## Table of contents:

1. Revised E/R diagram
2. Relational schemas

2.1 Institution

2.2 Researcher

2.3 Author

2.4 Editor

2.5 Reviewer

2.6 Publication

2.7 References

2.8 Events

2.9 Work

2.10 Write

2.11 Manage

2.12 Assign

2.13 Review

2.14 Conference

2.15 Journal

2.16 Attend

2.17 Subscriber

2.18 Subscribe

2.19 Issue

2.20 Submit

1. Fuctional Dependencies and Normailsation
2. Functional Components

4.1 Use Case/scenarios

4.1.1 Subscriber

4.1.2 Author

4.1.3 Reviewer

4.1.4 Editor

4.2 Algorithms

4.2.1

4.2.2

1. User Interface
2. Advanced Database Components

6.1 Reports

6.1.1 Total number of publications submitted

6.1.2 Total number of published publications

6.1.3 Number of reviews

6.1.4 number of subscribers

6.2 Views

6.2.1 Journal listings

6.2.2 Author listings

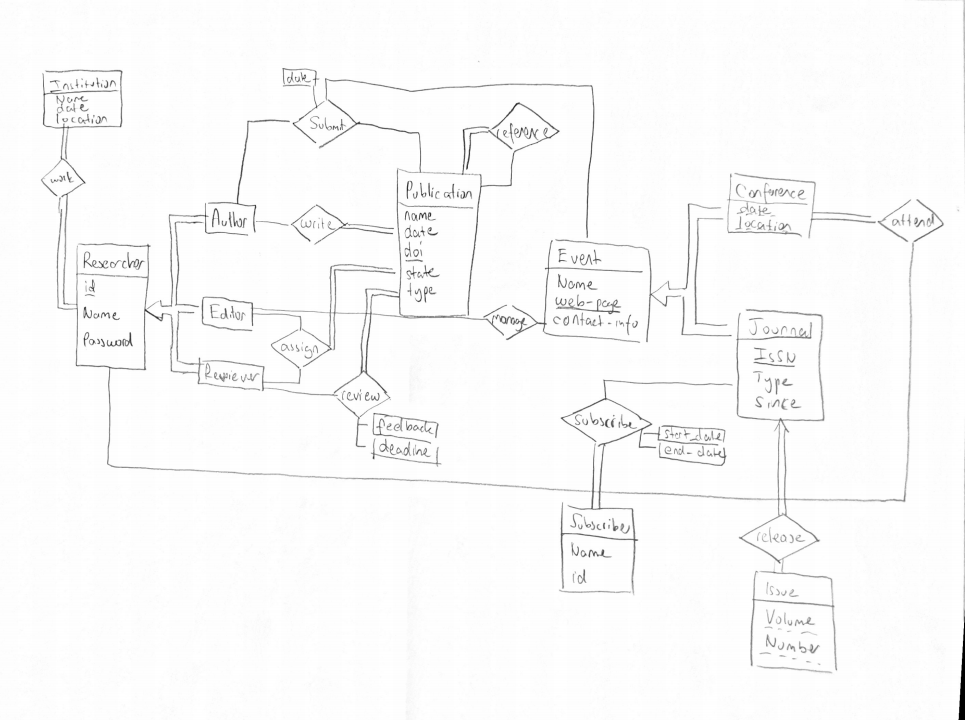
6.3 Triggers

6.4 Constraints

6.5 Stored procedures

1. Implementation

## 1. Revised E/R Model According to assistant's review, we revised our E/R model considering the feedback in points as follows:



## 2. Relational Schemas

## 2.1.Institution

**Relational Model:**

Institution (iName, foundationDate, location)

**Functional Dependencies:**

iName, foundationDate -> location

**Candidate Key:**

{ (iName, foundationDate) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE Instution(

iName varchar(32)

foundationDate DATE,

location varchar(32) NOT NULL

PRIMARY KEY (iName, foundationDate)

)

## 2.2.Researcher

**Relational Model:**

researcher (rID, rName, password)

**Functional Dependencies:**

rID -> name

rID -> password

**Candidate Key:**

{ (rID) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE researcher(

rID int PRIMARY KEY,

rName varchar(32) NOT NULL,

password int NOT NULL

)

## 2.3.Author

**Relational Model:**

author (rID)

**Functional Dependencies:**

No dependencies

**Candidate Key:**

{ ( rID) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE author(

rID int PRIMARY KEY,

FOREIGN KEY (rID) references researcher

)

## 2.4.Editor

**Relational Model:**

editor (rID)

**Functional Dependencies:**

No dependencies

**Candidate Key:**

{ ( rID) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE editor(

rID int PRIMARY KEY,

FOREIGN KEY (rID) references researcher

)

## 2.5.Reviewer

**Relational Model:**

reviewer (rID)

**Functional Dependencies:**

No dependencies

**Candidate Key:**

{ ( rID) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE reviewer(

rID int PRIMARY KEY,

FOREIGN KEY (rID) references researcher

)

## 2.6.Publication

**Relational Model:**

publication (doi, pName, writtenDate, state, type)

**Functional Dependencies:**

doi -> pName,

doi -> writtenDate,

doi -> type,

**Candidate Key:**

{ (doi) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE publication(

doi int PRIMARY KEY,

pName varchar(32) ,

writtenDate DATE,

state varchar(10) ,

type varchar(10),

)

## 2.7.References

**Relational Model:**

references (doi,refDoi)

**Functional Dependencies:**

No dependencies.

**Candidate Key:**

{ (doi,refDoi) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE references(

doi int,

refDoi int,

PRIMARY KEY(doi, refDoi) ,

FOREIGN KEY(doi) references publication,

FOREIGN KEY(refDoi) references publication,

)

## 2.8.Event

**Relational Model:**

event (webPage, eName, contactInfo)

**Functional Dependencies:**

webPage -> eName ,

webPage -> contactInfo,

**Candidate Key:**

{ (webPage) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE event(

webPage varchar(300) PRIMARY KEY,

eName varchar(32) NOT NULL,

contactInfo varchar(500)

)

## 2.9.Work

**Relational Model:**

work (iName,foundationName, rID)

**Functional Dependencies:**

No dependencies

**Candidate Keys:**

{ (iName,foundationName, rID) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE work(

iName varchar(32),

foundationDate DATE ,

rID int,

PRIMARY KEY(iName,foundationName, rID),

FOREIGN KEY(iName) references Institution,

FOREIGN KEY(foundationDate) references Institution,

FOREIGN KEY(rID) references researcher

)

## 2.10.Write

**Relational Model:**

write (rID, doi)

**Functional Dependencies:**

No dependencies

**Candidate Keys:**

{ (rID, doi) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE write(

rID int,

doi int,

PRIMARY KEY(rID, doi),

FOREIGN KEY(doi) references publication,

FOREIGN KEY(rID) references author

)

## 2.11.Manage

**Relational Model:**

manage (rID, webPage)

**Functional Dependencies:**

No dependencies

**Candidate Keys:**

{ (rID, webPage) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE manage(

rID int,

webPage varchar(300),

PRIMARY KEY(rID, webPage),

FOREIGN KEY(webPage) references event,

FOREIGN KEY(rID) references editor,

)

## 2.12.Assign

**Relational Model:**

assign (editor.rID, reviewer.rID, doi)

**Functional Dependencies:**

No dependencies

**Candidate Keys:**

{ (editor.rID, reviewer.rID, doi) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE assign(

editor.rID int,

reviewer.rID int,

doi int,

PRIMARY KEY(editor.rID, reviewer.rID, doi)

FOREIGN KEY(doi) references publication,

FOREIGN KEY(rID) references editor,

FOREIGN KEY(rID) references reviewer

)

**2.13.Review**

**Relational Model:**

review (reviewer.rID, doi,feedback,deadline)

**Functional Dependencies:**

Reviewer.rID,doi ->feedback

Reviewer.rID,doi ->deadline

**Candidate Keys:**

{ ( reviewer.rID, doi) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE review(

rID int,

doi int,

feedback varchar(1000),

deadline varchar(20),

PRIMARY KEY(rID, doi)

FOREIGN KEY(doi) references publication,

FOREIGN KEY(rID) references reviewer

)

## 2.14.Conference

**Relational Model:**

conference (webpage,location,date)

**Functional Dependencies:**

No dependencies

**Candidate Key:**

{ (webpage,location,date) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE conference(

webpage int PRIMARY KEY,

location varchar(300) PRIMARY KEY,

date DATE PRIMARY KEY ,

FOREIGN KEY (webpage) references event

)

## 2.15.Journal

**Relational Model:**

journal(webpage,ISSN,since,type)

**Functional Dependencies:**

webpage,ISSN -> since

webpage,ISSN -> type

**Candidate Key:**

{ (webpage, ISSN) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE journal(

webpage varchar(300),

ISSN int,

since int,

PRIMARY KEY(webpage, ISSN)

type varchar (32),

FOREIGN KEY (webpage) references event

)

**2.16.Attend**

**Relational Model:**

attend (rID,webpage, location,date)

**Functional Dependencies:**

No dependencies

**Candidate Keys:**

{ (rID, webpage, location,date) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE review(

rID int,

webpage varchar(300),

location varchar(300),

date DATE,

PRIMARY KEY (rID, webpage, location,date),

FOREIGN KEY(rID) references researcher,

FOREIGN KEY(webpage) references conference

FOREIGN KEY(location) references conference

FOREIGN KEY(date) references conference

)

## 2.17.Subscriber

**Relational Model:**

subscriber (subID,sName,password)

**Functional Dependencies:**

subID -> sName ,

**Candidate Key:**

{ (subID) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE subscriber(

subID int PRIMARY KEY,

sName varchar(50) NOT NULL,

password varchar(11) NOT NULL,

)

## 2.18.Subscribe

**Relational Model:**

subscribe (subID,webpage,ISSN,startDate,endDate)

**Functional Dependencies:**

subID,webpage,ISSN -> startDate

subID,webpage,ISSN -> endDate

**Candidate Key:**

{ (subID,webpage,ISSN) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE subscriber(

subID int,

webpage varchar(300),

ISSN int,

startDate DATE,

endDate DATE,

PRIMARY KEY(subID,webpage,ISSN),

FOREIGN KEY (subID) references subscriber,

FOREIGN KEY (webpage) references journal,

FOREIGN KEY (ISSN) references journal,

)

## 2.19.Issue

**Relational Model:**

issue (webpage,ISSN,volume,number)

**Functional Dependencies:**

No dependencies

**Candidate Key:**

{ (webpage,ISSN,volume,number) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE issue(

webpage varchar(300),

ISSN int,

volume int,

number int,

PRIMARY KEY(webpage,ISSN,volume,number),

FOREIGN KEY (webpage) references journal,

FOREIGN KEY (ISSN) references journal,

)

**2.20.Submit**

**Relational Model:**

submit (rID,doi,webpage,state,sDate)

**Functional Dependencies:**

rID,doi,webpage -> state

rID,doi,webpage -> sDate

**Candidate Key:**

{ (rID,doi,webpage) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE submit(

rID int,

doi int,

webpage varchar(300),

state varchar(8) NOT NULL,

sDate DATE,

PRIMARY KEY(rID,doi,webpage),

FOREIGN KEY (rID) references researcher,

FOREIGN KEY (doi) references publication,

FOREIGN KEY (webpage) references event,

)

**3. Functional Dependencies and Normalization of Tables**

In Section 2, functional depedencies and normal forms are indicated. . After our calculations, we conculuded that all of our relations are in Third Normal Form(3NF) and there is no ueed for decmposition.

## 4. Functional Components

**4.1. Use Case / Scenarios**

The program can be used by four types of users. These are subscriber, author, reviewer and editor. When they log in, the system determines the type of the user and grants access according to the user's type.

**4.1.1 Subscriber use case**

* Subscriber can log into the system by their id and their password.
* subscriber can see the list of all available journals.
* subscriber can see the list of jourals that they have subscribed to.
* subscriber can see the end dates of their subscriptions to journals.
* subscriber can see the list of issues of a journal that they have subscribed to.
* subscriber can view the issues of a journal that they have subscribed to.
* subscriber can subscribe to a journal available in the list of journals.
* subscriber can renew thir subscription to a journal.
* subscriber can search for a journal by its name.
* subscriber can search for a publication by its name or its author.
* after searching subscriber can view the detais of the publication such as its author and the issue/journal it appears but they cannot view the publication itself.

**4.1.2 Author use case**

* Author can log into the system by their id and their password.
* Author can see the list of all available journals.
* Author can see the list of issues of a journal.
* Author can search for a journal by its name.
* Author can search for a publication by its name or its author.
* After searching author can view the details and the publication itself.
* Author can submit a publication to a journal or a conference by uploading the publication and entering its details such as the list of citations to the system.
* Author can see the list of past conferences and view their details.
* Author can see the list of upcoming conferenes.
* Author can choose to attend an upcoming conference.
* Author can search for a conference by its name or date.
* Author can search for an author by their name and view thier details.

**4.1.3 Reviewer use case**

* Reviewer can log into the system by thier ID and their password.
* Reviewer can see the list of all available journals.
* Reviewer can see the list of issues of a journal.
* Reviewer can search for a journal by its name.
* Reviewer can search for a publication by its name or its author.
* After searching Reviewer can view the details and the publication itself.
* Reviewer can see the list of past conferences and view their details.
* Reviewer can see the list of upcoming conferenes.
* Reviewer can choose to attend an upcoming conference.
* Reviewer can search for a conference by its name or date.
* Reviewer can see the list of publications that they were assigned.
* Reviewer can see the details of the assignments they have such as their deadlines.
* Reviewer can give feedback to the publications that they were assigned to review.
* Reviewer can choose to reject or accept the publications that they were assigned.
* Reviewer can search for an author by their name and view thier details.

**4.1.4 Editor use case**

* Editor can log into the system by thier ID and their password.
* Editor can see the list of all available journals.
* Editor can see the list of issues of a journal.
* Editor can search for a journal by its name.
* Editor can search for a publication by its name or its author.
* After searching Editor can view the details and the publication itself.
* Editor can see the list of past conferences and view their details.
* Editor can see the list of upcoming conferenes.
* Editor can choose to attend an upcoming conference.
* Editor can search for a conference by its name or date.
* Editor can see the list of journals that they are editing.
* Editor can see the list of conferences that they are responsible for.
* Editor can see the list of submitted publications to journals that they are editing and conferences that they are responsible for.
* Editor can see the list of reviewers.
* Editor can search a reviewer by its name.
* Editor can assign a submitted publication to a reviewer.
* Editor can search for an author by their name and view thier details.
* Editor can choose which issue of the journal the accepted publications will appear in.
* Editor can create new events(journal or conference)

**4.2 Algorithms**

**4.2.1 Publication submisson algorithm (parameters: publication name, date author name, publication file, list of referenced publications)**

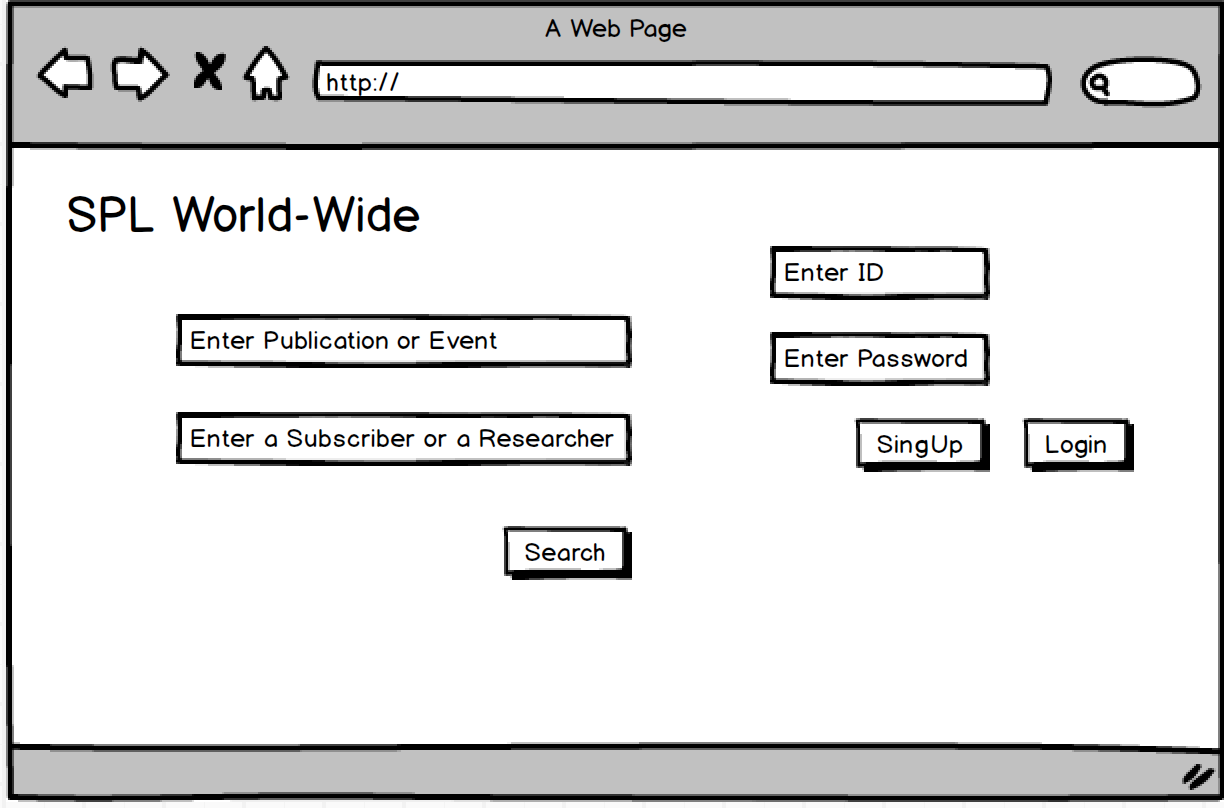
The authors will be able to submit a publication by uploading the text of the publication. After that they should specify which journal they are submitting it to and fill in other details such as the list of works cited. After submission, the state of the publication is initialised to "On-Review" and the foreign key referencing issue and reviewer will be initialised to null. When the editor assigns a reviewer, the foreign key referencing reviewer changes accordingly. When reviewer submits a feedback the author is notified. when the reviewer accepts or rejects the publication, the state changes accordingly and the editor and author is notified. When the editor publishes the paper, the foreign key referencing issue changes accordingly.

**4.2.2 Citation algorithm (parameters: list of referenced publication dois)**

When an author submits a publication, the system will read the list of references from the submission and add tuples to references table accordingly.

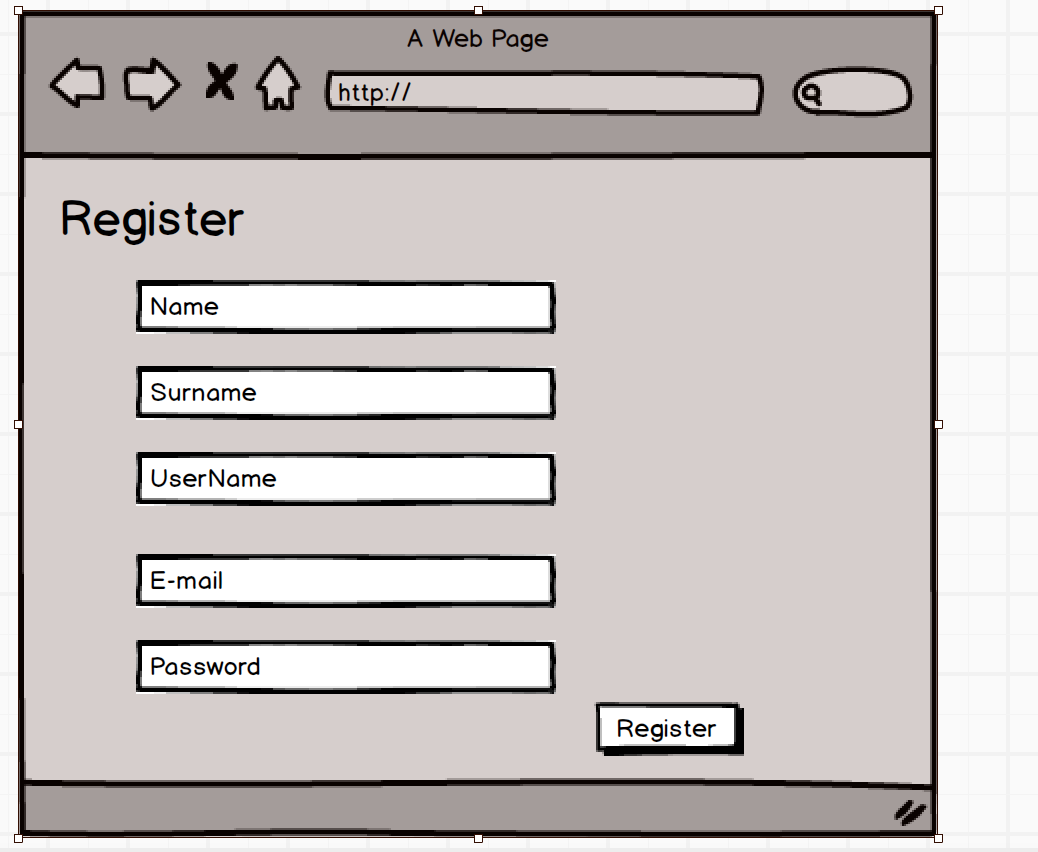
**5. User Interface Design And Corresponding SQL Statements**

**HOMEPAGE:**



**Description:** The homepage screen allows users to search journals, publications, subscribers and authors. It is not necessary to log in to the system to make a search. However to make a sunscription or an upload as an author the user has to log in the system.

**REGISTER:**



**Description:** Register screen will be opened when the signup button is clicked on the homepage. It will allow adding a new user to the system. It will ask from the user name,surname, username, e-mail and a password.

**SQL:**

Counting users to add a new one

Select Count(UserID)

From User

Checks of the existence

Select UserID

From User

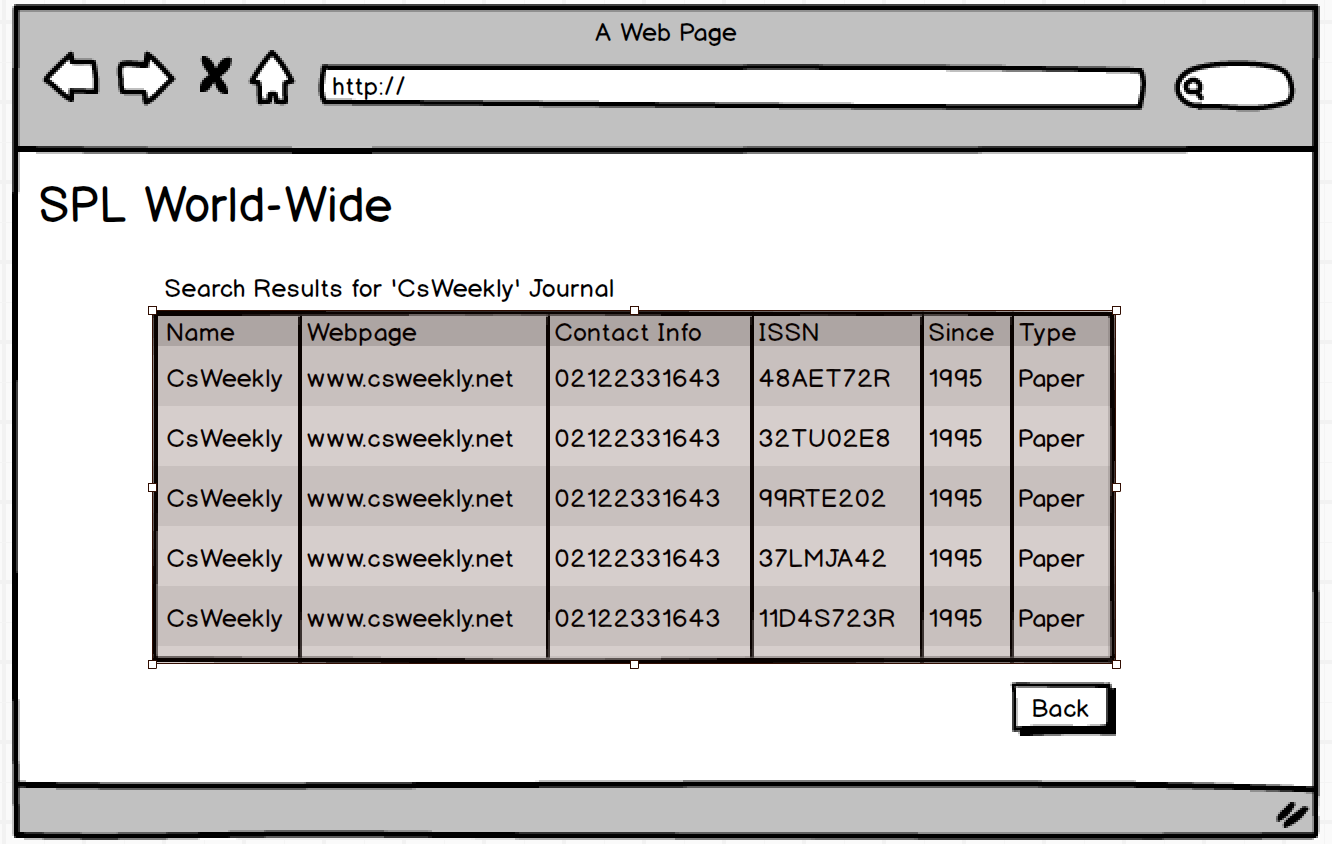
Where e-mail = email

Add user

Insert Into User(uname,usurname,username,uid,e-mail)

values(@name,@surname,@username,gid,@e-mail)

**Journal Search:**



**Description:** When the user enters a journal name in the first search box and presses the search button the results are listed as above.

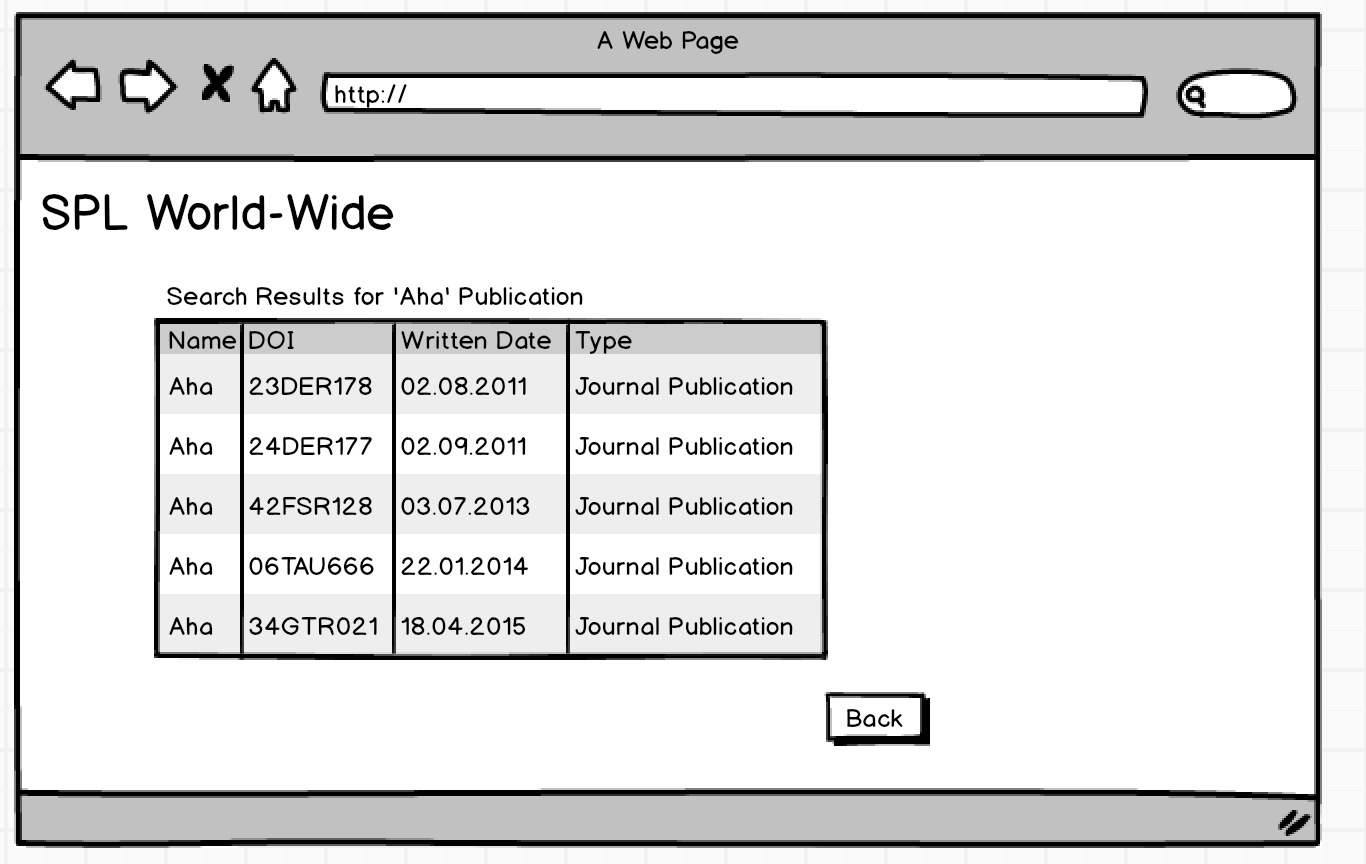
**SQL:**

SELECT e.name,e.webpage,e.contactinfo,j.issn,j.since,j.type

FROM Event as e natural join Joutnal as j

WHERE j.name = @name

**Search for Publication:**

**Description:** When the user enters a publication name in the first search box and presses the search button the results are listed as above. If the user clicks the DOI number he can access the Publication page

**SQL:**

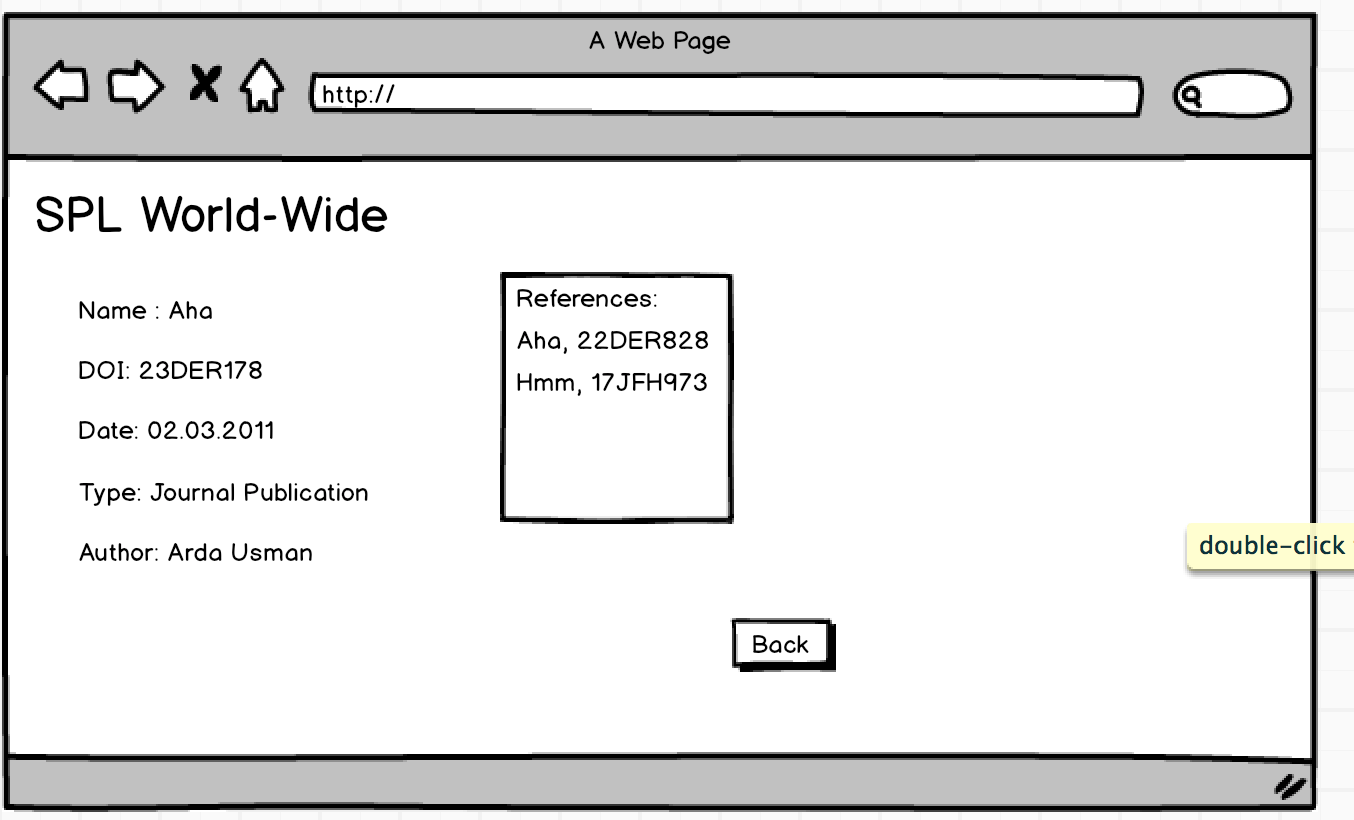
SELECT Publication.name, Publication.DOI, Publication.writtenDate, Publication.type

FROM PUBLICATION

WHERE PUBLICATION.name = @pname

GROUP BY PUBLICATION.type

**Publication Screen:**



**Description:** :When the user clicks a DOI number from the results a page for that particular publication appears on the screen

**SQL:**

SELECT p.pname,p.doi,p.writtendate,p.type,a.aname

FROM Publication as p , Author as a, Write as w

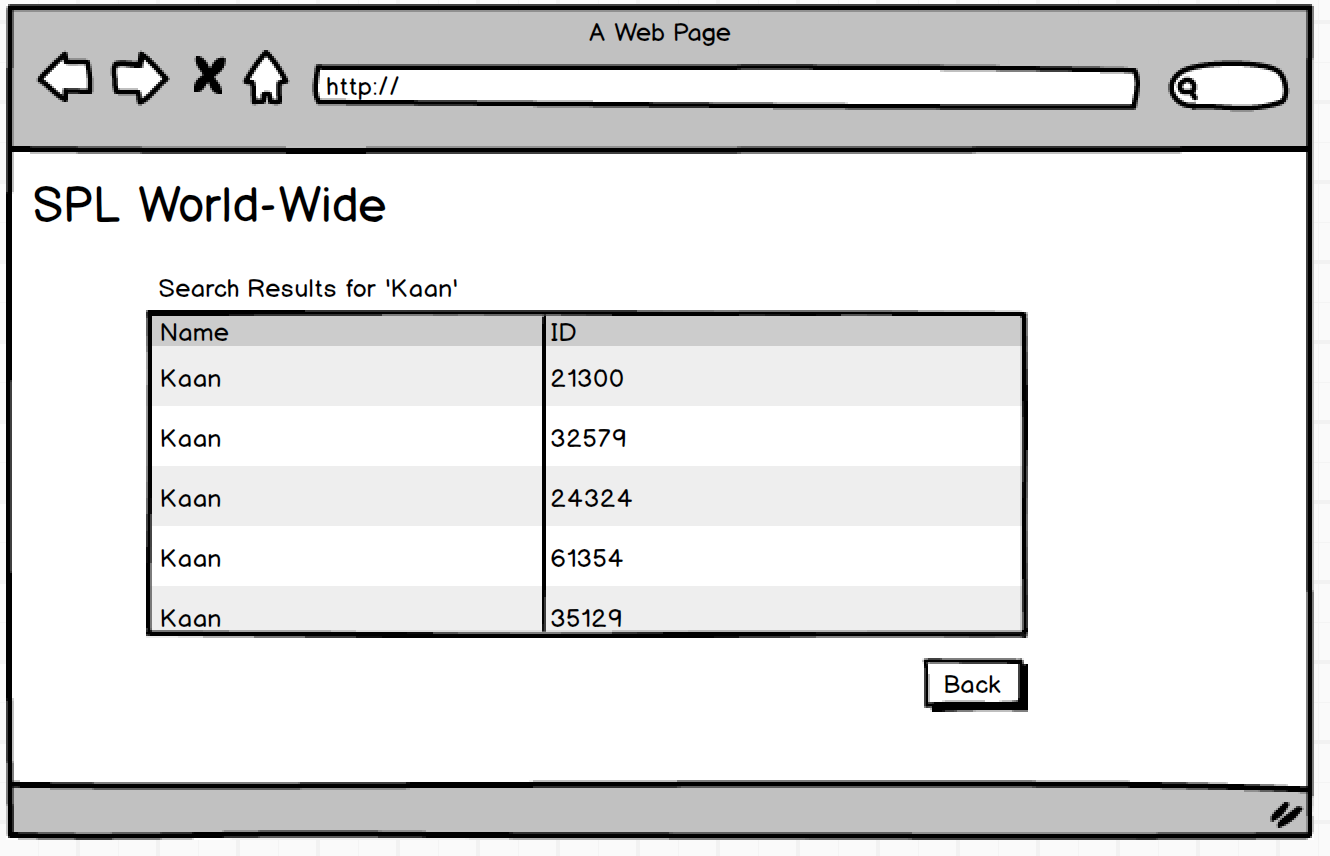
WHERE p.doi = w.doi and w.aid = a.aid and p.doi = @doi

references

SELECT p.doi,p.name

FROM Publication as p, Reference as r

WHERE p.doi = r.doi and p.doi = @doi



**User Search:**

**Description:** When the user enters a user name into the second search box. The results are listed as above. The user can access a preferred user from the list by clicking the ID number.

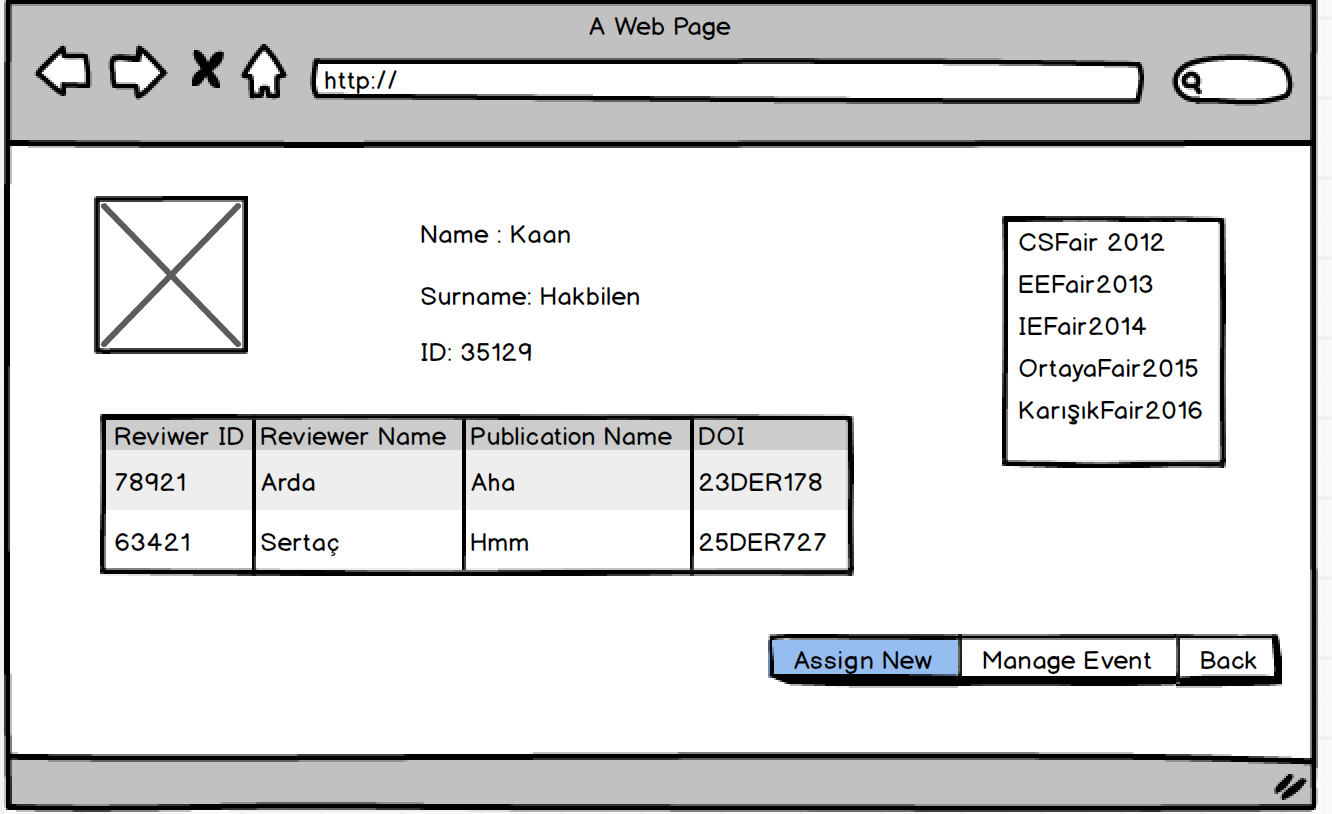
**SQL:**

SELECT r.rname,r.rid,s.sname,s.sid

FROM Researcher as r union Subscriber as s

WHERE r.rname = @rname or s.sname = @sname;

**Editor Screen:**



**Description:** When user clicks to a id in the search result table and that id belongs to an editor. A page like above appears on the screen

**SQL:**

SELECT e.name,e.id

FROM Editor as e

WHERE e.id = @id

assignment table

SELECT a.rid,a.rname,a.doi,p.pname

FROM Assign as a natural join Publication as p natural join Editor as e

WHERE a.eid = e.eid

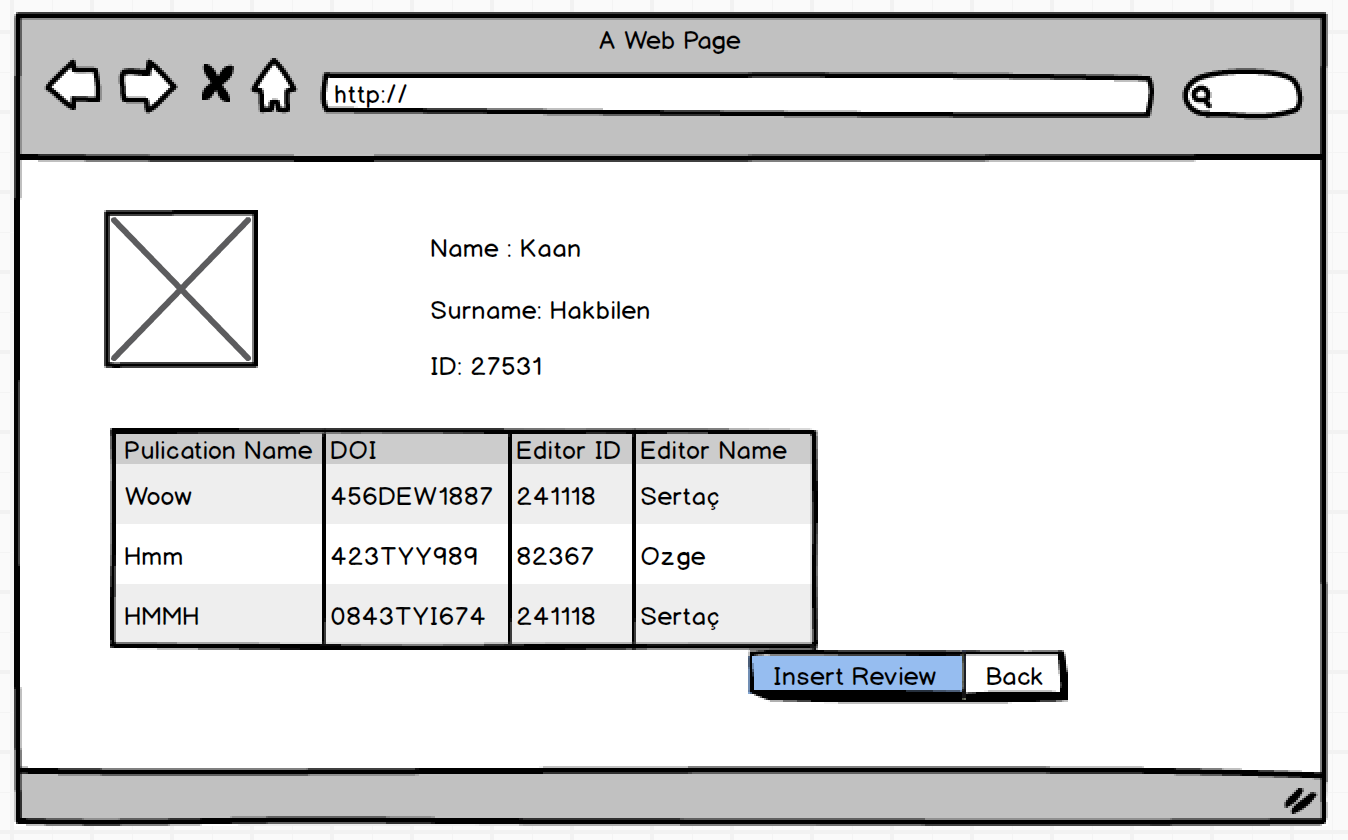
event list

SELECT e.evname

FROM Manage natural join Editor, Event as e

WHERE Manage.eid = Editor.eid

**Reviewer Screen:**



**Description:** When the user clicks to an id from the search results an that id belongs to a reviewer this page appears on the screen

**SQL:**

SELECT r.name,r.id

FROM Reviwer as r

WHERE r.id = @id

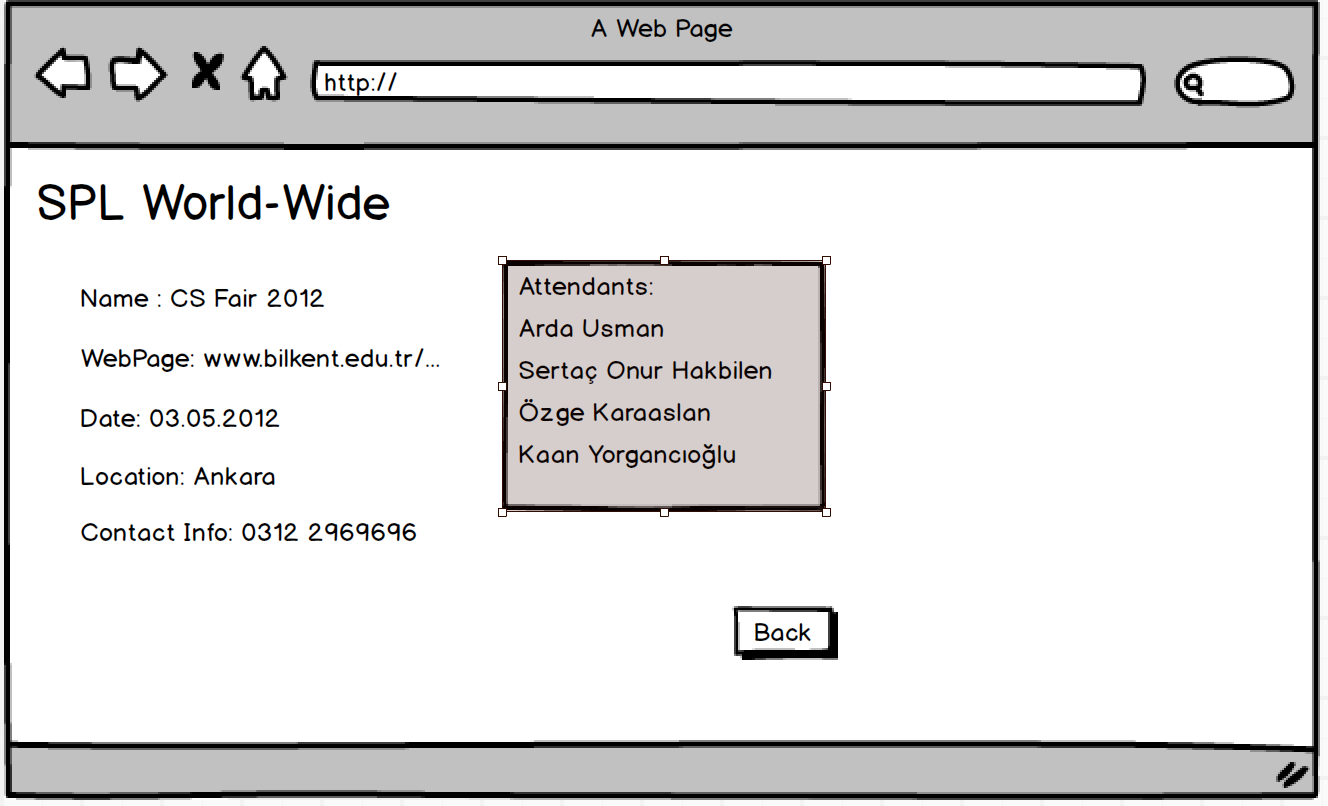
Review table

SELECT p.name,p.pid,e.eid,e.ename

FROM Publication as p, Editor as e, Assign as a

WHERE p.pid = a.pid and e.eid = a.eid and a.rid = @rid

**Conference Screen:**



**Description:**When the user clicks to a conference name from a list on the editor page this screen occurs

**SQL:**

SELECT c.name,c.webpage,c.contactinfo,c.location,c.date

FROM Conference as c

WHERE c.name = @name

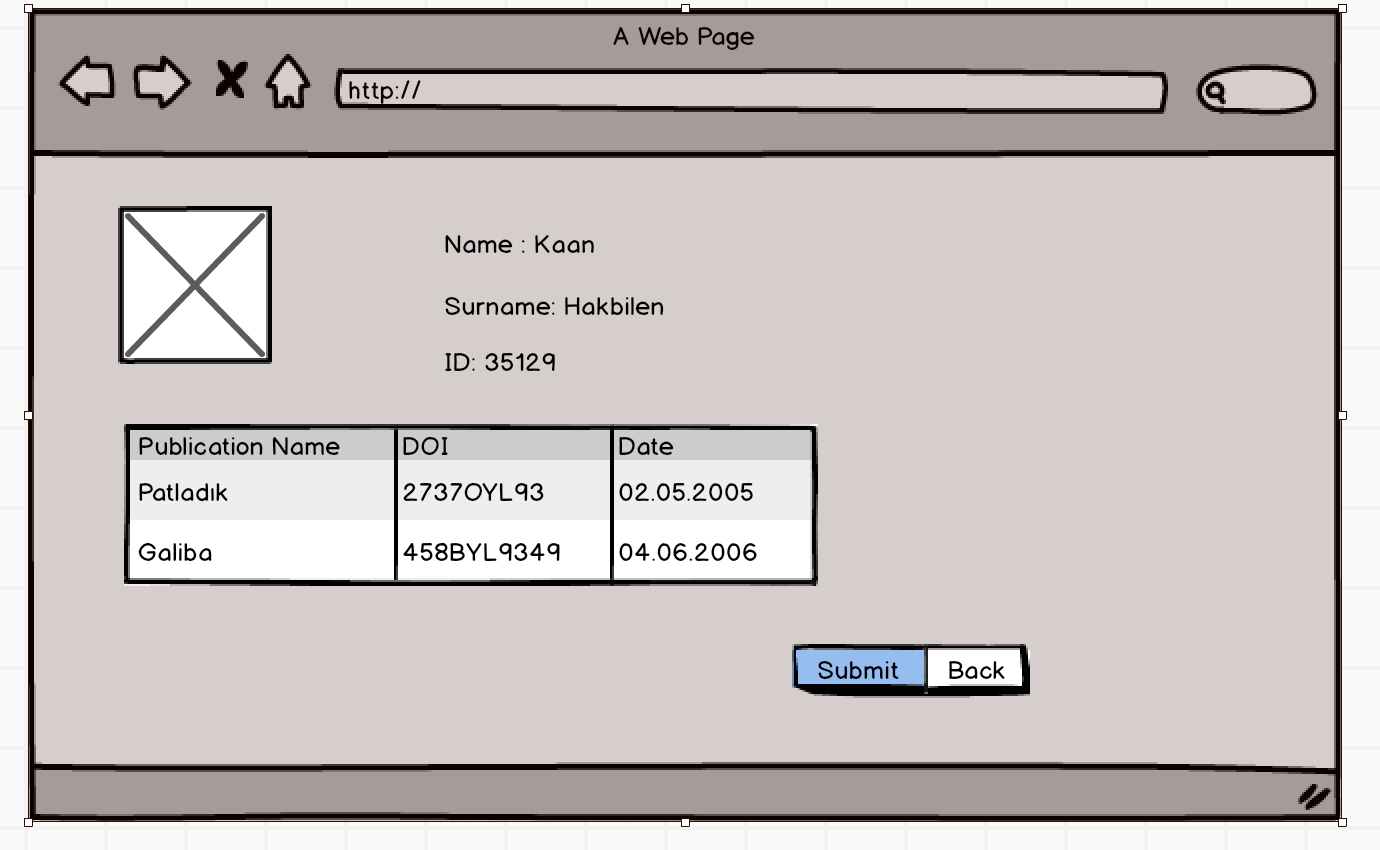
attendants

SELECT r.rname

FROM Conference natural join Attend, Researcher as r

WHERE Attend.rid = r.rid and Attend.webpage = Conference.@webpage

**Author Screen:**



**Description:** When the user clicks to an id in the results table and that id belongs to an author this page shows up

**SQL:**

SELECT a.aid,a.aid

FROM Author as a

WHERE a.aid = @aid

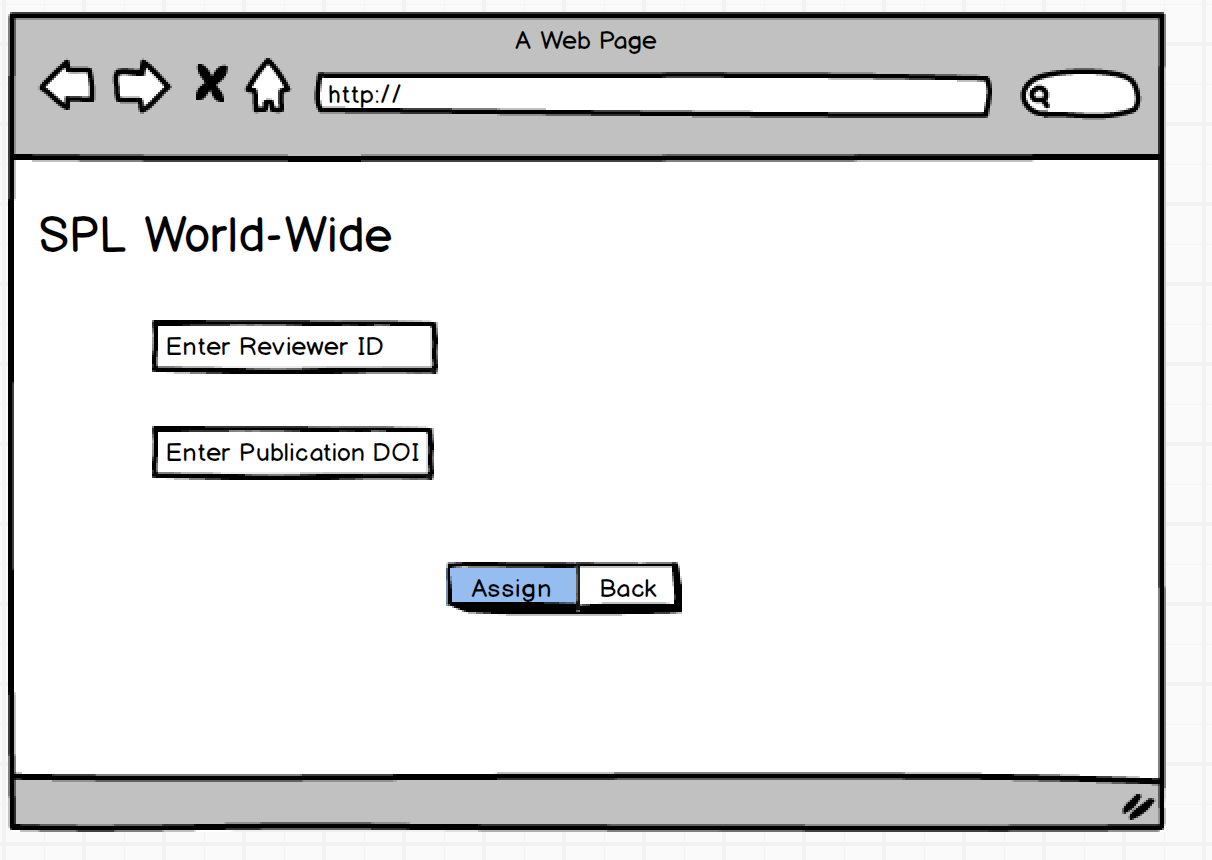
table

SELECT p.name,p.doi,s.date

FROM Author natural join Submit as s natural join Publication as p

WHERE s.aid = @aid

**Make Assignment:**



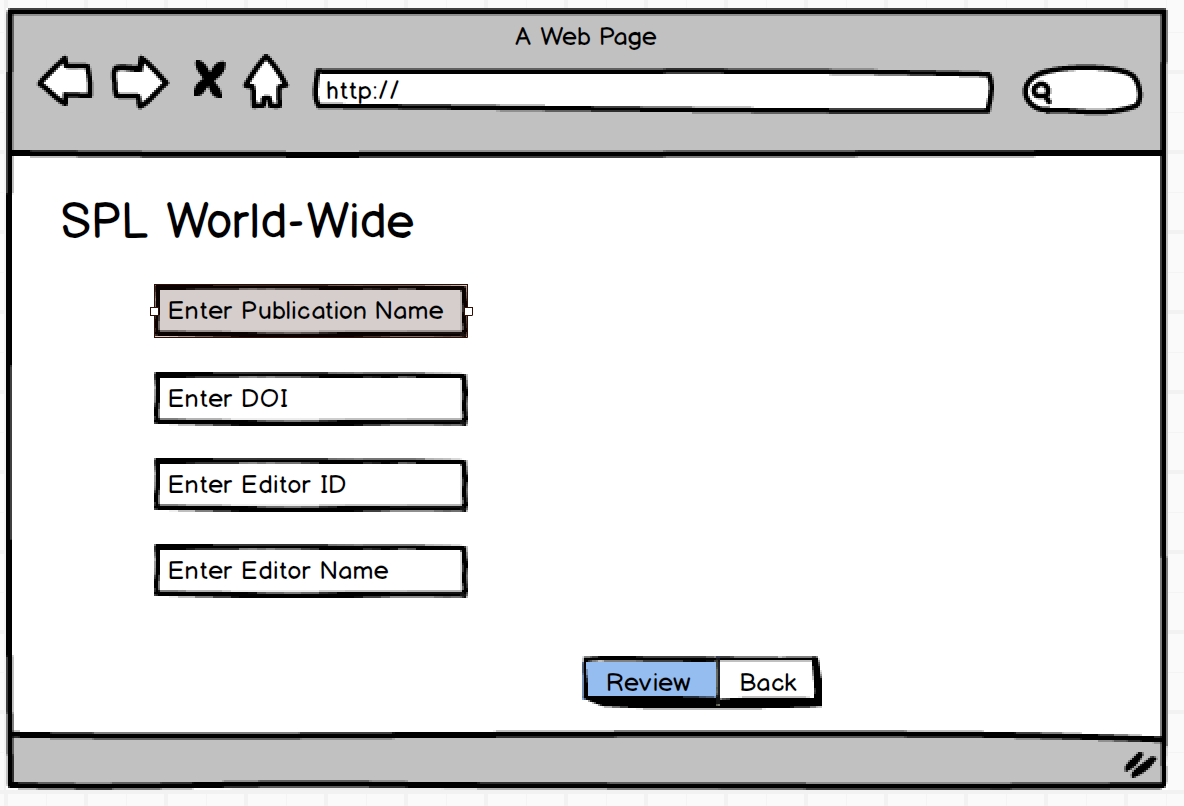
**Description:** When the editor clicks to the make assignment button this screen shows up that allows editor to make a new assignment to a reviewer

**SQL:**

INSERT INTO Assign(rid,doi,aid)

VALUES(@rid,@doi,aid)

**Insert Review:**

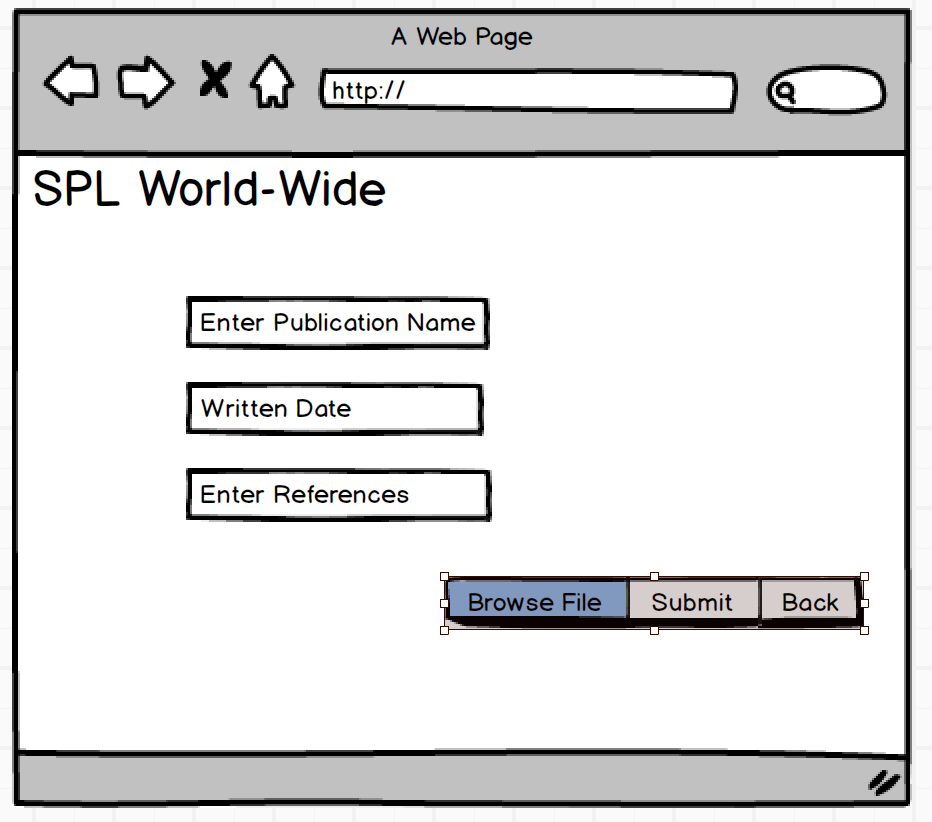


**Description:** When the reviewer wants to add a new review to the system and clicks the insert review button this screen shows ups

**SQL:**

INSERT INTO Review(doi,eid,rid,feedback,deadline)

VALUES(@doi,@eid,rid,@feedback@deadline)

**Submit Publication:**

**Description:** When the author wants to add a new publication to the system and clicks the submit button this screen shows ups

**SQL:**

INSERT INTO Publication(name,doi,writtenDate,state,type)

VALUES(name,doi,@writtenDate,state,type)

**6. Advanced Database Components**

**6.1 Reports**

**6.1.1 Total number of publications submitted**

WITH submissions\_by\_author (Author\_id, Author\_name, submissions)

AS(SELECT A.name, A.id, count(P.doi) AS submissions

FROM author A, submit S, Publication P

WHERE A.id = S.id

AND S.doi= P.doi

GROUP BY A.id)

**6.1.2 Total number of published publications**

WITH publications\_by\_author (Author\_id, Author\_name, publications)

AS(SELECT A.name, A.id, count(P.doi) AS publications

FROM author A, submit S, Publication P

WHERE A.id = S.id

AND S.doi= P.doi

AND P.state='published'

GROUP BY A.id)

**6.1.3 Number of reviews**

WITH reviews\_by\_reviewer (Reviewer\_id, reviewer\_name, reviews)

AS(SELECT R.name, R.id, count(P.doi) AS reviews

FROM reviewer R, review E, Publication P

WHERE R.id = E.id

AND E.doi= P.doi

GROUP BY R.id)

**6.1.4 Number of subscribers**

WITH subscribers\_by\_journal (ISSN, Journal\_name, subs)

AS(SELECT J.name, J.id, count(S.id) AS subs

FROM journal J, subsribe R, Publication P

WHERE J.id = R.ISSN

AND R.doi= S.doi

GROUP BY J.ISSN)

**6.2 Views**

**6.2.1 Journal listings**

All users can see the list of journals. Therefore this view will be used to list the journals.

CREATE VIEW Journal\_info AS

SELECT name, ISSN

FROM Journal

**6.2.2 Author listings**

All users can see the list of Authors . this view will be used to list the Authors.

CREATE VIEW Author\_info AS

SELECT name

FROM Authors

**6.3 Triggers**

* When an event is created a corresponding tuple is added to the manage/edit table.
* when the editor assigns a reviewer to a paper, a corresponding tuple is added to review table.
* when an author submits a paper, a tuple will be added to the writes table.
* When a reviewer accepts or rejects a paper, its state is changed accordingly.
* when an editor choses to publish an accepted paper in an issue a tuple is added to published Table.

**6.4 Constraints**

* The system cannot be used without logging in.
* Until a publication is published it cannot be seen by anyone except the author, the assigned reviewer and the editor.
* A researcher cannot attend to a past conference
* A subscriber can only view the issues of the journal he has subscribed to.
* An editor can only view the submissions to the journal/ conference he is responsible for
* A reviewer can only view the submissions he is assigned to

**6.5 stored Procedures**

* A procedure will be used to notify the author when a reviewer posts feedback to the paper.
* A procedure will be used to notify the reviewer when a new paper is assigned to him.
* A procedure will be used to create references once a paper is submitted.

**7. Implementation Plan**

At data layer we will use MySQL Server in our project as database management system. For application logic and user interface we will code in AngularJs.