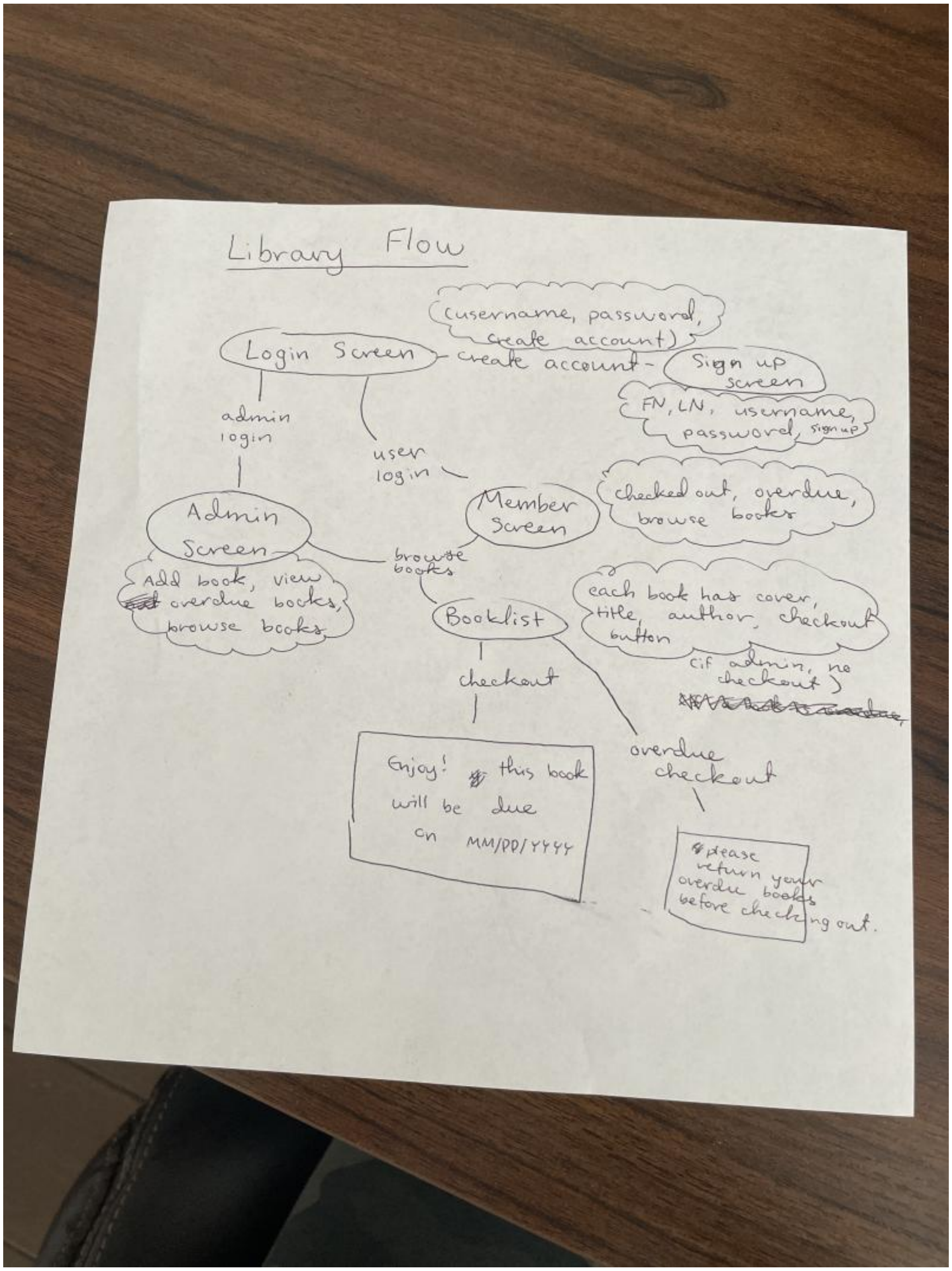
ELibrary Final - Writeup


Library Flow
(username, password, create account)
Login Screen
create account
Sign up screen
FN, LN, username, password, sign up
admin login
user login
Admin Screen
Member Screen
checked out, overdue, browse books
browse books
Add book, view overdue books, browse books
Booklist
each book has cover, title, author, checkout button
(if admin, no checkout)
checkout
overdue checkout
Enjoy! this book will be due on MM/DD/YYYY
please return your overdue books before checking out.

- ➔ Drawing is of the general structure of the application's screens and the user flow. Users will login and will open their library screens, differing on whether (if) they are admin access or not. Admin users will be able to view users and books, while Members can view/checkout/return books.
- ➔ The backend will consist of an abstract User class, which will be extended by the Member and AdminUser classes. User will include abstract methods to get

username, name and password information, as well as a concrete function that allows its “children” to open files.

- Interface CanOpenFile implemented by Users
- Interface HasTable implemented by Screens that display data
- When logging in or signing up, an exception will be thrown if proper input is not provided. These are caught by the screens and will show an error message.
- Member class will rely on JSON handling to instantiate and get member data (including books checked out), both reading and writing to Members.json.
- Since all actions involving books involve reading the file and messing with the book in the data, we use a general method for handling books which has a switch statement that dictates what to do with them.
- Recursive quicksort used to organize users alphabetically.
- If a user wants, they can press the wildcard button to get a random book checked out for them.
- A LinkedList is used as an intermediary between the CSV data and the 2D array used to display it in the GUI.

This program shows more advanced handling of data via reading and writing to files in JSON, while also connecting a graphical user interface to this data. The JSON file keeps track of all users even after the program has stopped running, and changes dynamically. As well as JSON, reading CSV data is also used to keep track of the books.