# Design Document

Dhruv Sangvikar
Net ID: dgs160230

The project was implemented using Ruby on Rails framework. For the front-end, it uses HTML, CSS and jQuery. For the back-end it uses Ruby language and the Rails framwork. It uses MySQL database. For the GUI fonts and images, it uses the font-awesome library.

## Why Ruby on Rails?

The MVC pattern of the Rails framework allows for a convenient architectural style for all the CRUD operations. Most of the library operations such as book search, book checkout, borrower creation come under one of the CRUD actions and binding of these tasks with the **C**ontroller part of the framework makes it straight forward to implement.

I have used the rails gem "Rails 3 models reverse engineering (RMRE)" to build the models from the database. This allowed me to create the database separately using MySQL without relying on the Rails Object Relational Mapping component (ActiveRecord) much.[1]

The front-end development using web technologies allows for a pretty and intuitive GUI development without relying on the underlying platform. Wherever convenient and possible, I have used Jquery to hide/show different elements of the page.

## Why MySQL?

MySQL being open source and freely available, as well as being cross platform, looked a suitable choice. Also, the "phpmyadmin" tool provides a good GUI client to interact with MySQL, which it made easy to perform some operations such as immediate

## How did I dump the data into the database?

I normalized and put in the data into the database by using python scripts. I have stored them in the db folder of the application for future use.

## Application Design Details

The application interfaces with the database using the Rails framework. The model mapping is done for the most part by the framework. Queries need to be put in manually for all the join operations – which is majority of them. Simple select operations using primary key or such field are handled directly by the inbuilt ORM functions. All the requirements specified the project document on elearning have been implemented and tested.

**Assumptions**

1. For simplicity, I have only used the 10 character ISBN. The database also stores only these 10 character ISBN.
2. To store card_id of borrowers, I have assumed that card_id would be integer. Wherever required and possible, I have displayed them as the 6 digit zero padded ids on the GUI. But the actual storage is done as int and this does not allow for the pre-padded zeros. Storing them as string would have been inefficient and auto incrementing them would have been complex in that case. The int representation solves these problems.
3. While searching for existing books checked out by a particular borrower, the search can be done as mentioned in the document. I have assumed when the name is given, it would be either the first name only or last name only or full name. Also, it will only return currently checked out books and not the borrowers history. There's no option to view the borrowers history.
4. By default, all the paid fines would be hidden and there is no option to view the paid fines.
5. Fines are stored with 2 decimal places. But if fine $1.50 or $1.00 it would be displayed as $1.5 and $1 respectively.
6. If a book is checked out, it will be shown on the search results page, but the borrow link would be disabled. If the URL is typed in directly, it is possible to reach the checkout page for that particular book. **But the loan would not be completed and an error message would be displayed.**
7. Book Images are directly accessed using the URL when the page is rendered. Sometimes, images fail to load or are unavailable. In that case the cover is blank. No default cover is used.

References:

1. Rails 3 models reverse engineering (RMRE) : https://github.com/bosko/rmre
2. Rails documentation : guides.rubyonrails.org/getting_started.html
3. Font- awesome : fontawesome.io/icons/