



CS 353 - Database Systems

**Project Title: Scientific Papers Data Management
System**

Project Design Report

Group No: 2

Busra Arabaci - 21401688 - Section 1

Gerard Hysa - 21503649 - Section 2

Volkan Sevinc - 21401106 - Section 1

Xhoana Aliu - 21500429 - Section 2

Spring 2018

1 Revised E/R Model (15p)	3
2 Relation Schemas(25p)	4
3 Functional Components(10p)	20
3.1 Use Cases/ Scenarios	20
3.2 Algorithms	22
3.3 Data Structures	23
4 User Interface Design and Corresponding SQL Statements(35)	24
5 Advanced Database Components(15p)	53
5.1 Views	53
5.1.1 Editor's View	53
5.1.2 Reviewer's View	53
5.1.3 Select Co-Authors View	53
5.1.4 Select Conference/Journal View	54
5.2 Stored Procedures	54
5.3 Reports	55
5.4 Triggers	55
5.5 Constraints	55
6 Implementation Plan	56

1 Revised E/R Model

- Publication entity is changed and renamed to Journal entity.
- Chair of conference is removed after assessed as redundant since it is an editor.
- Subscribe relation is connected directly with the journal since all of the user types can subscribe and are added the start and end date attributes.
- Has relations are added between journal and conference entities with editor and reviewer entities.
- Rate relation is added between user and paper together with the rating_points attribute. (This is the additional feature)
- Decide relation is added between editor and paper entities together with decision attribute.
- The relation between Reviewer and Editor is changed to assign relation. connected with paper in total participation from the paper part.
- User_login weak entity is removed and the password attribute is added to the User entity
- The relation between Institution and Author entities is made total participation from both sides.
- Has relation between Institution and Paper entities is removed.
- All the count attributes are removed from the diagram since they are redundant.
- Role attribute is added to the User entity to distinguish from the types of users when they log in. Also date of birth, photo attributes are added.
- Submit to Conference and Submit to Journal relations related with the Paper entity are added.
- Total participation on the side of Paper entity is added for review relation, write relation and decide relation.
- In the Paper entity the file attribute is added.
- Author id, editor id and reviewer id are added respectively to their entities.

2 Relation Schemas

User(email_address, name, lastname, password, date_of_birth, age, photo, role)

Author(email_address, name, lastname, password, date_of_birth, age, photo, role, author_id, avg_citations_per_paper, webpage)

Unique(author_id, webpage)

Reviewer(email_address, name, lastname, password, date_of_birth, age, photo, role, reviewer_id, webpage)

Unique(reviewer_id, webpage)

Editor(email_address, name, lastname, password, date_of_birth, age, photo, role, editor_id, webpage)

Unique(editor_id, webpage)

Paper(paper_id, title, abstract, date_of_publication, index_term, file)

Institution(institution_name, institution_webpage, about, avg_citations)

Unique(webpage)

Conference(conference_name, date, location, award_count, description)

Journal(ISSN, year_of_publication, name)

write(email_address, paper_id)

FK: email_address references to Author

FK: paper_id references to Paper

review(email_address, paper_id, review_content)

FK: email_address references to Reviewer

FK: paper_id references to Paper

decide(paper_id, email_address, decision)

FK: paper_id references to Paper

FK: email_address references to Editor

assign(paper_id, reviewer_email, editor_email)

FK: paper_id references to Paper

FK: reviewer_email references to Reviewer(email_address)

FK: editor_email references to Editor(email_address)

cites(paper_title, reference_title)

FK: paper_title references to Paper

FK: reference_title references to Paper(paper_title)

submit_to_conference(paper_id, conference_name, date, location)

FK: paper_id references to Paper

FK: conference_name references to Conference

submit_to_journal(paper_id, ISSN)
 FK: paper_id references to Paper
 FK: ISSN references to Journal
 collaborate(institution_name, collaborative_inst_name)
 FK: institution_name references to Institution
 FK: collaborative_inst_name references to Institution(institution_name)
 comment(email_address, paper_id, comment_content)
 FK: email_address references to User
 FK: paper_id references to Paper
 subscribe(email_address, ISSN, start_date, end_date)
 FK: email_address references to User
 FK: ISSN references to Journal
 has_author(email_address, institution_name)
 FK: email_address references to Author
 FK: institution_name references to Institution
 rate(email_address, paper_id, rating_points)
 FK: email_address references to User
 FK: paper_id references to Paper
 journal_has_editor (ISSN, email_address)
 FK: ISSN references to Journal
 FK: email_address references to Editor
 conference_has_editor (conference_name, date, location, email_address)
 FK: email_address references to Editor
 FK: conference_name, date, location reference to Conference
 journal_has_reviewer(ISSN, email_address)
 FK: ISSN references to Journal
 FK: email_adress references to Reviewer
 conference_has_reviewer(conference_name, date, location, email_address)
 FK: conference_name references to Conference
 FK: date references to Conference
 FK: location references to Conference
 FK: email_address references to Reviewer

User

Relational Model

User(email_address, name, lastname, password, date_of_birth, age, photo, role)

Functional Dependencies

email_address → name, lastname, password

Candidate Keys

{{email_address}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE user {  
    email_address          varchar(40) PRIMARY KEY,  
    name                   varchar(40) NOT NULL,  
    lastname               varchar(40) NOT NULL,  
    password               varchar(10) NOT NULL,  
    date_of_birth          date        NOT NULL,  
    age                    int          NOT NULL,  
    photo                  BLOB,  
    role                   varchar(40) NOT NULL  
};
```

Author

Relational Model

Author(email_address, name, lastname, password, date_of_birth, age, photo, role, avg_citations_per_paper, webpage)

Functional Dependencies

email_address → name, lastname, password, publication_year,
avg_citations_per_paper, author_id, webpage

Candidate Keys

{(email_address)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE author {  
    email_address          varchar(40) PRIMARY KEY,  
    name                   varchar(40) NOT NULL,  
    lastname               varchar(40) NOT NULL,  
    password               varchar(10) NOT NULL,  
    date_of_birth          date         NOT NULL,  
    age                    int          NOT NULL,  
    photo                  BLOB,  
    role                   varchar(40) NOT NULL  
    avg_citations_per_paper int,  
    author_id              int          UNIQUE,  
    webpage                varchar(40)  
};
```

Reviewer

Relational Model

Reviewer(email_address, reviewer_id, name, lastname, password, date_of_birth, age, photo, role, webpage)

Functional Dependencies

email_address → name, lastname, password, webpage

Candidate Keys

{(email_address)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE reviewer {  
    email_address          varchar(40) PRIMARY KEY,  
    name                   varchar(40) NOT NULL,  
    lastname               varchar(40) NOT NULL,  
    password               varchar(10) NOT NULL,  
    date_of_birth          date       NOT NULL,  
    age                    int        NOT NULL,  
    photo                  BLOB,  
    role                   varchar(40) NOT NULL,  
    reviewer_id            int        NOT NULL,  
    webpage                varchar(40)  
};
```


Editor

Relational Model

Editor(email_address, editor_id, name, lastname, password, date_of_birth, age, photo, role, webpage)

Functional Dependencies

email_address → name, lastname, password, webpage

Candidate Keys

{(email_address)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE editor {  
    email_address          varchar(40) PRIMARY KEY,  
    name                   varchar(40) NOT NULL,  
    lastname               varchar(40) NOT NULL,  
    password               varchar(10) NOT NULL,  
    date_of_birth          date       NOT NULL,  
    age                    int        NOT NULL,  
    photo                  BLOB,  
    role                   varchar(40) NOT NULL  
    editor_id              int        UNIQUE,  
    webpage                varchar(40) UNIQUE  
};
```

Paper

Relational Model

Paper(paper_id, title, abstract, date_of_publication, index_term, file)

Functional Dependencies

paper_id \rightarrow title, abstract, date_of_publication, index_term

Candidate Keys

{(paper_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE paper {  
    paper_id      varchar(40) PRIMARY KEY,  
    title         varchar(40) NOT NULL,  
    abstract      varchar(40) NOT NULL,  
    date_of_publication int    NOT NULL,  
    index_term    varchar(40),  
    file          BLOB  
};
```

Institution

Relational Model

Institution(institution_name, institution_webpage, about, avg_citations)

Functional Dependencies

institution_name \rightarrow institution_webpage, about, avg_citations

Candidate Keys

{(institution_name)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE institution {  
    institution_name  varchar(40) PRIMARY KEY,  
    institution_webpage varchar(40) NOT NULL,  
    about            varchar(40),  
    avg_citations    int  
};
```

Conference

Relational Model

Conference(conference_name, date, location, award_count, description)

Functional Dependencies

conference_name → date, location, award_count, description

Candidate Keys

{{conference_name}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE conference {  
    conference_name  varchar(40)  PRIMARY KEY,  
    date             int NOT NULL,  
    location         varchar(20)  NOT NULL,  
    about            varchar(40),  
    award_count      int,  
    description      varchar(40)  
};
```

Journal

Relational Model

Journal(ISSN, year_of_publication, name)

Functional Dependencies

ISSN → title, year_of_publication

Candidate Keys

{{ISSN}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE journal {  
    ISSN             varchar(40)  PRIMARY KEY,  
    year_of_publication int      NOT NULL,  
    name             varchar(40)  NOT NULL,  
};
```

Review

Relational Model

review(email_address, paper_id, review_content)

Functional Dependencies

ISSN \rightarrow title, year_of_publication

Candidate Keys

{(ISSN)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE review {  
    email_address    varchar(40)  PRIMARY KEY,  
    paper_id         varchar(40)  PRIMARY KEY,  
    review_content   varchar(300) NOT NULL,  
    FOREIGN KEY (email_address) references Reviewer,  
    FOREIGN KEY (paper_id) references Paper  
};
```

Assign

Relational Model

assign(paper_id,reviewer_email,editor_email)

Functional Dependencies

paper_id, reviewer_email \rightarrow editor_email

Candidate Keys

{(paper_id,reviewer_email)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE assign {  
    paper_id         varchar(40)  PRIMARY KEY,  
    reviewer_email   varchar(40)  PRIMARY KEY,  
    editor_email     varchar(40)  PRIMARY KEY  
    FOREIGN KEY (paper_id) references Paper,  
    FOREIGN KEY (reviewer_email) references Reviewer(email_address),  
    FOREIGN KEY (editor_email) references Editor(email_address)  
};
```

Submit to Journal

Relational Model

submit_to_journal(paper_id, ISSN)

Functional Dependencies

No dependencies.

Candidate Keys

{{paper_id,ISSN}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE submit_to_journal {  
    paper_id          varchar(40)  PRIMARY KEY,  
    ISSN              int          PRIMARY KEY,  
    FOREIGN KEY (paper_id) references Paper,  
    FOREIGN KEY (ISSN) references Journal  
};
```

Submit to Conference

Relational Model

submit_to_conference(paper_id,conference_name, date, location)

Functional Dependencies

No dependencies.

Candidate Keys

{{paper_id,conference_name, date, location}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE submit_to_conference {  
    paper_id          varchar(40)  PRIMARY KEY,  
    conference_name    varchar(40)  PRIMARY KEY,  
    date              date          PRIMARY KEY,  
    location           varchar(40)  PRIMARY KEY,  
    FOREIGN KEY (paper_id) references Paper,  
    FOREIGN KEY (conference_name) references Conference  
};
```

Write

Relational Model

write(email_address, paper_id)

Functional Dependencies

No dependencies.

Candidate Keys

{(email_address, paper_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE write {  
    email_address    varchar(40) PRIMARY KEY,  
    paper_id         varchar(40) PRIMARY KEY,  
    FOREIGN KEY (email_address) references Reviewer,  
    FOREIGN KEY (paper_id) references Paper  
};
```

Decide

Relational Model

decide(paper_id, email_address, decision)

Functional Dependencies

No dependencies.

Candidate Keys

{(paper_id, email_address, decision)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE decide {  
    paper_id         varchar(40) PRIMARY KEY,  
    email_address    varchar(40) PRIMARY KEY,  
    decision         varchar(10),  
    FOREIGN KEY (paper_id) references Paper,  
    FOREIGN KEY (email_address) references Editor  
};
```

Cites

Relational Model

cites(paper_title, reference_title)

Functional Dependencies

No dependencies.

Candidate Keys

{(paper_title, reference_title)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE cites {  
    paper_title      varchar(40)  PRIMARY KEY,  
    reference_title   varchar(40)  PRIMARY KEY,  
    FOREIGN KEY (paper_title) references Paper,  
    FOREIGN KEY (reference_title) references Paper  
};
```

Collaborate

Relational Model

collaborate(institution_name, collaborative_inst_name)

Functional Dependencies

No dependencies.

Candidate Keys

{(institution_name, collaborative_inst_name)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE collaborate {  
    institution_name      varchar(40)  PRIMARY KEY,  
    collaborative_inst_name  varchar(40)  PRIMARY KEY,  
    FOREIGN KEY (institution_name) references Institution,  
    FOREIGN KEY (collaborative_inst_name) references Institution  
};
```

Comment

Relational Model

comment(email_address, paper_id, comment_content)

Functional Dependencies

No dependencies.

Candidate Keys

{(email_address, paper_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE comment {  
    email_address    varchar(40) PRIMARY KEY,  
    paper_id         int    PRIMARY KEY,  
    comment_content  varchar(120),  
    FOREIGN KEY (email_address) references User,  
    FOREIGN KEY (paper_id) references Paper  
};
```

Subscribe

Relational Model

subscribe(email_address, ISSN, start_date, end_date)

Functional Dependencies

No dependencies.

Candidate Keys

{(email_address, ISSN)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE subscribe {  
    email_address    varchar(40) PRIMARY KEY,  
    ISSN             int    PRIMARY KEY,  
    start_date       date    NOT NULL,  
    end_date         date    NOT NULL  
    FOREIGN KEY (email_address) references User,  
    FOREIGN KEY (ISSN) references Journal  
};
```


Rate

Relational Model

rate(email_address, paper_id, rating_points)

Functional Dependencies

No dependencies.

Candidate Keys

{(email_address, paper_id)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE rate {  
    email_address    varchar(40) PRIMARY KEY,  
    paper_id         int    PRIMARY KEY,  
    rating_points    int,  
    FOREIGN KEY (email_address) references User,  
    FOREIGN KEY (paper_id) references Paper  
};
```

Conference has editor

Relational Model

conference_has_editor(conference_name, date, location, email_address)

Functional Dependencies

No dependencies.

Candidate Keys

{(conference_name, date, location, email_address)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE conference_has_editor {  
    conference_name  varchar(40) PRIMARY KEY,  
    email_adress     varchar(40) PRIMARY KEY,  
    date             date    PRIMARY KEY,  
    location         varchar(40) PRIMARY KEY,  
    FOREIGN KEY (conference_name) references Conference,  
    FOREIGN KEY (date) references Conference,  
    FOREIGN KEY (location) references Conference,  
    FOREIGN KEY (email_address) references Editor  
};
```

Conference has reviewer

Relational Model

conference_has_reviewer(conference_name,date,location,email_address)

Functional Dependencies

No dependencies.

Candidate Keys

{{conference_name, date, location, email_address}}

Normal Form

BCNF

Table Definition

```
CREATE TABLE conference_has_reviewer {  
    conference_name  varchar(40)  PRIMARY KEY,  
    email_adress     varchar(40)  PRIMARY KEY,  
    date             date          PRIMARY KEY,  
    location         varchar(40)  PRIMARY KEY,  
    FOREIGN KEY (conference_name) references Conference,  
    FOREIGN KEY (date) references Conference,  
    FOREIGN KEY (location) references Conference,  
    FOREIGN KEY (email_address) references Reviewer  
};
```

Journal has editor

Relational Model

journal_has_editor(ISSN, email_address)

Functional Dependencies

No dependencies.

Candidate Keys

{(ISSN, email_address)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE journal_has_editor {  
    ISSN            int    PRIMARY KEY,  
    email_adress    varchar(40) PRIMARY KEY,  
    FOREIGN KEY (ISSN) references Journal,  
    FOREIGN KEY (email_address) references Editor  
};
```

Journal has reviewer

Relational Model

journal_has_reviewer(ISSN, email_address)

Functional Dependencies

No dependencies.

Candidate Keys

{(ISSN, email_address)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE journal_has_reviewer {  
    ISSN            int    PRIMARY KEY,  
    email_adress    varchar(40) PRIMARY KEY,  
    FOREIGN KEY (ISSN) references Journal,  
    FOREIGN KEY (email_address) references Reviewer  
};
```

Has Author

Relational Model

has_author(email_address, institution_name)

Functional Dependencies

No dependencies.

Candidate Keys

{(email_address, institution_name)}

Normal Form

BCNF

Table Definition

```
CREATE TABLE has_author {  
    institution_name    varchar(40) PRIMARY KEY,  
    email_address       varchar(40) PRIMARY KEY,  
    FOREIGN KEY (email_address) references Author,  
    FOREIGN KEY (institution_name) references Institution  
};
```

3 Functional Components

Below we will discuss the functional components in three sections.

3.1 Use Cases/ Scenarios

User:

- User can login to the system.
- User can subscribe to journals.
- User can comment to papers.
- User can rate the papers.
- User can view list of all the papers, list of all the institutions, list of all the authors, list of all the journals and list of all the conferences.
- User can view detailed information about a specific paper.
- User can view comments of the papers.
- User can view detailed information about a specific author.
- User can view detailed information about a specific institution.
- User can search papers by writing keywords.
- User can download a paper.

NOTE: Author, editor and reviewer have the same use cases as the user. In addition to these use cases they have some other use cases explained below.

Author:

- Author can upload a paper by filling in the required information about the paper.
- Author can submit her/his paper to journals or conferences.
- Author can view and read reviews of the paper made by reviewers.
- Author can view the list of her/his uploaded/submitted papers.

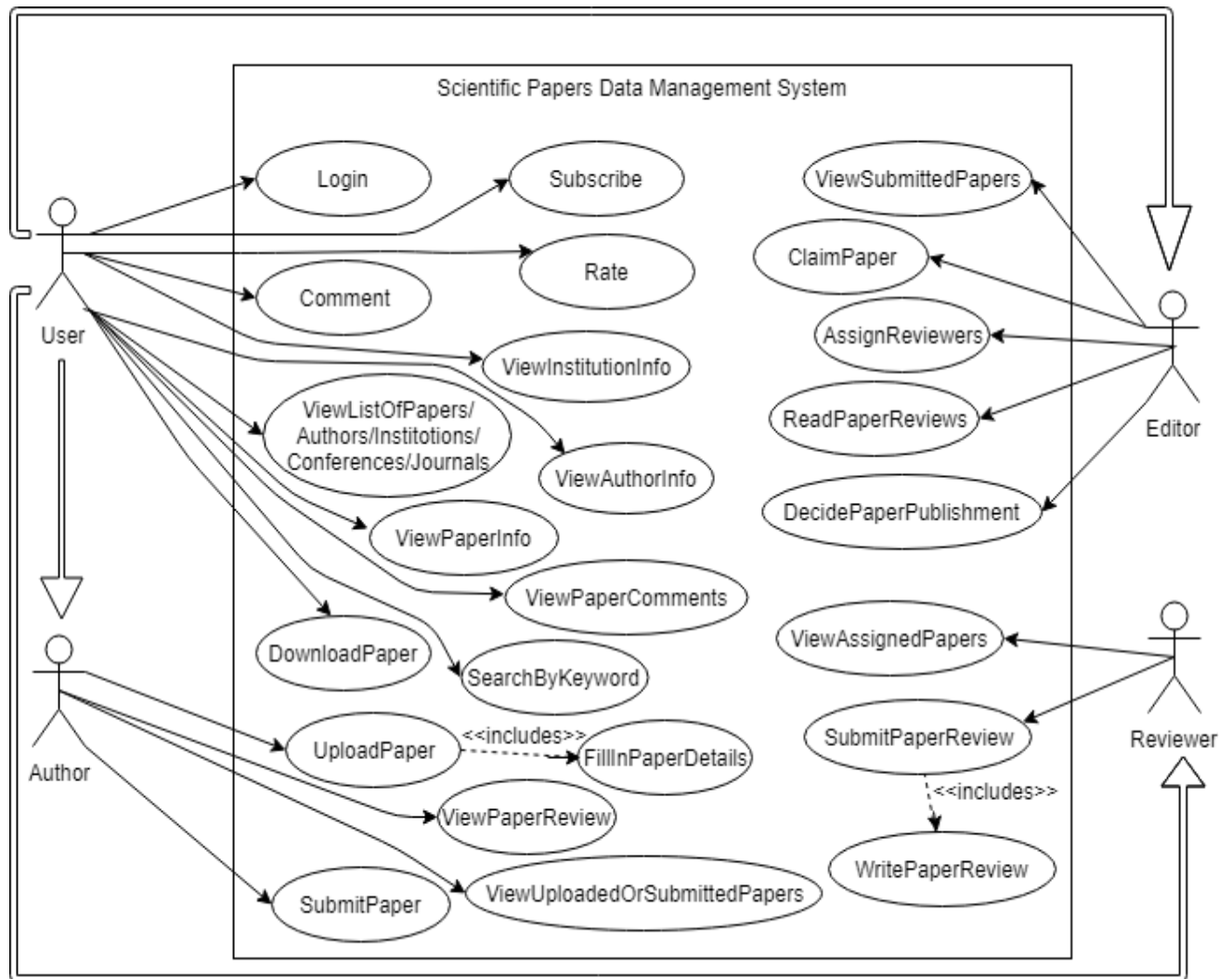
Editor:

- Editor can view the submitted papers to the journal/conference s/he is part of.
- Editor can claim papers to evaluate.
- Editor can assign reviewers to the claimed papers.
- Editors can assess and read the reviews of the papers s/he has claimed.
- Editors can decide about the publishment of a claimed paper paper.

Reviewer:

- Reviewer can get the assigned paper.

- Reviewer can submit the review of the assigned paper to the editors and authors by writing it in our system.



3.2 Algorithms

3.2.1 Paper-Related Algorithms

Every paper is uploaded and submitted by an author. It is very important to have information regarding the submission and upload processes of the papers as a paper is editable when uploaded but cannot be edited further once submitted to a journal. Also a paper can also be submitted to one journal and no more.

When the paper is uploaded by author, it can be edited by the author until it is submitted. No one else in the system can see these uploaded papers as they are subject to change. An uploaded paper cannot be cited and commented and therefore cannot affect *citation_count* attribute of the author or the institution. When the author decides to submit the paper, he/she selects a journal to submit to and cannot interfere further. The edit option should be disabled once a paper is submitted as editing already submitted papers will create problems on journal side.

3.2.2 Editor and Reviewer Related Algorithms

Once a paper is submitted, an editor will be able to see it in the “Submitted Papers” tab of her/his main page. Only editors can see this tab in their home pages. In this page, editors of a journal claim submitted papers. A paper can only be claimed by one editor and claimed papers are transferred from “Submitted Papers” tab to “Claimed Papers” tab. All editors of a journal can see “Submitted Papers” but only the editor who claimed the specific paper will be able to see it in the “Claimed Papers” tab. An editor will be able to assign three reviewers to each claimed paper in order to prevent confusion among reviewers.

A reviewer can only review papers that are assigned to her/him by the editors. Reviewers cannot see submitted papers and cannot edit their reviews once they submit the review.

3.2.3 Algorithms to Meet Logical Requirements

In order to prevent logical errors, some aspects of the system will have some safeguards. For example, submission, editing, reviewing and assignment of papers should be handled carefully.

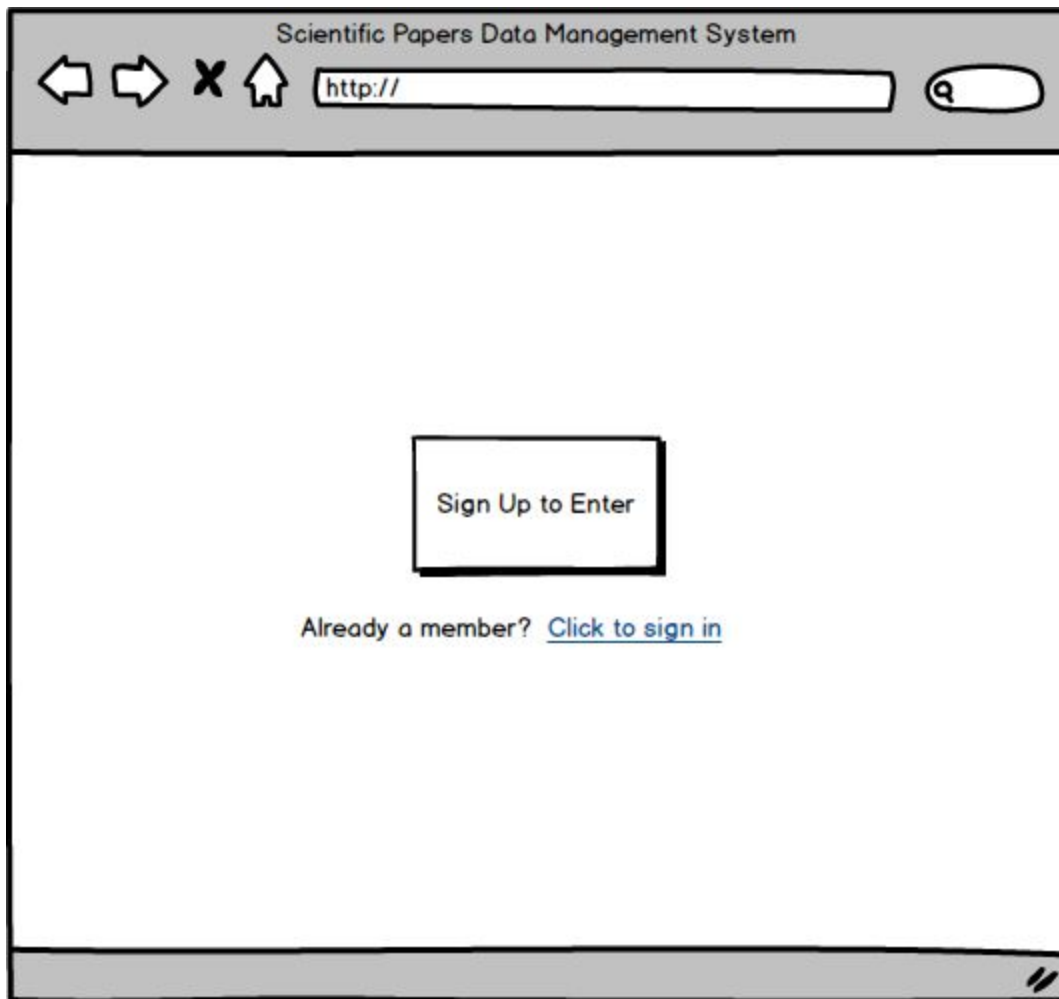
Because it would be impossible to track if submitted papers are edited after submission, editing of such papers will be disabled by the system. From that point, system will constantly change who can see the paper and edit it. Editors cannot claim an already claimed paper in order to prevent such errors. Only three reviewers can be assigned to a paper.

Uploaded papers and papers in the submission cycle cannot be counted as citations or publications yet and cannot be commented on by users in order to prevent possible numerical issues with publication and citation counts of institutions and authors.

3.3 Data Structures

For the attribute domains we use Numeric , String, date, int and BLOB data types in MySQL.

4 User Interface Design and Corresponding SQL Statements



First Screen (Fig.1)

This will be the first screen that a user sees. In order to use our application, the user should sign up or log in if s/he already has an account in our database.

Scientific Papers Data Management System

http://

Create a New Account

Name

Surname

E-mail

Password

Confirm Password

Account Type Subscriber ▼
Author

☒ I agree to the [terms of use](#)

Sign Up Screen (Fig. 2)

If the user wants to sign up s/he will have to fill in all the information in this page. According to the role the user chooses, other fields like “Institution name”, “Journal”, “Conference” will be required to be filled in by the authors, editors and reviewers.

```
INSERT INTO User(name,lastname,email_address, password,role)
VALUES(@name,@lastname,@email_address, @password,@role)
```

Scientific Papers Data Management System

http://

Sign In

E-mail

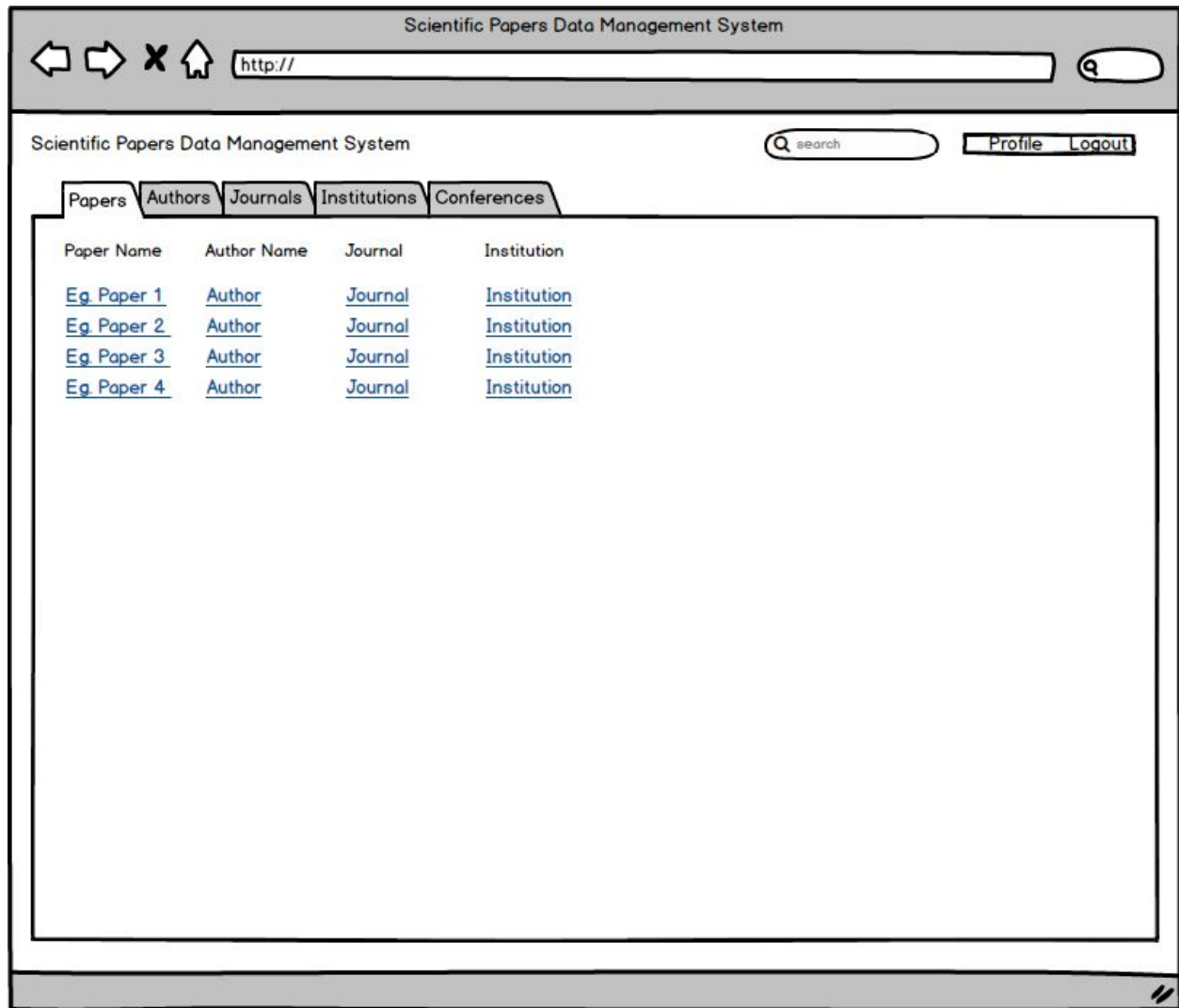
Password

Sign In

Sign In (Fig. 3)

This will be the sign in page and a user will be able to login by using her/his email address and password.

```
SELECT email_address,password
FROM User
WHERE email_address = @email_address and password = @password
```

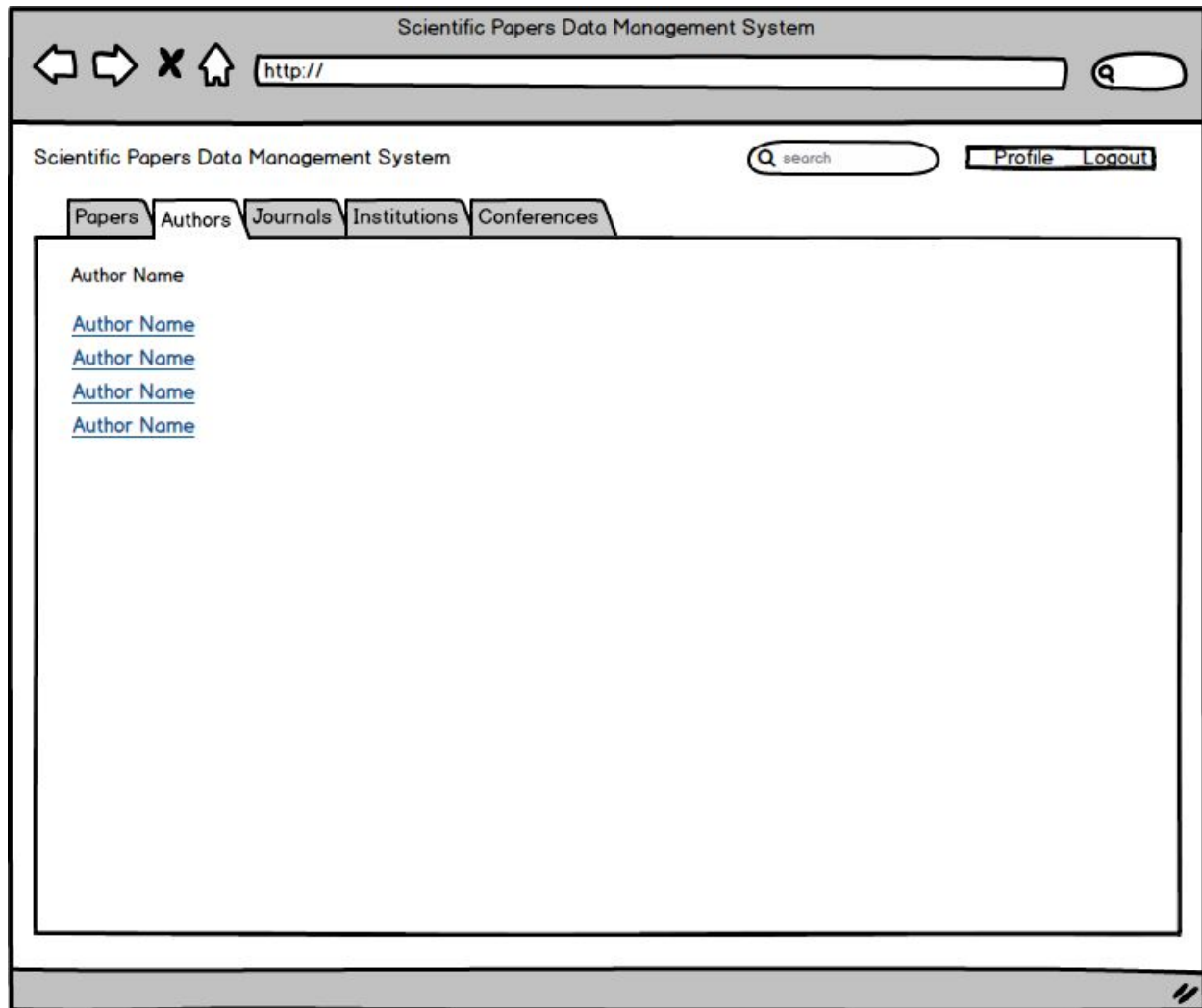


Homepage (Fig. 4)

The homepage of our application will contain the list of all the papers in our database.

Listing the papers:

```
SELECT paper_id, title  
FROM Paper  
ORDER BY title
```

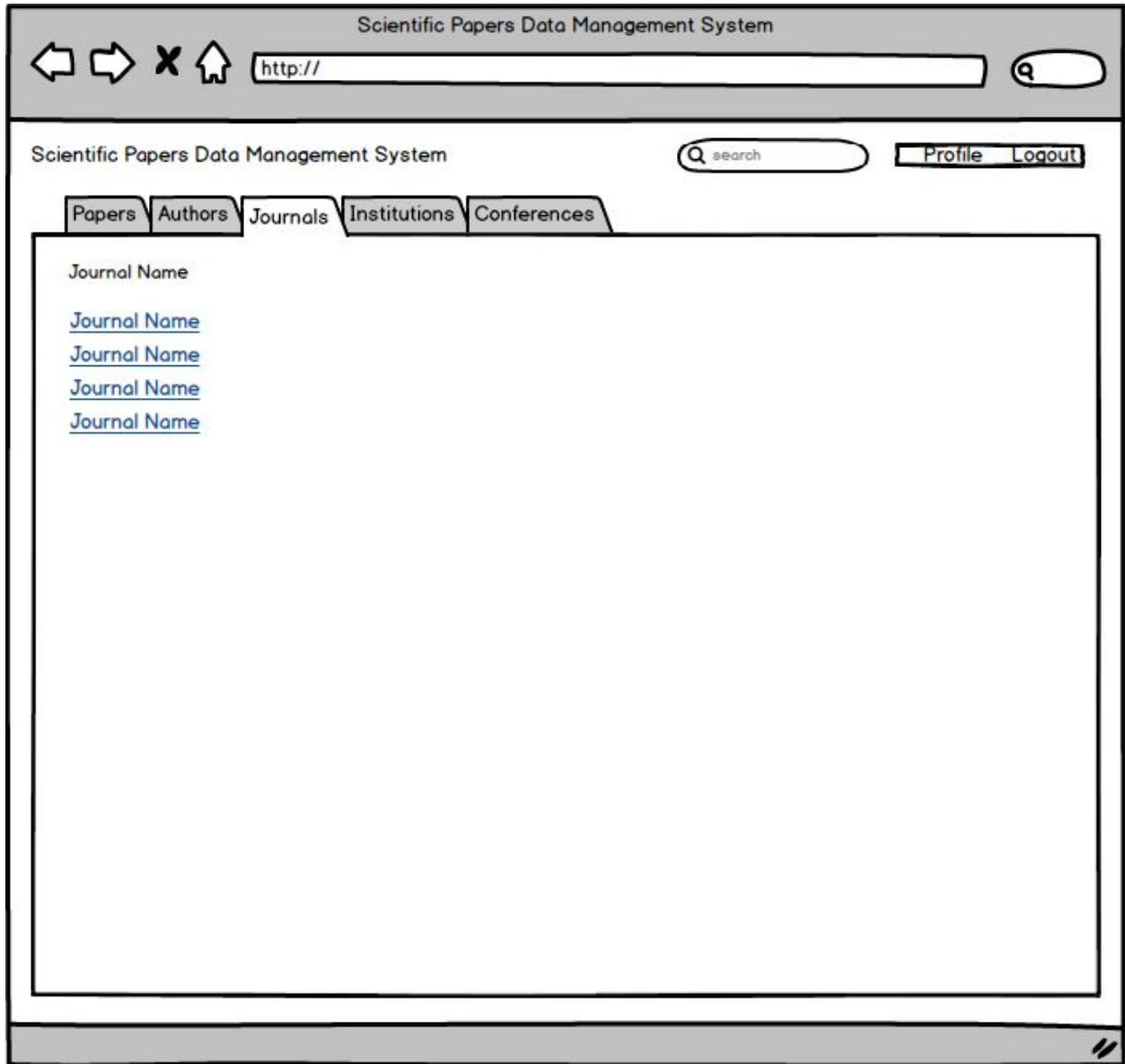


Author's List(Fig.5)

In this tab the names of all the authors will be displayed with the link to their webpages.

Listing the authors:

```
SELECT DISTINCT(name, lastname)
FROM Author
ORDER BY lastname
```

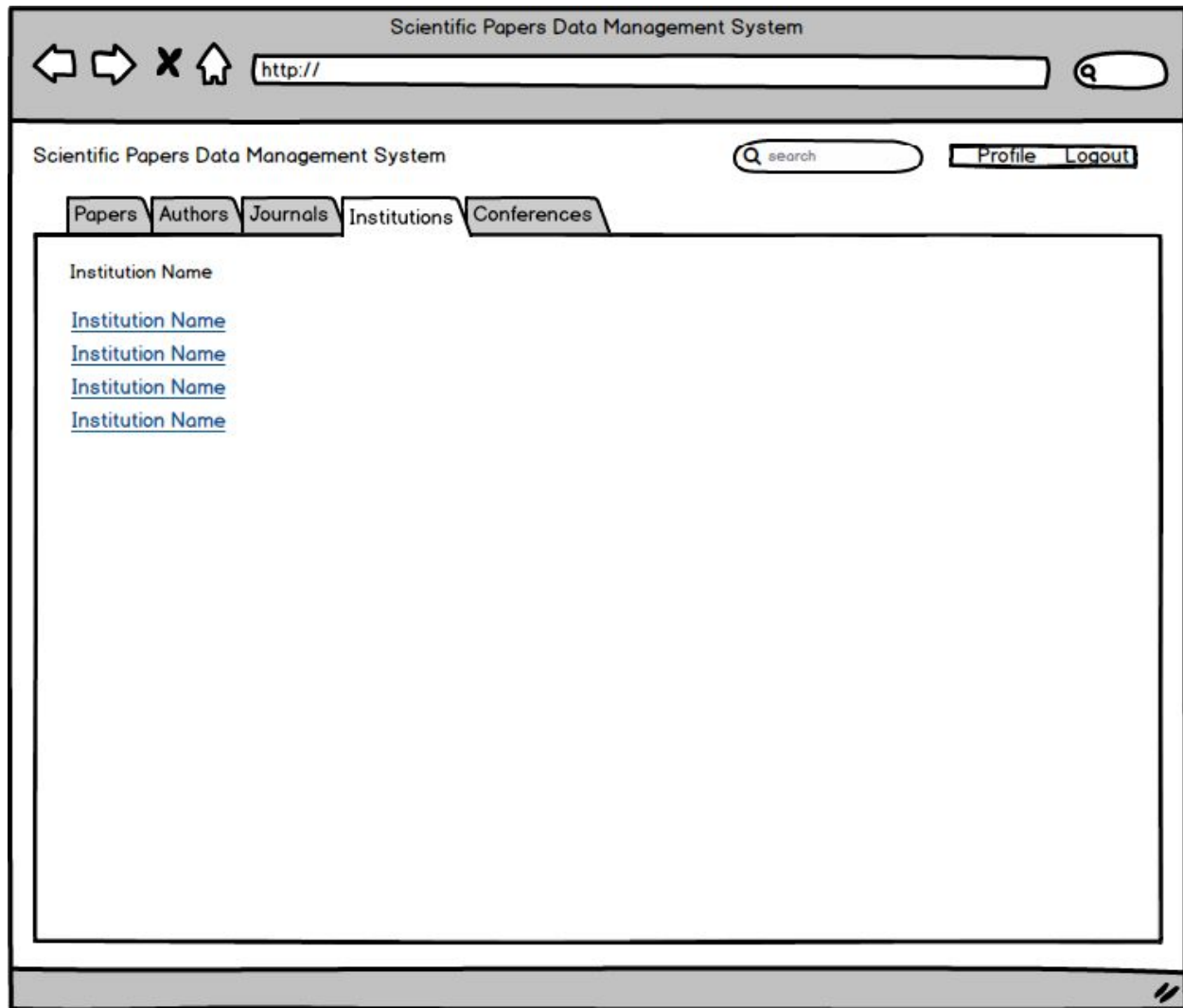


Journal's List(Fig.6)

In this tab the names of all the journals will be displayed with the link to the list of papers in that journal.

Listing the journals:

```
SELECT name  
FROM journal_list
```

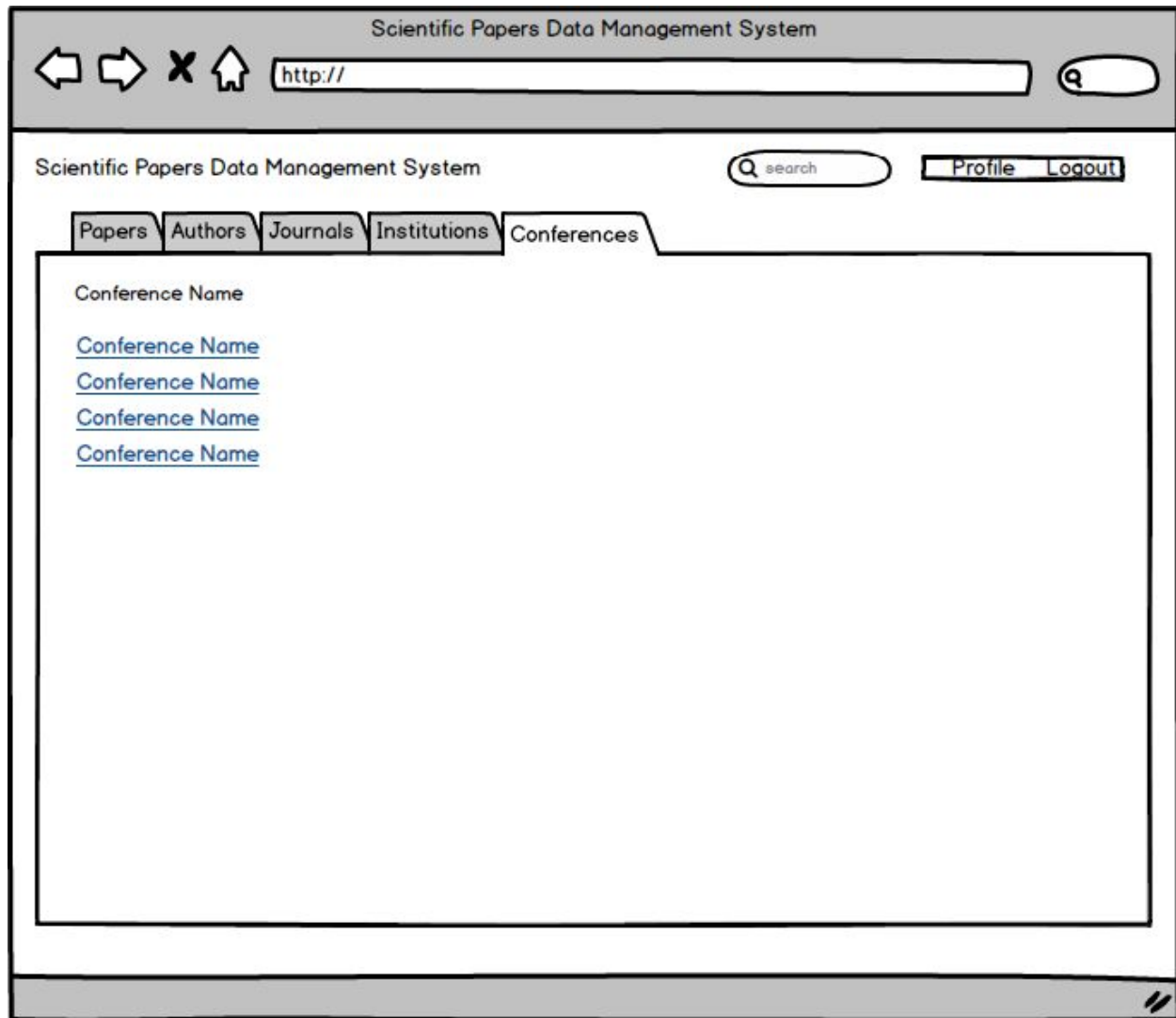


List of Institutions(Fig.7)

In this tab the list of all the institutions is displayed with the respective links to their pages where the details of the institution and the papers are shown.

Listing the institutions:

```
SELECT DISTINCT(institution_name)
FROM Institution
ORDER BY institution_name
```



List of conferences (Fig. 8)

In this tab the list of all the conferences is displayed with the respective links to their pages where the details of the conference and the papers published are shown.

Listing the conferences:

```
SELECT *  
FROM conference_list
```

Searching (all results):

```
CREATE TABLE temp_search  
{  
    keyword    VARCHAR(40),  
};
```



```
INSERT INTO temp_search(keyword)
VALUES(@keyword)
```

```
WITH paper_results(paper_id) AS
  (SELECT DISTINCT(paper_id)
   FROM   paper_terms AS P, temp_search AS T
   WHERE  P.index_term = T.keyword)
WITH author_results(author_name, author_lastname) AS
  (SELECT DISTINCT(name,lastname)
   FROM   Author AS A, temp_search AS T
   WHERE  T.keyword = A.name OR T.keyword = A.lastname)
WITH institution_results(institution_name) AS
  (SELECT DISTINCT(institution_name)
   FROM   Institution AS I, temp_search AS T
   WHERE  T.keyword = I.instituion_name)
WITH journal_results(journal_name) AS
  (SELECT DISTINCT(name)
   FROM   Journal AS J, temp_search AS T
   WHERE  T.keyword = J.name)
WITH conference_results(conference_name) AS
  (SELECT DISTINCT(name)
   FROM   Conference AS C, temp_search AS T
   WHERE  T.keyword = C.conference_name)
```

```
SELECT *
FROM   paper_results, author_results, institution_results, journal_results,
conference_results
```

Displaying search results for papers:

```
SELECT DISTINCT(title)
FROM   paper_results, Paper
WHERE  Paper.paper_id = paper_results.paper_id
```

Displaying search results for authors:

```
SELECT author_name, author_lastname
FROM   author_results
```

Displaying search results for institutions:

```
SELECT institution_name
```

```
FROM      institution_results
```

Displaying search results for journals:

```
SELECT    journal_name
FROM      journal_results
```

Displaying search results for institutions:

```
SELECT    conference_name
FROM      conference_results
```

```
//Then temp_search is deleted
```

```
DROP TABLE temp_search
```

The screenshot shows a web browser window titled "Scientific Papers Data Management System". The address bar shows "http://". The page header includes the system name, a search bar with a magnifying glass icon, and links for "Profile" and "Logout". Below the header is a navigation menu with tabs for "Papers", "Authors", "Journals", "Institutions", and "Conferences". The "Papers" tab is selected. The main content area displays a paper entry with the following fields: "Paper Name", a "Download" button, a star rating (4 stars out of 5), and a link "Rank This Paper". Below these are the "Abstract:", "Author: [Author Name](#)", "Date Of Publication: DD/MM/YYYY", and "Reselased On: [Journal/Conference Name](#)". At the bottom of the main content area is a "Write a Comment..." text box and a list of three comments: "Comment 1 - Comment Writer", "Comment 2 - Comment Writer", and "Comment 3 - Comment Writer".

Details of a specific paper(Fig. 9)

In this page are shown all the details of a specific paper where all the comments are shown. The user can write a comment or download the paper. Also the user can rate the paper with stars choosing from a max 5-star rating.

```
SELECT paper_id, title, abstract, file,name, lastname  
FROM Paper natural join write natural join Author  
WHERE paper_id = @paper_id
```

Download

```
SELECT paper_file,  
FROM Paper  
WHERE paper_id = @paper_id
```

Comment

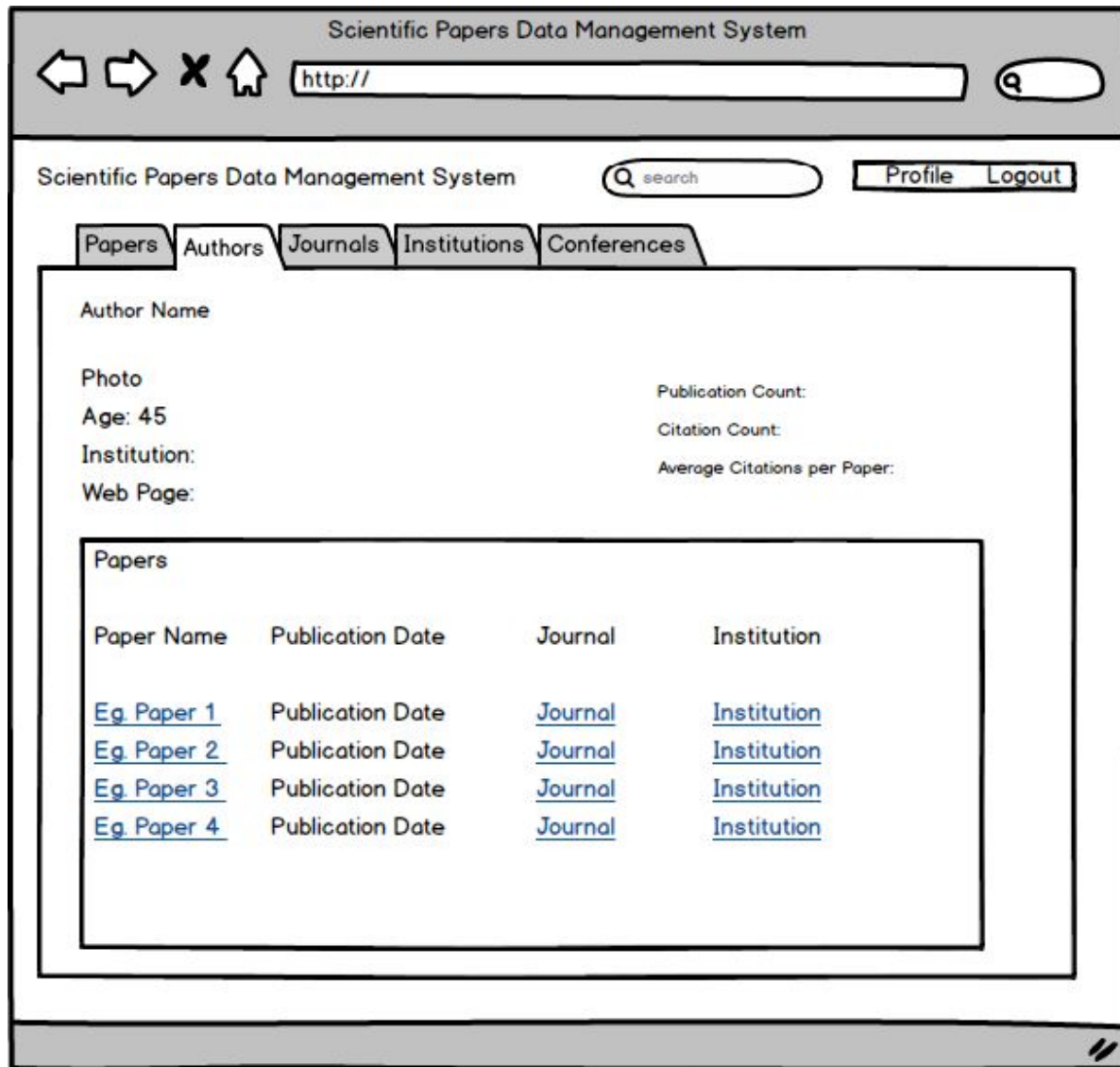
```
INSERT INTO comment (email_address, paper_id, comment_content)  
VALUES(@email_address, @paper_id, @comment_content)
```

See comments

```
SELECT paper_id, paper_title, comment_content  
FROM Paper natural join comment  
WHERE paper_id = @paper_id
```

Rate

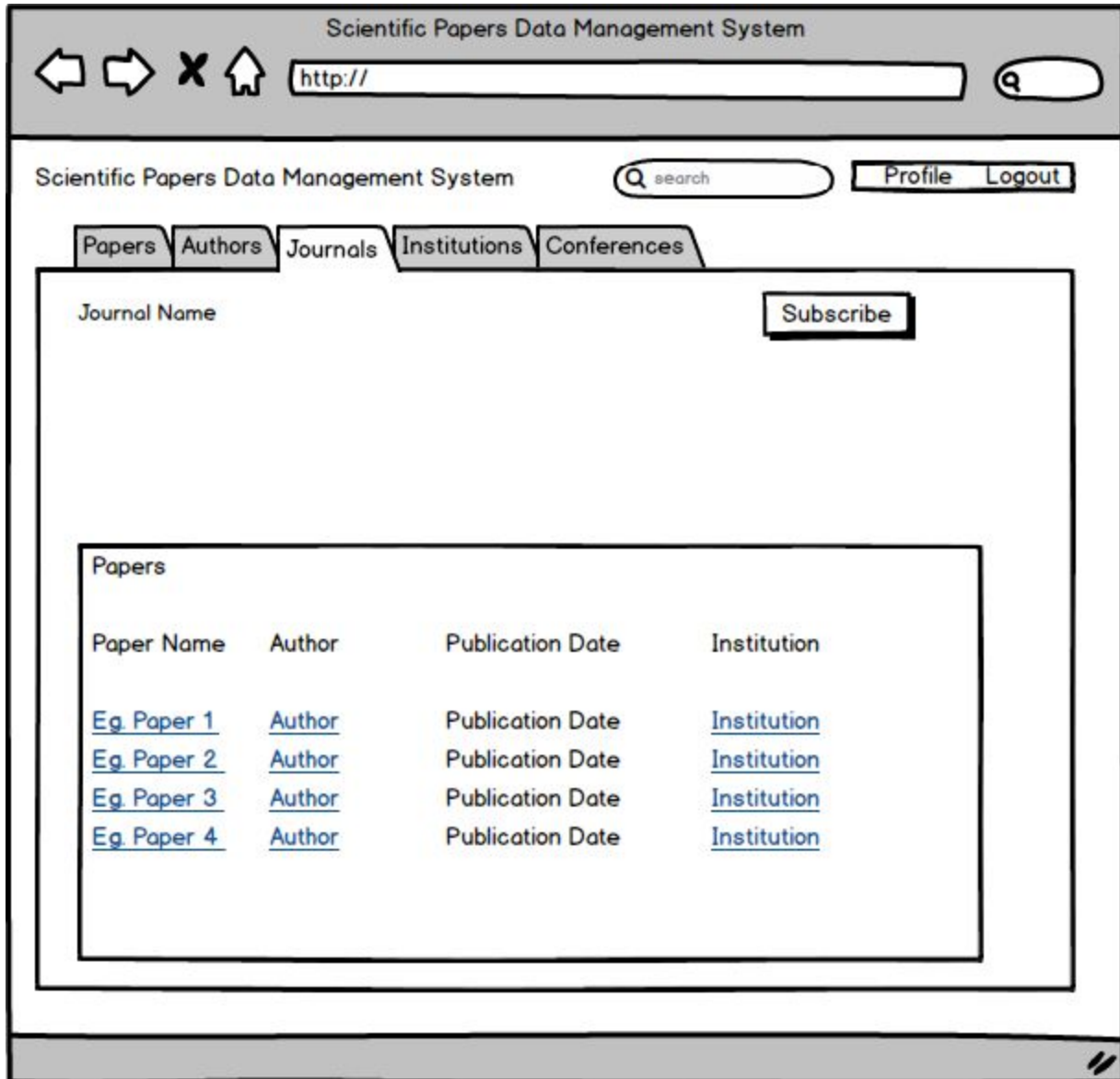
```
INSERT INTO rate (email_address, paper_id, rating_points)  
VALUES(@email_address, @paper_id, @rating_points)
```



Author page (Fig.10)

In this page a specific author's details are shown together with the papers that he/she has written. The papers that are shown here will have their details about the publication.

```
SELECT name, lastname, Author.webpage, institution_name, paper_id, paper_title
FROM write natural join has_author natural join Author,
WHERE name = @name
```



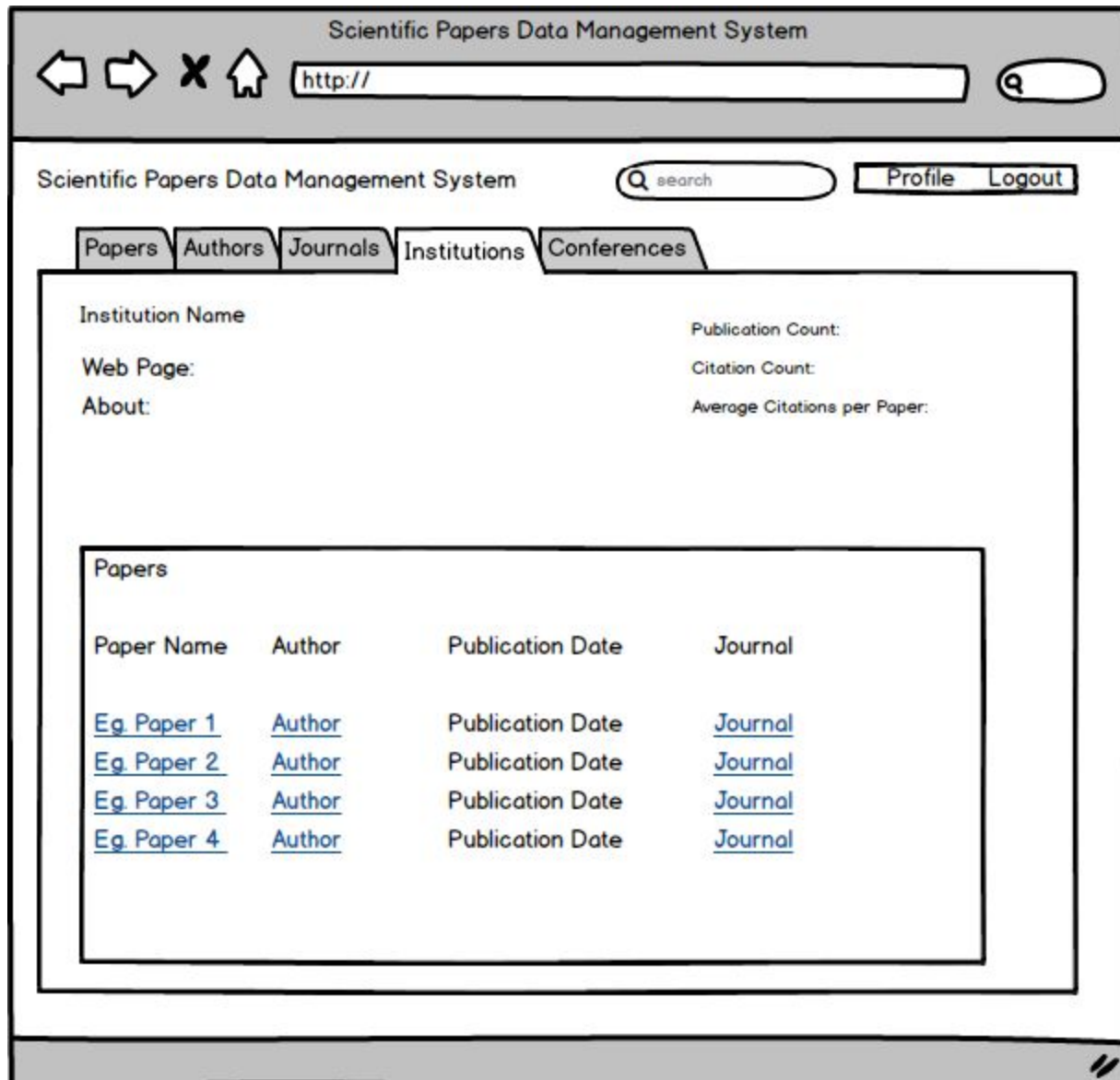
Journal page (Fig. 11)

This page contains information about a specific journal and the list of its papers.

```
SELECT J.ISSN, J.name, J.year_of_publication, P.paper_id, P.paper_title
FROM Journal as J natural join papers_of_journal P,
WHERE J.ISSN= @ISSN
```

Subscribe

```
INSERT INTO subscribe(email_address, ISSN, start_date, end_date),
VALUES(@email_address, @ISSN, @start_date, @end_date)
```



Institutions page(Fig.12)

In this page is shown a specific institution with its details and the authors that are supported by this institution together with some details about the paper and publication.

```
SELECT W.title, A.name, A.lastname
FROM Institution as I natural join has_author as A natural join write as W
WHERE I.institution_name = @institution_name
ORDER BY A.lastname
```

Scientific Papers Data Management System

http://

Scientific Papers Data Management System

search

Profile Logout

Papers Authors Journals Institutions Conferences

Conference Name

Award Count:

Date: DD/MM/YYYY

Location:

Description:

Papers

Paper Name	Author	Institution
Eg. Paper 1	Author	Institution
Eg. Paper 2	Author	Institution
Eg. Paper 3	Author	Institution
Eg. Paper 4	Author	Institution

Conference page (Fig.13)

In this page is shown a specific conference with its details and the papers that are published in this conference together with the respective authors and institutions.

```
SELECT P.title
FROM papers_of_conferences as P
WHERE S.conference_name = @conference_name
```

LOGGED IN AS AUTHOR

The screenshot shows a web browser window titled "Scientific Papers Data Management System". The address bar shows "http://". The page header includes a search bar with a magnifying glass icon and the text "search", and links for "Profile" and "Logout". Below the header is a navigation menu with tabs: "Papers", "Authors", "Journals", "Institutions", "Conferences", "Upload Paper" (which is active), and "My Papers". The main content area of the "Upload Paper" page contains the following form elements:

- "Paper Title:" followed by a text input field with the placeholder text "Enter Paper Title".
- "Paper Abstract:" followed by a text input field with the placeholder text "Enter Abstract".
- An "Upload File" button with a small upward arrow icon, followed by ".pdf".
- "Upload" and "Cancel" buttons at the bottom.

Upload Paper (Fig.14)

In this page the author can upload a paper by entering its details and choosing the file from its computer to upload it.

```
INSERT INTO Paper(paper_id, title, abstract, date_of_publication, index_term, file)
VALUES(@paper_id, @title, @abstract, @date_of_publication, @index_term, @file)
```

```
INSERT INTO write(email_address, paper_id)
VALUES(@email_address, @paper_id)
```


The screenshot shows a web browser window titled "Scientific Papers Data Management System". The address bar contains "http://". The page header includes a search bar with a magnifying glass icon and the text "search", and links for "Profile" and "Logout". Below the header is a navigation menu with tabs: "Papers", "Authors", "Journals", "Institutions", "Conferences", "Upload Paper", and "My Papers". The "Upload Paper" tab is selected, leading to the "Submit Paper" form. The form contains two dropdown menus: "Collaborated Authors" with a value of "Author List", and "Submit to" with a value of "Journal List". At the bottom of the form are two buttons: "Submit" and "Cancel".

Add co authors & Submit Paper (Fig.15)

When a paper will be submitted in this page the author can choose the co authors from a dropdown list and also the author can choose where to submit her/his paper.

```
INSERT INTO write(email_address, paper_id)
VALUES(@email_address, @paper_id)
```

Add references

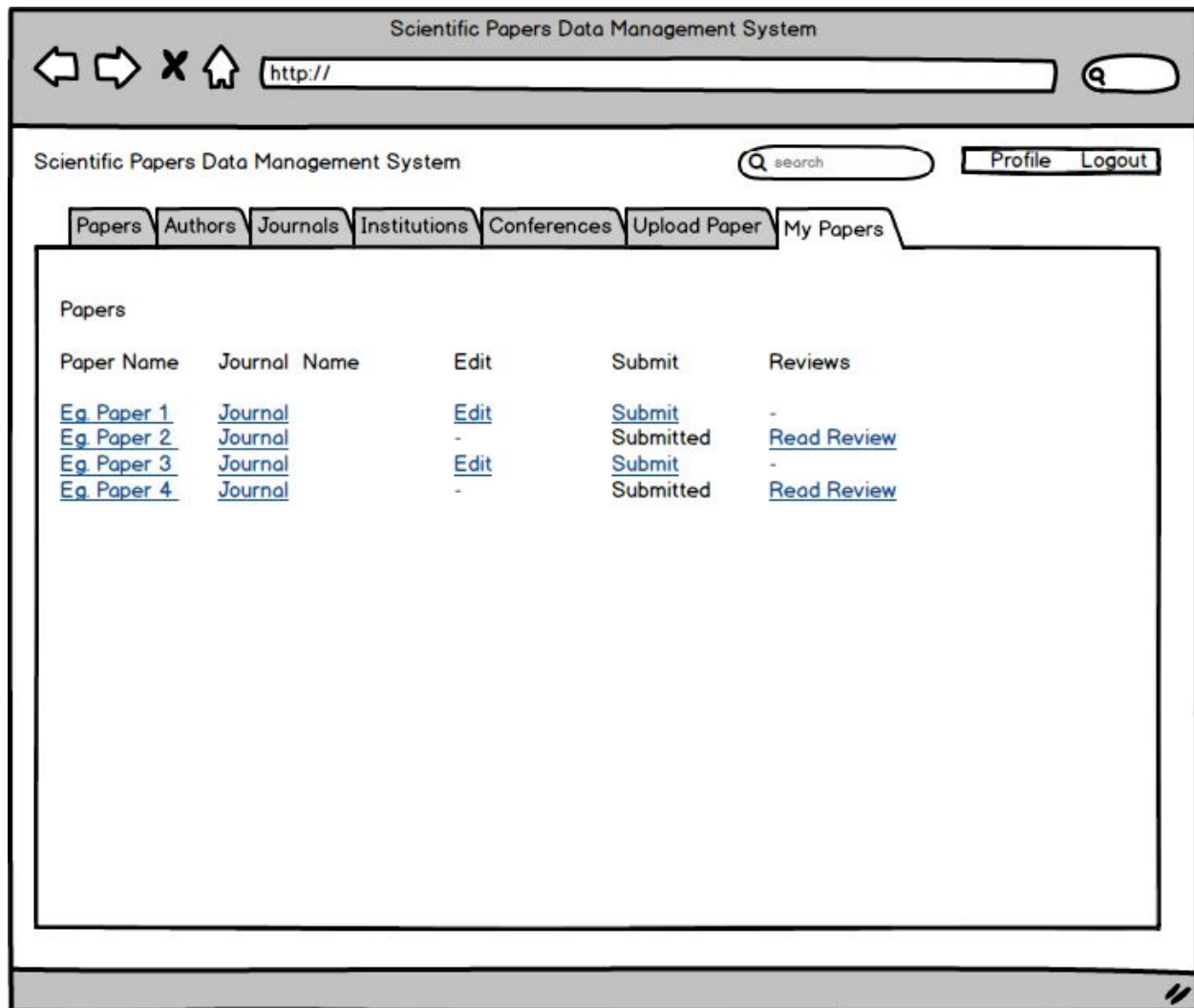
```
INSERT INTO cites(paper_title, reference_title)
VALUES(@paper_title, @reference_title)
```

Submit to Journal

```
INSERT INTO submit_to_journal(ISSN,paper_id)
VALUES(@ISSN, @paper_id)
```

Submit to Conference

```
INSERT INTO submit_to_conference(conference_name,date,location,paper_id)
VALUES(@conference_name, @date,@location,@paper_id)
```



List of uploaded papers(Fig.16)

In this page the author can see the uploaded papers and submitted papers. Here the author can also read the reviews of the submitted papers, edit the upload papers and then submit them.

```
SELECT paper_title
FROM write natural join Paper
WHERE email_address = @email_address
```

List of submitted papers to conference

```
SELECT paper_title
FROM submit_to_conference natural join Paper natural join write
WHERE email_address = @email_address
```

List of submitted papers to journal

```
SELECT paper_title
FROM submit_to_journal natural join Paper natural join write
WHERE email_address = @email_address
```

Read reviews

```
SELECT review_content
FROM review
WHERE paper_id = @paper_id
```

The screenshot displays a web browser window titled "Scientific Papers Data Management System". The browser's address bar shows "http://". The page header includes a search bar with a magnifying glass icon and the text "search", and two buttons labeled "Profile" and "Logout". Below the header is a navigation menu with tabs for "Papers", "Authors", "Journals", "Institutions", "Conferences", "Upload Paper", and "My Papers". The "My Papers" tab is currently selected. The main content area is titled "Edit Paper" and contains two input fields: "Paper Title:" with a text box containing "Enter Paper Title", and "Paper Abstract:" with a text box containing "Enter Abstract". At the bottom of the form are two buttons: "Save Changes" and "Cancel".

Edit Uploaded paper Info (Fig.17)

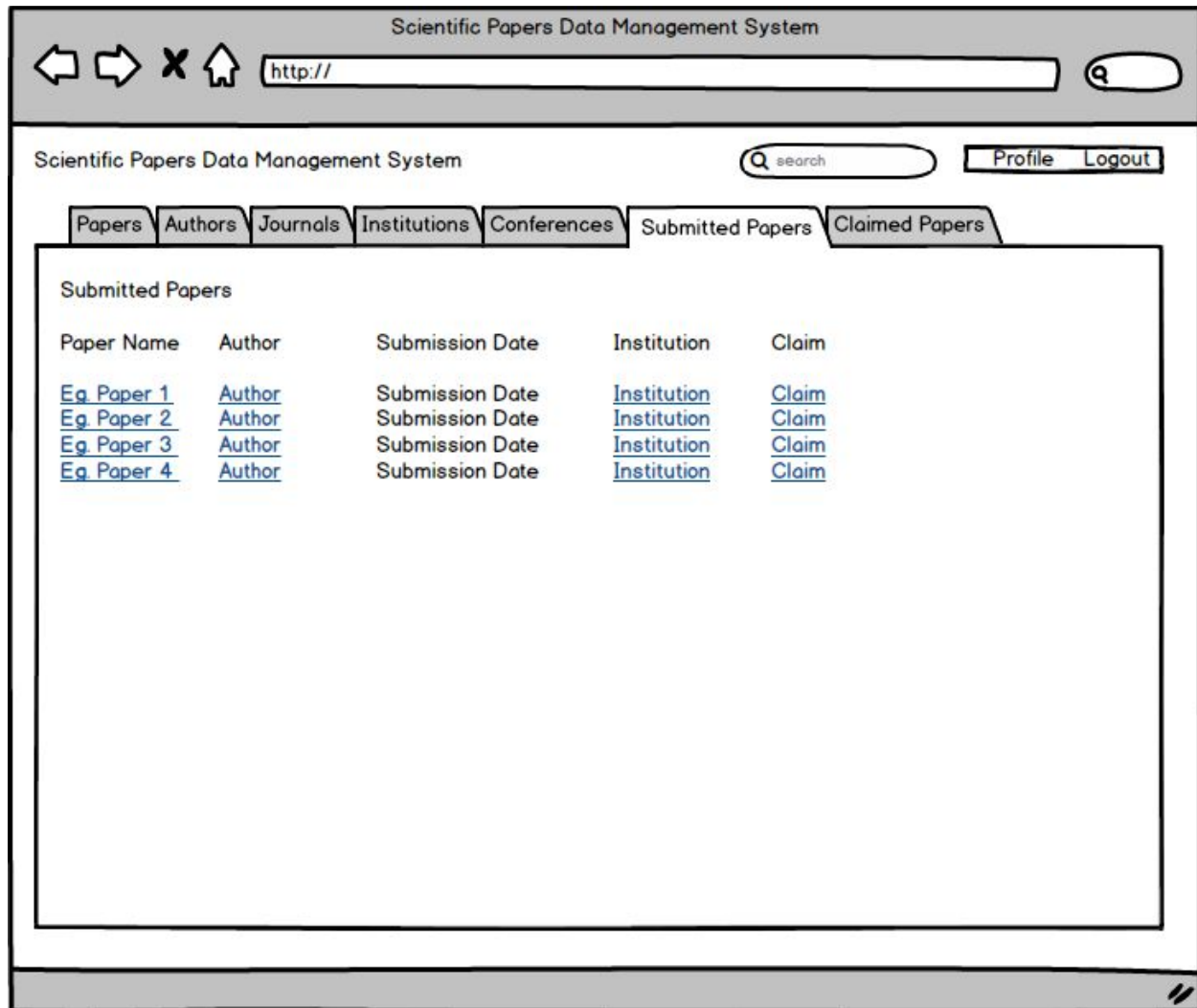
In this page the author can edit the papers that have been uploaded and then save the changes.

UPDATE Paper

SET title = @title, abstract = @abstract, index_term = @index_term, file = @file

WHERE paper_id = @paper_id

LOGGED IN AS EDITOR

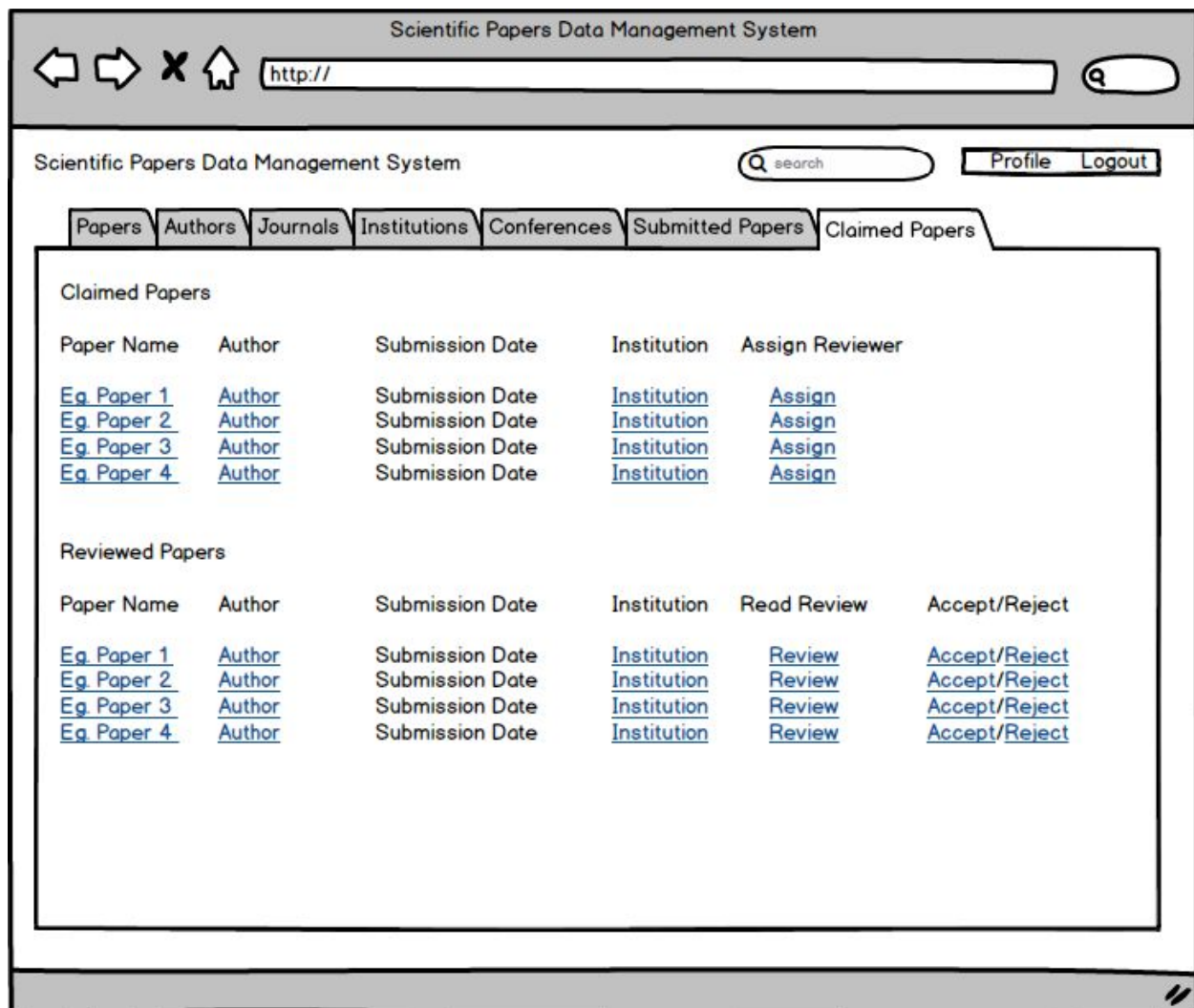


Submitted papers (Fig.18)

In this page the editor can see the papers that have been submitted by the author with some of their details and then can claim them.

```
SELECT *  
FROM submitted_paper_editorC  
WHERE email_address = @email_address
```

```
SELECT *  
FROM submitted_paper_editorJ  
WHERE email_address = @email_address
```

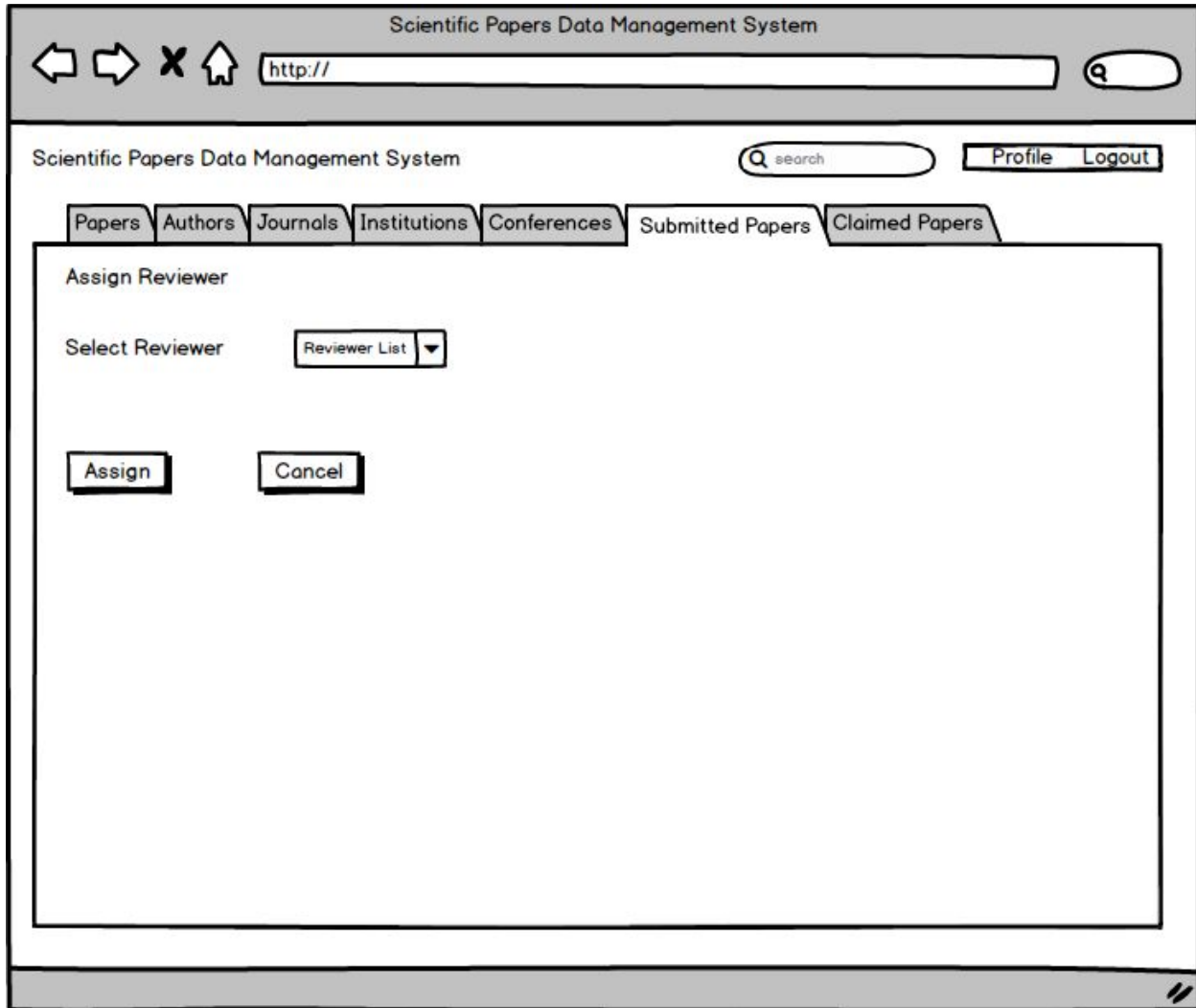


Claimed papers (Fig.19)

In this page the editor can see the papers that he/she has claimed and then can assign them to reviewers. Also the reviewed papers can be seen here where the editor can read the reviews and then decides to accept the paper to be published or not.

```
SELECT C.title
FROM submitted_paper_editorC as C natural join decide as D
WHERE D.decision = "NULL"
```

```
SELECT J.title
FROM submitted_paper_editorJl as J natural join decide as D
WHERE D.decision = "NULL"
```



Assign reviewers(Fig. 20)

In this page the editor can assign the reviewers to a paper. Here the editor selects the reviewer from a dropdown list and assigns him/her.

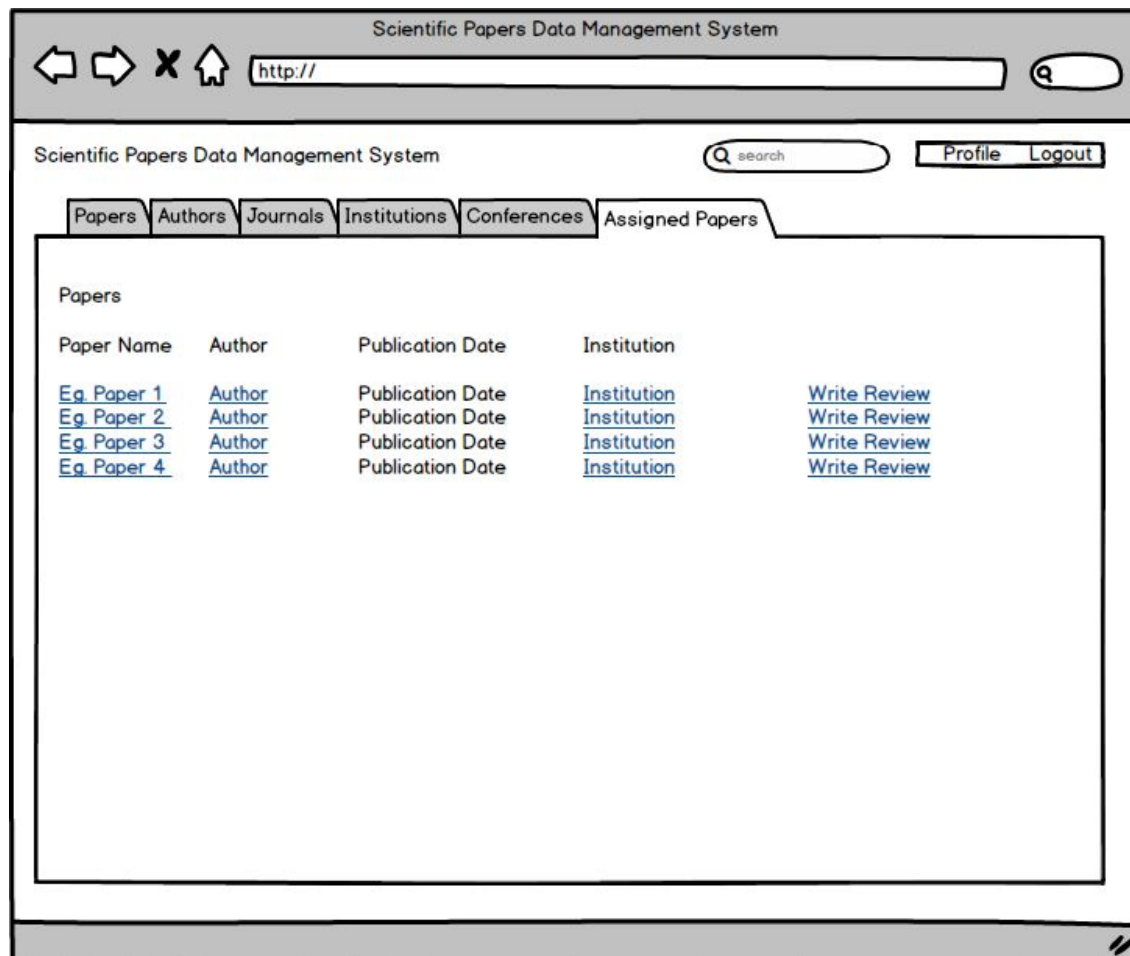
```
INSERT INTO assign(paper_id,reviewer_email,editor_email)
VALUES(@paper_id, @reviewer_email,@editor_email)
```

```
Read reviews
SELECT review_content
FROM review
WHERE paper_id = @paper_id
```

Paper info

```
SELECT paper_id, title, abstract, paper_file,name,lastname
FROM Paper natural join write natural join Author
WHERE paper_id = @paper_id
```

LOGGED IN AS REVIEWER



Assigned papers(Fig.21)

In this page the reviewer can see the papers that are assigned to him to be reviewed.

```
SELECT      *
FROM        assigned_papers
WHERE email_address = @email_address
```


Scientific Papers Data Management System

Scientific Papers Data Management System

Q search Profile Logout

Papers Authors Journals Institutions Conferences Assigned Papers

Paper Name Download Submit Review

Abstract:
 Author: [Author Name](#)
 Date Of Publication: DD/MM/YYYY
 Reselased On: [Journal/Conference Name](#)

Write Review Here...

Save Changes Cancel

Assigned Paper Review(Fig.22)

In this page the reviewer can see a specific paper assigned to him with its details. The reviewer can download the paper and also can write its own reviews here and submit them.

Write review:

```
INSERT INTO uploaded_reviews
VALUES (@email_address, @paper_id, @review_content))
```

Edit review:

```
UPDATE uploaded_reviews
SET review_content = @review_content
WHERE paper_id = @paper_id
```

Submit review:

INSERT INTO review

SELECT *

FROM uploaded_reviews

WHERE email_address = @email_address AND paper_id = @paper_id

DELETE FROM uploaded_reviews

WHERE email_address = @email_address AND paper_id = @paper_id

The screenshot displays a web browser window titled "Scientific Papers Data Management System". The address bar shows "http://". The page header includes a search bar with a magnifying glass icon and the text "search", and two buttons labeled "Profile" and "Logout". Below the header, there are five tabs: "Papers", "Authors", "Journals", "Institutions", and "Conferences". The "Authors" tab is currently selected. The main content area contains a form for editing a profile. The form has four labeled input fields: "Name:" with a placeholder "Enter New Name", "Photo" with a placeholder "Upload Photo", "Date Of Birth:" with a placeholder "DD/MM/YYYY" and a calendar icon, and "Password:" with a placeholder "Enter New Password". At the bottom of the form are two buttons: "Save Changes" and "Cancel".

Edit Profile Page (Fig.23)

This page is shown when the user clicks “Edit Profile” and s/he can update the personal information. When the user clicks “Save Changes” all the changes are saved in database. The email address cannot be changed. Depending on the role of the user there will be additional information to be edited like institution name for authors and journal/conference information for authors, editors and subscribers.

UPDATE User

```
SET name = @name, lastname = @lastname, password = @password,
date_of_birth = @date_of_birth, age = @age, photo = @photo, role = @role
WHERE email_address = @email_address
```

UPDATE Author

```
SET name = @name, lastname = @lastname, password = @password,  
date_of_birth = @date_of_birth, age = @age, photo = @photo, role = @role, webpage  
= @webpage  
WHERE email_address = @email_address
```

UPDATE has_author

```
SET institution_name = @institution_name  
WHERE email_address = @email_address
```

UPDATE journal_has_author

```
SET ISSN = @ISSN  
WHERE email_address = @email_address
```

UPDATE conference_has_author

```
SET conference_name = @conference_name, date = @date, location =  
@location  
WHERE email_address = @email_address
```

UPDATE Editor

```
SET name = @name, lastname = @lastname, password = @password,  
date_of_birth = @date_of_birth, age = @age, photo = @photo, role = @role, webpage  
= @webpage  
WHERE email_address = @email_address
```

UPDATE journal_has_editor

```
SET ISSN = @ISSN  
WHERE email_address = @email_address
```

UPDATE conference_has_editor

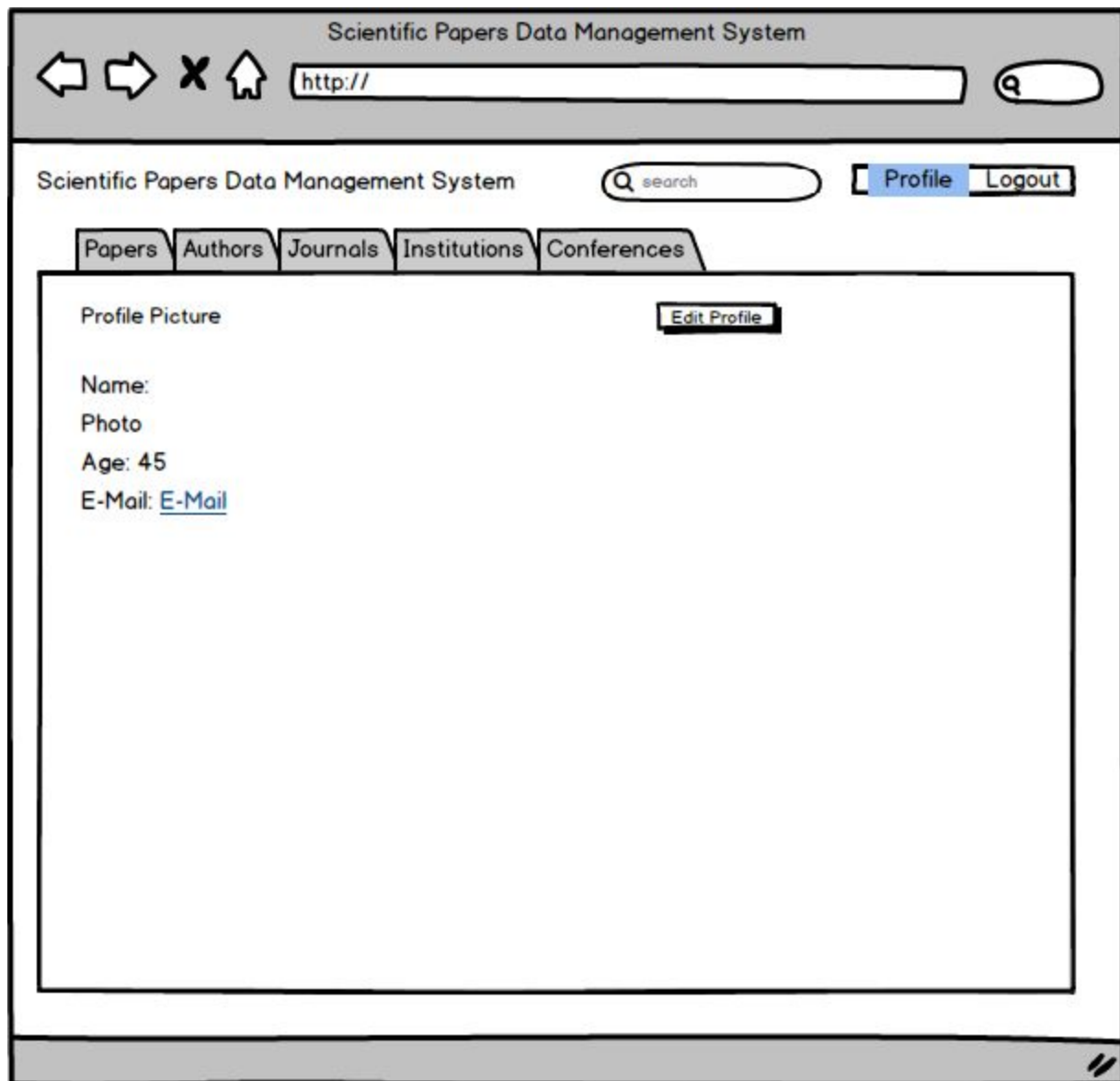
```
SET conference_name = @conference_name, date = @date, location =  
@location  
WHERE email_address = @email_address
```

UPDATE Reviewer

```
SET name = @name, lastname = @lastname, password = @password,  
date_of_birth = @date_of_birth, age = @age, photo = @photo, role = @role, webpage  
= @webpage  
WHERE email_address = @email_address
```

```
UPDATE journal_has_reviewer
  SET ISSN = @ISSN
  WHERE email_address = @email_address
```

```
UPDATE conference_has_reviewer
  SET conference_name = @conference_name, date = @date, location =
@location
  WHERE email_address = @email_address
```



User's Profile(Fig.24)

In the editor's profile page will be shown also the edit's count together with the conferences and journals that the editor takes part. In the reviewers profile page the review count and the journals and conferences that the reviewer takes part are shown. In the author's profile page will be shown the list of papers in addition to the components above.

```
SELECT (*)
FROM User
WHERE email_address = @email_address
```

5 Advanced Database Components(15p)

5.1 Views

5.1.1 Editor's View

An editor cannot see all the submitted paper but just the papers that are submitted to the journal/conference s/he is part of. This view will be used in UI in Fig.18.

```
create view submitted_paper_editorC as
    SELECT P.title
    FROM Paper as P natural join submit_to_conference as S natural join decide as D
    WHERE P.paper_id not in (SELECT paper_id
                             FROM decide)
```

```
create view submitted_paper_editorJ as
    SELECT P.title
    FROM Paper as P natural join submit_to_journal as S natural join decide as D
    WHERE P.paper_id not in (SELECT paper_id
                             FROM decide)
```

5.1.2 Reviewer's View

A reviewer can not review or see the papers that are not assigned to her/him. This view will be used in the UI in Fig.21

```
create view assigned_papers as
```

```

SELECT      AReviewer_email,P.title
FROM        assign as A natural join Paper as P
GROUP BY    AReviewer_email

```

5.1.3 Select Co-Authors View

When an author uploads her/his paper he/she will be shown the list of the authors in our database just by their name and lastname (no other information about them will be shown there). This view will be used in the UI in Fig.15.

```

create view coauthors_list as
SELECT name, lastname
FROM Author
ORDER BY name

```

5.1.4 Select Conference/Journal

When an author wants to submit her/his paper, s/he will be shown the list of conferences/journals from which s/he will choose to submit. This view will be used in the UI in Fig.15.

```

create view conference_list as
SELECT C.conference_name, C.date, C.location
FROM Conference as C
ORDER BY C.date

```

```

create view journal_list as
SELECT J.ISSN, J.name
FROM Journal as J
ORDER BY J.name

```

5.1.5 Papers of Conference/Journal View

When a journal is selected only the name of the journal is displayed. This view will be used in UI in Fig.11.

```

create view papers_of_journals as
SELECT S.ISSN, D.paper_id, P.paper_title,
FROM submit_to_journal as S natural join decide as D natural join Paper as P,
WHERE D.decision = "YES"
GROUP BY S.ISSN

```

When a conference is selected only the conference_name, date and location will be displayed. This view will be used in UI in Fig.13.

create view papers_of_conferences as

```
SELECT S.conference_name, S.date, S.location, D.paper_id, P.paper_title
FROM submit_to_conference as S natural join decide as D natural join Paper as
P,
WHERE D.decision = "YES"
GROUP BY S.conference_name, S.date, S.location
```

5.2 Stored Procedures

We plan to use stored procedures when submitting a paper. For each new submission, we need to create a submitted paper tuple. For each paper published, we increase the number of publications for institutions and journals. This procedure is always the same for any publishing process. Therefore we decided to store this process as a procedure.

A similar procedure will be used for citations count with some minor differences. Because one paper can only be submitted once but can be cited infinitely, we used a procedure to update the citation count of papers for each new submission. This will be done by checking the citation part of newly submitted papers and updating the citation count of related paper, author, journal and institution accordingly. This process will also stay same for all papers and therefore using a procedure for it is logical.

A procedure will also be used for rating management. When a paper is rated by a user, her/his vote will be added to the total vote value of the paper and then be divided to the total vote count for the said paper. This averaging procedure will always be the same for each paper and therefore using a procedure will provide useful.

5.3 Reports

5.3.1 Total Number of Citations

```
WITH authors_and_institutions(avg_citations, paper_id)
AS(SELECT a.avg_citations, p.paper_id FROM author a NATURAL JOIN institution i
NATURAL JOIN paper p )

SELECT sum(avg_citations)
```



```
FROM authors_and_institutions
GROUP BY paper_id
```

5.3.2 Total Number of Reviews

```
WITH reviewers_and_papers(review_content, paper_id)
AS(SELECT r.review.content, p.paper_id FROM reviewer r NATURAL JOIN paper p )

SELECT sum(review.content)
FROM reviewers_and_papers
GROUP BY paper_id
```

5.3.3 Total Number of Papers

```
SELECT count(paper_id)
FROM paper
```

5.3.4 Total Number of Comments

```
WITH users_and_papers(comment_content, paper_id)
AS(SELECT u.comment_content, p.paper_id FROM user u NATURAL JOIN paper p )

SELECT count(comment_content)
FROM users_and_papers
GROUP BY paper_id
```

5.3.5 Total Number of Papers Submitted to Conferences

```
WITH conferences_and_papers(conference_name, paper_id)
AS(SELECT c.conference_name, p.paper_id FROM conference c NATURAL JOIN
paper p )

SELECT count(paper_id)
FROM conferences_and_papers
GROUP BY conference_name
```

5.3.6 Total Number of Papers Submitted to Journals

```
WITH journals_and_papers(ISSN, paper_id)
AS(SELECT j.ISSN, p.paper_id FROM journal j NATURAL JOIN paper p )
```

```
SELECT count(paper_id)
FROM journals_and_papers
GROUP BY ISSN
```

5.3.7 Total Number of Reviewers in Journals

```
WITH journals_and_reviewers(ISSN, reviewer_id)
AS(SELECT j.ISSN, r.reviewer_id FROM journal j NATURAL JOIN reviewer r )
```

```
SELECT count(reviewer_id)
FROM journals_and_reviewers
GROUP BY ISSN
```

5.3.8 Total Number of Editors in Journals

```
WITH journals_and_editors(ISSN, editor_id)
AS(SELECT j.ISSN, e.editor_id FROM journal j NATURAL JOIN editor e )
```

```
SELECT count(editor_id)
FROM journals_and_editors
GROUP BY ISSN
```

5.3.9 Total Number of Authors in Institutions

```
WITH authors_and_institutions(author_id, institution_name)
AS(SELECT a.author_id, i.institution_name FROM author a NATURAL JOIN institution i)
```

```
SELECT count(author_id)
FROM authors_and_institutions
GROUP BY institution_name
```

5.4 Triggers

- When a paper is uploaded, it will be available for submission.
- When a paper is submitted, it will be available for editors to see.
- When a submission is claimed by an editor, it will be shown in the “Claimed Papers” page.
- When a reviewer is assigned to a paper by an editor, reviewer will be able to add reviews to that specific paper.
- When a paper is published by an editor, publication count of institutions and journals will be updated accordingly.
- When a paper is cited, citation count of relevant parties such as author and paper will be updated accordingly.

- When a comment is posted by a user, comments part of a paper will be updated accordingly.
- When a rating is given to a paper by any user, rating on the paper page will be updated accordingly.

5.5 Constraints

- Everyone who wants to use our application should register/login to the system.
- An editor can assign at most 3 reviewers to a paper.
- A reviewer can be assigned to at most 5 papers at a time.
- An editor and a reviewer should provide the journal title or conference name where he/she is part of in order to see the submitted papers to that specific journal or conference.
- An author can submit her/his paper at most one specific journal/conference at a time.
- If the paper of the author is rejected, he/she will have the paper to the uploaded papers and can submit it to another conference/journal.

6 Implementation Plan

To implement our database system we will use MySQL. To implement the functions of the application we will use PHP. We will use HTML, CSS and JavaScript for front-end design of the web application.