We chose to create our GUI in NetBeans and database in Microsoft SQL Server. All of the back-end work was done in Java and SQL and the front-end was done with JForms. We chose NetBeans, instead of other Java IDEs like Eclipse, because NetBeans has the functionalities to open the entire structure of the project at once whereas something like Eclipse requires extensions to be able to do so. Our choice of using JForms was to make the program more accessible to people as that is a widely accepted platform. We also used GitHub to collaborate on the entire project and make sure the changes that were being made were accurately delivered to everyone. Whether someone used the GitHub webpage, the Git command line or GitHub Desktop, this method facilitated the process of collaboration because sharing everything was simple and we could work on our respective areas in separate branches of the same repository and not conflict with each other. This also reduced errors because we wouldn’t merge with master until our branch was good to go.

Our main goal for this project was to make the program as user friendly as possible. So, we implemented multiple tabs for the user to quickly maneuver through without difficulty. We also implemented only one search bar for efficiency and quickness of results. We implemented stored procedures for the database in order to execute with parameters that will populate the searches and tables taking out the need for inline SQL. The schema for the overall database is basically the same the schema given in the rubric.

We began with utilizing the JForm feature of NetBeans to create our table, page tabs and buttons and created the necessary objects in Java to link up with the table and properly load when called upon. When we went to implement the one search bar, we did run into some issues. Having one search bar resulted in the program only taking one input through that search bar, but the use of stored procedures passed that one input to compare with all three parameters (ISBN, book, author). This resulted in the program getting no return value because no ISBN would ever match with the value stored in the variables for book and author. The fix was pretty simple. We just went back to our query and realized we were using ANDs instead of ORs which resulted in the issue. Once that was addressed, the search function was good to go. We were also debating on whether to implement a static or live search bar. At first, we thought using a static search bar would be a better option for the program since we are querying the database and not just the local tables. But we then decided that instead of querying the database we should filter the tables and only query when the database needs to be updated. After this, it was clear that a live search bar would be the best avenue to go with as it would make the results load faster with more efficiency and make the overall functionality of search simpler.

We also had some issues getting the database to connect with our Java code through the LibraryConnection file. We set parameters that were exclusive to each of us that needed to be changed to match an existing user which prevented each of us from connecting after pulling another’s changes. This was probably the only downfall of using GitHub as every time we pulled, we would have the reset the parameters in order to run the program. Though it was not too much a hassle, it was inefficient so we ended up changing the parameters to fix the issue and also because we realized that the structure might cause issues when it comes to grading. We also implemented warnings in the LibraryConnection file. The idea behind it is, you cannot see the warnings until you go through the result set. So, on the first try, it tries to get a warning. Then it goes through all results from the query and tries to get a warning object if it finds one. This was a little more complex than it needed to be, but we eventually got everything synced up and working together.

As mentioned previously, overall, we tried to keep everything as simple as possible because having a user-friendly program was our first priority. Most of our design decisions were straight forward, although a few needed reevaluating. Using JForms gave us the freedom to design the interface to our satisfaction. Although Python is a stronger objected orientated language, the team’s past experience with Java helped with problem-solving and error-resolving and obviously the contents of the course helped with everything SQL related.