DEPI-JAVA Final Project

<u>Library Management System</u>

Team:

- Tasneem Abdeltawab Abdelaziz Kotb   < tasneemabdeltawab205@gmail.com >
- Nadin Mokhtar Elsaid Elbadwy kandil   < nadinkandil19@gmail.com >
- Aya Ebrahim Mohamed Shehawy   < ayai65291@gmail.com   >
- Yasmeen Khaled Mohammed Abd Elhafez < jassmmin2003@gmail.com >

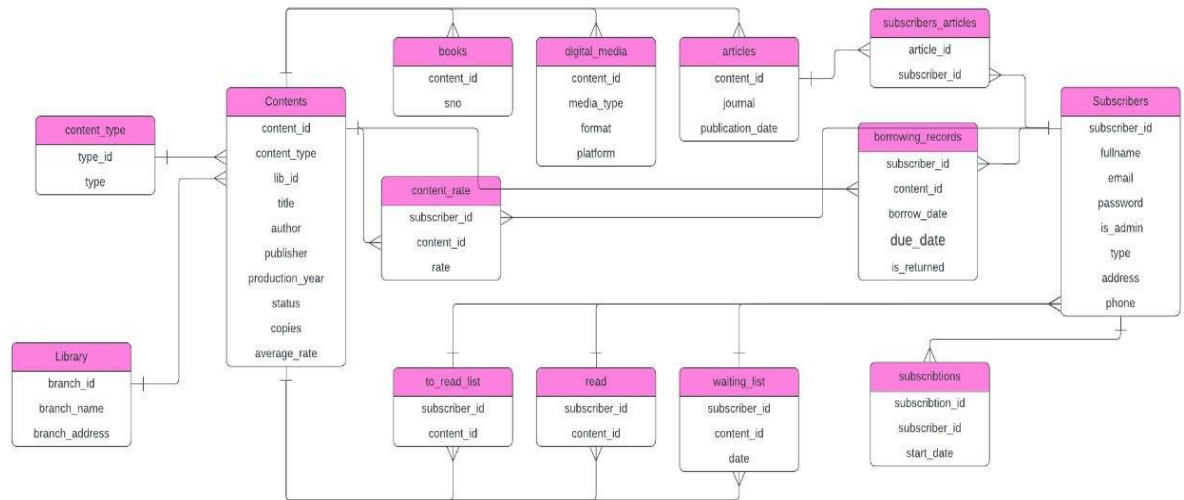**Instructor name:** Dr/Mina Younan

<u>Main scenario:</u>

Library Management System

Develop an application to manage Library borrowing and returns activities. Consider the following information gathered during the analysis process

Design:

- **Basic Classes**: Library, Contents, Books, Articles, Digital Media, Subscribers, Borrowing Records, and have the following relationships:
- **A library** content can be a book, an article in a journal, or digital media.
- **A subscriber** can borrow any of the library contents, for three weeks, golden subscribers can borrow for three months, after the lease time is over, a fee accrue on the subscriber until the borrowed item is returned.
- **A Librarian/administrator** can print a report about all borrowed books and rank them according to audience interest rates, can save/load data on and from files.

- **Database:**

Your extra suggestions are welcome, especially to organize main function, display a menu to the user to:

- Adding/editing/deleting instances belonging to each class,

- Subscriber browsing library contents and select items to borrow.

- Subscriber returning borrowed item and check his balance and pay any late fee if any.

- Admin can print reports of overdue borrowed items.

library.txt

Library ID	Name
1	Main Campus Library
2	CS Library
3	Engineering Library

items.txt

Item ID	Library ID	Category	Title	Author	Publisher	Production Year	Status	Copies
1	1	book	Introduction to Java	Daniel Liang	Pearson	2010	On shelf	10
2	...	...	...	...	...	...	...	...

subscribers.txt

Subscriber ID	Type	Name	Address	Phone	Email
1	regular	Mohamed	...	...	...
2	golden	Ahmed	...	...	...
3	regular	Mostafa	...	...	...

Borrowing.txt

subscriber ID	Borrow Date	content ID	Return Date	Fee
1	10/1/2013	1	...	0
1	10/1/2013	1	...	50
2	12/2/2013	3	...	...
2	13/2/2013	3	...	...

Extra Features:

- Library view made as interface, and UserMainPage and AdminMainPage are made from it.
- All contents can be borrowed using the same function "borrow" (Polymorphism)
- Search for Contents using filters
- Add Contents to wish list "to_read_list"
- Add Contents to read list "the already read books"
- Add content to waiting list to get notified if available.
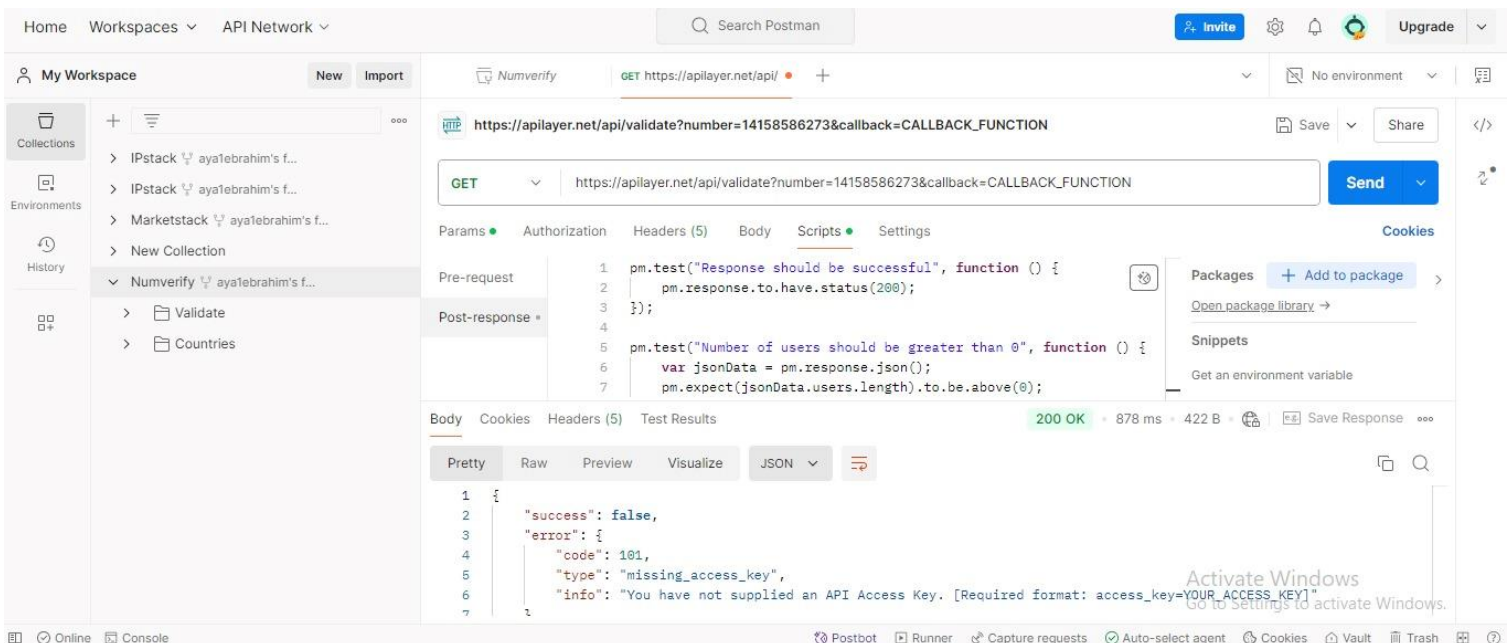- Get notified before the end of the borrowing period by email.

Source code:

Tasneem205/Library_Management_System: Library Management System as a DEPI project (github.com)

note: not all the team used GitHub, so some edits are made by someone and committed by Tasneem205.

API usage:

We tried 3 deferent APIs and tested them using postman

Home   Workspaces ∨   API Network ∨   Search Postman   Invite   Upgrade ∨

My Workspace   New   Import   Overview   Getting started   New Collection   GET https://api.ipstack.com/ •   +   ∨   No environment ∨

Collections

> IPstack aya1ebrahim's f...   https://api.ipstack.com/72.229.28.185,110.174.165.78?language=ru   Save ∨   Share

> IPstack aya1ebrahim's f...

∨ New Collection   GET ∨   https://api.ipstack.com/72.229.28.185,110.174.165.78?language=ru   Send ∨

Environments   This collection is empty
                   Add a request to start working.   Params •   Authorization   Headers (5)   Body   Scripts •   Settings   Cookies

History   Pre-request   1 pm.test("Number of users should be greater than 0", function ()
                                                              2   var jsonData = pm.response.json();
                                            Post-response •   3   pm.expect(jsonData.users.length).to.be.above(0);
                                                              4 });

        Packages   + Add to package
        Open package library →
        Snippets
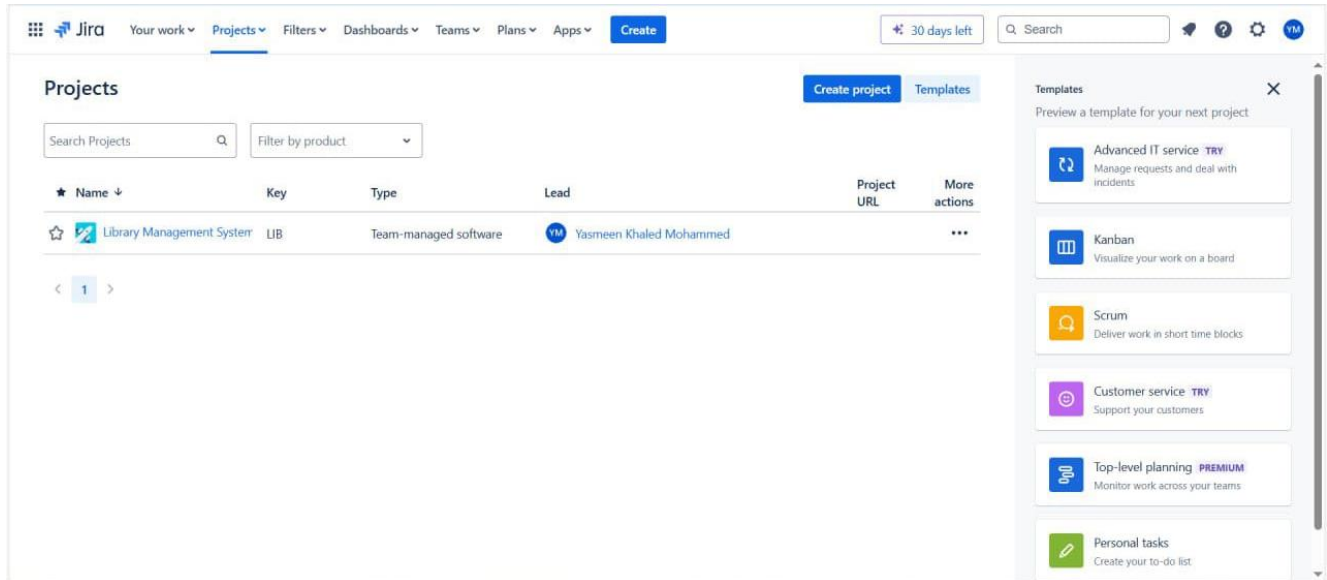   Get an environment variable
        Get a global variable

Body   Cookies   Headers (5)   Test Results (0/1)   200 OK · 902 ms · 422 B   Save Response

Pretty   Raw   Preview   Visualize   JSON ∨

1 {
2   "success": false,
3   "error": {
4   "code": 101,
5   "type": "missing_access_key",
6   "info": "You have not supplied an API Access Key. [Required format: access_key=YOUR_ACCESS_KEY]"

Activate Windows

Online   Console   Postbot   Runner   Capture requests   Auto-select agent   Cookies   Vault   Trash
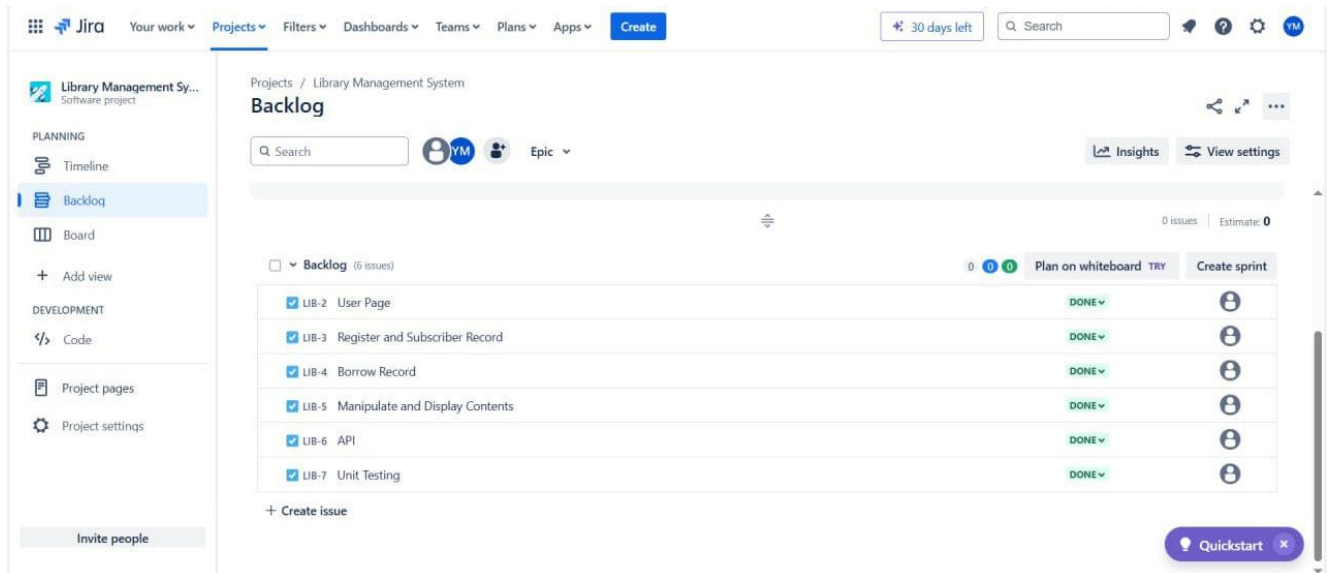

Home   Workspaces ∨   API Network ∨   Search Postman   Invite   Upgrade ∨

My Workspace   New   Import   Marketstack   GET https://api.marketstack. •   +   ∨   No environment ∨

Collections

> IPstack aya1ebrahim's f...   https://api.marketstack.com/v1/eod?symbols=DJI.INDX   Save ∨   Share

> IPstack aya1ebrahim's f...

∨ Marketstack aya1ebrahim's f...   GET ∨   https://api.marketstack.com/v1/eod?symbols=DJI.INDX   Send ∨

  > End-of-Day Data
  > Intraday Data   Params •   Authorization   Headers (5)   Body   Scripts •   Settings   Cookies
  > Real-Time Updates
  > Historical Data   Pre-request   1 pm.test("Response should be successful", function () {
  > Splits Data   2   pm.response.to.have.status(200);
  > Dividends Data   Post-response •   3 });
  > Tickers   4
  > Exchanges   5 pm.test("Number of users should be greater than 0", function () {
  > Currencies   6   var jsonData = pm.response.json();
  > Timezones   7   pm.expect(jsonData.users.length).to.be.above(0);
                                          8 });
> New Collection

        Packages
        Open package library →
        Snippets
        Get an environment variable
        Get a global variable

Body   Cookies   Headers (15)   Test Results   401 Unauthorized · 1236 ms · 1.08 KB   Save Response

Pretty   Raw   Preview   Visualize   JSON ∨

1 {
2   "error": {
3   "code": "missing_access_key",
4   "message": "You have not supplied an API Access Key. [Required format: access_key=YOUR_ACCESS_KEY]"
5   }
6 }

Activate Windows
Go to Settings to activate Windows.

Online   Console   Postbot   Runner   Capture requests   Auto-select agent   Cookies   Vault   Trash

Management tool (JIRA):

We used JIRA to manage the project using agile concepts, here is an example: