

CS 353 - Database Systems

Project Design Report

PURE DIGITAL LIBRARY

Group 4

Kaan Aktürk **-** 21401120

Pelin Elbin Günay - 21402149

Hareem Larik - 21503645

Emine Ayşe Sunar – 21502099

Contents

[**1.** **Revised E/R** 3](#_Toc510441617)

[**2.** **Table Schemas** 6](#_Toc510441618)

[**3.** **Functional Components** 11](#_Toc510441619)

[**3.1 Use Cases/Scenarios** 11](#_Toc510441620)

[**3.1.1 Author** 11](#_Toc510441621)

[**3.1.2 Editor** 11](#_Toc510441622)

[**3.1.3 Reviewer** 12](#_Toc510441623)

[**3.1.4 Subscriber** 12](#_Toc510441624)

[**4.** **User Interface Design and Coresponding SQL Statements** 13](#_Toc510441625)

[**4.1 Sign In Page** 13](#_Toc510441626)

[**4.2 Sign up page** 14](#_Toc510441627)

[**4.3** **Search Page** 15](#_Toc510441628)

[**4.4 View Page** 17](#_Toc510441629)

[**4.5 Profile Page** 19](#_Toc510441630)

[**4.5.1 Author Pages** 20](#_Toc510441631)

[**4.5.1.1 My Papers Page** 21](#_Toc510441632)

**4.5.1.2 Submit Paper Page**…………………………………………………………………….22

[**4.5.2 Editor Pages** 24](#_Toc510441633)

[**4.5.3 Reviewer Pages** 26](#_Toc510441634)

[**4.5.4 Regular User Pages** 28](#_Toc510441635)

[**5.** **Advanced Database Components** 30](#_Toc510441636)

[**5.1** **Constraints** 30](#_Toc510441637)

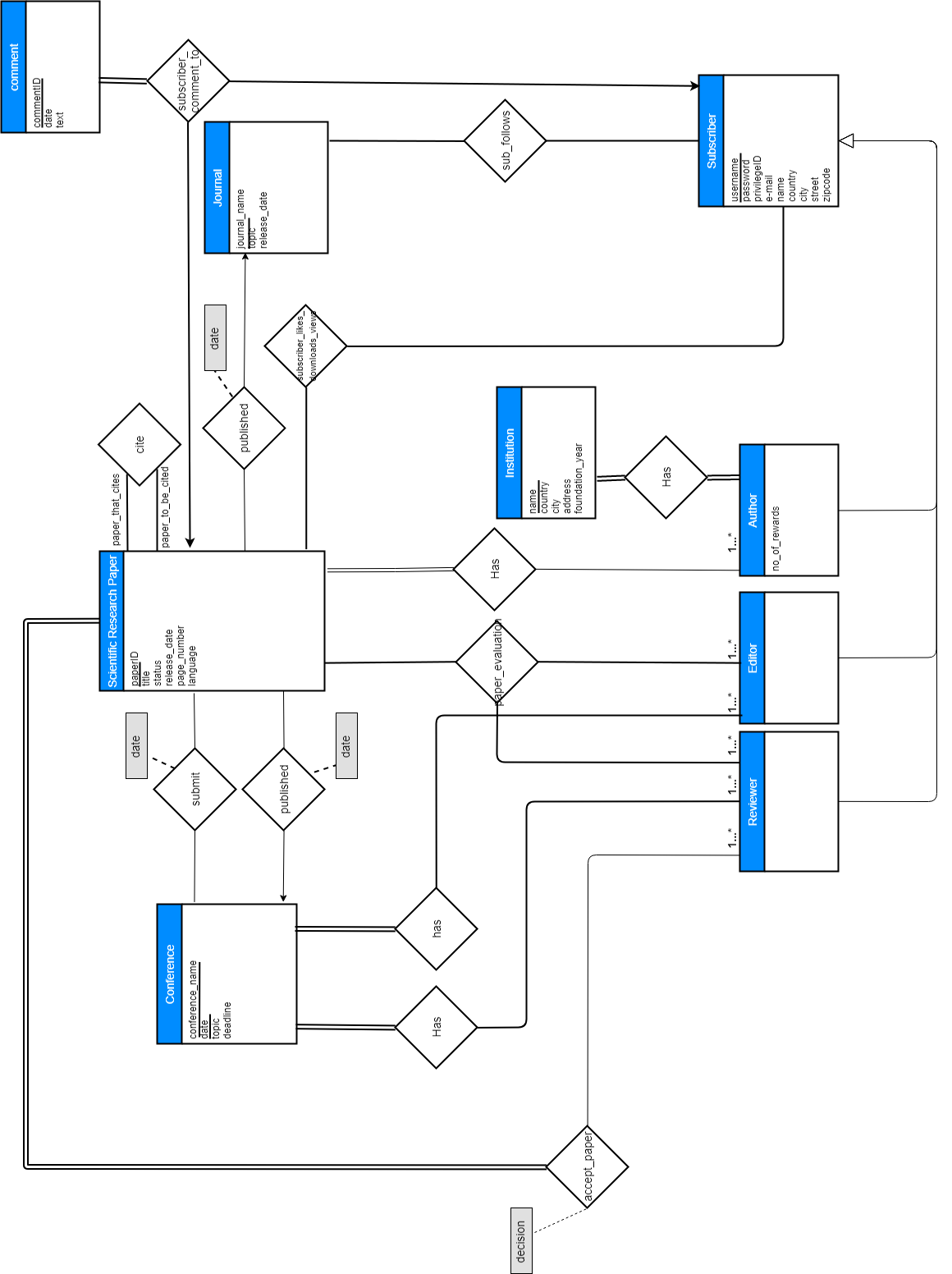
[**5.2 Triggers** 30](#_Toc510441638)

[**5.3 View** 31](#_Toc510441639)

[**5.4 Report** 32](#_Toc510441640)

# **Revised E/R**

Figure 1 Revised Entity Diagram



According to the TA’s feedback, we made the following changes in our ER diagram:

* We removed the person entity and merged it with the subscriber entity. So, we transferred the attributes of the person entity into subscriber.
* We removed the Identity Entity and added this as an attribute named privilegID into the subscriber entity.
* We removed the Teacher and Student entities because they has no specific attributes and were unnecessary.
* Instead of keeping the comment count inside the Scientific Research Paper identity, we added a new entity named comment. Since a subscriber could make only a single comment for a specific paper, we connected the Subscriber, Comment and Scientific Research Paper entities with a ternary relationship where the comment has total participation and the Subscriber and Scientific Research Paper entities have a one to many relation with the Comment entity.
* We removed the Journal Subscriber entity and connected the Journal and Subscriber with a sub\_follows relation. This relation is a many to many relation since a subscriber can follow multiple journals.
* We changed the cardinality between the Institution and the Author entities with full participation because in our database an Author must belong to an institution and an Institution’s information is only kept if an author belong to that institution.
* We added a subscriber\_likes\_downloads\_views relation between the Subscriber and Scientific Research Paper which keeps the information of whether the subscriber liked, downloaded or viewed a paper in single tuple. The relation is many to many because a subscriber could like, download and view many papers.
* We added a date attribute to the published relations between the Scientific Research Paper-conference and Scientific Research Paper-Journal relations. We also added a date attribute to the submit relation between Scientific Research Paper-Conference since the date of these relations are important.
* We added a cite relation to Scientific Research Paper which turns back to itself. This relation is needed because in out library the author will be able to select which papers to cite.
* We removed the Editor Team entity and instead connected the Editor entity with the Conference entity with a relation named has.
* We added a ternary relationship between Scientific Research Paper, Editor and Reviewer. This relation keeps the information of which editor assigned which reviewer to which Scientific Research Paper.
* We removed the Acceptance entity and instead connected the Reviewer to the Scientific Research Paper with a relation named accept\_paper. This relation has an attribute named decision which corresponds to the decision the reviewer makes to a paper.

# **Table Schemas**

**create table** subscriber (

username **varchar**(20) **primary key**,

password **varchar**(20) **not null**,

privilegedID **int**,

email **varchar**(40) **not null**,

name **varchar**(40) **not null**,

country **varchar**(30),

city **varchar**(40),

street **varchar**(40),

zip-code **int**

);

**create table** scientific\_research\_paper (

paperID **int** **primary key**,

title **varchar**(100) **not null**,

status **varchar**(10) **not null**,

page\_number **int** ,

language **varchar**(20),

release\_date **varchar**(10)

);

**create table** journal (

journal\_name **varchar**(40) **primary key**,

topic **varchar**(15),

release\_date **varchar**(10)

);

**create table** conference (

conference\_name **varchar**(40),

date **varchar**(10),

topic **varchar**(15),

deadline **varchar**(10),

**primary key** (conference\_name, date)

);

**create table** paper\_publish\_journal (

paperID **int** **primary key**,

journal\_name **varchar**(40),

publication\_date **varchar**(10),

**foreign key** (journal\_name) **references** journal,

**foreign key** (paperID) **references** scientific\_research\_paper

);

**create table** paper\_publish\_conference (

paperID **int** **primary key**,

conference\_name **varchar**(40),

publication\_date **varchar**(10),

**foreign key** (conference \_name) **references** conference,

**foreign key** (paperID) **references** scientific\_research\_paper

);

**create table** institution (

institution\_name **varchar**(40) **primary key**,

foundation\_year **varchar**(10),

country **varchar**(30),

city **varchar**(40),

street **varchar**(40),

zip-code **int**

);

**create table** subscriber\_comment\_paper (

date **varchar**(10),

text **varchar**(140),

username **varchar**(20),

paperID **int**,

**primary key** (paperID, username),

**foreign key** ( username) **references** subscriber,

**foreign key** (paperID) **references** scientific\_research\_paper

);

**create table** subscriber\_follows\_journal (

username **varchar**(20),

journal\_name **varchar**(40),

**primary key** (username, journal\_name)

);

**create table** subscriber\_likes\_downloads\_views\_paper (

username **varchar**(20),

paperID **int,**

isDownloaded **int**,

isLiked **int**,

isViewed **int**,

**primary key** (username, paperID)

**foreign key (**username**) references** subscriber (username)

**foreign key (**paperID**) references** scientific\_research\_paper (paperID)

);

**create table** author\_institution (

username **varchar**(20),

institution\_name **varchar**(40),

**primary key** (username, institution\_name)

);

**create table** author\_has\_paper (

username **varchar**(20),

paperID **int**,

**primary key** (username, paperID)

);

**create table** paper\_citation (

paper\_that\_cites **int**,

paper\_to\_be\_cited **int**,

**foreign key** (paper\_that\_cites, paper\_to\_be\_cited) **references** scientific\_research\_paper,

**primary key** (paper\_that\_cites, paper\_to\_be\_cited)

);

**create table** paper\_submit\_conference (

conference\_name **varchar**(40),

paperID **int**,

date **varchar(40)**,

**foreign key** (paperID ) **references** scientific\_research\_paper,

**foreign key** (conference\_name) **references** conference,

**primary key** (conference\_name, paperID )

);

**create table** conference\_reviewer (

username **varchar**(20),

conference\_name **varchar**(40),

**primary key** (conference\_name, username)

**foreign key** (conference\_name) **references** conference,

**foreign key** (username) **references** subscriber

);

**create table**  conference\_editor (

username **varchar**(20),

conference\_name **varchar**(40),

**primary key** (conference\_name, username)

**foreign key** (conference\_name) **references** conference,

**foreign key** (username) **references** subscriber

);

**create table**  paper\_evaluation (

reviewer\_username **varchar**(20),

editor\_username **varchar**(20),

paperID **int**,

**primary key** (reviewer\_username, editor\_username),

**foreign key** (reviewer\_username, editor\_username) **references** subscriber

**foreign key** (paperID ) **references** scientific\_research\_paper

);

**create table**  paper\_acceptance (

reviewer\_username **varchar**(20),

paperID **int**,

decision **int**,

**primary key** (username, paperID),

**foreign key** (reviewer\_username) **references** subscriber,

**foreign key** (paperID ) **references** scientific\_research\_paper

);

# **Functional Components**

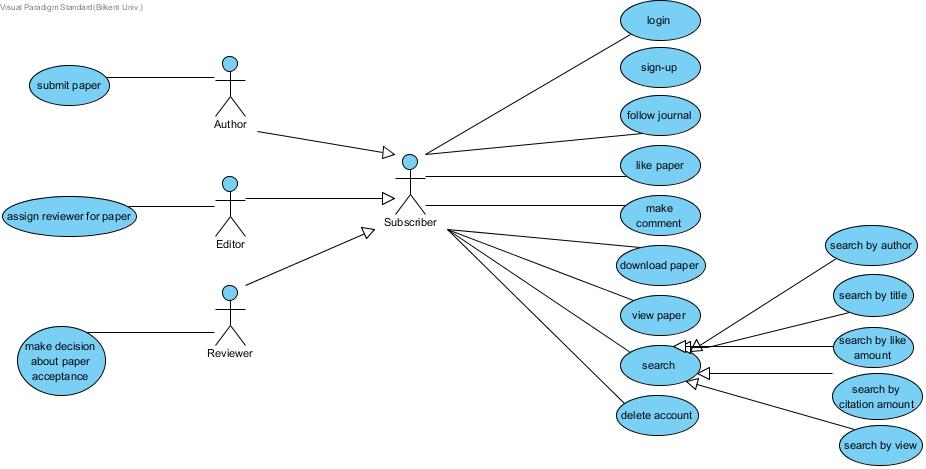


Figure 2 Use Case Diagram

In Pure Digital Library,there are three types of users that are author, editor, reviewer. All of them is a child of subscriber. They have same duties as a subscriber and also other duties that are special for them. Users should register and login to the system to use it.

## **3.1 Use Cases/Scenarios**

### **3.1.1 Author**

* Author can submit paper to the reviewer to be reviewed.

### **3.1.2 Editor**

* Editor can assign the submitted paper by author and decide whether it will be published or not.

### **3.1.3 Reviewer**

* Reviewers should be able to decide whether the source will be published or not.

### **3.1.4 Subscriber**

* Subscriber can login into the library with username, password.
* Subscriber can sign-up into the library with username, password, email, name, country, city, street, zip-code.
* Subscriber can delete his/her library account by clicking delete button.
* Subscriber can follow a journal that if the paper has been published in a journal
* Subscriber can like a paper by clicking the like paper.
* Subscriber can make a comment to the paper.
* Subscriber can download the paper.
* Subscriber can be directed to the paper view page when the user clicks on the link of the paper in the search result page.
* Subscriber can search the paper by searching the author name, by searching title of paper, by searching like amount, by searchingcitation amount, by searching view.

# **User Interface Design and Coresponding SQL Statements**

## **4.1 Sign In Page**

In order to utilize from the library, users have to have library account. The ones who have already an account can enter their username and password to access the library data. Others can get membership by pressing the sign-up button.

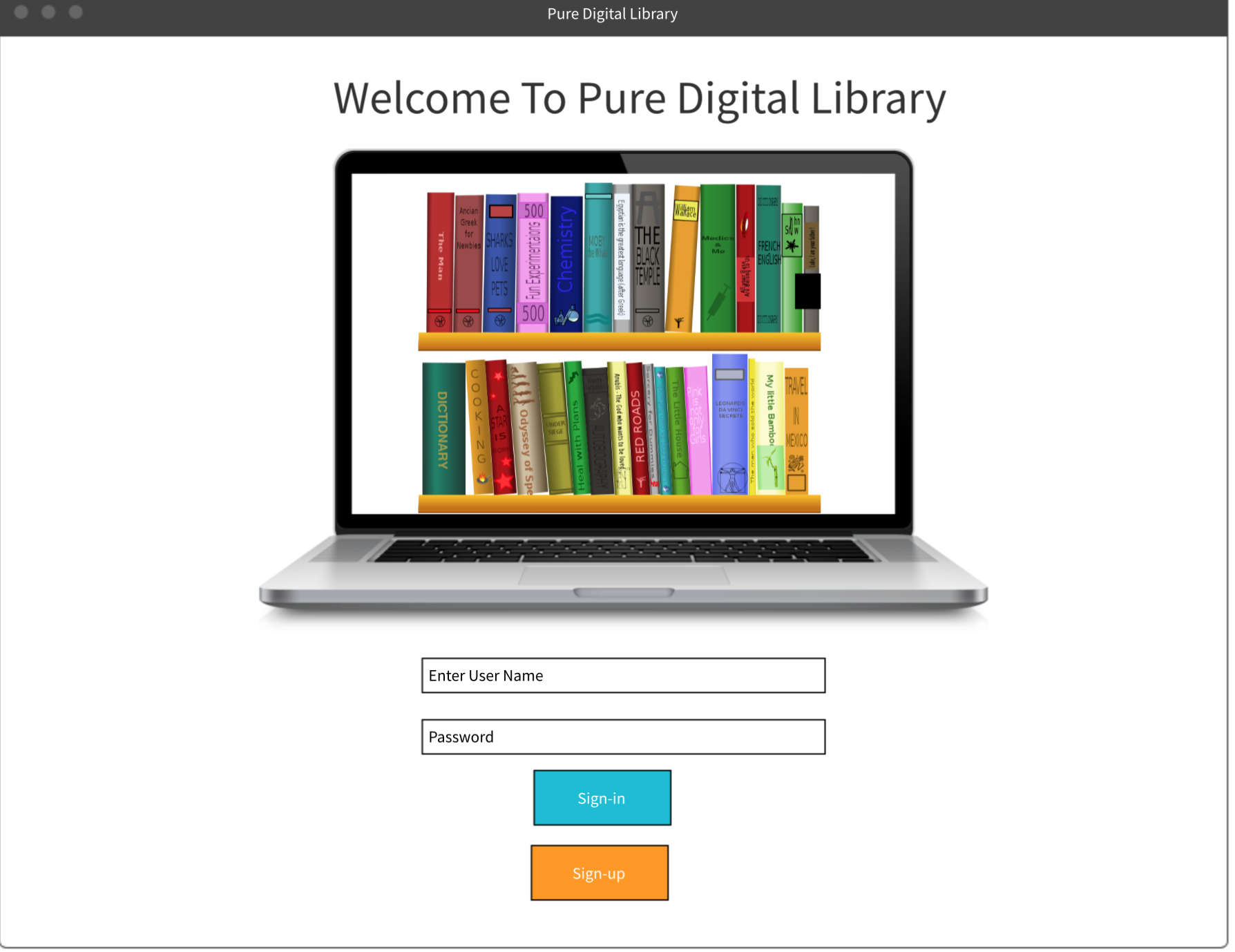
****

Figure 3 Sign In Page

## **4.2 Sign up page**

In this page, users create their library account by entering required information that are displayed below on figure.



Figure 4 Sign Up Page

e.g: username: “Kaan\_Fredrick\_87”, password: “fedo123”, privilegedID: 1 (Author), email: “kaan.fredrick@bilkent.edu.tr”, name = “Kaan Fredrick”, country: “Netherlands”, city: “amsterdam”, street: “Greenway Street”, zip-code: 02341.

When adding a new subscriber:

**insert into** subscriber

**values** (“Kaan\_Fredrick\_87”, “fedo123”, 1, “kaan.fredrick@bilkent.edu.tr”, “Kaan Fredrick”, “amsterdam”, “Greenway Street”, 02341);

## **Search Page**

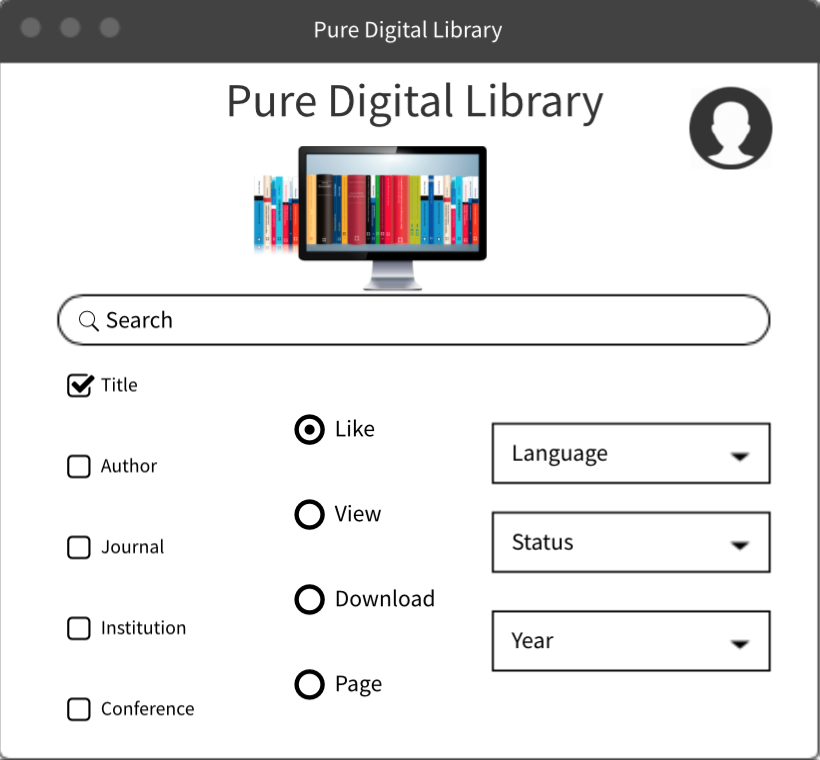
****

Figure 5 Search by title Page

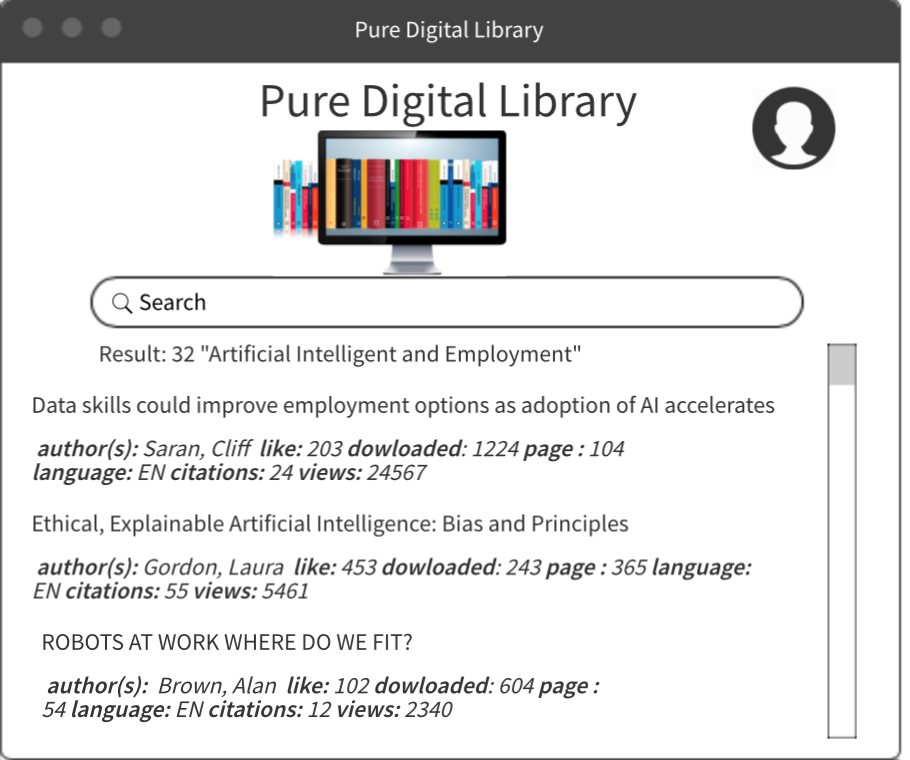
****

Figure 6 Results Page

Search by paper title

Since when searching a paper, according to our user interface design, the information of like number, download number and view number associated with the paper that is searched will be listed. So we must find the like, download and view numbers of the papers. The user could also select filters when searching for papers. The filters would be set to default values and the user will be able to change the filters.

Note: The isLiked, isDownloaded and isViewed are 0 or 1.

input: $title

$filter.language

$filter.status

$filter.year

To get the like, download and view number information of the listed papers according to the filters and the selected order:

**with** paper\_info**(**paperID, like\_no, download\_no, view\_no**) as**

**select** SR.paperID, sum(SR.isLiked) as like\_number, sum(SR. isDownloaded) as download\_number, sum(SR. isViewed) as view\_number

**from** scientific\_research\_paper SR **natural join** subscriber\_likes\_downloads\_views\_paper S

**where** SR.title like %$title% and SR.language = $filter.language and SR.status = $filter.status and SR.release\_date = $filter.year

**group by** SR.paperID **)**

**select** S.title, S.release\_date, P.like\_no, P.download\_no, P.view\_no

**from** scientific\_research\_paper S **natural join** paper\_info P

**order by** like\_no **desc**;

To get the information of the citation numbers of the listed papers:

**with** paper\_citation\_no**(**paperID, citation\_number**) as**

**select** SR.paperID, count(distinct PC. paper\_to\_be\_cited) as citation\_number

**from** scientific\_research\_paper SR, paper\_citation PC

**where** SR.title like %$title% and SR.paperID = PC. paper\_that\_cites and SR.language = $filter.language and SR.status = $filter.status and SR.release\_date = $filter.year

**group by** SR.paperID )

**select** S.title, S.release\_date, P.citation\_number

**from** scientific\_research\_paper S **natural join** paper\_citation\_no P;

## **4.4 View Page**

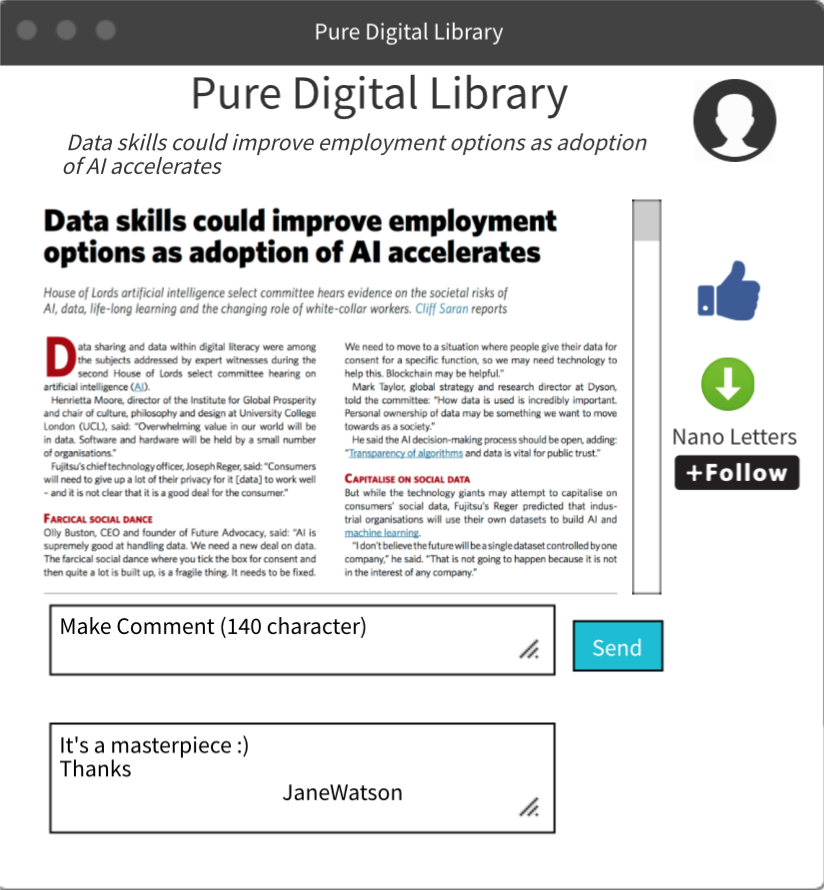
****

Figure 7 View Paper Page

When the user clicks on the link of the paper in the search result page, the user will be directed to the paper view page. After the user has clicked the paper’s link in the result page, the isViewed attribute will be updated to 1 for that specific user and paper in the subscriber\_likes\_downloads\_views\_paper table. Also the isDownloaded and isLiked attributes will be updated to 1 if the user clicks the buttons accordingly. The user will also be able to follow the journal the paper has been published in if the paper has been published in a journal.

The isViewed attribute will be updated to 1 for the tuple containing the information of that specific user and paperID:

**update** subscriber\_likes\_downloads\_views\_paper

**set** isViewed = 1

**where** username = $username and paperID = $paperID

If the user presses the like button, the isLiked attribute will be updated to 1 for the tuple containing the information of that specific user and paperID:

**update** subscriber\_likes\_downloads\_views\_paper

**set** isLiked = 1

**where** username = $username and paperID = $paperID

If the user downloads the paper, the isDownlaoded attribute will be updated to 1 for the tuple containing the information of that specific user and paperID:

**update** subscriber\_likes\_downloads\_views\_paper

**set** isDownloaded= 1

**where** username = $username and paperID = $paperID

when the user follows a journal, a new tuple will be inserted into the table subscriber\_follows\_journal:

**insert into** subscriber\_follows\_journal

**values**($username, $journal\_name)

when the user comments under a paper a tuple will be added to the subscriber\_comment\_paper table:

**insert into** subscriber\_comment\_paper

**values**($date, $text, $username, $paperID)

## **4.5 Profile Page**

The profile page is not related to sql code directly. When the user clicks one of the buttons, the button will direct the user to a new page. A user can delete his or her account. There are different types of users, thus the sql tables related to each user type is different and when deleting these users, they must be deleted from the tables they are being used in.

Common for all users:

**delete from** subscriber

**where** username = $username

**delete from** subscriber\_likes\_downloads\_views\_paper

**where** username = $username

**delete from** subscriber\_follows\_journal

**where** username = $username

**delete from** subscriber\_comment\_paper

**where** username = $username

Deletions specific for authors:

**delete from** author\_institution

**where** username = $username

**delete from** author\_has\_paper

**where** username = $username

Deletions specific for editors:

**delete from** conference\_editor

**where** username = $username

**delete from** paper\_evaluation

**where** editor\_username = $username

Deletions specific for reviewers:

**delete from** conference\_reviewer

**where** username = $username

**delete from** paper\_evaluation

**where** reviewer\_username = $username

### **4.5.1 Author Pages**

If the user has Author privileges, in the profile page, there will be two additional buttons which will direct the users to the following pages:

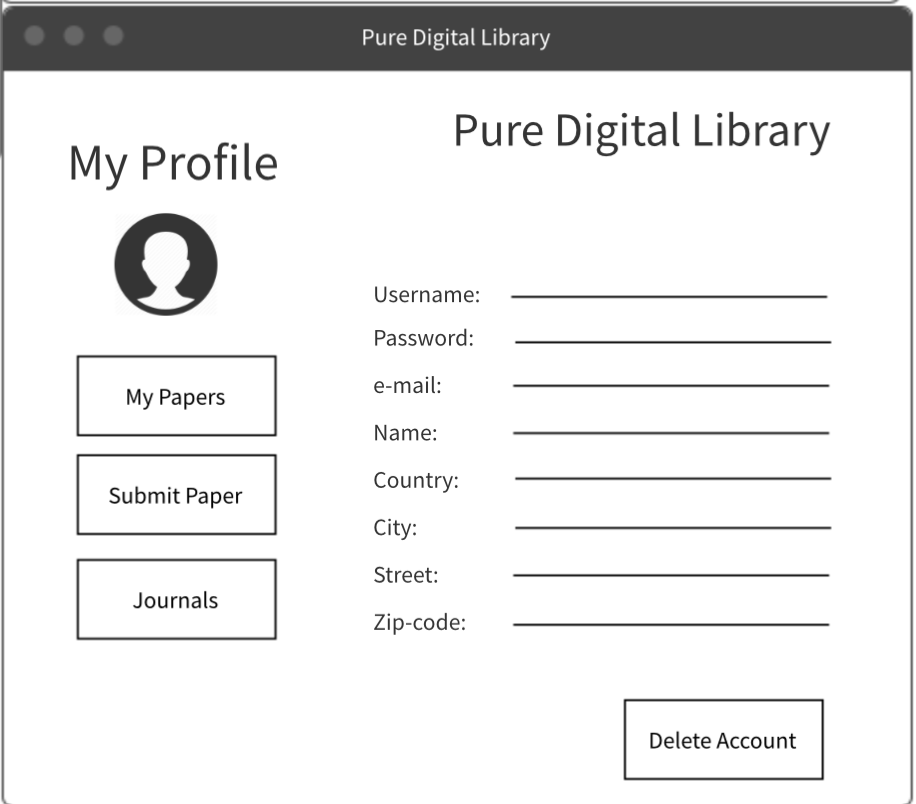


Figure 8 Author Profile Page

### **4.5.1.1 My Papers Page**

The Author will be able to see the papers s/he has written and the like, download and view numbers of each paper separately. Additionally the author will be able to see the usernames of the people who liked her paper.

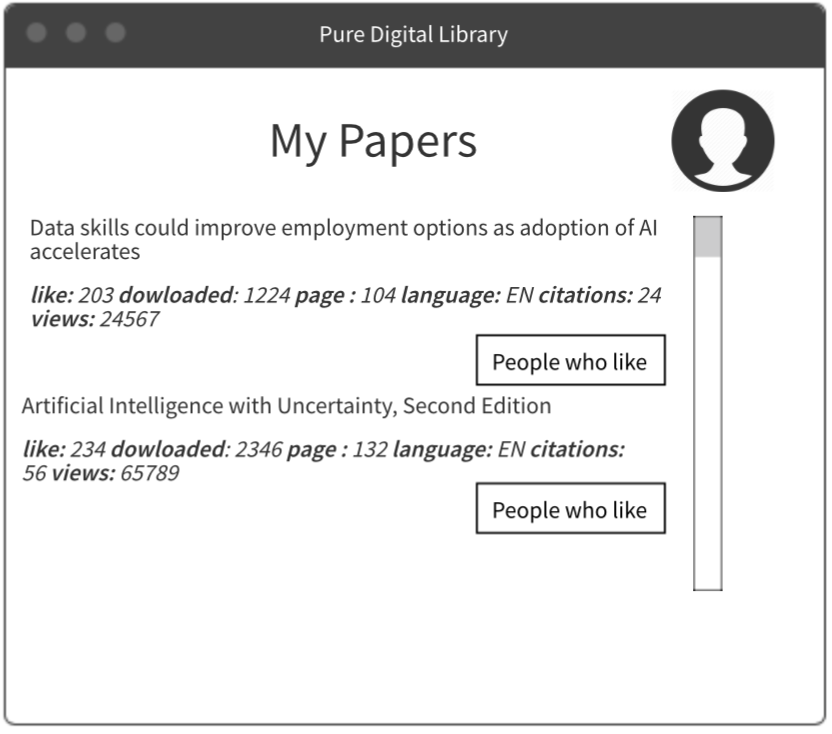


Figure 9 My Papers Page

**with** paper\_info**(**paperID, like\_no, download\_no, view\_no**) as**

**(select** A.paperID, sum(SR.isLiked) as like\_number, sum(SR. isDownloaded) as download\_number, sum(SR. isViewed) as view\_number

**from** author\_has\_paper A **natural join** scientific\_research\_paper SR **natural join** subscriber\_likes\_downloads\_views\_paper S **using** (paperID)

**where** A.username = $username

**group by** A.paperID**)**

**select** S.title, P.like\_no, P.download\_no, P.view\_no, S.status

**from** scientific\_research\_paper S **natural join** paper\_info P

see who liked which paper:

**select** username

**from** subscriber\_likes\_downloads\_views\_paper

**where** paperID = $paperID and isLiked = 1

#### **4.5.1.2. Submit Paper Page**

After the user clicks the submit paper button, s/he will be able to submit a paper to a conference. The author will enter the related information about the paper. Three tables will be modified after the user submits the information. There will be a function called generatePaperID which will generate new paper id’s for every new submitted paper. The status will be initially 0.

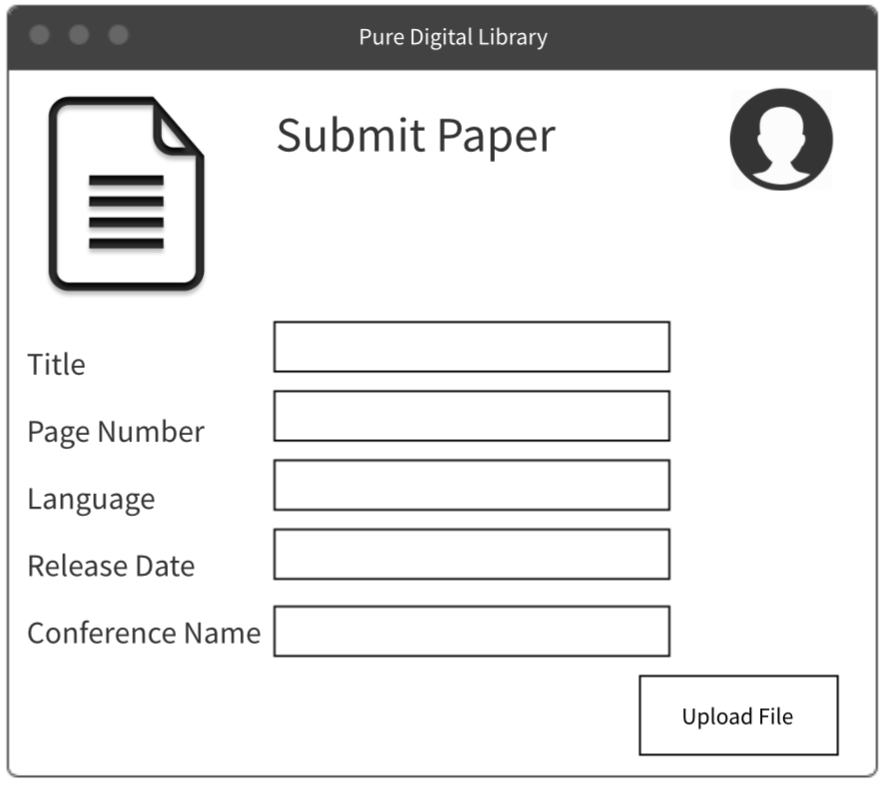


Figure 10 Submit Paper Page

Input: $title, $username, $conference\_name, $page\_number, $language, $release\_date

**insert into** scientific\_research\_paper

**values** ($generatedPaperID, $title, 0, $page\_number, $language, $release\_date)

**insert into** paper\_submit\_conference

**values** ($conference\_name, $generatedPaperID, $date)

**insert into** author\_has\_paper

**values** ($username, $generatedPaperID)

Author can add citation to her paper (The searching and selection of the papers are similar to the process we’ve described in the Search page so we are not writing it here again):

**insert** **into** paper\_citation

**values**($generatedPaperID, $selectedPaperID)

### **4.5.2 Editor Pages**

If the user has editor privileges, in the profile page, there will be an additional button which will direct the editor to the following page:

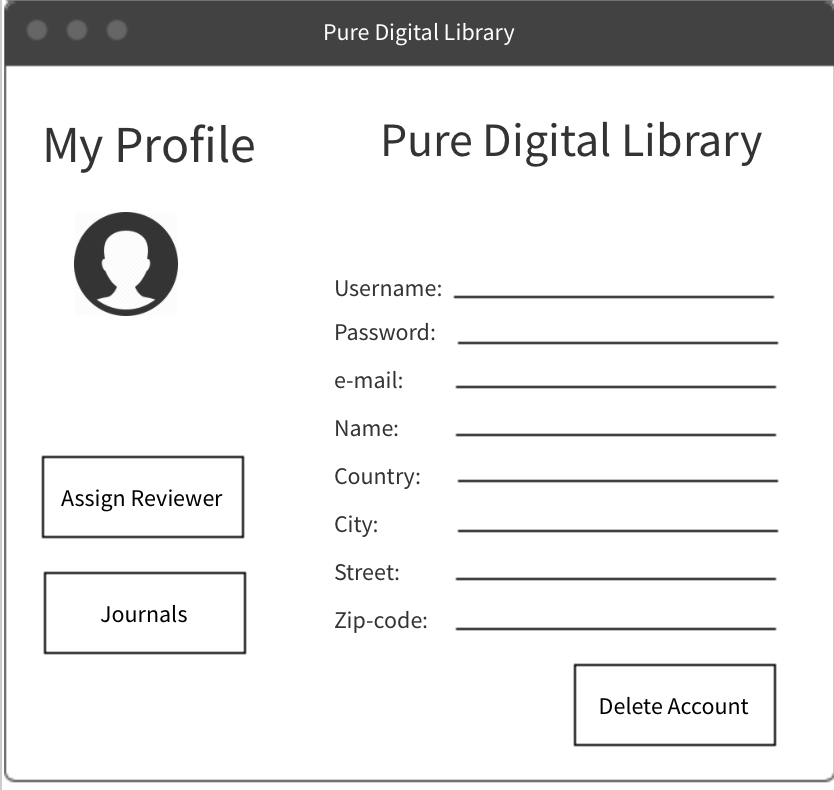
****

Figure 11 Editor Profile Page

#### **4.5.2.1 Select Reviewer Page**

In this page, the editor will be able to assign papers to reviewers. First the editor will be able to see the list of the reviewers according to the conference name selected and then will select a reviewer to assign papers to review. So the information of the list of reviewers are needed and after assigning a reviewer to a paper, a tuple must be added to the paper\_evaluation table.

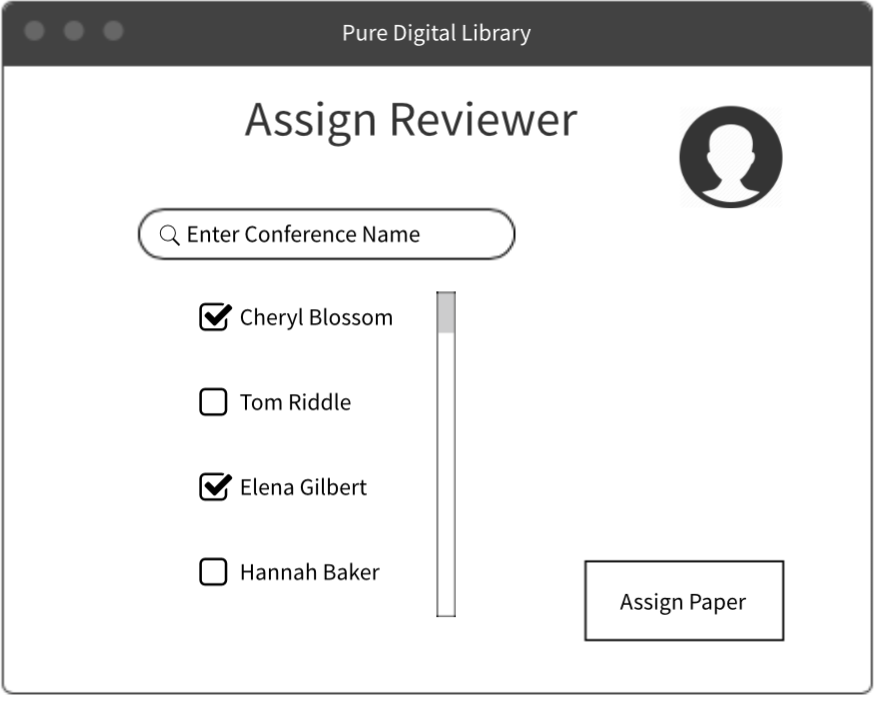


Figure 12 Assign Reviewer Page

List reviewer names :

**select** username

**from** conference\_reviewer

**where** conference\_name = $conference\_name

Insert reviewer to paper\_evaluation table:

**insert into** paper\_evaluation

**values**($reviewer\_username, $editor\_username, $paperID)

### **4.5.3 Reviewer Pages**

If the user has reviewer privileges, in the profile page, there will be an additional button which will direct the reviewer to the following page:

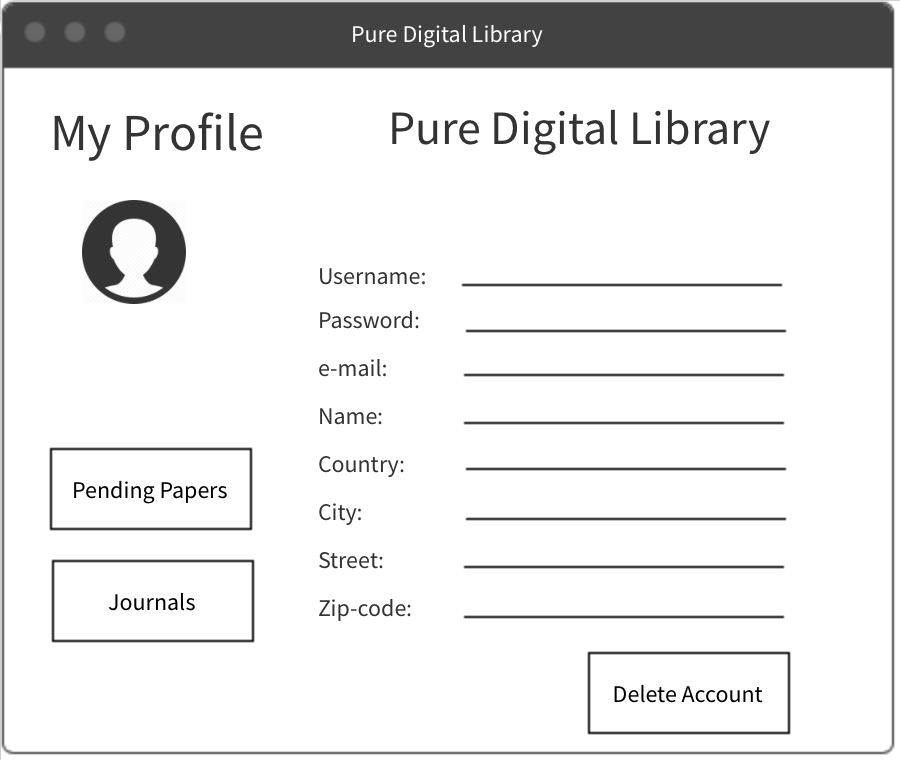


Figure 13 Reviewer Profile Page

#### **4.5.3.1 Pending Papers Page**

In this page the reviewer will be able to see the papers assigned to herself and make a decision about the acceptance of the paper.

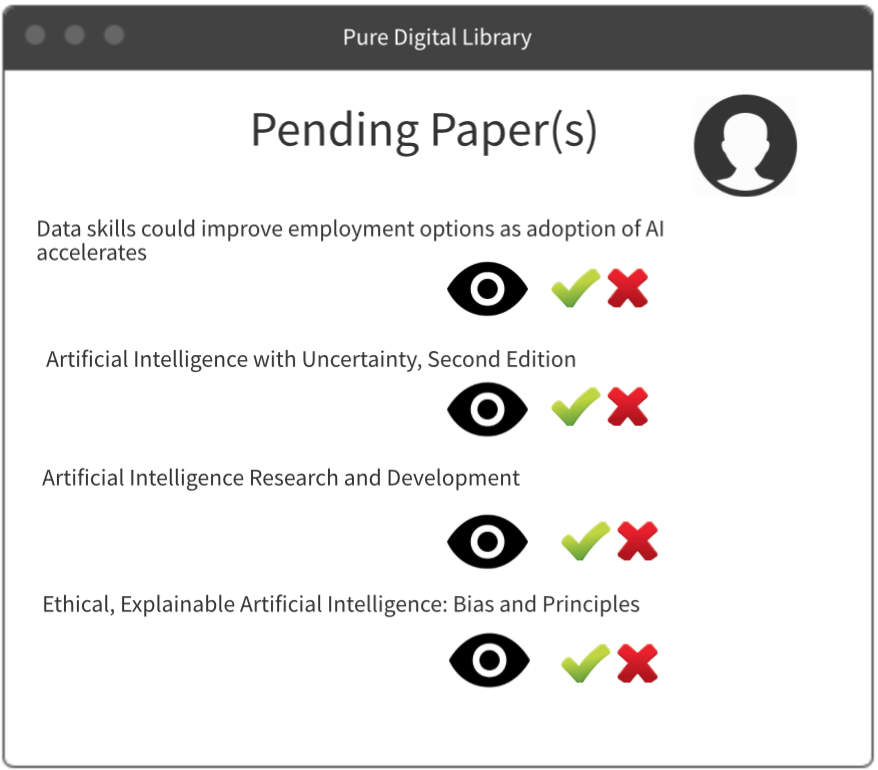


Figure 14 Pending Papers Page for Reviewer

To list the papers assigned to the reviewer we will use this sql query:

**select** S.title

**from** scientific\_research\_paper S **natural join** paper\_evaluation E

**where** E.reviewer\_username = $username

Inserting the reviewer’s decision to the paper\_acceptance table (The decision is 0 for rejection or 1 for acceptance):

**insert into** paper\_acceptance

**values**($username, $paperID, $decision)

After the decision has been made, the paper will be deleted from the pending papers list:

**delete** **from** paper\_evaluation

**where** reviewer = $username **and** paperID = $paperID

### **4.5.4 Regular User Pages**

This profile has no additional operations, therefore it is for the regular users which are neither author, editor nor reviewer. There is a single button where the user could see the journals s/he subscribed. This operation is also available for the other user types.

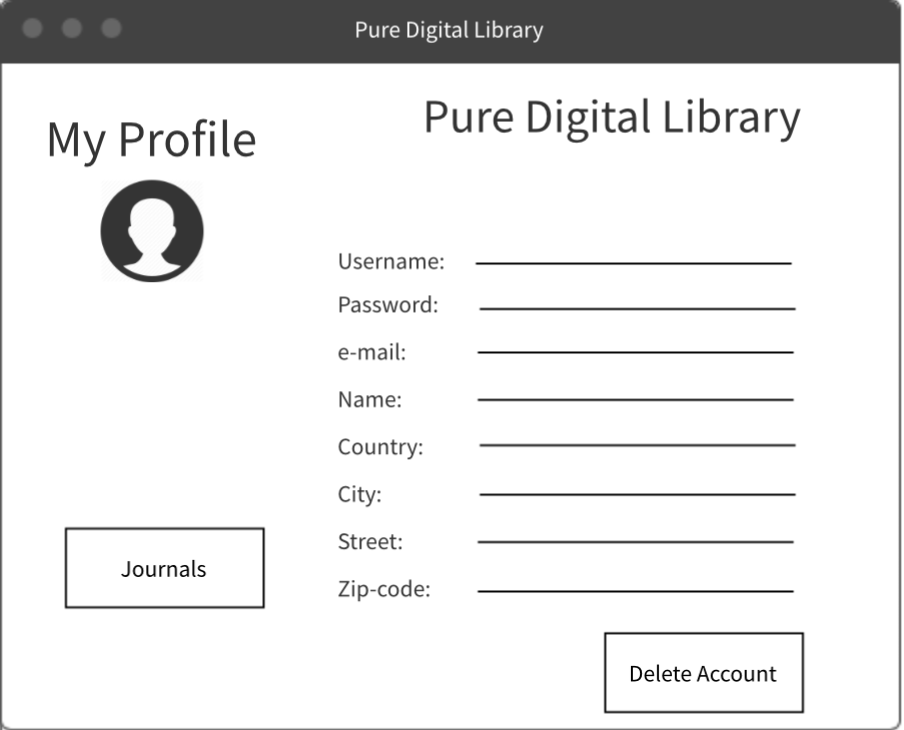


Figure 15 Regular user profile page

#### **4.5.4.1 Subscribed Journals Page**

The user will be able to see the list of the journals s/he subscribed to.

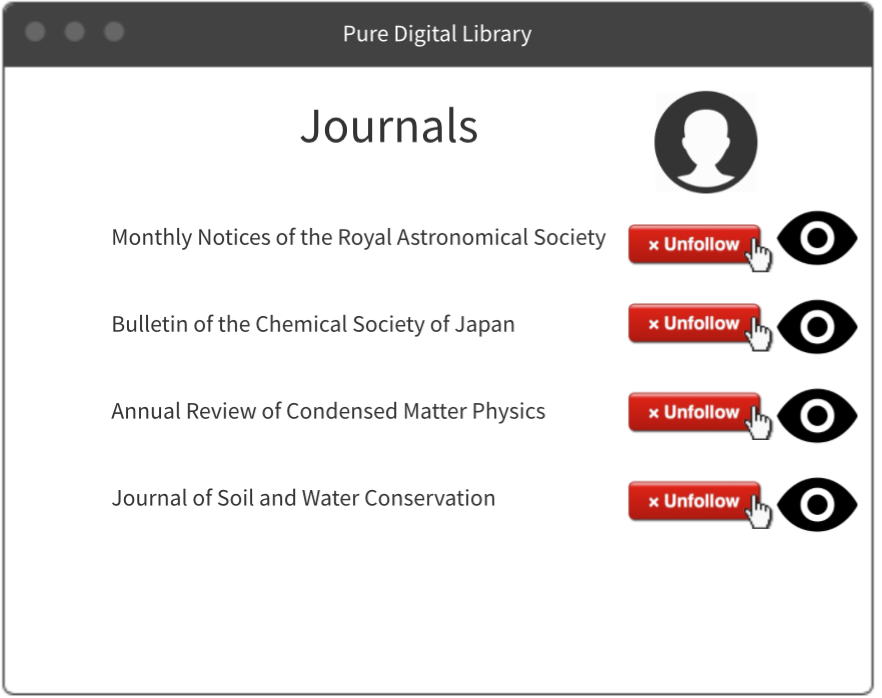


Figure 16 Followed Journals Page

To list all of the journals the user is following we will use this sql query:

**select** jounal\_name

**from** subscriber\_follows\_journal

**where** username = $username

If the user selects the unfollow button then the tuple holding the information of the subscriber following that journal will be deleted from the table subscriber\_follows\_journal:

**delete** **from** subscriber\_follows\_journal

**where** username = $username

# **Advanced Database Components**

## **5.1** **Constraints**

* Every user has a unique username.
* User can like an article only once.
* Every author must be associated with an institution.
* Read count of an article will only increment when the user views that article for the first time.
* A comment can only be 140 characters.
* User can only add a single comment for an article, in order not to spam the comment section.
* Services of the system cannot be utilized without logging into the system
* The privileged ID of subscribers will determine their roles accordingly, which are author, reviewer, editor or regular user.
* Deadline of the conference must be earlier than the date of conference.
* The paper can only be published in a single conference or journal.

## **5.2 Triggers**

* When an author submits a paper to a conference, a new tuple will be inserted to the author\_has\_paper, paper\_submit\_conference and scientific\_research\_paper tables. Also during submission the papers selected by the author as cited papers will be added to the table paper\_citation.
* The number of yes and no decisions will be calculated from the paper\_acceptance table and if the yes number is bigger than the no number the paper’s status will be changed to published and this information will be added to the paper\_publish\_conference.
* When a user subscribes to a journal, the user’s information will be added to the subscriber\_follows\_journal.
* When a comment is made, the information of the comment is added to the subscriber\_comment\_paper
* If the user likes/downloads/views a paper, then the related tuple will be updated, accordingly.
* When an editor assigns a paper to a reviewer, the information will be added as a new tuple into the paper\_evaluation table. Also, when the reviewer makes a decision about a paper then that decision will be added to the paper\_acceptance.
* When the reviewer finalizes his/her decision the tuple containing the information about the assignment will be removed from paper\_evaluation table.

## **5.3 View**

* An editor can only see the list of the reviewers of the conference he/she affiliated to. This view will be used in the Select Reviewer Page which is a specific page for editors only. The goal of this view is to be able to list the reviewer names for the editor. After viewing the list, the editor could select which reviewers to assign for a paper. This view is necessary because it would be irrelevant for an editor to be able to see editors from conferences the editor does not belong to.

**create view** editor\_list\_reviewer **as**

**select** username

**from** conference\_reviewer

**where** conference\_name **in** (**select** conference\_name

**from** conference\_editor)

* An Author can see the usernames of the people who liked his/her papers and other user types cannot see which user liked which paper. Also other authors cannot see the usernames of the users who have liked other authors’ papers. So being able to see the usernames who have liked a paper is a privilege given to the author of that specific paper. This view is used in the “My Papers Page”. When the author clicks the “people who like” button s/he will be able to see the usernames who have liked the paper.

**create view** people\_who\_liked **as**

**select** username

**from** subscriber\_likes\_downloads\_views\_paper

**where** isLiked = 1 **and** paperID **in** (**select** paperID

**from** author\_has\_paper)

## **5.4 Report**

* The first, second and third highest downloaded papers’ Id’s:

Download number of each paper:

**with** temp(paperID, download\_count) **as**

(**select** paperID, count(\*)

**from** subscriber\_likes\_downloads\_views\_paper

**group by** paperID)

Highest downloaded paper:

**select** paperID

**from** temp

**where** donwload\_count = (**select**  **max**(download\_count)

**from** temp)

Second highest downloaded paper:

**select** paperId

**from** temp T1

**where** (**select** count(\*)

**from** temp T2

**where** T2.download\_count > T1.download\_count) = 1

Third highest downloaded paper:

**select** paperId

**from** temp T1

**where** (**select** count(\*)

**from** temp T2

**where** T2.download\_count > T1.download\_count) = 2

* The accepted paper count of each author:

Note: There are four status for the papers in out database.

0 → submitted

1 → on review (pending paper)

2 → accepted

3 → rejected

**select** username, count(\*) as accepted\_paper\_count

**from** author\_has\_paper A **natural join** scientific\_research\_paper S

**group by** A.username

**having** S.status = 2

* The total number of papers in each state:

**select** count(\*) as status\_count

**from** scientific\_research\_paper

**group by** status