**CS 6360.501 Database Design**

**System Architecture, Design Decision’s & Assumptions**

The Library System Management project was developed using Python IDLE (v3.6.5) and the Tkinter library available in it for GUI. The database management system was implemented using PostgreSQL 12. The Library Management Application is a local desktop application built using python Tkinter for the front end Graphical User Interface (GUI) and back end with python connected to the PostgreSQL database. The following are the tables used in the database:

1. BOOK
2. BOOK\_AUTHORS
3. AUTHORS
4. BORROWER
5. BOOK\_LOANS
6. FINES

During the design of the database, some data from the .csv files was imported directly into the tables. But for the other tables, the data cannot be directly loaded. So the following decisions were made:

* Created an authors\_temp table and stored all the authors names. As there was many repetitive name but in authors each author should have only one author\_id. So the distinct author\_name was imported into AUTHORS from the authors\_temp table.
* To import data into the BOOK\_AUTHORS, we need the auto generated author\_id and the ISBN of the book written by that author. So to implement this, a new table book\_authors\_temp was created and the ISBN and the author\_name was impoted into the table.

Now to insert values into BOOK\_AUTHORS, we select the author\_id and ISBN from AUTHORS and book\_authors\_temp tables and insert it into BOOK\_AUTHORS based on the condition if the author\_name in both the tables are the same.

* From the given .csv dataset, ISBN10 has been chosen in this project for all the ISBN fields.
* For the given 25,000 books there was many author\_name repeating. So only the distinct author\_name was impoted but it is mapped to all the books.
* The loan\_id (BOOK\_LOANS) and card\_id (BORROWER) are defined as SERIAL. So the values auto increment upon inserting tuples in the table.

**GUI**

The Graphical User Interface is designed using Python’s Tkinter library. This library consists of Labels, Buttons, and Textbox etc. which are easy to customize according to the user requirement. We also use Python’s in-built library Treeview to display multiple columns and to get the view of a table. So when we fetch the results from the database we store it in the treeview list and use it to display in the GUI.

**Database**

To implement this Library System Management project, PostgreSQL was used. The given database schema was implemented with the following constraints for the tables.

1. BOOK

{ isbn varchar(10) primary key,

title varchar(1000)

}

1. AUTHORS

{ author\_id serial primary key,

name varchar(1000)

}

1. BOOK\_AUTHORS

{ author\_id int foreign key,

isbn varchar(10) foreign key

}

1. BORROWER

{ card\_id serial primary key,

Ssn varchar(20),

Bname varchar(100),

Address varchar(250),

phone varchar(30)

}

1. BOOK\_LOANS

{ loan\_id serial primary key,

isbn int foreign key,

card\_id foreign key,

date\_out date,

date\_in date,

due\_date date

}

1. FINES

{ loan-id int foreign key,

fine\_amt decimal,

paid Boolean

}

All the data from the .csv files was parsed and inserted into the respective databases using the import wizard tool available in PostgreSQL.

**System Architecture**

C:\Users\Nitesh\Downloads\Untitled Diagram.png