

**Bilkent University**

**Department Of Computer Engineering**

**CS 353**

**FINAL REPORT**

***‘SPLWORLD-WIDE’***

# https://github.com/ardausman/cs353Termproject.git

**PROJECT GROUP MEMBERS**:

* KAAN YORGANCIOĞLU
* SERTAÇ ONUR HAKBİLEN
* ÖZGE KARAASLAN
* TURGAY ARDA USMAN

Contents

[1. Brief Description 1](#_Toc450304050)

[Todays word education becomes one of the most important constant in our lives. People try 1](#_Toc450304051)

[2. Final E/R Diagram 2](#_Toc450304052)

[3. Relational Schemas 3](#_Toc450304053)

[3.1.Institution 3](#_Toc450304054)

[3.2.Researcher 3](#_Toc450304055)

[3.3.Author 4](#_Toc450304056)

[3.4.Editor 4](#_Toc450304057)

[3.5.Reviewer 5](#_Toc450304058)

[3.6.Publication 5](#_Toc450304059)

[3.7.References 6](#_Toc450304060)

[3.8.Event 6](#_Toc450304061)

[3.9.Work 7](#_Toc450304062)

[3.10.Write 8](#_Toc450304063)

[3.11.Manage 8](#_Toc450304064)

[3.12.Assign 9](#_Toc450304065)

[3.13.Review 9](#_Toc450304066)

[3.14.Conference 10](#_Toc450304067)

[3.15.Journal 10](#_Toc450304068)

[3.16.Attend 11](#_Toc450304069)

[3.17.Subscriber 12](#_Toc450304070)

[3.18.Subscribe 12](#_Toc450304071)

[3.19.Issue 13](#_Toc450304072)

[3.20.Submit 14](#_Toc450304073)

[4. Implementation Details 15](#_Toc450304074)

[5. Advanced Database Features 15](#_Toc450304075)

[5.1. Secondary indices 15](#_Toc450304076)

[5.2. Views 15](#_Toc450304077)

[6. User Manual 16](#_Toc450304078)

# 1. Brief Description

# Todays word education becomes one of the most important constant in our lives. People try

to find new things with assistance of their education and publish their findings in a scientific format.

Our project is a scientific papers digital library. If we think about a digital library there are lots of

information which are for keeping track of scientific publications, authors, journals, conferences,

institutions and subscribers.

We are going to build this system to keep track of every document in digital library.

Scientific papers digital library management system is a system that provide a huge variety of

publications. A publication may be a journal/conference paper or a technical report and these

publications can be in different states; on submit, accepted on review or rejected. These features

can be seen by users.

To provide all functionalities of the system database will be used because the system

should store huge amount of data and a data management system will be required to add data

change informations about them and find the perfect selection according to users input. Finding a

paper without such a system will be very hard because there are hundreds of scientific paper

which are about same problem or solution. Database usage also provides ease of use because

users only type letters to find their keywords to find the related scientific papers. There will be 2

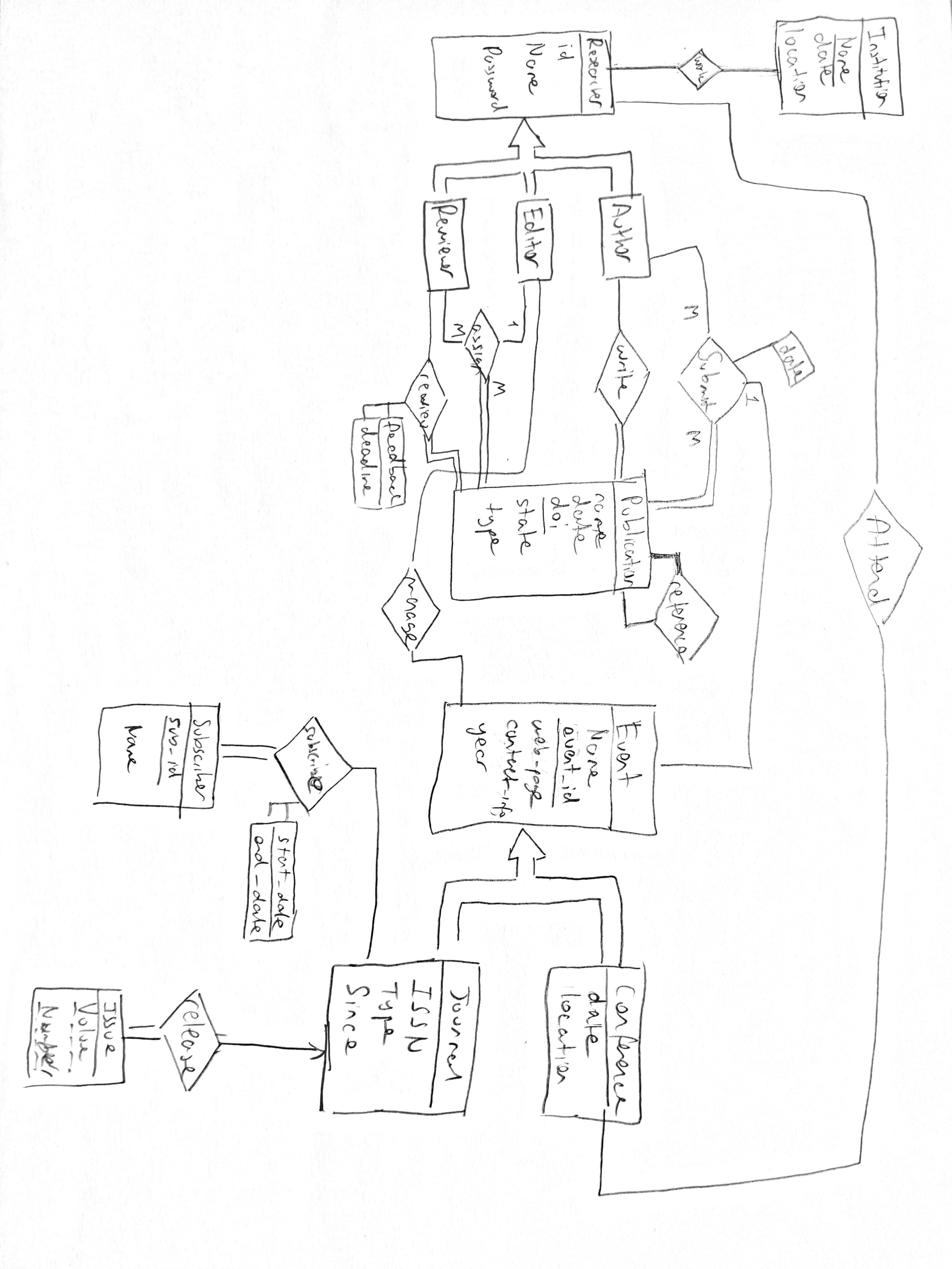
types of user which are admins and users. However not all users have to have user id because

papers will be open to everyone if they are not restricted. User id will be necessary only for

submitting paper to system. Admins will be the only person who give permission or change papers

states in the system and it will also provides security in our system.

# 2. Final E/R Diagram



# 3. Relational Schemas

### 3.1.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)

)

### 3.2.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

)

### 3.3.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

)

### 3.4.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

)

### 3.5.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

)

### 3.6.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),

)

### 3.7.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,

)

### 3.8.Event

**Relational Model:**

event (eName, event-id, webpage, contactInfo, year)

**Functional Dependencies:**

event-id —> eName ,

event-id -> contactInfo,

event-id -> webpage

event-id -> contactInfo

event-id -> year

**Candidate Key:**

{ (event-id) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE event(

eName varchar(32) NOT NULL,

event-id int PRIMARY KEY,

webPage varchar(300),

contactInfo varchar(500)

year DATE,

)

### 3.9.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, foundationDate) references Institution,

FOREIGN KEY(rID) references researcher

)

### 3.10.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

)

### 3.11.Manage

**Relational Model:**

manage (rID, event-id)

**Functional Dependencies:**

No dependencies

**Candidate Keys:**

{ (rID, event-id) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE manage(

rID int,

webPage varchar(300),

PRIMARY KEY(rID, event-id),

FOREIGN KEY(event-id) references event,

FOREIGN KEY(rID) references editor,

)

### 3.12.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(rID),

FOREIGN KEY(rID) references reviewer

)

### 3.13.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

)

### 3.14.Conference

**Relational Model:**

conference (event-id,location,date)

**Functional Dependencies:**

event-id -> location

even-idt-> date

**Candidate Key:**

{ (event-id) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE conference(

event-id int PRIMARY KEY,

location varchar(300),

date DATE,

FOREIGN KEY (event-id) references event

)

### 3.15.Journal

**Relational Model:**

journal(event-id,ISSN,since,type)

**Functional Dependencies:**

event-id-> since

event-id-> type

**Candidate Key:**

{ (event-id) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE journal(

event-id int PRIMARY KEY,

ISSN int,

since int,

type varchar (32),

FOREIGN KEY (Event-id) references event

)

### 3.16.Attend

**Relational Model:**

attend (rID,event-id)

**Functional Dependencies:**

No dependencies

**Candidate Keys:**

{ (rID, event-id) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE attend(

rID int,

event-id int,

PRIMARY KEY(rID)

PRIMARY KEY(event-id)

FOREIGN KEY(rID) references researcher

FOREIGN KEY(evet-id) references conference

)

### 3.17.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,

)

### 3.18.Subscribe

**Relational Model:**

subscribe (subID,event-id,startDate,endDate)

**Functional Dependencies:**

subID,event-id -> startDate

subID,event-if -> endDate

**Candidate Key:**

{ (subID,event-id) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE subscriber(

subID int,

webpage varchar(300),

ISSN int,

startDate DATE,

endDate DATE,

PRIMARY KEY(subID,event-id),

FOREIGN KEY (subID) references subscriber,

FOREIGN KEY (event-id) references journal,

)

### 3.19.Issue

**Relational Model:**

issue (event-id,volume,number)

**Functional Dependencies:**

No dependencies

**Candidate Key:**

{ (event-id,volume,number) }

**Normal Form:**

3NF

**Table Definition:**

CREATE TABLE issue(

webpage varchar(300),

ISSN int,

volume int,

number int,

PRIMARY KEY(event-id,volume,number),

FOREIGN KEY (webpage) references journal,

FOREIGN KEY (ISSN) references journal,

)

### 3.20.Submit

**Relational Model:**

submit (rID,doi,event-id,state,sDate)

**Functional Dependencies:**

rID,doi,event-id -> state

rID,doi,event-id -> sDate

**Candidate Key:**

{ (rID,doi,event-id) }

**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,event-id),

FOREIGN KEY (rID) references researcher,

FOREIGN KEY (doi) references publication,

FOREIGN KEY (event-id) references event(event-id),

)

# 4. Implementation Details

We have developed SPL World-Wide website for connecting to for CS 353 lecture. According to manage webpage and database we use different programming languages and programs to complete our project.

* We use PHP and HTML5 as our languages
* We use developed our project using PhpMyAdmin and Sublime Text 3.
* We create our database in MySQL.
* We connected queries to MySQL from PHP language with PHP’s object oriented mysql extension. The MySQLi Extension (MySQL Improved) is a relational database driver used in the PHP programming language to provide an interface with MySQL databases.
* According to design report feedback, we changed our event information because we see that some of the events may not have any email information so we insert event id according to email primary key. We reviewed our relations between our entities.

# 5. Advanced Database Features

### 5.1. Secondary indices

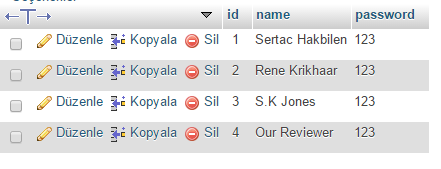
In this project indices, were used in many unique attributes. Thus we had secondary indices as well as we used primary indices.

#### 5.1.1. Subscriber Table

Email attribute used as second index. İn this table.

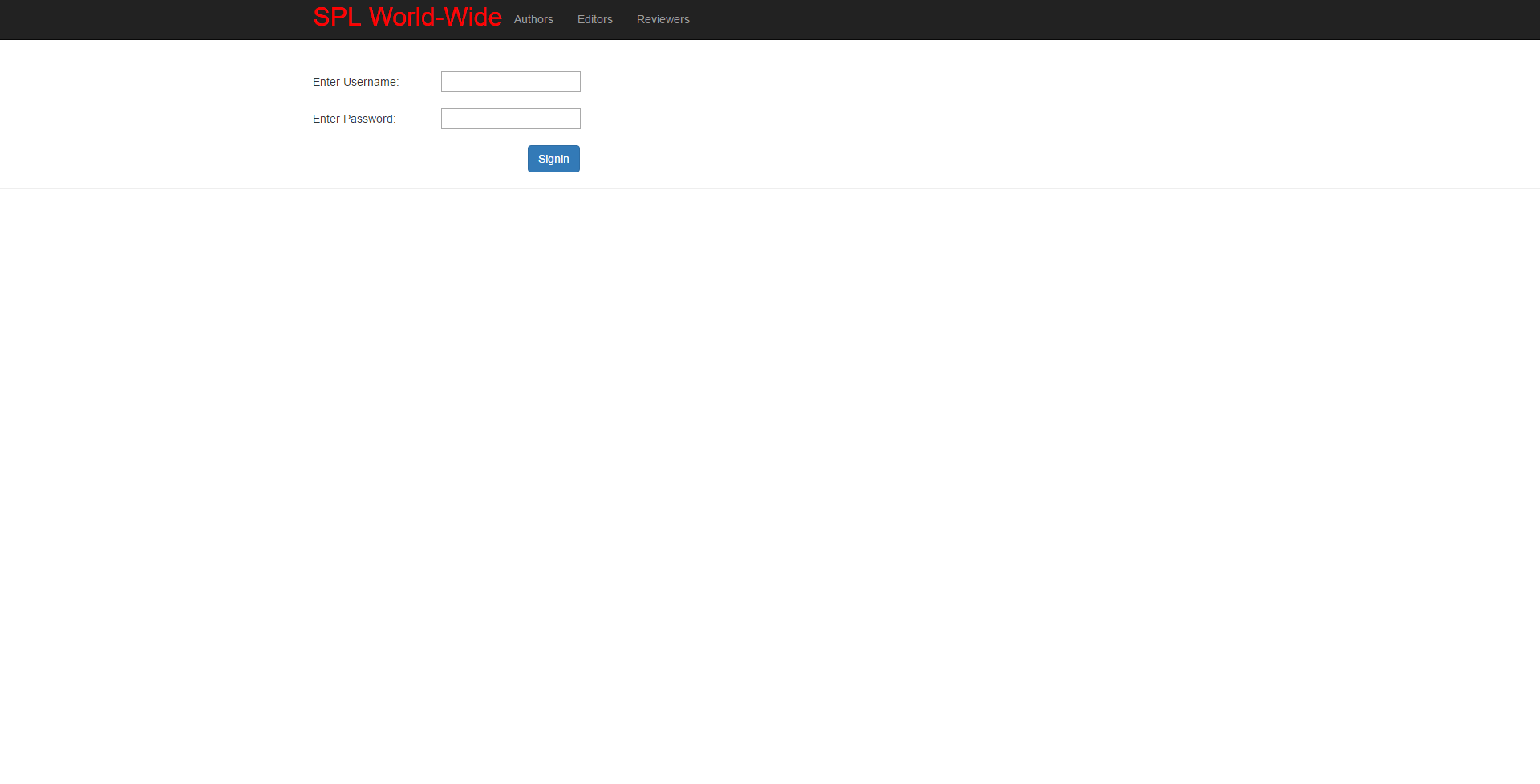
### 5.2. Views

We used a researchview for our website. Since users will visit and login the website very often we formed a view so that we will not need to join tables for every login operation. We joined the researcher-author,researcher-editor, researcher-reviewer tables and this join operation have completed. We gathered author, editor and reviewer tables. We only included ID, name, password to these tables since any more attributes will only cause redendancy.



# 6. User Manual

**HOMEPAGE:**

****

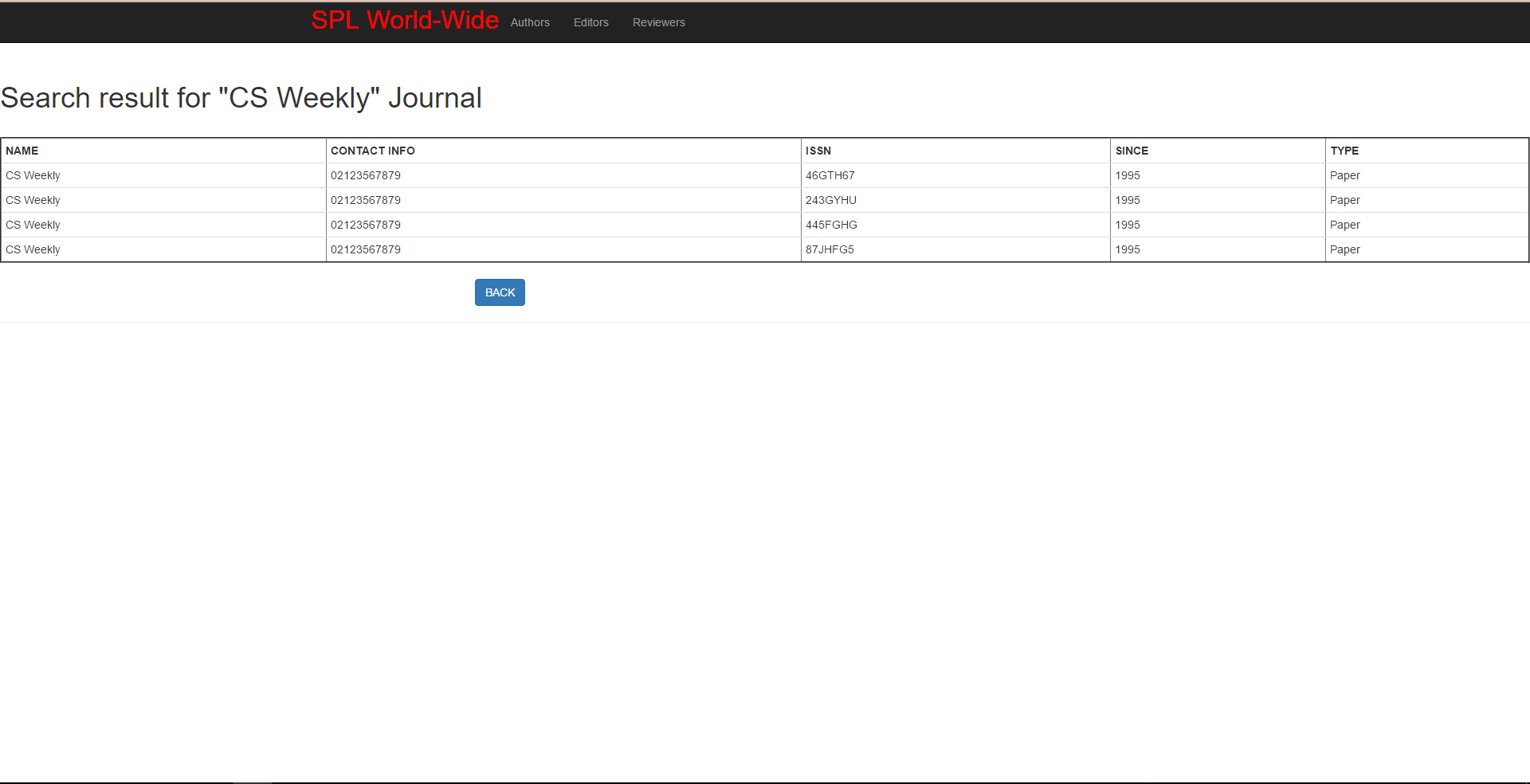
**Description:** The homepage screen waits for a signed user to sign in to the system. A search part for this screen will be added at the implementation which will allow non-signed users and signed users to search for publications and users and conferences

**REGISTER:**



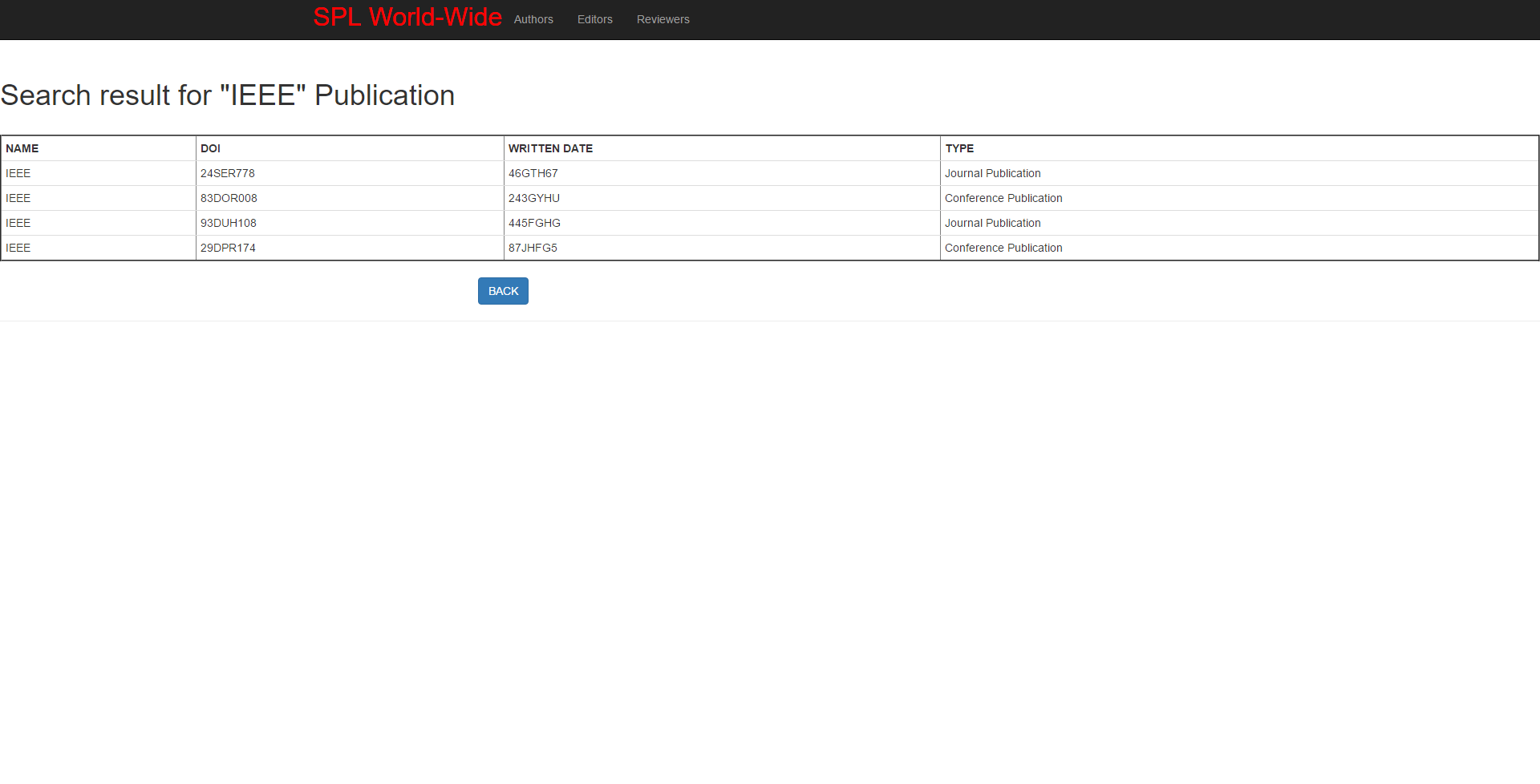
**Description:** Register screen will be opened when the signup button is clicked on the homepage. It will allow adding a new user to the system. It will ask from the user name,surname, username, e-mail and a password.

**Journal Search:**

****

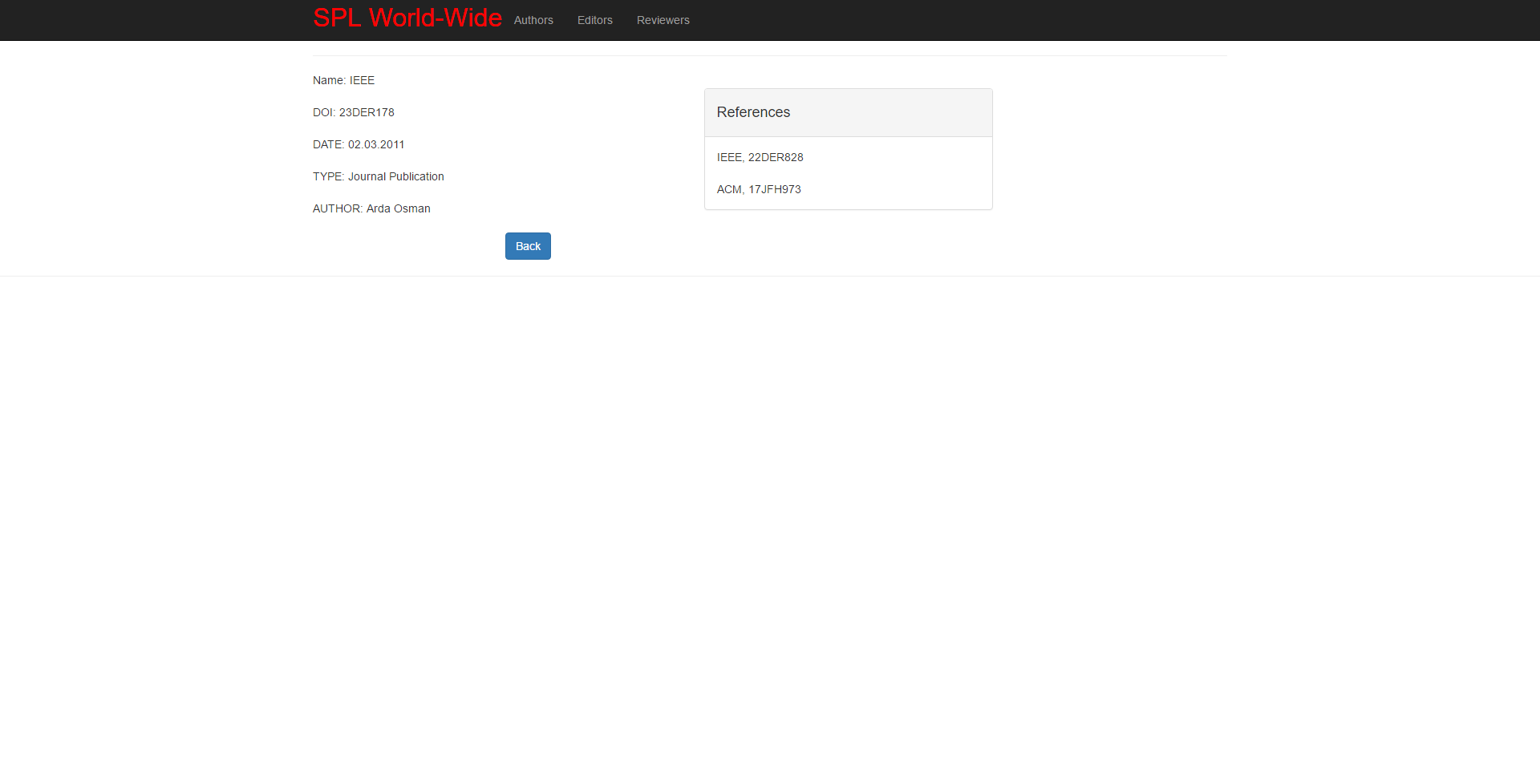
**Description:** When the user enters a journal name in the search box for journal searches and presses the search button the results are listed as above. Back button allows user to go back to the search screen to make a new search.

**Search for Publication:**

****

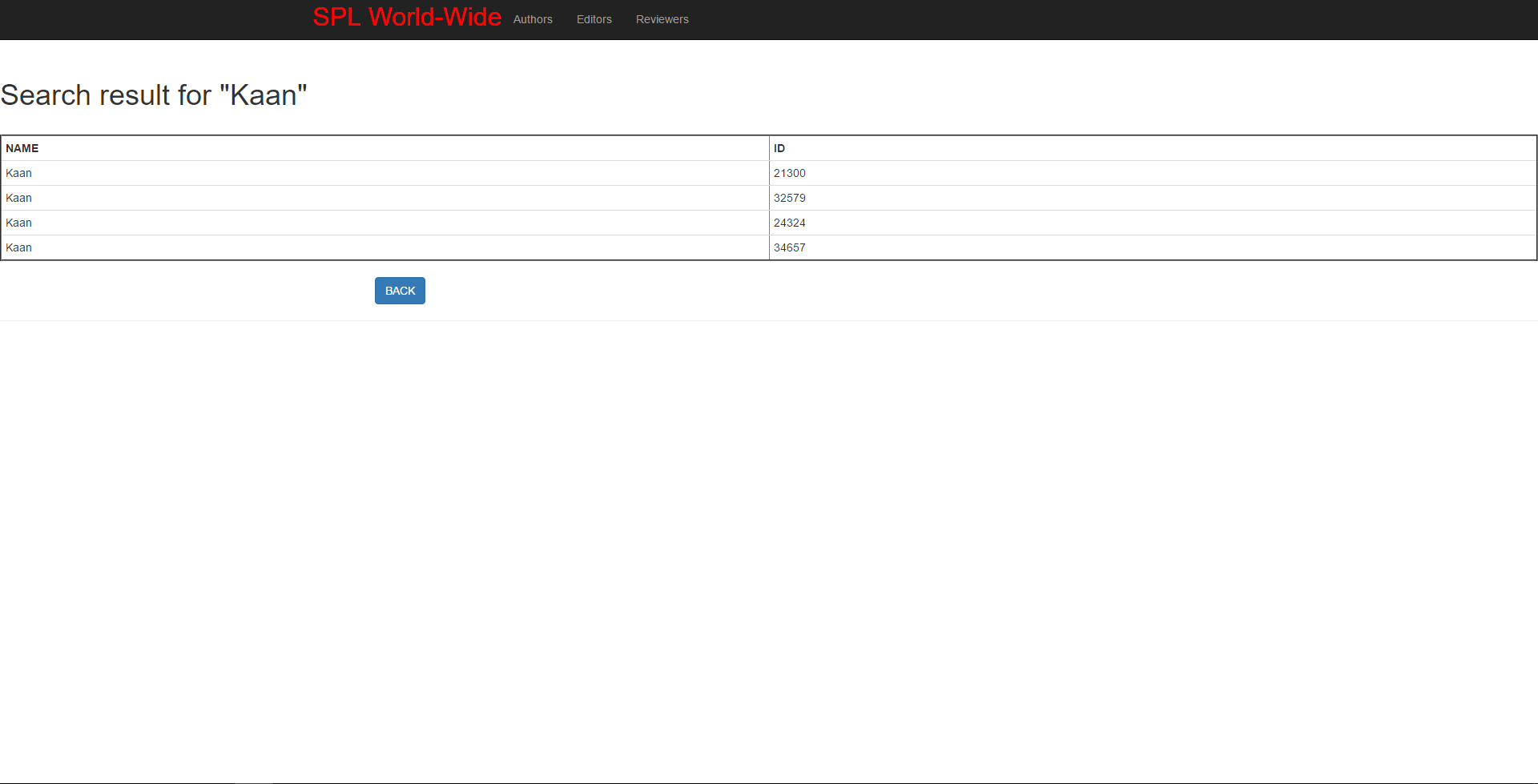
**Description:** When the user enters a publication name in the search box for the publication searches and presses the search button the results are listed as above. If the user clicks the DOI number he can access the Publication page

**Publication Screen:**

****

**Description:** :When the user clicks a DOI number from the results a page for that particular publication appears on the screen. This screen shows detailed information about the particular publication

**User Search:**

****

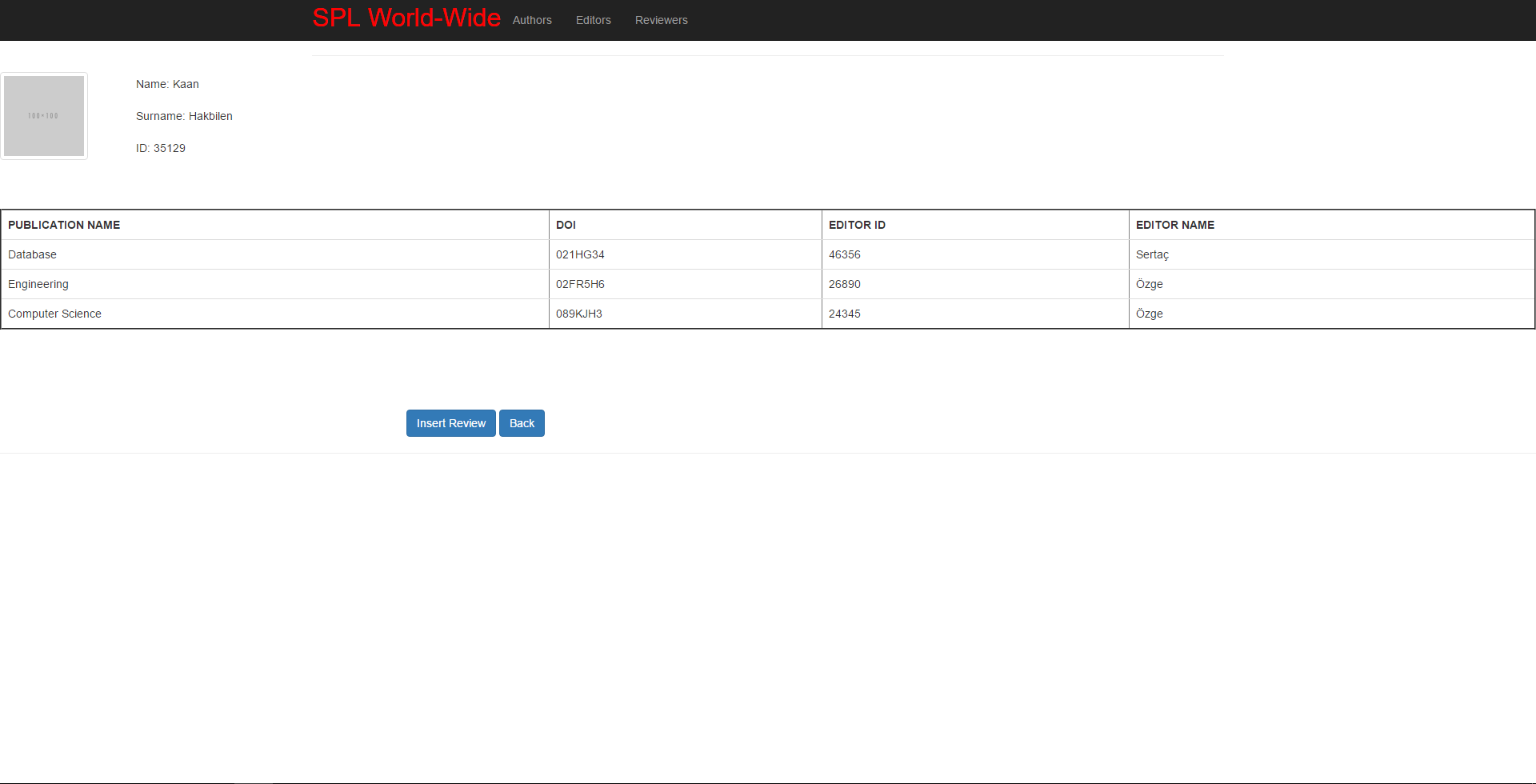
**Description:** When the user enters a user name into the search box for the user searches. The results are listed as above. The user can access a preferred user from the list by clicking the ID number.

**Editor Screen:**

****

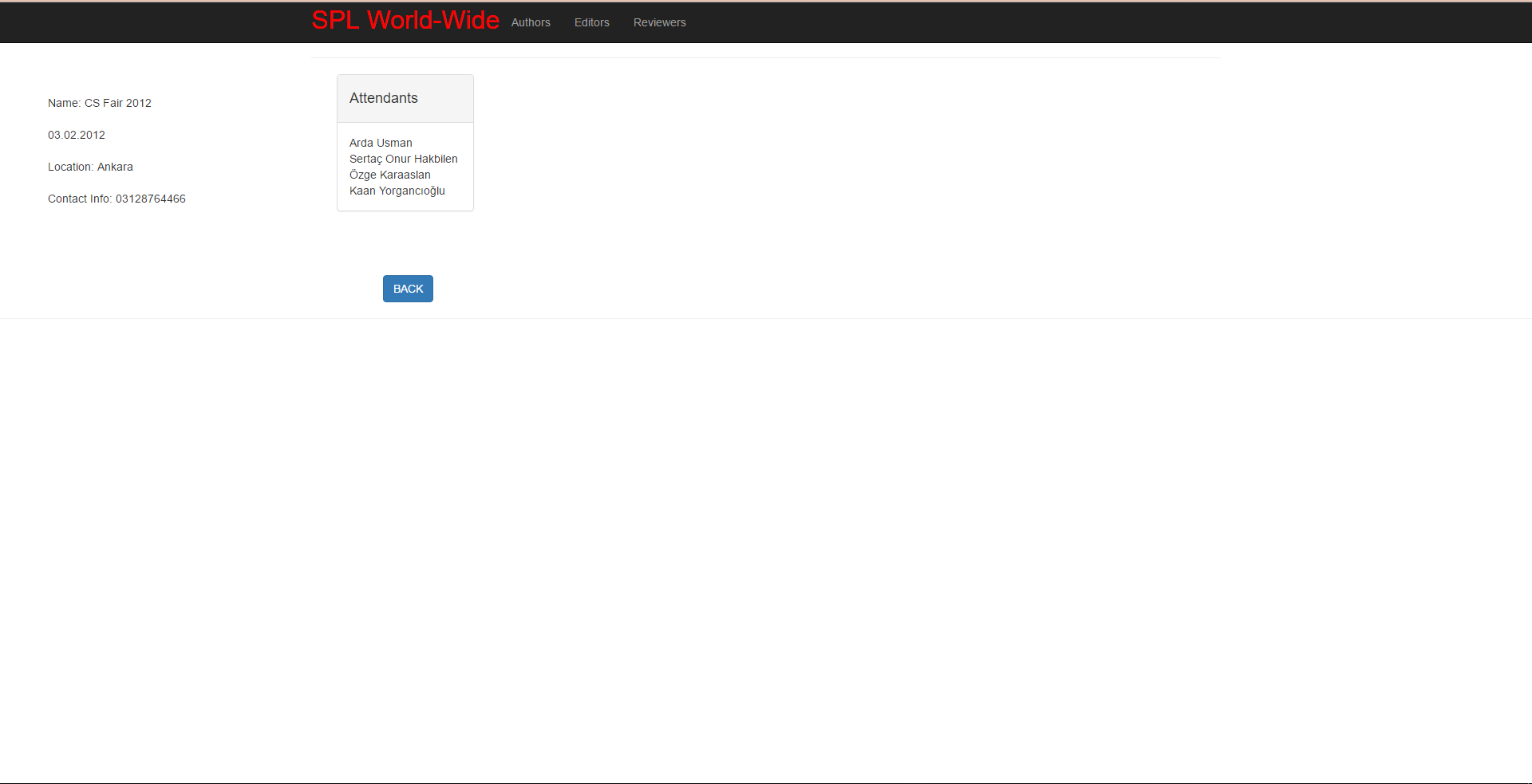
**Description:** When user clicks to a id in the search result table and that id belongs to an editor. A page like above appears on the screen. It allows editor to assign a new assignment, look submitted publications.

**Reviewer Screen:**

****

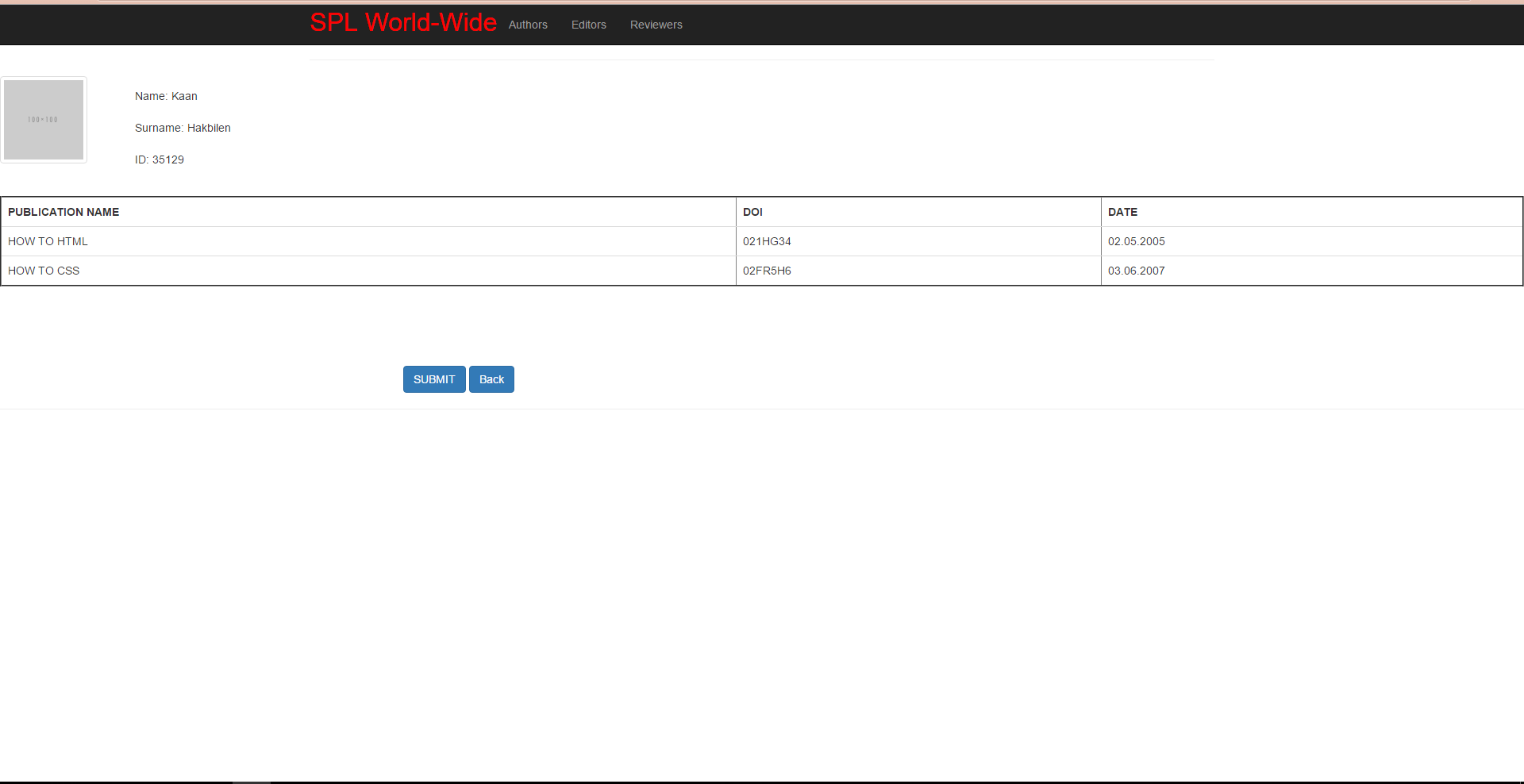
**Description:** When the user clicks to an id from the search results an that id belongs to a reviewer this page appears on the screen. A reviewer can add a new review to the system via this page.

**Conference Screen:**

****

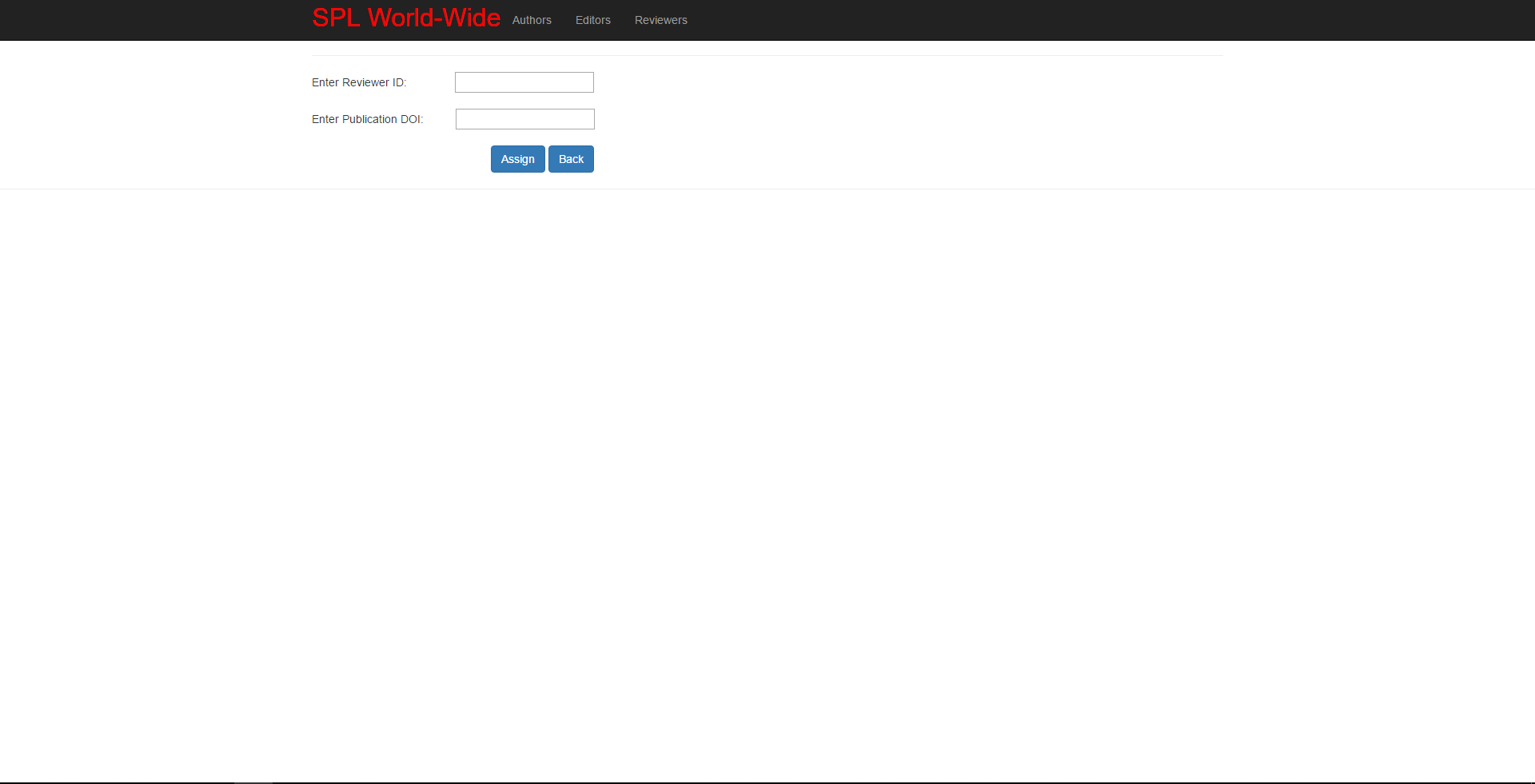
**Description:**When the user clicks to a conference name from a list on the editor page this screen occurs

**Author Screen:**

****

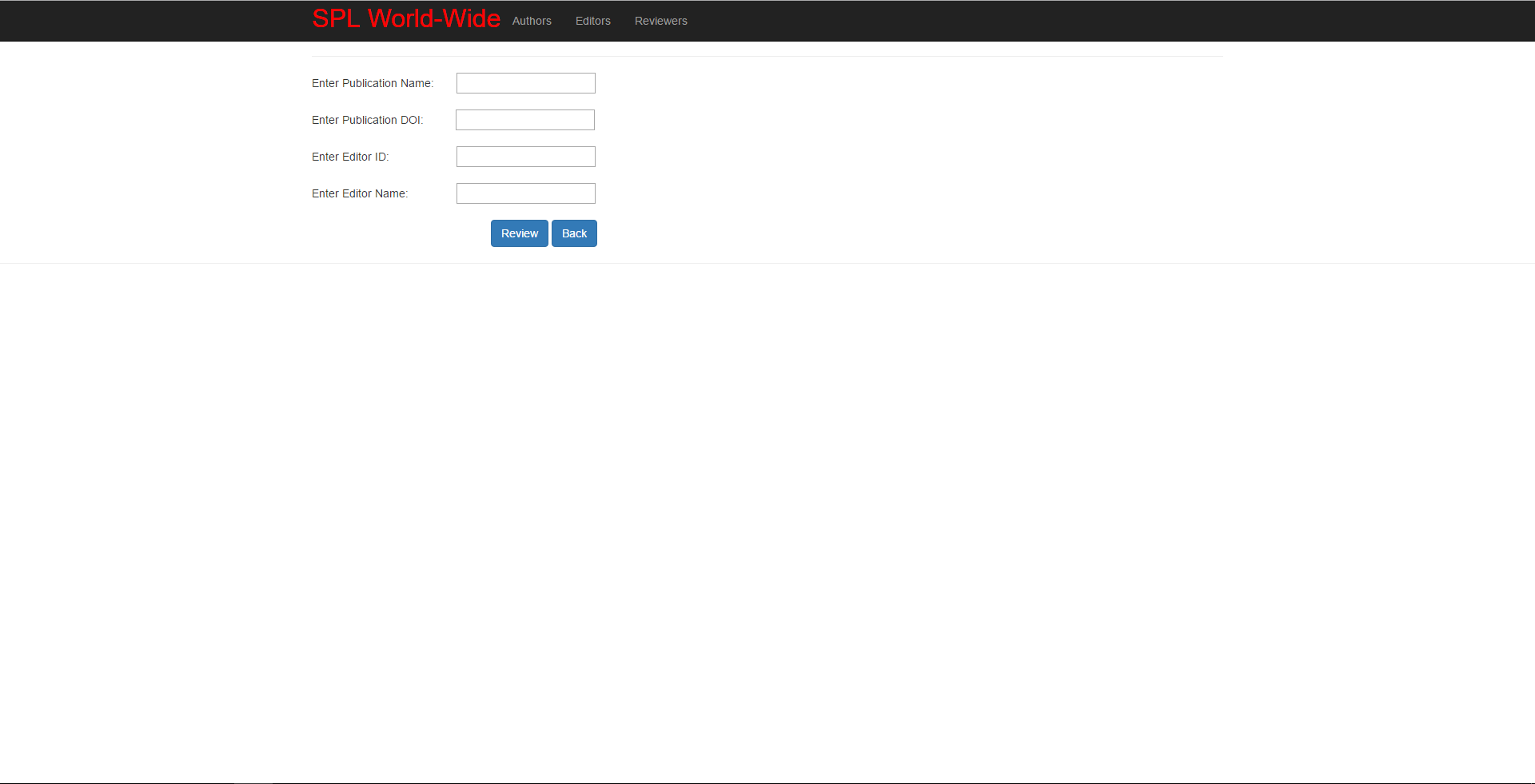
**Description:** When the user clicks to an id in the results table and that id belongs to an author this page shows up. An author can add a new work of his into the system from this page.

**Make Assignment:**



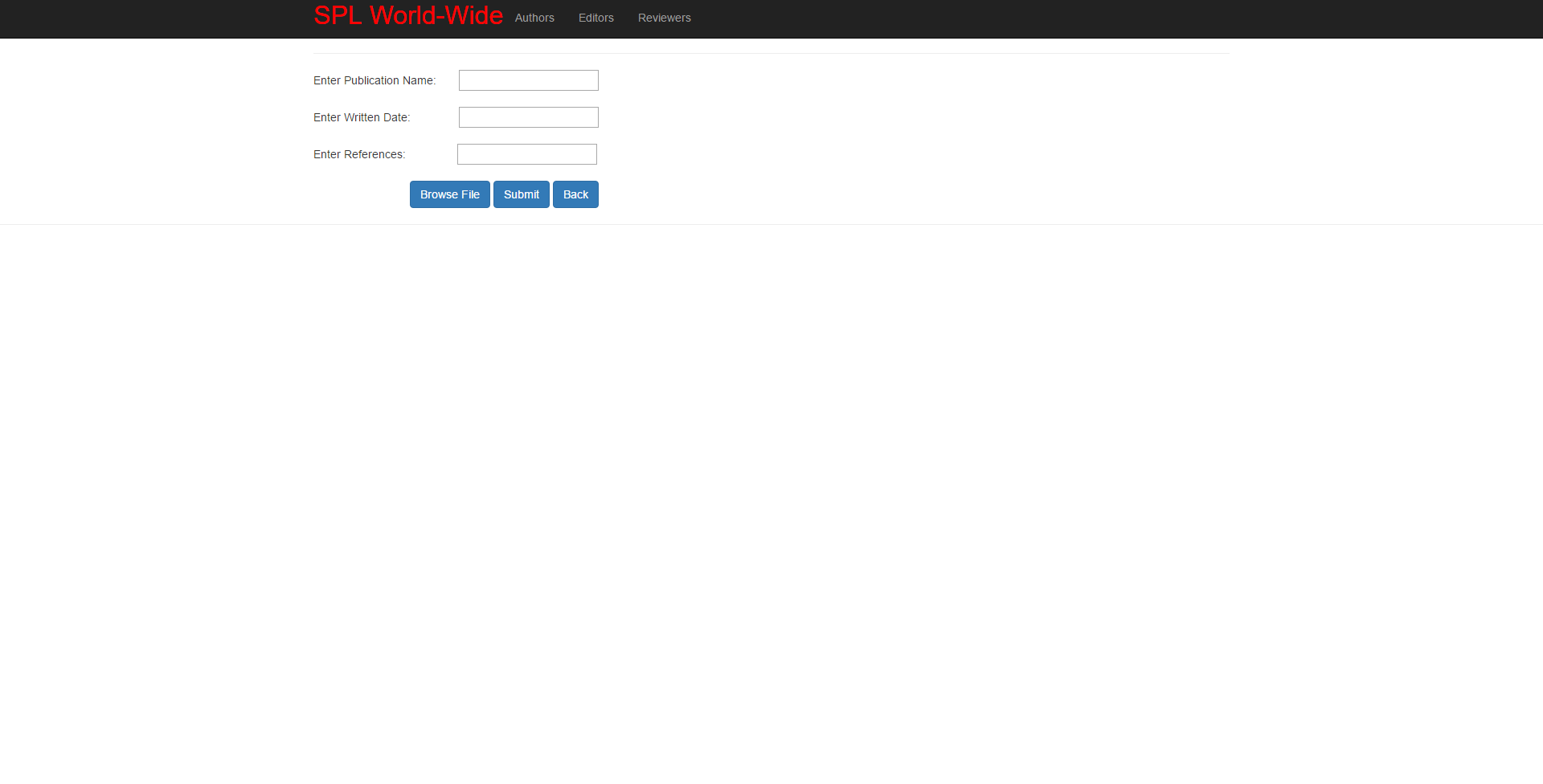
**Description:** When the editor clicks to the make assignment button this screen shows up that allows editor to make a new assignment to a reviewer

**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

**Submit Publication:**

****

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