Library Management System
Project Design Document

Introduction

This project is a library management system where users interact with a graphical user interface to manipulate items in a library database. This system is intended for librarian users who can search for books, check out books, check in books, create new borrowers, and handle fines via a GUI that connects to the backend via Django.

The technologies this project uses include Python, HTML/CSS, and the Django web framework.

Design Decisions

- Python Programming Language as Main Language
  - Python was chosen as its implementation is the simplest to implement for a class project and allows for the integration of the Django web framework.
- HTML and CSS Templates
  - Basic HTML and CSS are used in order to keep the main user webpage simple and easy to use, as aesthetics in terms of looks is out of the scope of the project. Additionally, it allows for a simple, customizable user interface for the required commands for the user.
- Choosing SQLite for Relational Database Management
  - Since our implementation is simple Python and the website for our demonstration is hosted locally, a simple SQLite implementation with files fits within the scope of the project.
- Integrating Django Web Framework
  - The Django framework is the simplest to implement web functionality that connects to the backend core Python code. Django's feature also allows for quicker development of the project.
- Separation of Logic
  - Our project clearly separates the logic of the core programs, which manipulate the database, and Django, which calls the core functions based on user interaction with a GUI. This allows for better scalability if we need to add more functionality, as well as more readability for other developers to understand the program.
- Using ISBN-10 instead of ISBN-13 for Database
  - Uses ISBN-10 as the primary key for books to be handled in the project for simplicity.
- Core Files Manipulating Database

We use our core files to manipulate the database instead of using Django ORM, since our programs were already built utilizing sqlite3 from Python to query and make changes to the database without affecting the schema.

System Architecture

We use the Model-View-Template system architecture since our implementation features the Django web framework.

Model Layer

The core programs in the model layer are responsible for using the information passed to them by Django in the view layer to make SQL queries to the database or to update it by adding items or making changes to the database.

View Layer

Django handles the view layer by receiving the user input from the webpage (get/ post methods) and sending that information to the core programs in the model layer.

Template Layer

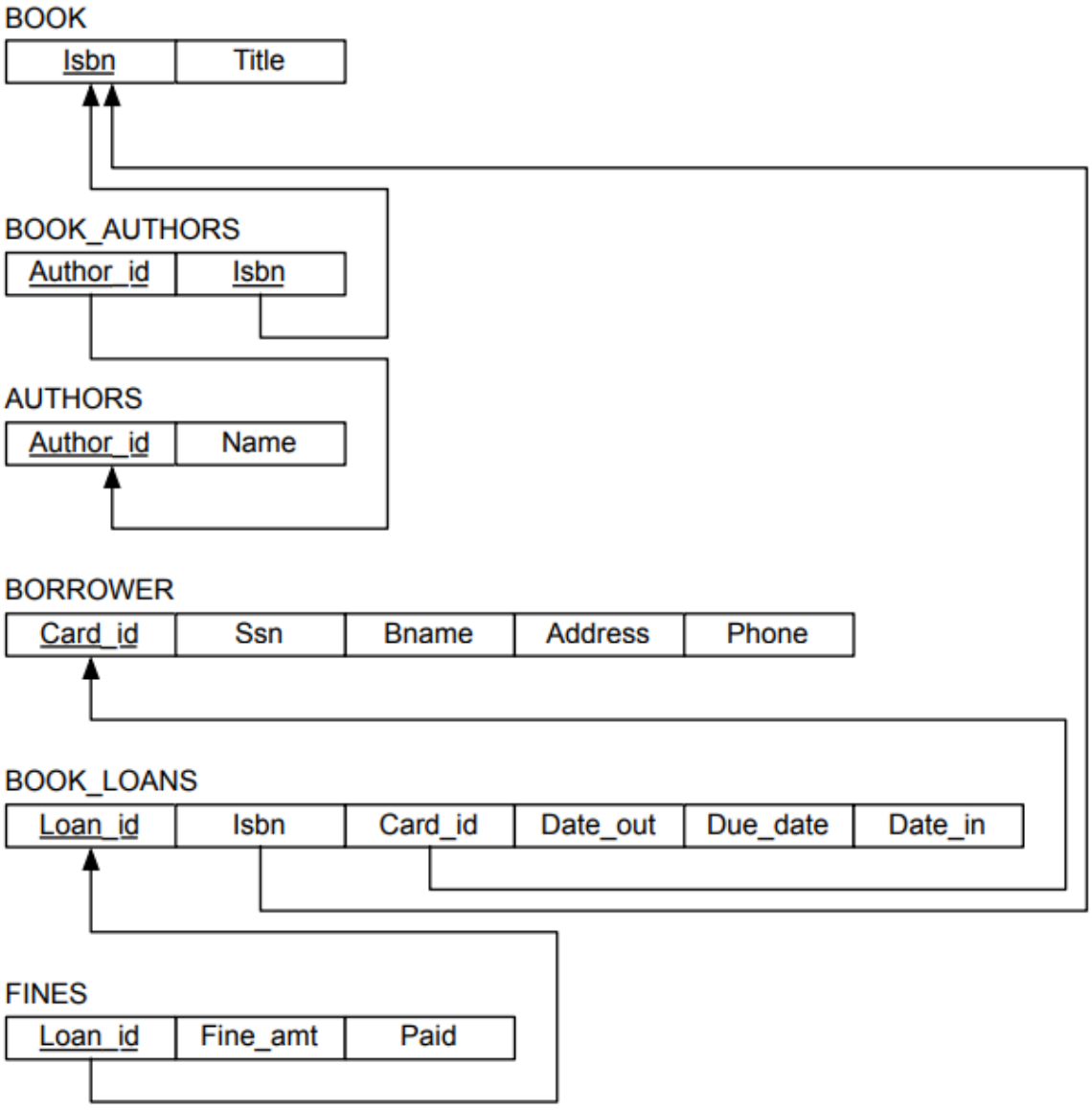
We use HTML Templates that receive the data from the view layer to then render that information to the web browser.

Assumptions

- All users of this system will be Librarians with the same level of system access
    - No need to create a super-user
    - No need to create a borrower-facing homepage
- System environment is being run on Python 3.10 or later
- System environment has Django 5.2.9 or later
- System environment contains sqlalchemy and pandas packages installed
- Book titles will not be longer than 255 characters
- Author names will not be longer than 255 characters
- Borrower names will not be longer than 255 characters
- Library uses ISBN-10 for ISBN in database

Database

Schema:


BOOK
Isbn
Title
BOOK_AUTHORS
Author_id
Isbn
AUTHORS
Author_id
Name
BORROWER
Card_id
Ssn
Bname
Address
Phone
BOOK_LOANS
Loan_id
Isbn
Card_id
Date_out
Due_date
Date_in
FINES
Loan_id
Fine_amt
Paid