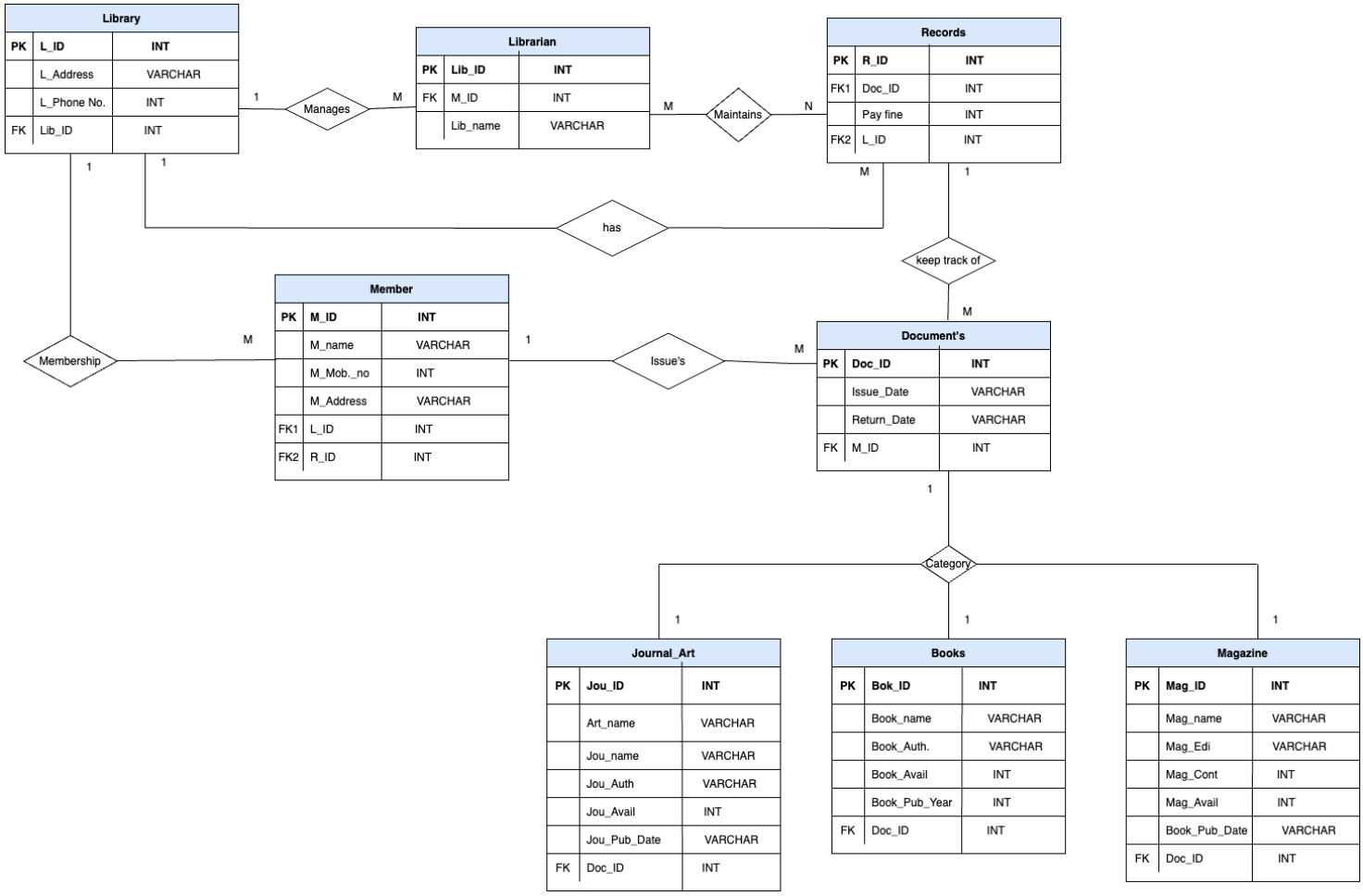
PHASE 1 - ER DIAGRAM (Major modifications has been done in all phases, below 2 ER diagrams are just for explaining why we changed our er diagram, do not consider this for grading, this 3rd ER diagram must be considered for grading)



The ER Diagram was Submitted as Phase 1 where we can see the flow of the project visual, which seems to be great. For initial stage of development this was our basic understanding of the project to start with. As per the comments and mistakes highlighted by the TA, we came to know our mistakes about the project.

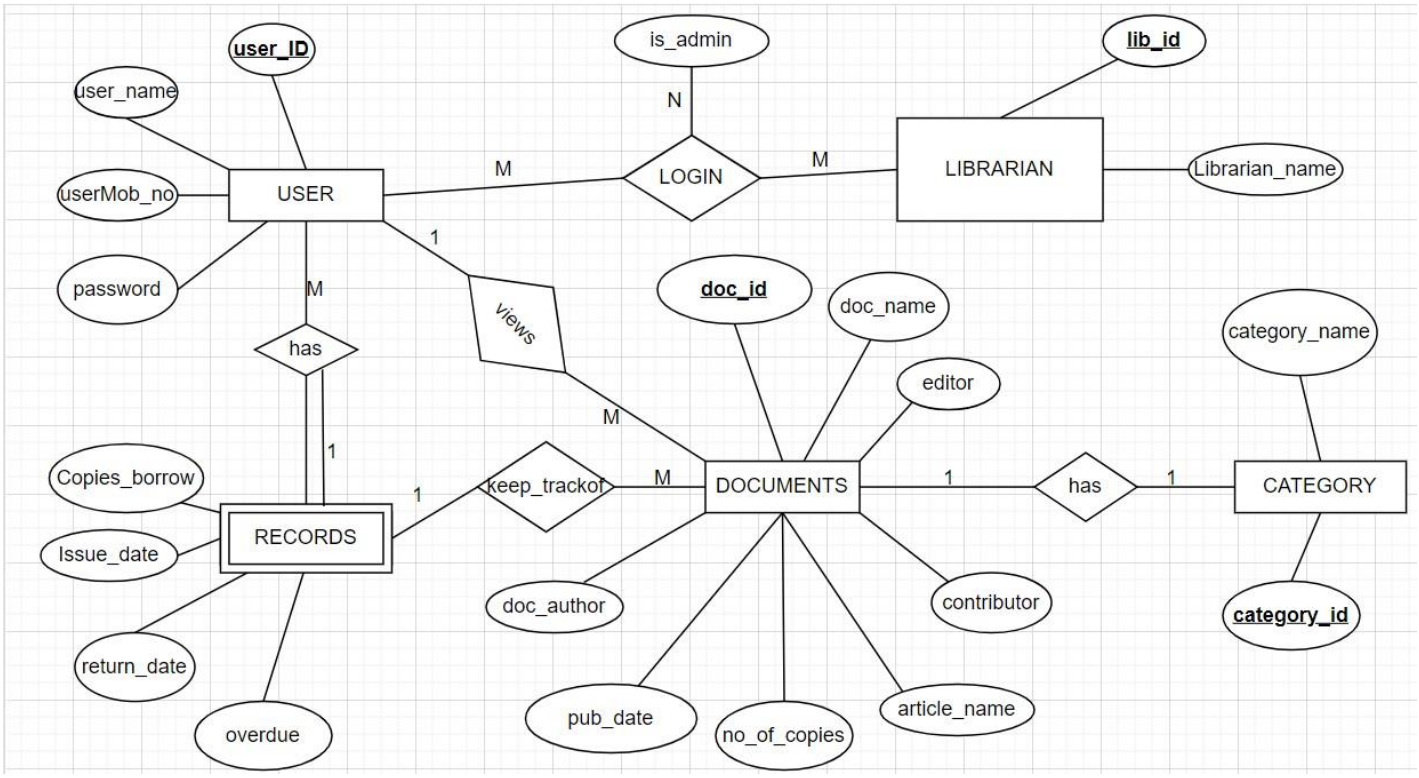
- Lack of professionalism
- All attributes were not covered
- Lack of functionalities.

As per the gaps highlighted, we started working and realized that our flow was lacking a lot professionally, which means we were facing a lot of issue converting it to project.

- We were not able to identify the difference between 2 types of users,
- not able to segregate documents as per their category
- User table was not connected with records table.

We removed the library table as it wasn't required and created an extra category table with the help of which we can identify the difference between all the documents. After a lot of brainstorming and thinking in a logical manner we came up with an ER which was submitted in phase 2 as given below.

PHASE 2 - ER DIAGRAM



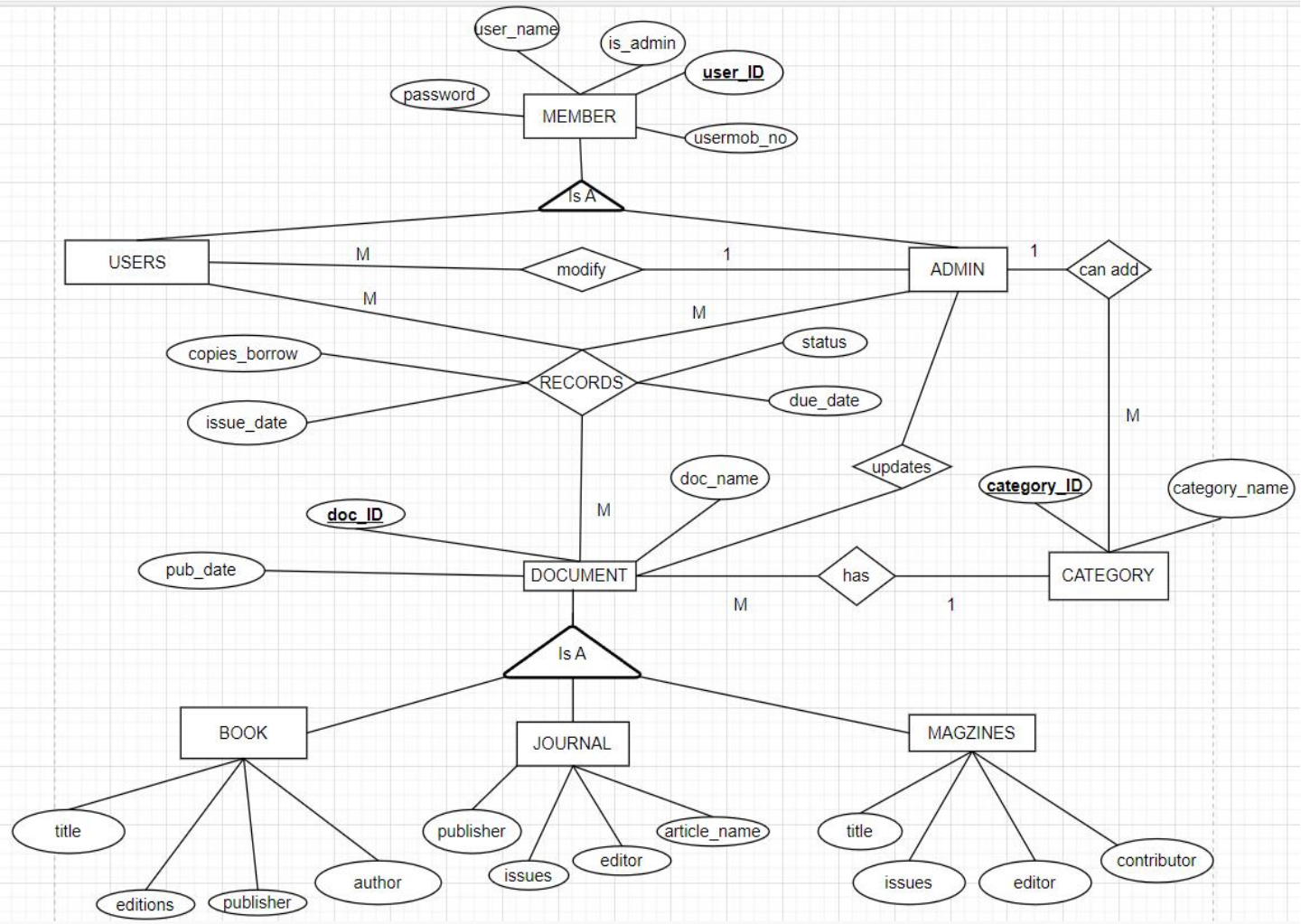
In Phase 2 we added the missing attributes which were copies of the documents, statue/overdue of the records, contributors etc. Later on, we realized there were a lot of attributes that were common between different types of documents (ie Document name/title, Published date, author, Publisher). So instead of making 3 different table we optimized the backend and merged all the table and named and DOCUMENT. We the help of we saved huge amount of repetitive code writing in our implementation and software started responding faster. To differ the difference between the document we need a table on basis of which it can define the difference, so we created a CATEGORY TABLE. Phase 2 diagrams was somewhat looking logically correct but couldn't convey the flow of our system because we were not able to show our implementations flow through our ER diagram. This was the duration when we met TA and cleared our doubt related to our project.
She highlighted mistakes about: -

- Redundant attribute
- Maximum attributes were covered but few attributes were missed.
- And couldn't understand the flow looking at the Diagram.

As per the guidance we started improving out ER diagram and implemented it. We removed the relation Login and librarian which wasn't making any sense. Used is_admin attribute (boolean) in the User table to identify the difference between user and admin so that we can give appropriate rights to the actual user of this software.

After all the improvement we had our final ER Diagram which is logically as well as visually correct. Can understand the flow of the project with the help of it.

PHASE 3 (Special note this is the final ER Diagram for the project)



The users of the table are related to Document table with the help of Records relation ie many to many relation which has played an important role in project. Record as a relation forms a table which doesn't have any of its own primary key but has all the primary keys of the entity surrounded around it (ie USER, DOCUMENT, ADMIN). Record cannot be represented as an individual strong entity as it does not have its own primary key.

There are 2 types of users for our project

- User
- Admin

User

- User our software can use this software to check the availability of documents
- Can Issue a document looking at the availability of the document.
- Can decide its own return date of the document (cannot exceed later than 2 weeks based on the issue date)
- Can issue as much as he wants (cannot take more than 3 copies and cannot issue a document that has been already issue by the user)
- Can search the document as per the category and as per attributes related to the document.

Admin

- Admin also can search for a specify documents
- Admin can update the user of the software, (ie can Add a user, Delete a user, Modify the user details.)
- Admin can maintain the document quantity availability (ie Add a document, Delete a Document, Modify the Document quantity or details as per the inventory of the library)
- Admin Can even Add different section/category of the document (ie if the document is not a book, magazine or journal)
- Admin can keep track of the records which user has returned the document which user has past the due date on basis of which he can penalize the user with a fine.

This all has been explained in the ER diagrams. And how to documents table have be separated can be clearly visualized with the help of ER diagram above.

PHASE 2:-

Strong Entities:

USERS (**user_ID**, user_name, password, is_admin ,usermob_no)

DOCUMENT(**doc_ID**, doc_name, doc_author, pub_date, no_of-copies, article_name, editor, contributor, issues, publisher, editions)

CATEGORY(**category_ID**, category_name)

Relationships:

USERS (**user_ID**, user_name, password, is_admin ,usermob_no)

DOCUMENT(**doc_ID**, doc_name, doc_author, pub_date, no_of-copies, article_name, editor, contributor, issues, publisher, editions, cat_ID)

CATEGORY(**category_ID**, category_name)

RECORDS(**user_ID** , **doc_ID**, copies_borrow, issue_date, status, due_date)

SQL SCHEMA :-

```
CREATE DATABASE "Library Management System"
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'English_United States.1252'
    LC_CTYPE = 'English_United States.1252'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1
    IS_TEMPLATE = False;

CREATE TABLE IF NOT EXISTS public."CATEGORY"
(
    "category_ID" integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 4 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    category_name character(50) COLLATE pg_catalog."default" NOT NULL,
```

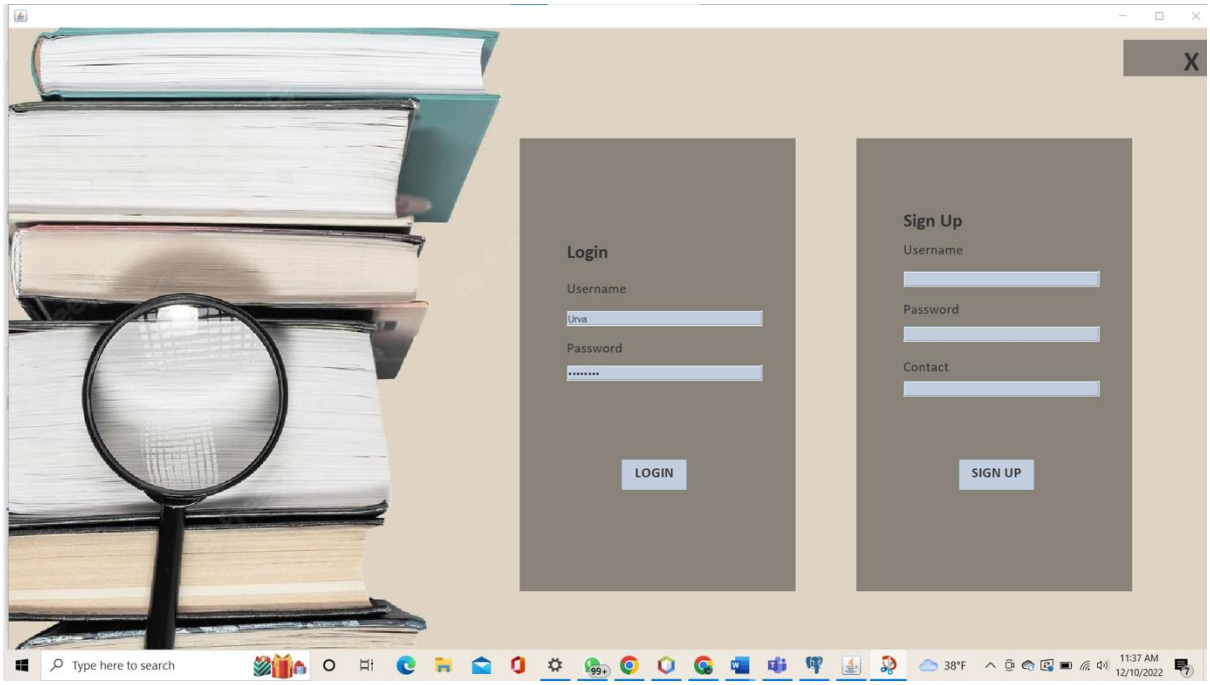
```
    CONSTRAINT "CATEGORY_pkey" PRIMARY KEY ("category_ID")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."CATEGORY"
    OWNER to postgres;

CREATE TABLE IF NOT EXISTS public."DOCUMENTS"
(
    "doc_ID" integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    doc_name character varying(50) COLLATE pg_catalog."default",
    doc_author character(50) COLLATE pg_catalog."default",
    pub_date date,
    "no_of-copies" integer,
    article_name character varying(50) COLLATE pg_catalog."default",
    editor character(50) COLLATE pg_catalog."default",
    contributor character(50) COLLATE pg_catalog."default",
    "cat_ID" integer NOT NULL,
    "Issues" character varying COLLATE pg_catalog."default",
    "Publiser" character varying COLLATE pg_catalog."default",
    editions character varying COLLATE pg_catalog."default",
    CONSTRAINT "DOCUMENTS_pkey" PRIMARY KEY ("doc_ID"),
    CONSTRAINT "cat_ID" FOREIGN KEY ("cat_ID")
        REFERENCES public."CATEGORY" ("category_ID") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
```

```
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."DOCUMENTS"
    OWNER to postgres;

CREATE TABLE IF NOT EXISTS public."USERS"
(
    "user_ID" integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 0 MINVALUE 0 MAXVALUE 2147483647 CACHE 1 ),
    user_name character varying(50) COLLATE pg_catalog."default",
    usermob_no double precision,
    password character varying COLLATE pg_catalog."default",
    is_admin boolean NOT NULL DEFAULT false,
    CONSTRAINT "USERS_pkey" PRIMARY KEY ("user_ID")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."USERS"
    OWNER to postgres;

CREATE TABLE IF NOT EXISTS public."RECORDS"
(
    "docu_ID" integer,
    "user_ID" integer,
    copies_borrow integer,
    issue_date date,
```

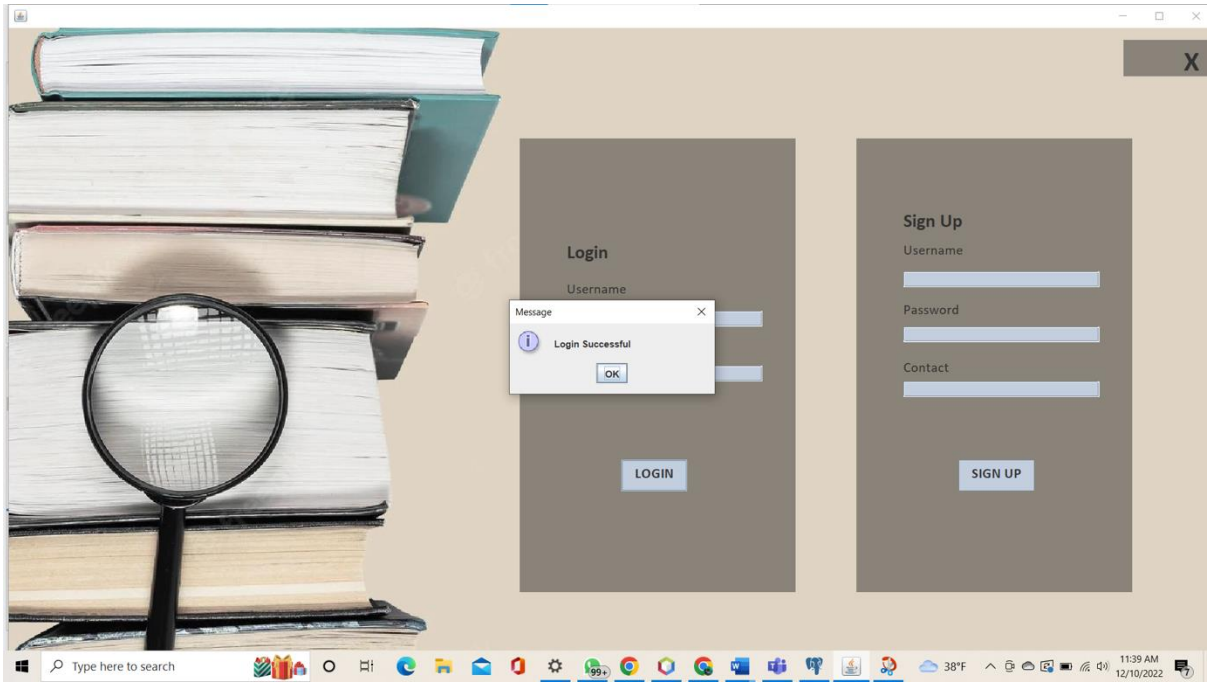
```
    due_date date,
    status character varying COLLATE pg_catalog."default",
    CONSTRAINT "docu_ID" FOREIGN KEY ("docu_ID")
        REFERENCES public."DOCUMENTS" ("doc_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID,
    CONSTRAINT "user_ID" FOREIGN KEY ("user_ID")
        REFERENCES public."USERS" ("user_ID") MATCH SIMPLE
        ON UPDATE CASCADE
        ON DELETE CASCADE
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."RECORDS"
    OWNER to postgres;
```

LOGIN PAGE:-

This is the login page. We have a common login page for user and admin as well. User/ Member can put his credentials and login. Same with the admin, Admin can put his credentials and login.

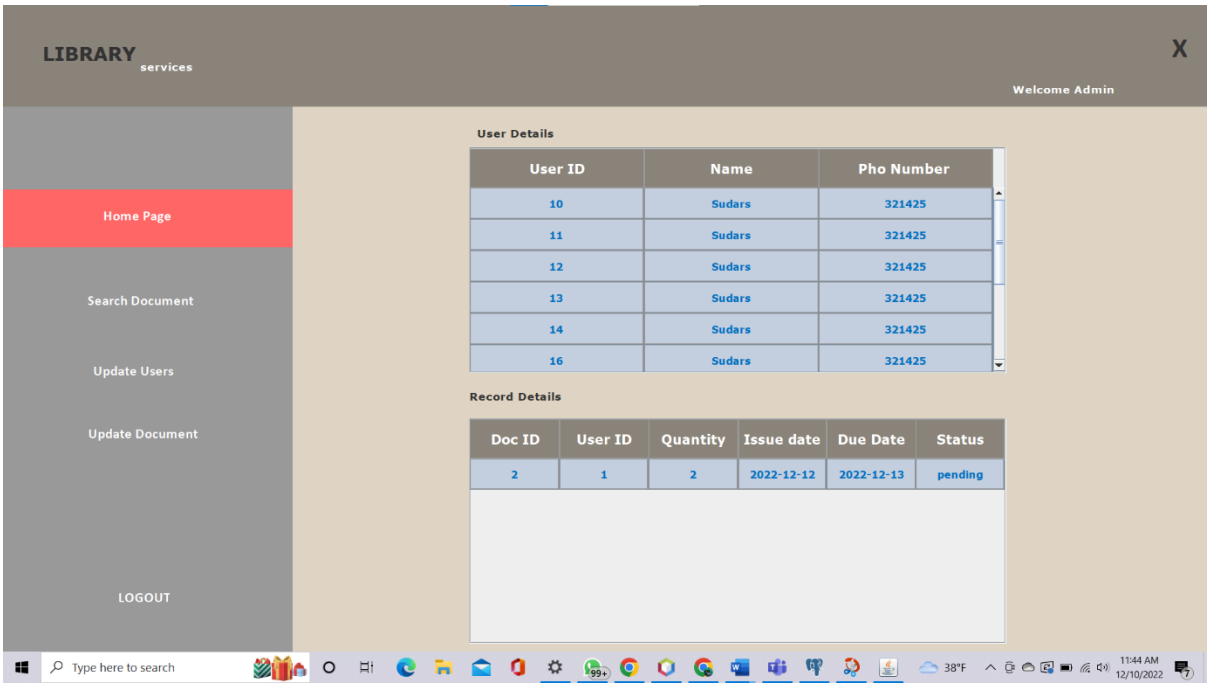


LOGIN SUCCESSFUL:-



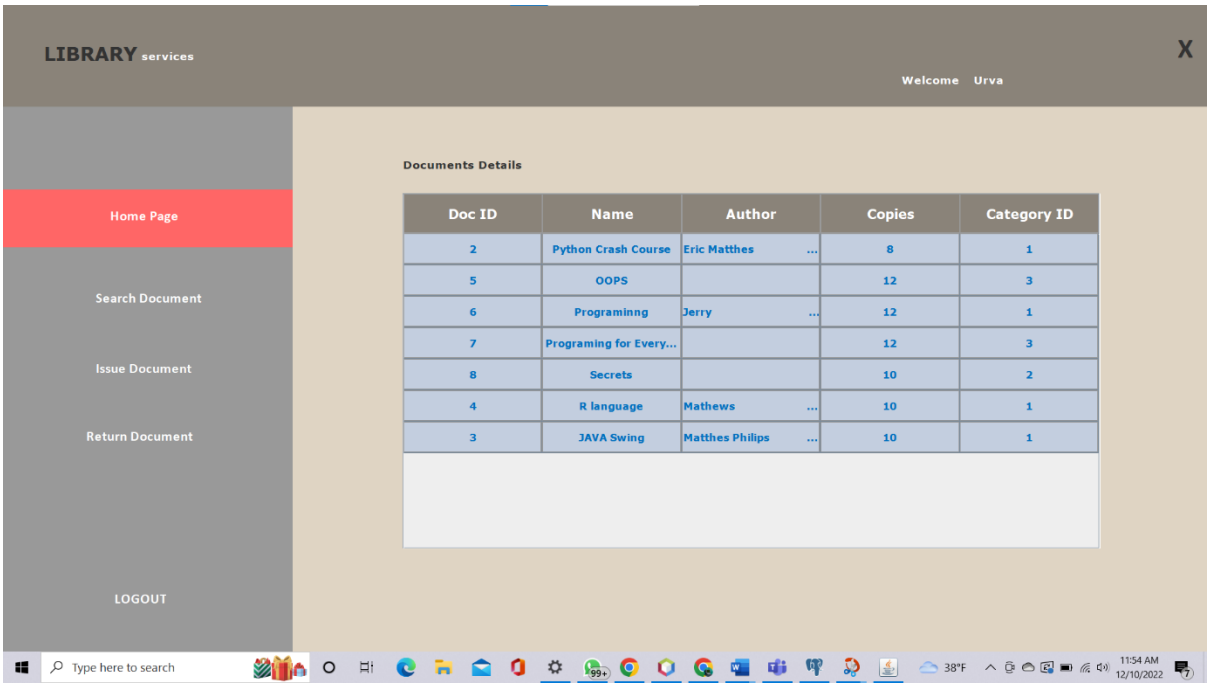
ADMIN HOME PAGE:-

After logging as an admin this is admin homepage. Admin has different privileges. Admin can first of all search for all documents available. After can view user details. In update document we have implemented different functionalities of admin which are, admin can modify a document, admin can add new documents, admin can delete documents.



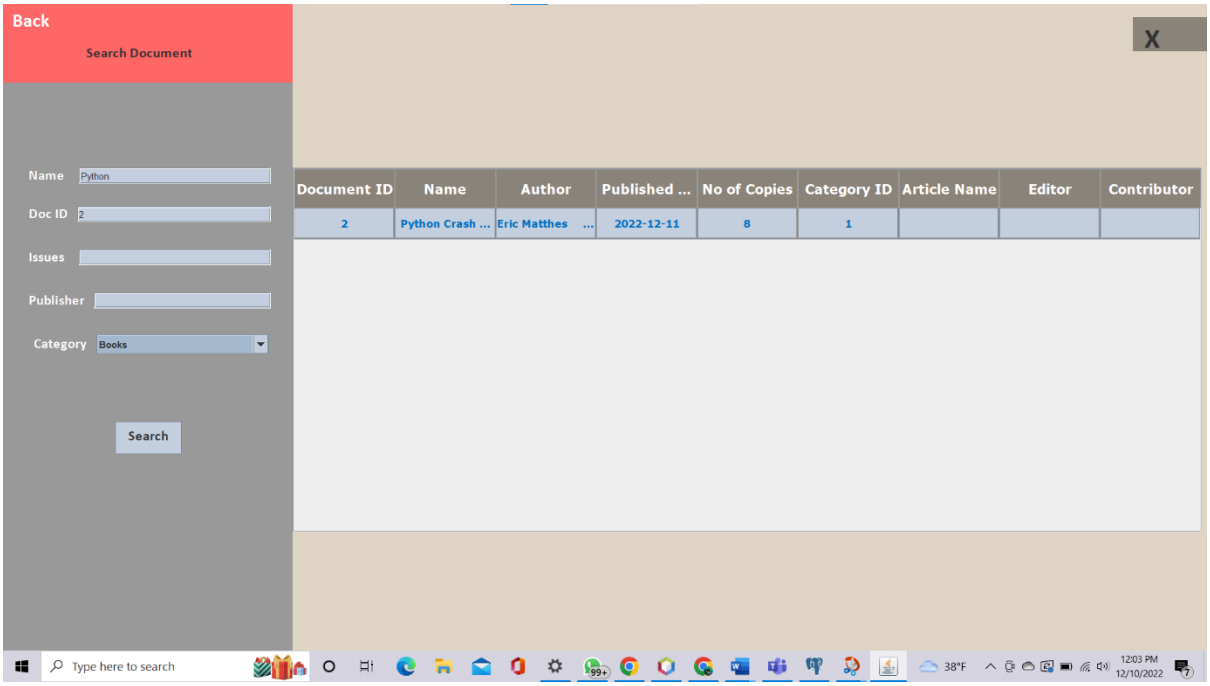
USER OR MEMBER HOME PAGE:-

This is the homepage of the user. A user/member can have overview of documents. A user/member can search for a particular document. User can issue a document and user has to return a document within 2 weeks. If a user/member is not able to return a document within 2 week a penalty will be charged and documents will be shown as pending.

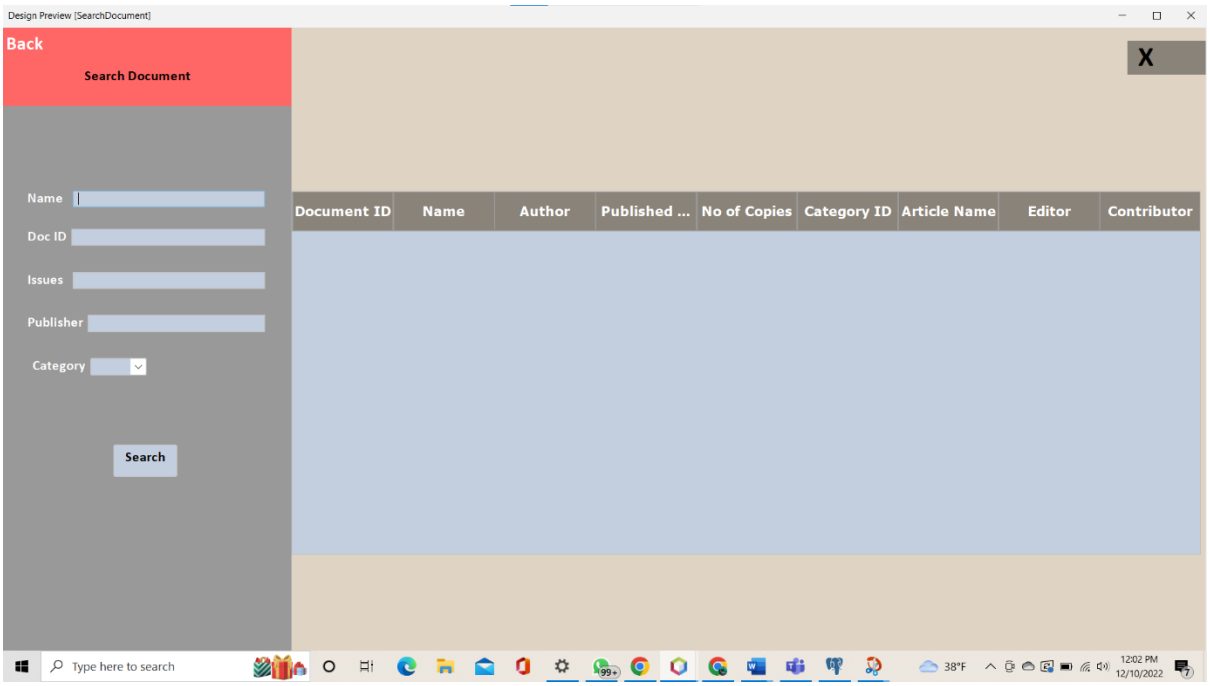


SEARCH DOCUMENT:-

This is how the user/member can search for the documents. User can search via a name or we have a drop down list from category where the user/member can select from a category of books , different categories are books , journals, magazines etc.



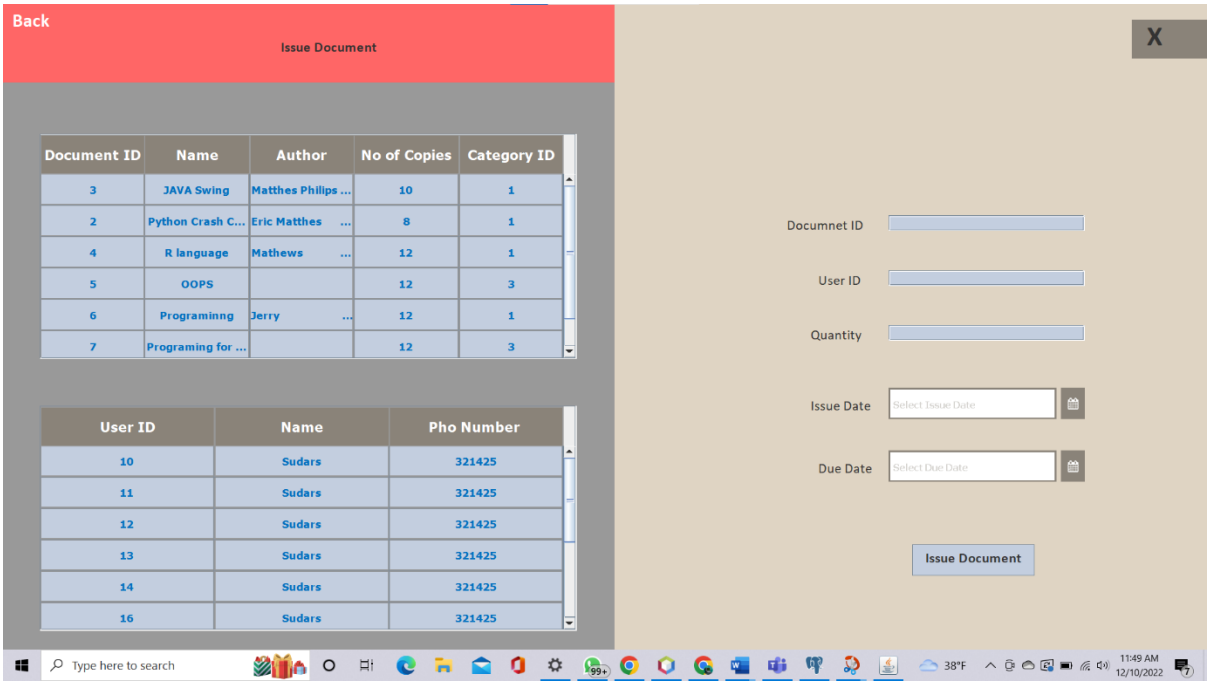
Additionally the user can search the document through the name, DOC ID , ISSUES, publisher category. If a particular document exists it will reflect here.



After searching the name of the book as python you can see that the book is available with author name, published year, no. of copies and category id.

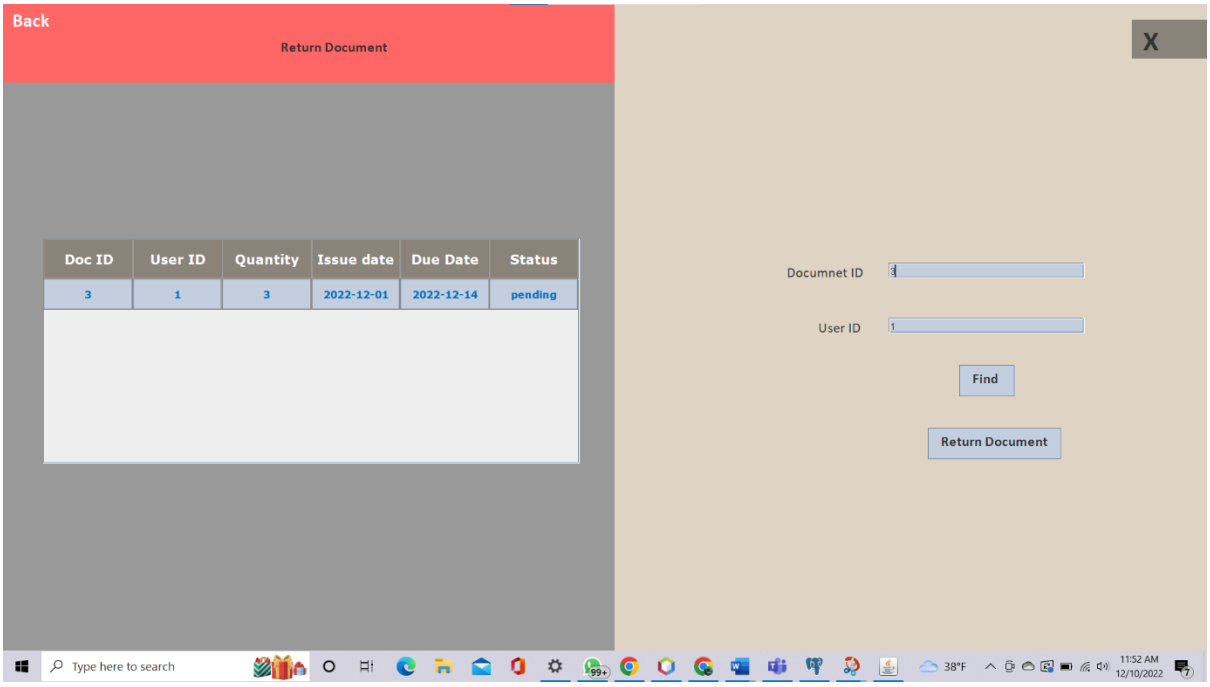
ISSUE DOCUMENT:-

After user can see that the particular document is available, he can go in the issue document section where he is able to issue the document. While issuing the document the user/member has to Provide the document ID, his user id and the issue date. A record will be created in the database. The user/member is bound to return the book within 2 weeks or else he will be penalized. User cannot take 1 document more than once and copies quantity cannot be more than 3 it will throw an error.

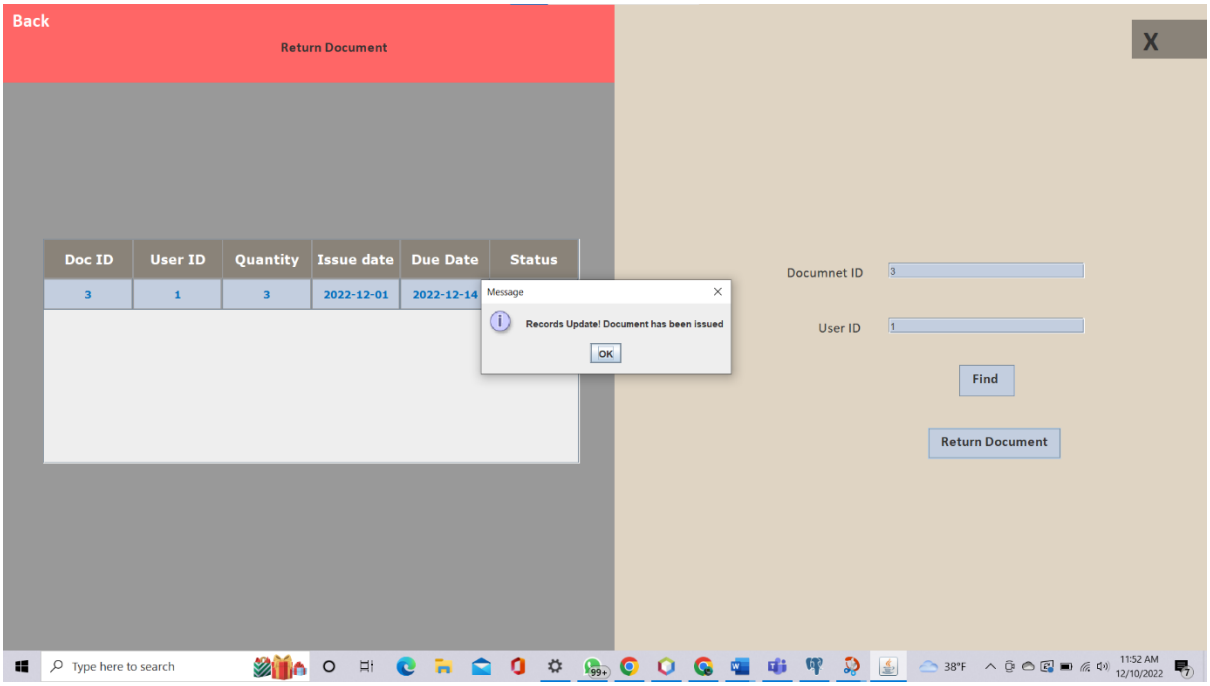


USER CAN RETURN THE DOCUMENT:-

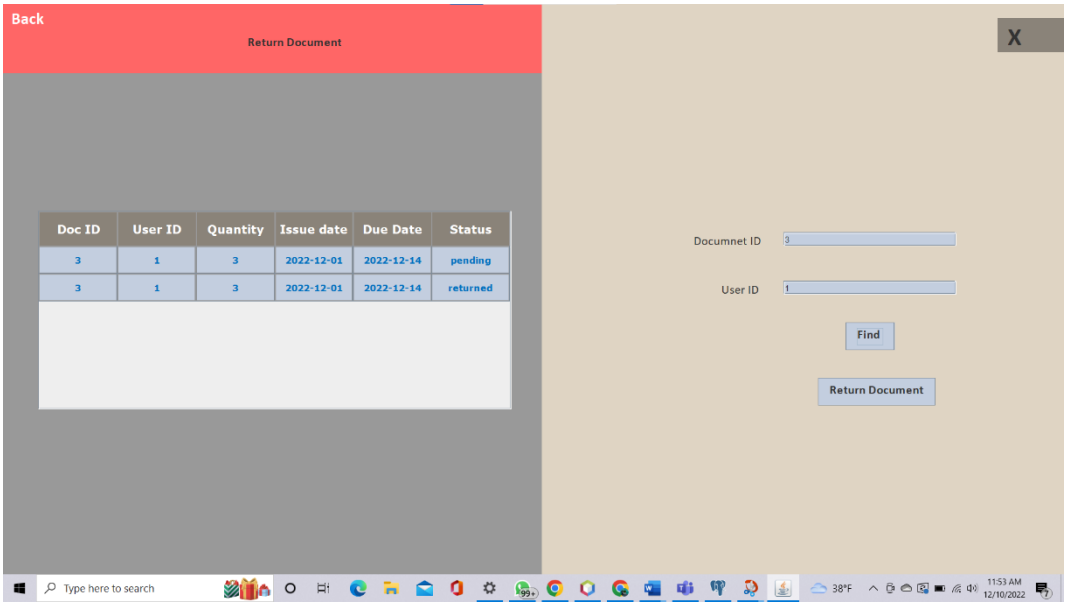
A user/member has to update the return date of the document.



After issuing the book a return date is created in the database. The document which is issued , the quantity of the book or the no. of copies is reduced.

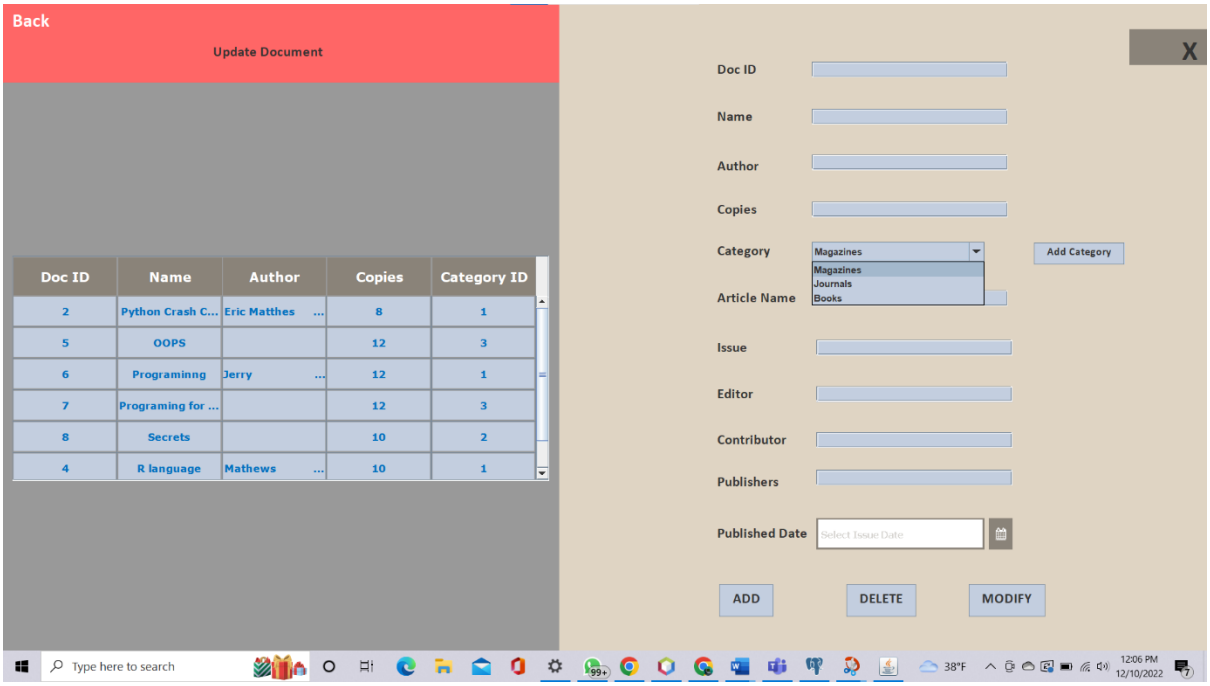


After successfully returning the document you can see that the status of the book changed from pending to returned.

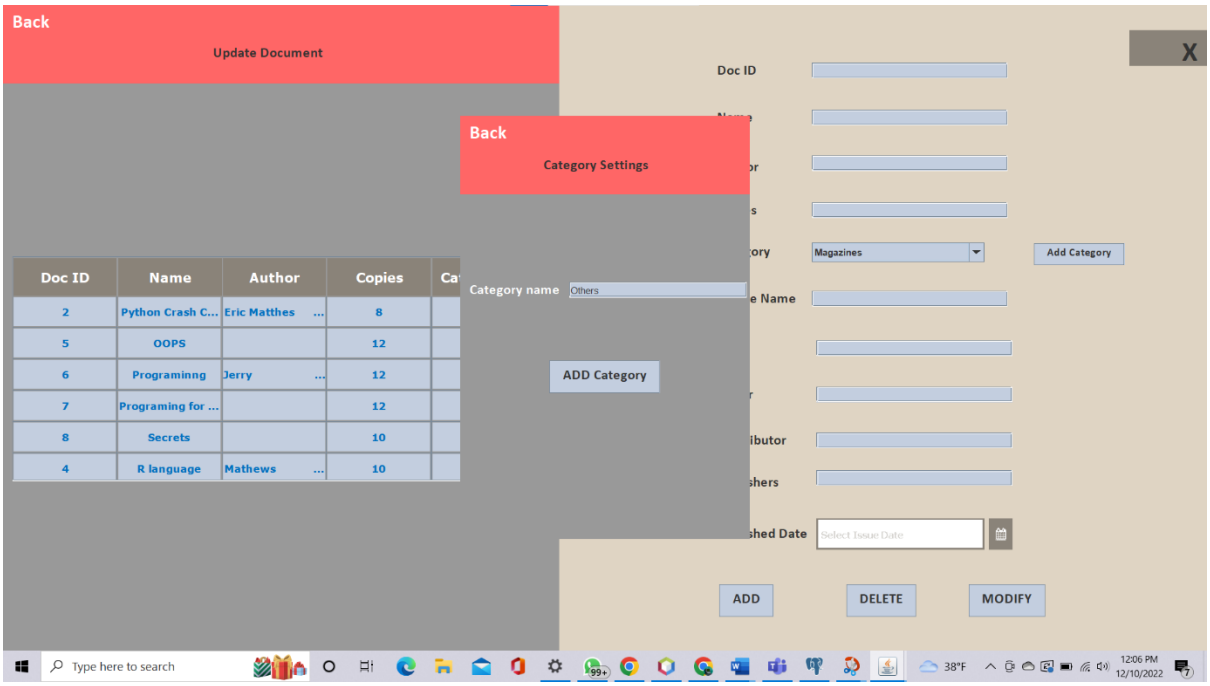


ADMIN CAN ADD A CATEGORY (EXTRA CREDITS):-

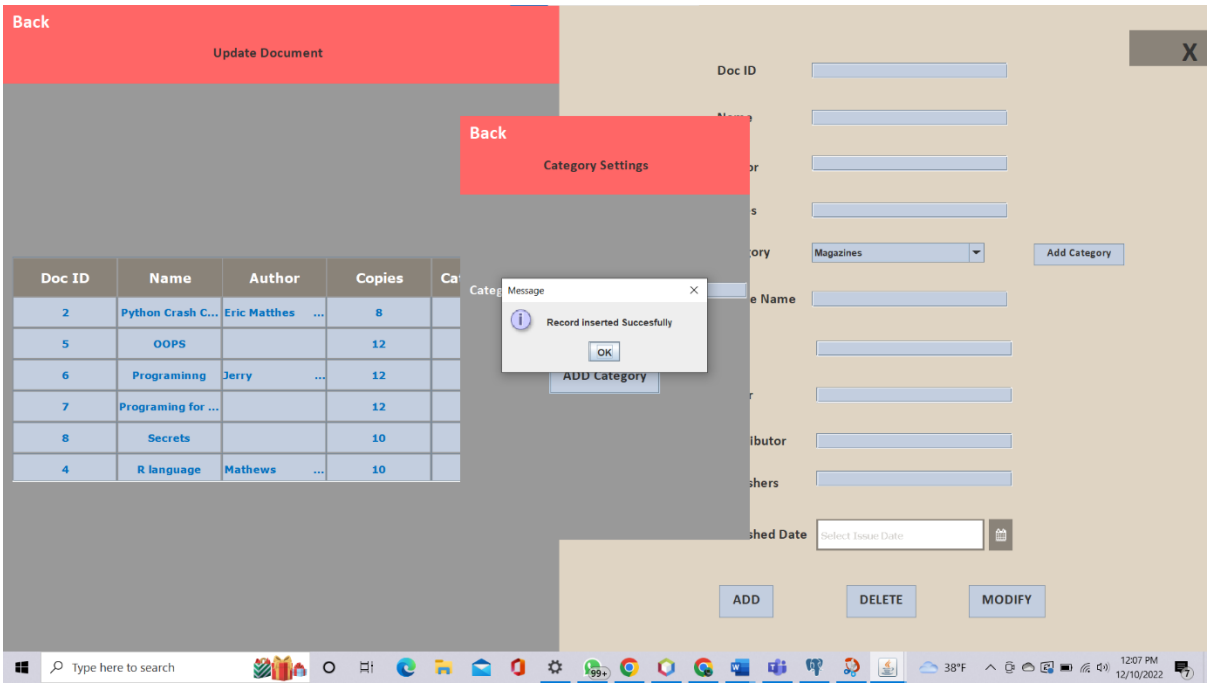
Here the admin can add a new category apart.



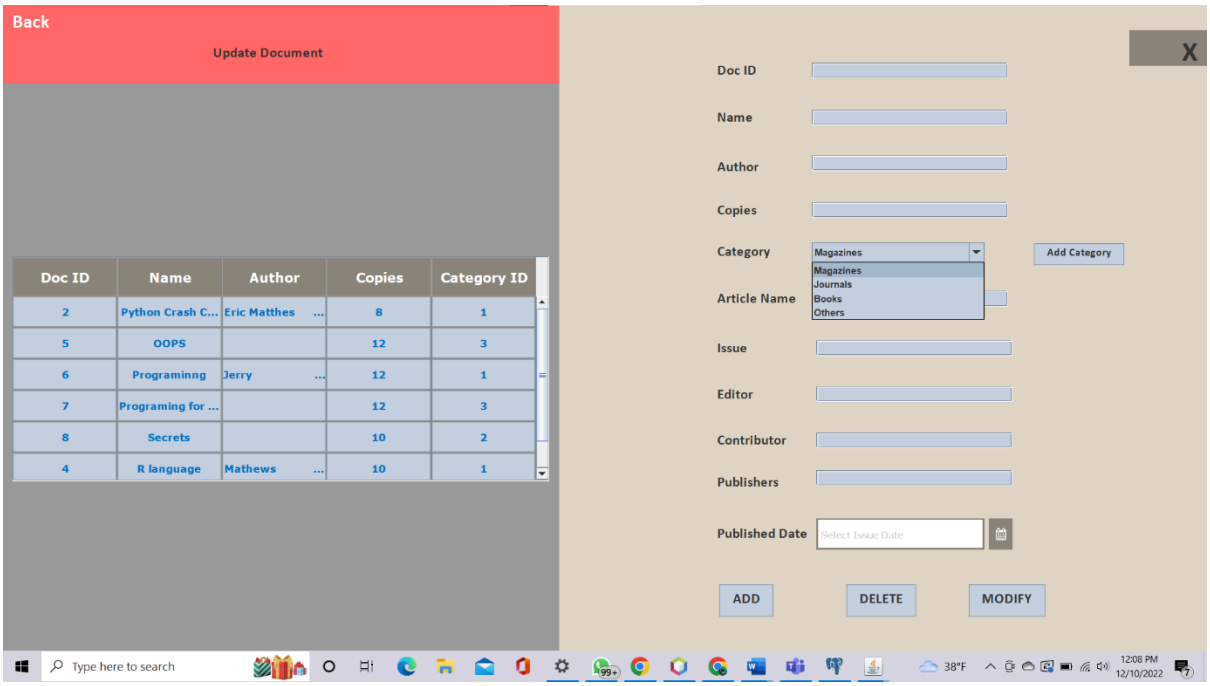
We have created a new category which is named is other and added to the category list.



You can see that the category has been added successfully.

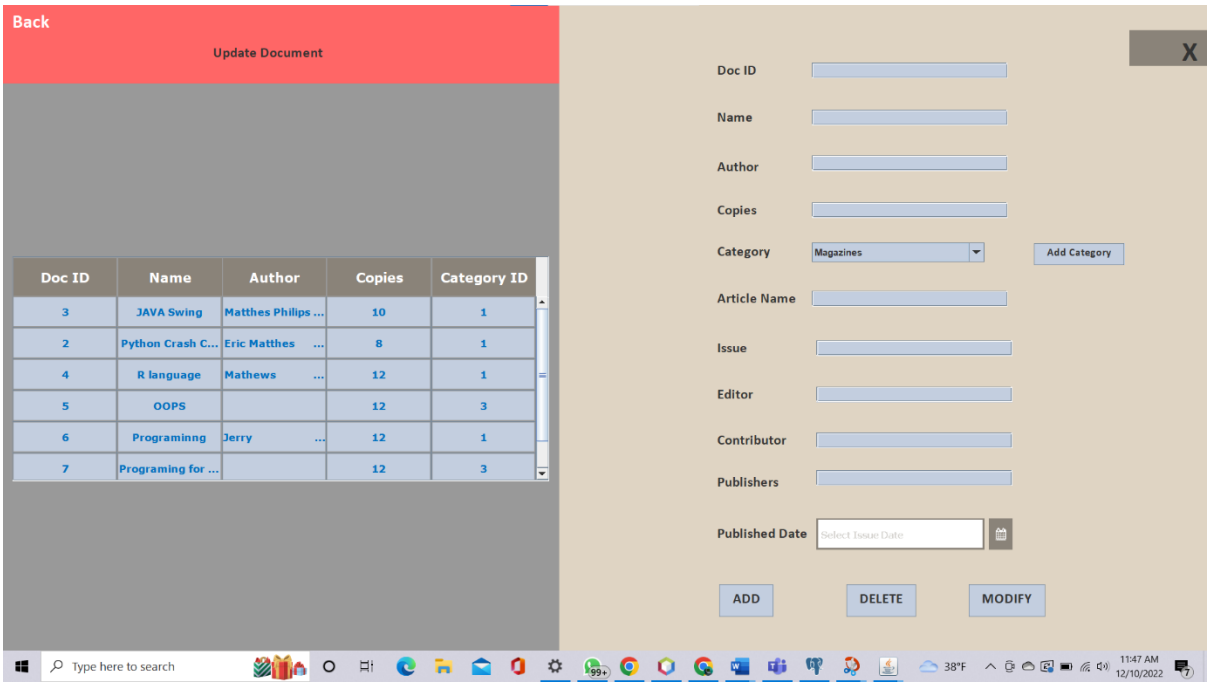


After adding the new category we can see that , in the drop down list there is a new category called as others.



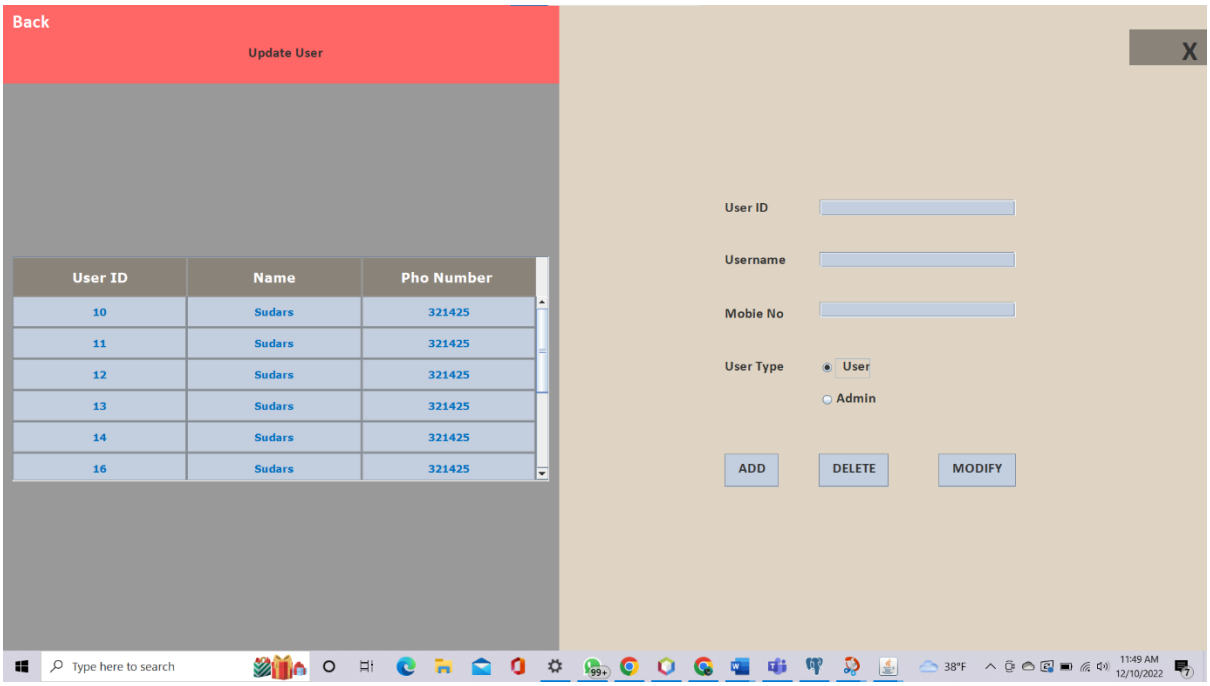
ADMIN CAN UPDATE DOCUMENT:-

Admin can update the documents available . Such as the admin can update the number of the copies.



ADMIN CAN UPDATE USER:-

Admin has the privilege to modify the user. Admin can modify the user such as he can change his mobile no, admin can also add a user or delete a user. Once the new user has been added the default password that user will be 12345678 which he or she can update later on.


Back
Update User
X
User ID
Username
Mobile No
User Type
User
Admin
ADD
DELETE
MODIFY
User ID
Name
Pho Number
10
Sudars
321425
11
Sudars
321425
12
Sudars
321425
13
Sudars
321425
14
Sudars
321425
16
Sudars
321425
Type here to search
38°F
11:49 AM
12/10/2022