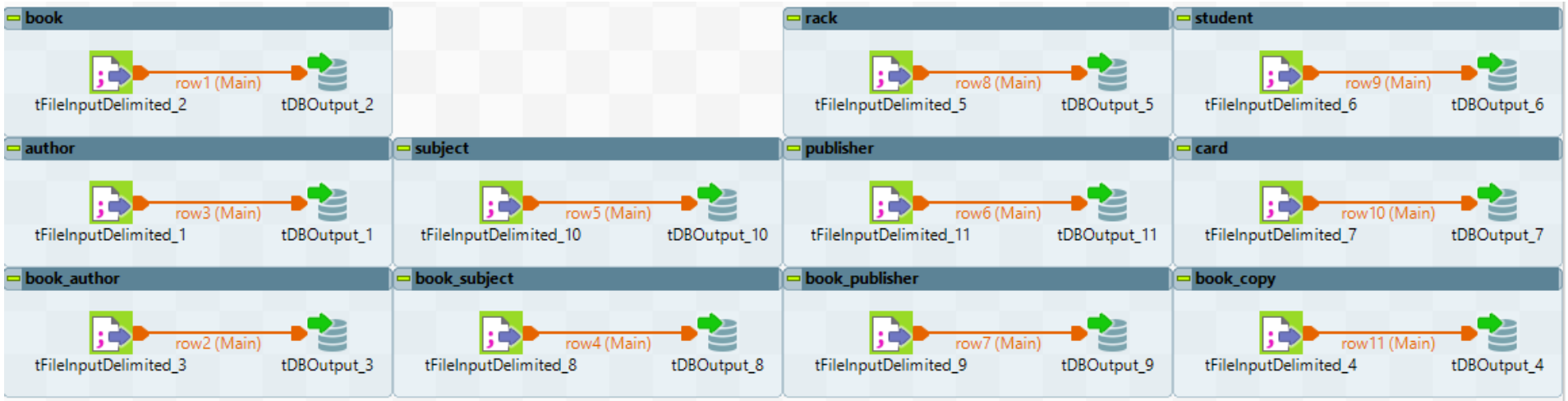
Appendix 1. Conceptual model



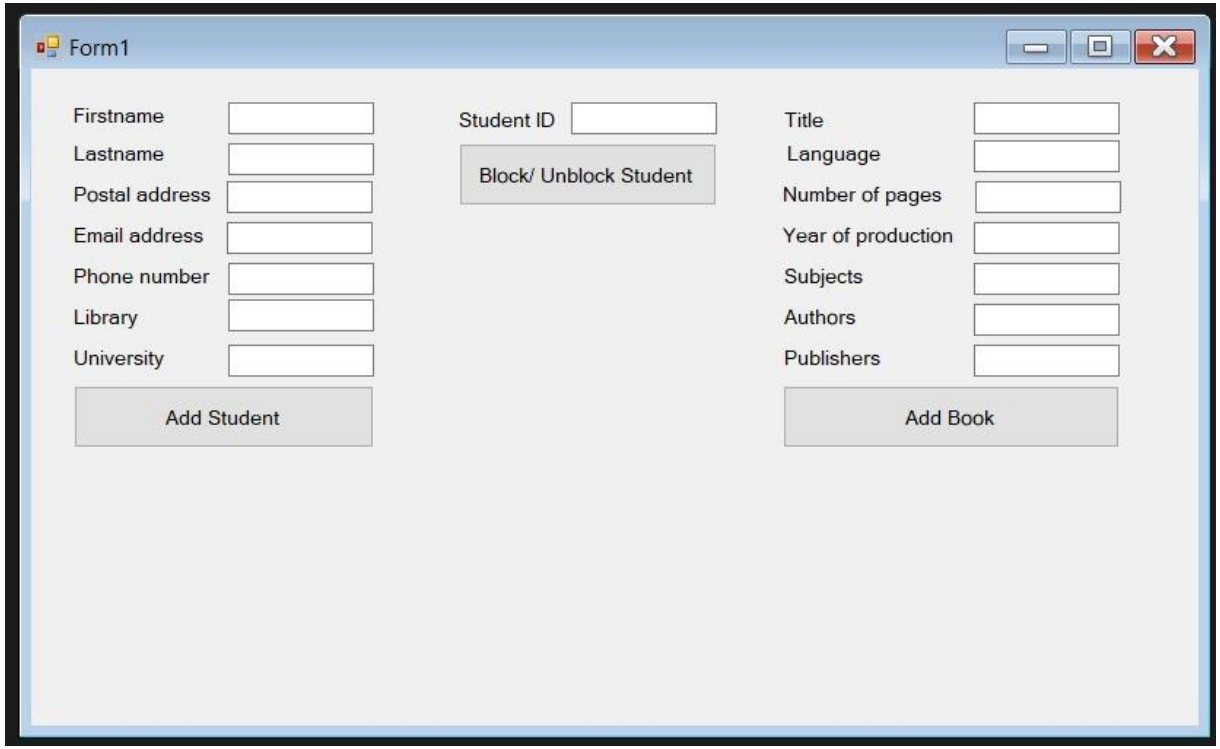
Appendix 2. Physical model

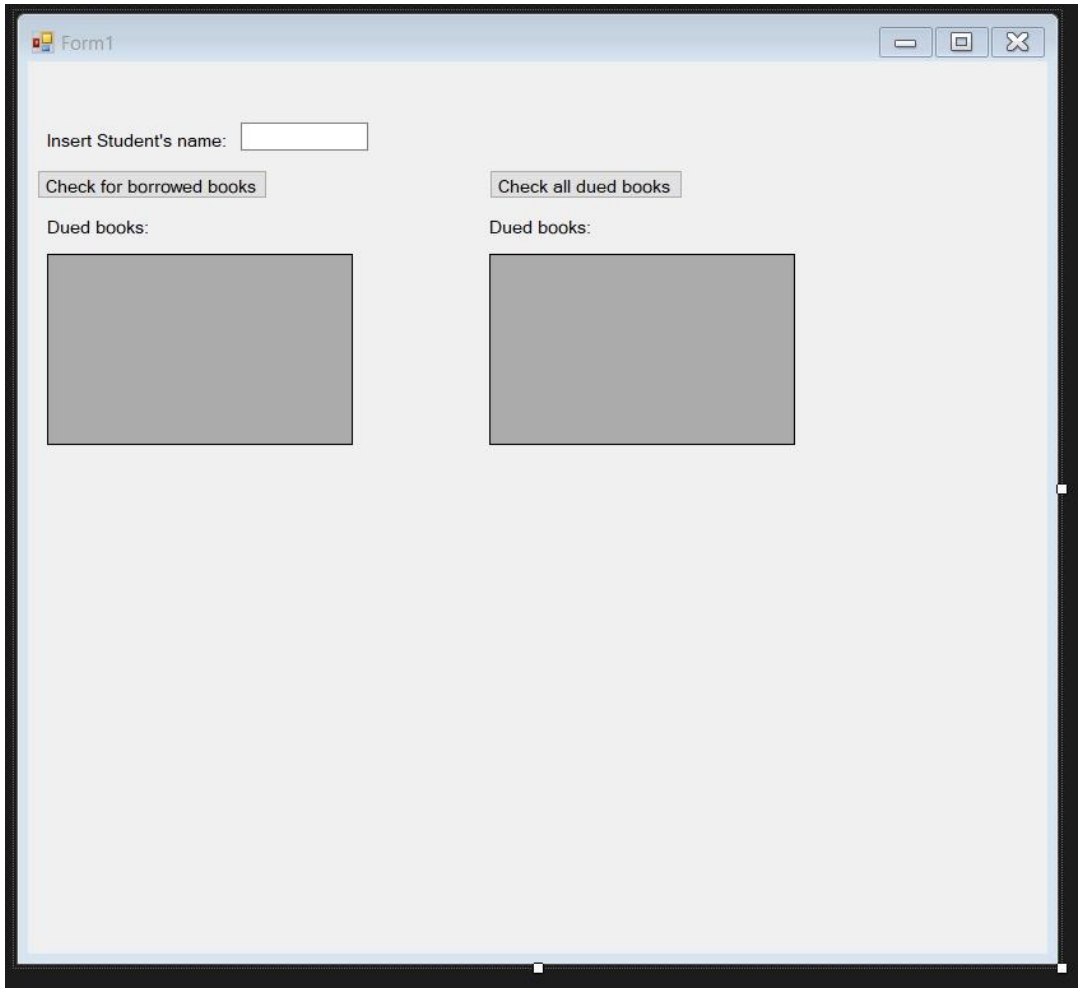


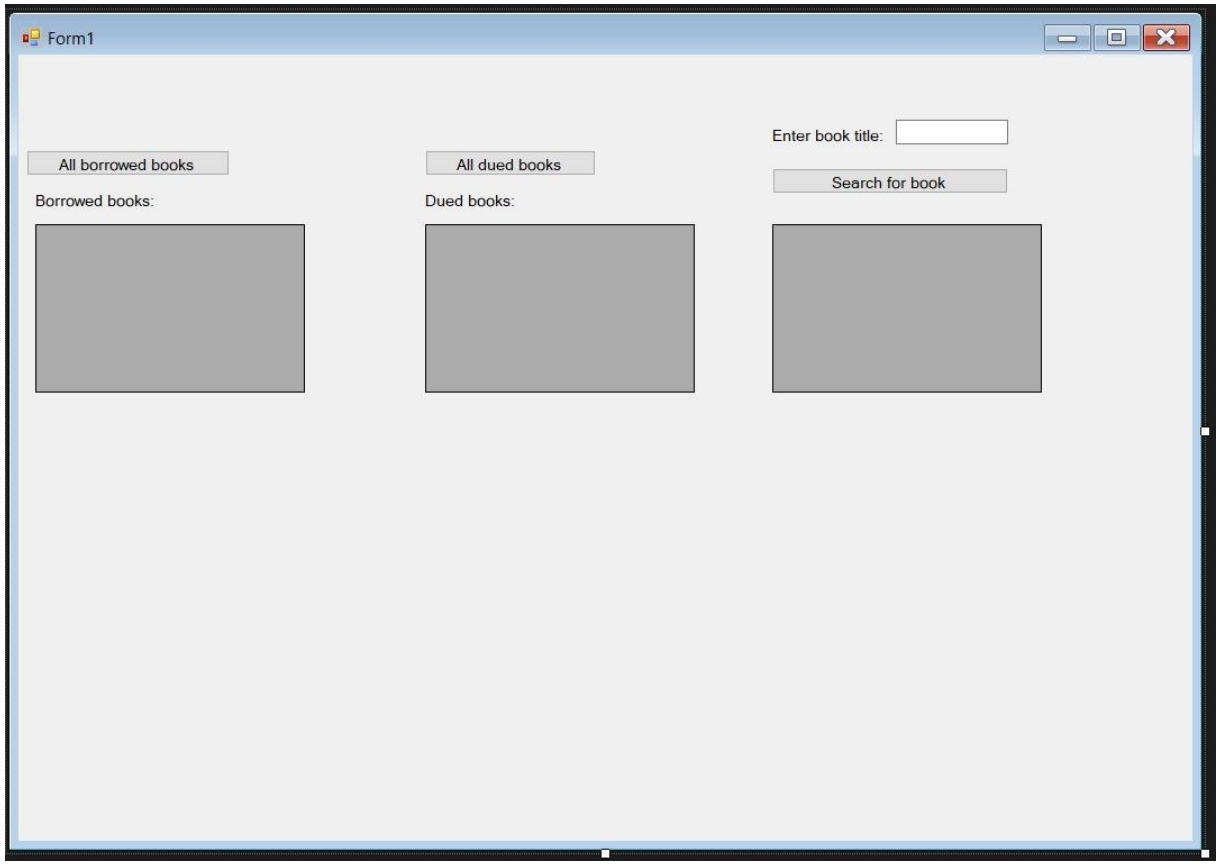
Appendix 3. Talend subjobs



Appendix 4. Graphic interface




Form1
Insert Student's name:
Check for borrowed books
Check all dued books
Dued books:
Dued books:


Form1
Enter book title:
All borrowed books
All dued books
Search for book
Borrowed books:
Dued books: