**Phase I – Design and Analysis**

**Project Name**: Library Book and Employee Management System

**Application Description**: This application will provide a way for library employees, visitors, and management to interact digitally with all the resources available in a library. It should support a maximum of 1000 users, and 1000 library resources. There should be a database (textfile) that holds all the library books and videos available to the public. There should also be another database with the employees of the organization. There does not need to be databases for the resource category or the authors because we will just can just use the same resource database for this info (just grab the info associated with each of the records since you will be doing that anyways to create record objects most likely). The initial application is a login screen as soon as you run it. You will either login as an existing user, or choose the option to create a new user. The prompt will tell you when no users exist and then it will run you through the setup prompt to create a user if you choose. There exist three types of users which will be implemented via inheritance as follows: admins, employees, and basic users. A basic user (customer) will only have the permissions to login, search for a book by title, by author name, resource ID, as well as view the status of the different library resources (checked out or available). A user should also be able to sort the library books by ID number and filter the books by author, and by category (filtering means only displaying books by a certain author - and only displaying books of a certain genre). This class does not cover sorting via characters or strings. This implementation should not involve filtering via characters, an example implementation might be mapping from an author to an author ID or category to a category ID and filtering on those associated numbers instead. This will be done by resource category, and author classes where each will have their own type of id associated with them (most likely). An Employee user can do everything that a basic user can as well as modify the availability field for a resource (for example changing the status of a book to checked out). An admin can do everything that a basic user and an employee can do while also being able to add and delete employees. There will be a command prompt whenever ANY user logs in, with a list of commands available to them. Textfile databases will be created by default if none exist upon the first user logging in. If there exist no users in the start of the database – there will be a prompt that tells the user this and asks them if they want to create a new user. After this is done, it will take them to the login screen.

**Object Descriptions:**

**Author:** Several of these objects will be created and then stored into a structure of your choosing upon the application login for however many authors are found in the database. The only attributes needed here are First name, Last name, and an author ID. These objects will be created just to keep track of the different authors when we create new resources. Since one must loop through each resource that exists in the DB to run this application, it might be best to just check at each iteration if the author and author ID exist in the structure you use to store authors and if it doesn’t, create a new one.

**Resource Category:** Multiple instances of this class will be created to keep track of the different categories when we create new resources (Non-fiction, fiction, etc. ). These objects will be created just to keep track of the different categories when we create new resources. Since one must loop through each resource that exists in the DB to run this application, it might be best to just check at each iteration if the resource type and resource type ID exist in the structure you use to store resource types and if it doesn’t, create a new one.

**Resource:** This could range everywhere from a book to a vhs. It will have an id number, a resource category with a resource category id to filter on- (for example, 1 for non-fiction), the resource name, the resource type (book, vhs), the resource author, the resource author ID, the year published, the status field which will be checked out or available, and a checkedOutBy field that contains the ID of the user who has checked out that resource.

**Basic User:** The attributes for this user will be as follows. A first name, last name, username, password, a User ID (which will need to be randomly generated to make sure no collisions onto that value), an email address, a mailing address, and a field for the resource they currently have checked out. All users can only checkout one thing at a time. The commands available to a basic user are display resources, sort by ID, sort by author, sort by category (non-fiction, fiction). Remember users can’t checkout or return books, an employee must do this for verification.

**Employee User:** An employee user must have all the same fields as a basic user. A first name, last name, username, password, a User ID (which will need to be randomly generated or incremented on creation to make sure no collisions onto that value), an email address, a mailing address, and a field for the resource they currently have checked out. The only difference between a basic user and an employee user, is that an employee user can update resource fields (change a book’s name, change the status of a book to checked out / available etc.) The commands available to an employee user are display resources, sort by ID, sort by author, sort by category (non-fiction, fiction), and modify resource (edit status field, edit checkedOutBy field).

**Admin User:** An admin user must have all the same fields as a basic user and an employee user. A first name, last name, username, password, a User ID (which will need to be randomly generated or incremented in order to make sure no collisions onto that value), an email address, a mailing address, and a field for the resource they currently have checked out. The only difference between an admin user and an employee user, is that an admin user can delete, modify, and add employees. The commands available to an admin user are display resources, sort by ID, sort by author, sort by category (non-fiction, fiction), modify resource, add user, modify user, and delete user. An admin can’t delete an employee if they currently have a book checked out.

**Validation:**

• Proper email format, xyz@gmail.com (regex) when creating a user.

• Addresses for users should be more than 5 characters and contain letters and numbers.

• New usernames must be unique

• If you choose to create a resource, the author name and ID should be unique if you decide to not use an author that currently exists.

• If you choose to create a new resource, and the category of the resource does not exist already, the resource category ID and name should be unique.

• Passwords for users should have 4 letters and 4 numbers. They don’t have to be encrypted

• Employee IDs should be unique when creating new ones

• Resource IDs should be unique when creating new ones

• Author IDs should be unique when creating new resources with new authors

• Resource category IDs should be unique when creating new resources with new authors

• When an employee decides to modify the status (checked out or available) of a resource it should be either “checked out” or “available”

• When an employee decides to modify the checkedOutBy field of a resource, it should be the user ID of someone who exists in the user database.

• User name and password upon the login screen should match that of a username and password that exists in the DB.

• Modifying resources should not modify the entire database but hopefully just the line that matters

• Modifying employees shouldn’t modify the entire database but hopefully just the line that matters

**User Interface:** The program for the most part can be a CLI. Anytime you run a command, it should output the results of that command and then display the options available to the user again.

**Statechart Diagram:**