CSCI 5448 - Sanketh, Tauseef, Gaurav, Nicholas

# Library Management System

## Project Part 4: Final Report

**1. List the features you implemented (from part 1 and 2 where you listed the features you were planning to implement/requirements you listed) Show Final class diagram, discuss benefits of designing before coding.**

In part 1 of our Project we drafted certain functionality that our application will certainly have (minimum functionality requirement). Those were:

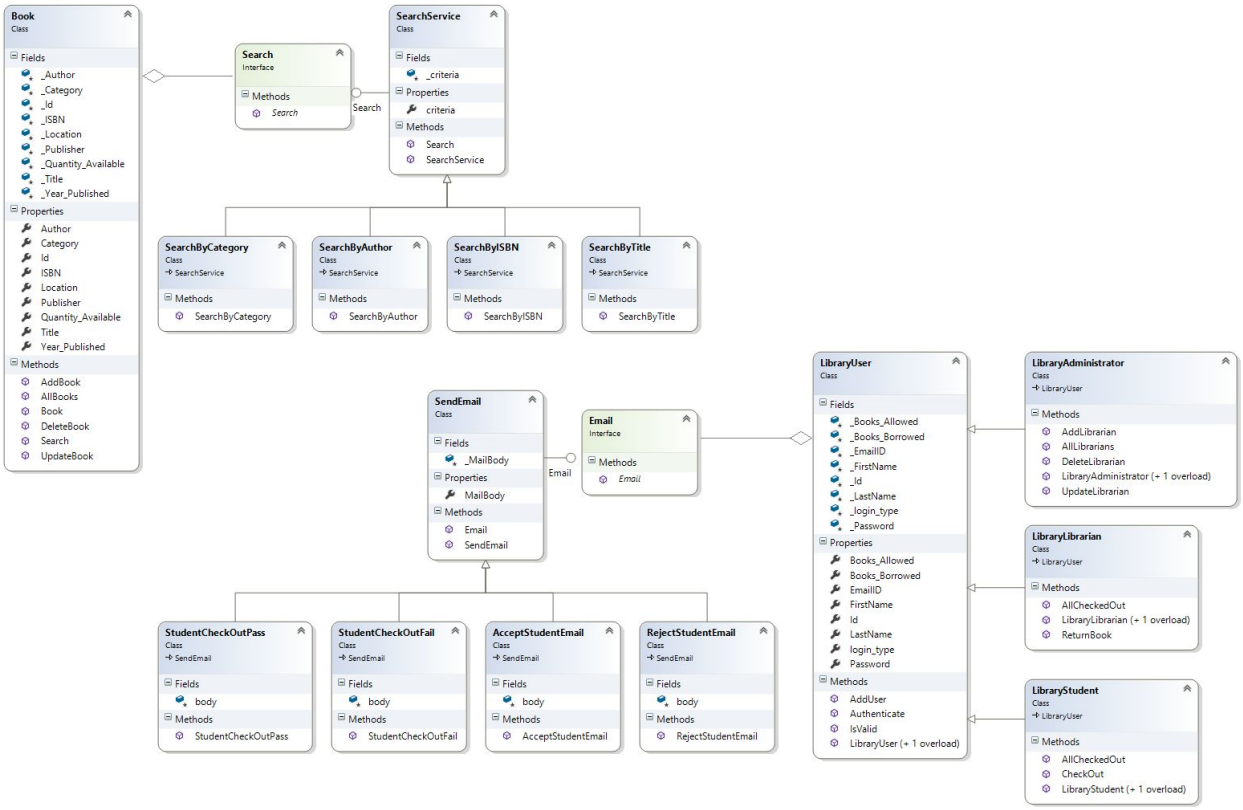
1. Student Signup
2. Student Login
3. Librarian Login
4. Admin Login
5. Admin adding a Librarian
6. Admin can add, edit, update books
7. Student can checkout books
8. Librarian can issue books to student checking his credentials

There were some stretch goals as well:

1. Student and Librarian Search functionality
2. Librarian manages Number Of Available books for Student

In our final implementation of the project, we have been able to cover all the minimum functionality requirements as stated above. Unfortunately, we have not been able to cover the stretch goals in the implementation. But we have other functionality that we have implemented which can be considered as “changes in requirements” for the project.

Here is our final Class diagram.



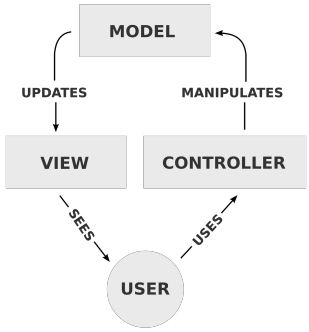

This is one of the few courses that we design before we code. To focus on the Design aspect of the rather than getting into coding details, our team chose to keep the project deliverables relatively simple. Over the period of the course our team members have spent significant time brainstorming over design and we have realised few things along the way:

1. Design is hard. Not as simple as it looks.
2. Design is a completely different ballgame compared to coding. It's more of Software Engineering is not about just code.

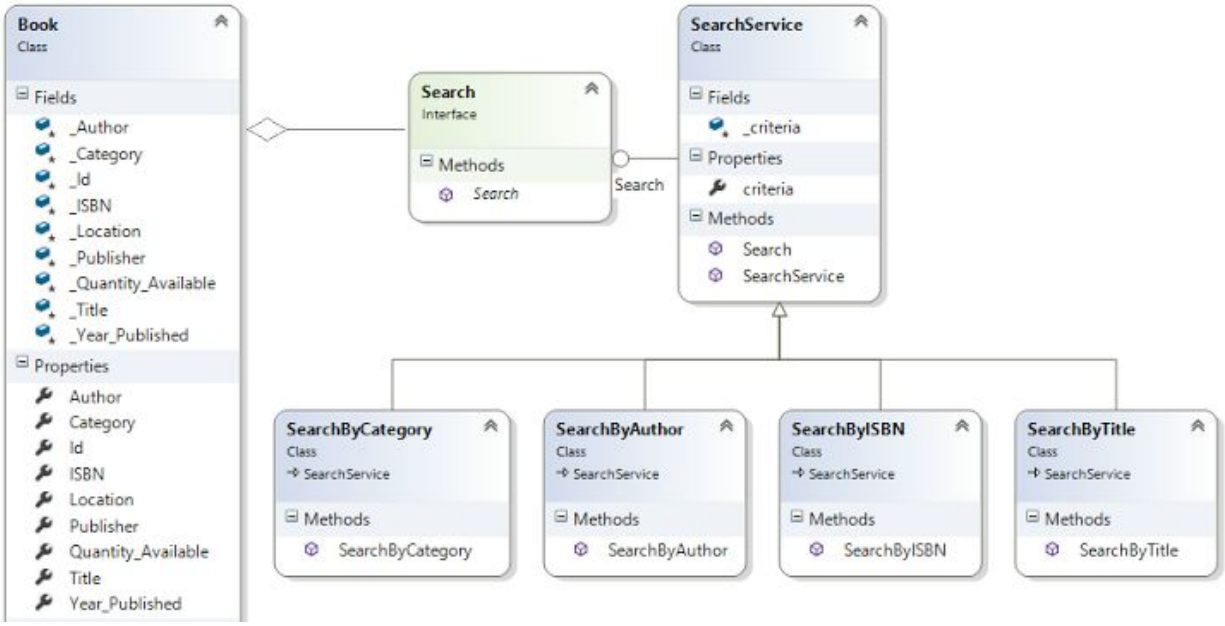
3. Design helps us give structure to our ideas before we translate it to code.
4. Design helps us develop software that can withstand changes in terms of requirements.
5. Design helps us develop software that is scalable.
6. Design is Programming language agnostic, therefore we can later use the design in a different language which has same features (different OO languages).
7. Once we have a design/architecture template, it can be reused to solve different problems, thus saving a software team time and maximise productivity.
8. Since documentation of UML/OCL diagrams becomes part of the process, it helps management give a direction to teams and lead them to complete projects within time and budget.

**2. Design patterns we used in the Library Management System. Did you make use of any design patterns in the implementation of your final prototype? If so, how? If not, where could you make use of design patterns in your system?**

As it can be seen from the Class Diagram, the overarching architectural design pattern used in our project is the MVC ( Model-View-Controller) pattern. We use the ASP.NET framework and MS SQL for the implementation. The **model** contains classes that store data that is retrieved according to commands from the controller and displayed in the view. The **controller** contains classes and interfaces that can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model. The **view** in the ASP.NET does not contain any specific classes as such , but uses the .cshtml templates to serve pages




In addition we have used some design patterns , for example the **bridge pattern** where the book class uses the the Search interface search method , within its own search method, to dynamically change the search criteria and alter which criteria is executed as shown below

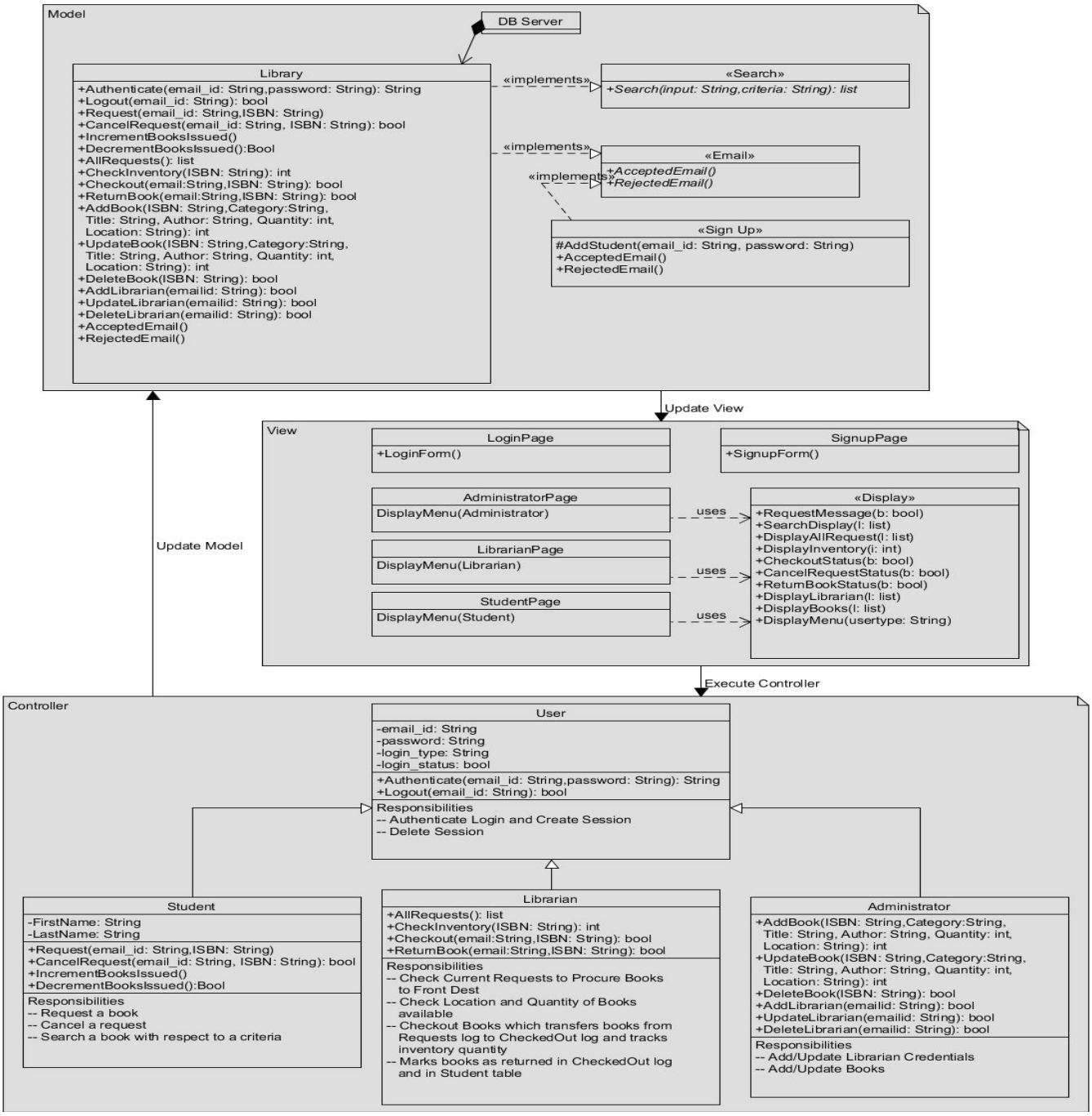



We realised after we had code finalised that we could have used the **Factory pattern** in 2 instances, but we decided to leave it at that due to time constraints.

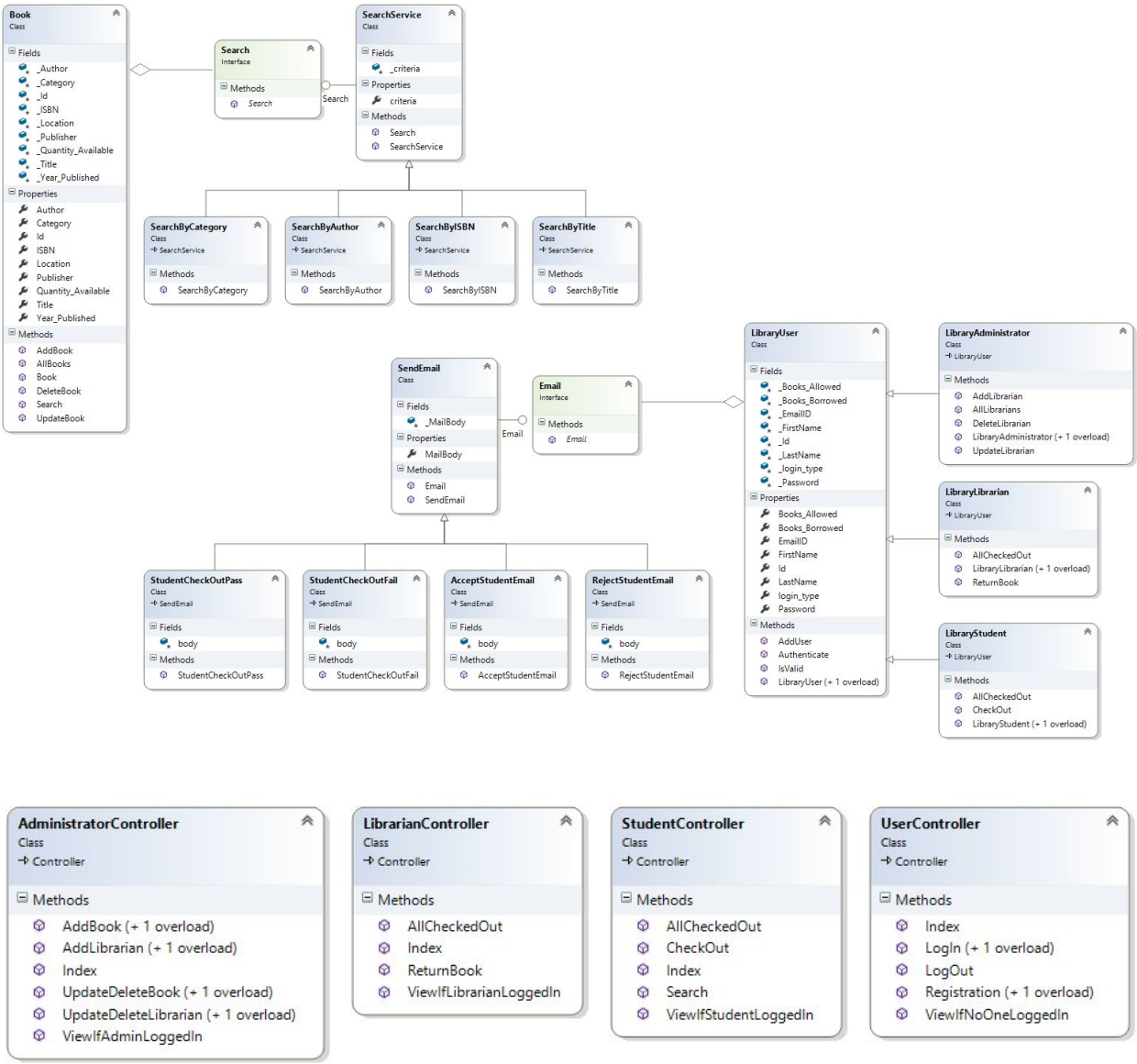
```
public List<Book> Search(String term,String criteria)
{
    Website.Models.Search search;
    switch (criteria)
    {
        case "Author": search = new Models.SearchByAuthor(); break;
        case "ISBN": search = new Models.SearchByISBN(); break;
        case "Category": search = new Models.SearchByCategory(); break;
        default: search = new Models.SearchByTitle(); break;
    }
    return search.Search(term);
}
```

**3. Compare your Part 2 class diagram and your final class diagram - include part 2 class diagram and point out the necessary changes. You can answer why you didn't realize you needed things during the design and/or how next time you will be able to make a better design after this learning experience.**

Shown below is the class diagram from Part 2 of our project




This is our new class diagram (shown below)




From the old class diagram, we had to change a couple of things:

1. We had a monster Library class which violated many object oriented design principles for example the single responsibility principle.
2. Though we used the inheritance in the Controller right with the User being the parent class and the Student and others being child classes, we added in a lot of unnecessary methods which was not specified in the Requirements.
3. The View classes had to change because the classes defined did not make sense in the context of a framework such as ASP.NET

4. There was no clear distinction between student, admin and librarian in the Model part of the design.

During the first run of the design, we didn't realise these because :

1. We were not sure about using the ASP.NET framework which is a framework for MVC that has its own classes which our defined classes can inherit and that implements a lot of functionality for us. This changes a lot of the design decisions.
2. We were not exposed to many design patterns, and couldn't have recognised the need for one immediately at that point of time.
3. It was a first run for us and we were ok with making mistakes so that we could revise and learn from them.

Having taken this course and had first hand experience trying to implement a project with design as focus and since we have spent a lot of time discussing design decisions in brainstorm sessions, we think we will have a better eye for software design in the future. Some of us are cursed with the design eye forever.

**4. What did you learn about analysis and design?**

Over the period of this course and working on the project, our team learnt some valuable lessons:

1. We can avoid some common mistakes that we could make during coding by thinking about the design beforehand
2. Mastering analysis and design is more about practice rather than listening to lectures.
3. We could all agree that the analysis and design part was harder than we thought it would be.
4. Analysis and design gives us a proper structure on paper as well as in our brains to tackle software engineering projects better.