Software Requirements Specification

Version 1.0

<<Annotated Version>>

June 16, 2020

KINDERGARTEN Automate System

Laila AL Hafez

Lana Ghali

Rama AL Hafez

Submitted in partial fulfillment

Of the requirements of

CS 310 Software Engineering

<<Any comments inside double brackets such as these are *not* part of this SRS but are comments upon this SRS example to help the reader understand the point being made.

Refer to the SRS Template for details on the purpose and rules for each section of this document.

This work is based upon the submissions of the Spring 2004 CS 310. The students who submitted these team projects were rama al hafez, Laila al hafez, and lana ghali. >>

# List of Figures

[Figure 1 - System Environment 4](#_Toc77487669)

[Figure 2 - Article Submission Process 6](#_Toc77487670)

[Figure 3 - Editor Use Cases 8](#_Toc77487671)

[Figure 4 - Logical Structure of the Article Manager Data 23](#_Toc77487672)

Table of contant

[List of Figures i](#_Toc43393388)

[1.0. Introduction 1](#_Toc43393389)

[1.1. Purpose 1](#_Toc43393390)

[1.2. Scope of Project 1](#_Toc43393391)

[1.3. Glossary 1](#_Toc43393392)

[1.4. References 2](#_Toc43393393)

[1.5. Overview of Document 2](#_Toc43393394)

[2.0. Overall Description 3](#_Toc43393395)

[2.1 System Environment 3](#_Toc43393396)

[2.2 Functional Requirements Specification 4](#_Toc43393397)

[2.2.1 Manger Use Case 4](#_Toc43393398)

[2.2.2 Accountant Use Case 5](#_Toc43393399)

[2.2.3 Employee Use Case 6](#_Toc43393400)

[2.3 User Characteristics 9](#_Toc43393401)

[2.4 Non-Functional Requirements 9](#_Toc43393402)

[3.0. Requirements Specification 12](#_Toc43393403)

[3.1 External Interface Requirements 12](#_Toc43393404)

[3.2 Functional Requirements 12](#_Toc43393405)

[3.2.1 Entering teacher data 12](#_Toc43393406)

[3.2.2 Registration of premiums 12](#_Toc43393407)

[3.2.3 Student registration 13](#_Toc43393408)

[3.2.4 Student registration information 13](#_Toc43393409)

[3.2.5 Print the final calendar 14](#_Toc43393410)

[3.2.7 : Send Weekly Report 15](#_Toc43393411)

[3.3 Detailed Non-Functional Requirements 15](#_Toc43393412)

[3.3.1 Logical Structure of the Data 15](#_Toc43393413)

[3.3.2 Security 17](#_Toc43393414)

# 

# 1.0. Introduction

## 1.1. Purpose

The purpose of this document is to present a detailed description of the

Kindergarten Automate System. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both the stakeholders and the developers of the system.

## 1.2. Scope of Project

This software system will be a Kindergarten Automate System for a local employee .This system will be designed to maximize the employee’s productivity by providing tools to assist in automating the internal system of kindergarten, which would otherwise have to be performed manually. By maximizing the employee’s work efficiency and production the system will meet the employee’s needs while remaining easy to understand and use.

More specifically, this system is designed to allow an employee to manage all about

student and teacher affairs completely and quickly on kindergarten automate system. The

software will facilitate communication between employees and teachers and parents.

This system will Enter and handle teacher data, enter new student data, and enter quick

access data for parents. The system is linked to an application with the parents.

It registers premiums and prints an invoice. This application needs a database

management system to store student data.

## 1.3. Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Active Article | The document that is tracked by the system; it is a narrative that is planned to be posted to the public website. |
| Employee | Person who is working in a company. |
| Database | Collection of all the information monitored by this system. |
| Teacher | Person who teach student in school. |
| Field | A cell within a form. |
| Application | Software will work on mobile. |
| Invoice | Paper containing how much money you paid. |
| Premiums | Dividing a sum of money into several small amounts. |
| Software Requirements Specification | A document that completely describes all of the functions of a proposed system and the constraints under which it must operate. For example, this document. |
| Logo | It's a symbol that indicates something. |
| kindergarten | Place where children under six years come to learn and enjoy. |

## 1.4. References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

## 1.5. Overview of Document

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.

# 2.0. Overall Description

## 2.1 System Environment

manger

accountant

parents

DB

system

application

Kindergarten automate System

employee

Figure 1 - System Environment

The kindergarten automates System has four active actors and one cooperating system.

The Manger, Accountant, Employee accesses the System through the local network directly. Parents accesses the System through the internet.

<< The division of the Kindergarten Automate System into two component parts, the Application and the System is an example of using domain classes to make an explanation clearer. >>

## 2.2 Functional Requirements Specification

This section outlines the use cases for each of the employee, manger and accountant separately. All of them have only one use case apiece and they are main actor in this system.

### 2.2.1 Manger Use Case

#### Use case: Entering teacher data

**Diagram:**

Manger

Entering teacher data

**Brief Description**

The manger accesses the system and enter teacher’s data.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Manger has already accessed the local network.

1. Teacher comes to the kindergarten.
2. Teacher give his information to the manger.
3. The Manger enter the teacher information to the system.
4. The system will store the information in database.

**Xref:** Section 3.2.1, Entering teacher data

system

Enter information

storing

Figure 2 – Entering teachers data Process

The *Entering teacher’s data Process* state-transition diagram summarizes the use cases listed below. A Teacher give his information for consideration. The Manger enters it into the system and assigns him. The system returns his information to data base. the manger will decide which class will teacher give

### 2.2.2 Accountant Use Case

Use case: **Registration of premiums**

**Diagram:**

Accountant

Registration of premiums

**Brief Description**

The accountant records the student installments and prints the invoice in the event of payment.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Accountant has already connected to the local network and verify the student's presence in the database.

1. The employee records the student in database.
2. The parent come to the accountant and pay money.
3. The accountant records the premium in the system.
4. The accountant prints the invoice for parent.

**Xref:** Section 3.2.2, Registration of premiums

### 2.2.3 Employee Use Case

The Employee has the following sets of use cases:

Student registration

employee

Student registration information

Print the final calendar

Send notice

Send weekly report

Figure 3 - Employee Use Cases

**Update Information use cases**

#### Use case: Student registration

**Diagram:**

Employee

Student registration

**Brief Description**

The employee record the student in the system.

**Initial Step-By-Step Description**

Before this use case can be initiated, the employee has already connected to the local network.

1. The parent comes to the kindergarten.
2. The employee searches within the database to find out if the student already exists or no.
3. If the student exists in database, the employee class it into a pre-existing student.
4. Else the employee class the student into a new student.

**Xref:** Section 3.2.3, Student registration.

**Use case:** Student registration information

**Diagram:**

Employee

Student registration information

**Brief Description**

The Employee records student information in database and dealing with it from collecting, modifying and others.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Employee has already accessed the local network.

1. The Employee records the student's absence and attendance.
2. The Employee records the student's marks and places them in the final calendar.
3. The Employee records a weekly report to the family on the student's condition.

**Xref:** Section 3.2.4 Student registration information .

#### Use case: Print the final calendar

**Diagram:**

Employee

Print the final calendar

**Brief Description**

The Employee print the final calendar.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Employee has already accessed the local network and has already recorded the student’s marks .

1. The Employee accesses the database.
2. The employee brings student marks.
3. The employee places the marks in the final calendar.
4. The employee prints the final calendar.

**Xref:** Section 3.2.5, Print the final calendar

#### Use case: Send Notice

Employee

Send notice

**Diagram:**

**Brief Description**

The Employee sends a notice to the parents.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Employee has already accessed the local network. The parent has already the application on there phone.

1. The Employee accesses the database.
2. The employee records the student’s absence.
3. The employee sends a notice to the parent.

**Xref:** Section 3.2.6 Send Notice

#### Use case: Send Weekly Report

**Diagram:**

Employee

Send weekly report

**Brief Description**

The Employee send a weekly report to the parents.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Employee has already accessed the local network and access to the database. The parent has already the application on there phone.

1. The Employee accesses the system.
2. The system fetches the student record from the database.
3. The employee writes the report about the student's condition.
4. The employee sends the report to the parent.
5. The process is repeated every week.

**Xref:** Section 3.2.7, Send Weekly Report

## 2.3 User Characteristics

The employee is expected to familiar with dealing with Windows 7 and above

operating systems and has knowledge of dealing with the database.

The accountant is expected to familiar with dealing with Windows 7 and above

operating systems, must be able to count money and be able to handle the printer.

The manger is expected to familiar with dealing with Windows 7 and above

operating systems

The detailed look of these pages is discussed in section 3.2 below.

## 

## 2.4 Non-Functional Requirements

The kindergarten automate system will be on a local server with response speed not exceeding 20 milliseconds capability. The system work in windows 7 and above.

There must be a database management system. The system should be easy to use.

The presence of an application associated with the system. The presence of a color

printer.

# 3.0. Requirements Specification

## 3.1 External Interface Requirements

The only link to an external system is the link to the Historical Society (HS) Database to verify the membership of a Reviewer. The Editor believes that a society member is much more likely to be an effective reviewer and has imposed a membership requirement for a Reviewer. The HS Database fields of interest to the Web Publishing Systems are member’s name, membership (ID) number, and email address (an optional field for the HS Database).

The *Assign Reviewer* use case sends the Reviewer ID to the HS Database and a Boolean is returned denoting membership status. The *Update Reviewer* use case requests a list of member names, membership numbers and (optional) email addresses when adding a new Reviewer. It returns a Boolean for membership status when updating a Reviewer.

## 3.2 Functional Requirements

The Logical Structure of the Data is contained in Section 3.3.1.

### 3.2.1 Search Article

|  |  |
| --- | --- |
| **Use Case Name** | Search Article |
| **XRef** | Section 2.2.1, Search Article  SDD, Section 7.1 |
| **Trigger** | The Reader assesses the Online Journal Website |
| **Precondition** | The Web is displayed with grids for searching |
| **Basic Path** | 1. The Reader chooses how to search the Web site. The choices are by Author, by Category, and by Keyword. 2. If the search is by Author, the system creates and presents an alphabetical list of all authors in the database. In the case of an article with multiple authors, each is contained in the list. 3. The Reader selects an author. 4. The system creates and presents a list of all articles by that author in the database. 5. The Reader selects an article. 6. The system displays the Abstract for the article. 7. The Reader selects to download the article or to return to the article list or to the previous list. |
| **Alternative Paths** | In step 2, if the Reader selects to search by category, the system creates and presents a list of all categories in the database.   1. The Reader selects a category. 2. The system creates and presents a list of all articles in that category in the database. Return to step 5.   In step 2, if the Reader selects to search by keyword, the system presents a dialog box to enter the keyword or phrase.   1. The Reader enters a keyword or phrase. 2. The system searches the Abstracts for all articles with that keyword or phrase and creates and presents a list of all such articles in the database. Return to step 5. |
| **Postcondition** | The selected article is downloaded to the client machine. |
| **Exception Paths** | The Reader may abandon the search at any time. |
| **Other** | The categories list is generated from the information provided when article are published and not predefined in the Online Journal database. |

### 3.2.2 Communicate

|  |  |
| --- | --- |
| **Use Case Name** | Communicate |
| **XRef** | Section 2.2.2, Submit Article; Section 2.2.3, Submit Review  SDD, Section 7.2 |
| **Trigger** | The user selects a *mailto* link. |
| **Precondition** | The user is on the *Communicate* page linked from the Online Journal Main Page. |
| **Basic Path** | This use case uses the *mailto* HTML tag. This invokes the client email facility. |
| **Alternative Paths** | If the user prefers to use his or her own email directly, sufficient information will be contained on the Web page to do so. |
| **Postcondition** | The message is sent. |
| **Exception Paths** | The attempt may be abandoned at any time. |
| **Other** | None |

### 3.2.3 Add Author

|  |  |
| --- | --- |
| **Use Case Name** | Add Author |
| **XRef** | Section 2.2.4, Update Author  SDD, Section 7.3 |
| **Trigger** | The Editor selects to add a new author to the database. |
| **Precondition** | The Editor has accessed the Article Manager main screen. |
| **Basic Path** | 1. The system presents a blank grid to enter the author information. 2. The Editor enters the information and submits the form. 3. The system checks that the name and email address fields are not blank and updates the database. |
| **Alternative Paths** | If in step 2, either field is blank, the Editor is instructed to add an entry. No validation for correctness is made. |
| **Postcondition** | The Author has been added to the database. |
| **Exception Paths** | The Editor may abandon the operation at any time. |
| **Other** | The author information includes the name mailing address and email address. |

### 3.2.4 Add Reviewer

|  |  |
| --- | --- |
| **Use Case Name** | Add Reviewer |
| **XRef** | Section 2.2.4, Update Reviewer  SDD, Section 7.4 |
| **Trigger** | The Editor selects to add a new reviewer to the database. |
| **Precondition** | The Editor has accessed the Article Manager main screen. |
| **Basic Path** | 1. The system accesses the Historical Society (HS) database and presents an alphabetical list of the society members. 2. The Editor selects a person. 3. The system transfers the member information from the HS database to the Article Manager (AM) database. If there is no email address in the HS database, the editor is prompted for an entry in that field. 4. The information is entered into the AM database. |
| **Alternative Paths** | In step 3, if there is no entry for the email address in the HS database or on this grid, the Editor will be reprompted for an entry. No validation for correctness is made. |
| **Postcondition** | The Reviewer has been added to the database. |
| **Exception Paths** | The Editor may abandon the operation at any time. |
| **Other** | The Reviewer information includes name, membership number, mailing address, categories of interest, and email address. |

### 3.2.5 Update Person

|  |  |
| --- | --- |
| **Use Case Name** | Update Person |
| **XRef** | Sec 2.2.4 Update Author; Sec 2.2.4 Update Reviewer  SDD, Section 7.5 |
| **Trigger** | The Editor selects to update an author or reviewer and the person is already in the database. |
| **Precondition** | The Editor has accessed the Article Manager main screen. |
| **Basic Path** | 1. The Editor selects Author or Reviewer. 2. The system creates and presents an alphabetical list of people in the category. 3. The Editor selects a person to update. 4. The system presents the database information in grid form for modification. 5. The Editor updates the information and submits the form. 6. The system checks that required fields are not blank. |
| **Alternative Paths** | In step 5, if any required field is blank, the Editor is instructed to add an entry. No validation for correctness is made. |
| **Postcondition** | The database has been updated. |
| **Exception Paths** | If the person is not already in the database, the use case is abandoned. In addition, the Editor may abandon the operation at any time. |
| **Other** | This use case is not used when one of the other use cases is more appropriate, such as to add an article or a reviewer for an article. |

### 3.2.6 Update Article Status

|  |  |
| --- | --- |
| **Use Case Name** | Update Article Status |
| **XRef** | Section 2.2.4, Update Article  SDD, Section 7.6 |
| **Trigger** | The Editor selects to update the status of an article in the database. |
| **Precondition** | The Editor has accessed the Article Manager main screen and the article is already in the database. |
| **Basic Path** | 1. The system creates and presents an alphabetical list of all active articles. 2. The Editor selects the article to update. 3. The system presents the information about the article in grid format. 4. The Editor updates the information and resubmits the form. |
| **Alternative Paths** | In step 4, the use case *Enter Communication* may be invoked. |
| **Postcondition** | The database has been updated. |
| **Exception Paths** | If the article is not already in the database, the use case is abandoned. In addition, the Editor may abandon the operation at any time. |
| **Other** | This use case can be used to add categories for an article, to correct typographical errors, or to remove a reviewer who has missed a deadline for returning a review. It may also be used to allow access to the named use case to enter an updated article or a review for an article. |

### 3.2.7 Enter Communication

|  |  |
| --- | --- |
| **Use Case Name** | Enter Communication |
| **XRef** | Section 2.2.4, Receive Article; Section 2.2.4, Receive Review  SDD, Section 7.7 |
| **Trigger** | The Editor selects to add a document to the system. |
| **Precondition** | The Editor has accessed the Article Manager main screen and has the file of the item to be entered available. |
| **Basic Path** | 1. The Editor selects the article using the *3.2.6, Update Article Status* use case. 2. The Editor attaches the file to the grid presented and updates the respective information about the article. 3. When the Editor updates the article status to indicate that a review is returned, the respective entry in the Reviewer table is updated. |
| **Alternative Paths** | None |
| **Postcondition** | The article entry is updated in the database. |
| **Exception Paths** | The Editor may abandon the operation at any time. |
| **Other** | This use case extends *3.2.6, Update Article Status* |

### 3.2.8 Assign Reviewer

|  |  |
| --- | --- |
| **Use Case Name** | Assign Reviewer |
| **XRef** | Section 2.2.4, Assign Reviewer  SDD, Section 7.8 |
| **Trigger** | The Editor selects to assign a reviewer to an article. |
| **Precondition** | The Editor has accessed the Article Manager main screen and the article is already in the database. . |
| **Basic Path** | 1. The Editor selects the article using the *3.2.6, Update Article Status* use case. 2. The system presents an alphabetical list of reviewers with their information. 3. The Editor selects a reviewer for the article. 4. The system updates the article database entry and emails the reviewer with the standard message and attaches the text of the article without author information. 5. The Editor has the option of repeating this use case from step 2. |
| **Alternative Paths** | None. |
| **Postcondition** | At least one reviewer has been added to the article information and the appropriate communication has been sent. |
| **Exception Paths** | The Editor may abandon the operation at any time. |
| **Other** | This use case extends *3.2.6, Update Article Status.* The Editor, prior to implementation of this use case, will provide the message text. |

### 3.2.9 Check Status

|  |  |
| --- | --- |
| **Use Case Name** | Check Status |
| **XRef** | Section 2.2.4, Check Status  SDD, Section 7.9 |
| **Trigger** | The Editor has selected to check status of all active articles. |
| **Precondition** | The Editor has accessed the Article Manager main screen. |
| **Basic Path** | 1. The system creates and presents a list of all active articles organized by their status. 2. The Editor may request to see the full information about an article. |
| **Alternative Paths** | None. |
| **Postcondition** | The requested information has been displayed. |
| **Exception Paths** | The Editor may abandon the operation at any time. |
| **Other** | The editor may provide an enhanced list of status later. At present, the following categories must be provided:   1. Received but no further action taken 2. Reviewers have been assigned but not all reviews are returned (include dates that reviewers were assigned and order by this criterion). 3. Reviews returned but no further action taken. 4. Recommendations for revision sent to Author but no response as of yet. 5. Author has revised article but no action has been taken. 6. Article has been accepted and copyright form has been sent. 7. Copyright form has been returned but article is not yet published.   A published article is automatically removed from the active article list. |

### 3.2.10 Send Communication

|  |  |
| --- | --- |
| **Use Case Name** | Send Communication |
| **XRef** | Section 2.2.4, Send Response; Section 2.2.4, Send Copyright  SDD, Section 7.10 |
| **Trigger** | The editor selects to send a communication to an author. |
| **Precondition** | The Editor has accessed the Article Manager main screen. |
| **Basic Path** | 1. The system presents an alphabetical list of authors. 2. The Editor selects an author. 3. The system invokes the Editor’s email system entering the author’s email address into the *To:* entry. 4. The Editor uses the email facility. |
| **Alternative Paths** | None. |
| **Postcondition** | The communication has been sent. |
| **Exception Paths** | The Editor may abandon the operation at any time. |
| **Other** | The standard copyright form will be available in the Editor’s directory for attaching to the email message, if desired. |

### 3.2.11 Publish Article

|  |  |
| --- | --- |
| **Use Case Name** | Publish Article |
| **XRef** | Section 2.2.4, Publish Article  SDD, Section 7.11 |
| **Trigger** | The Editor selects to transfer an approved article to the Online Journal. |
| **Precondition** | The Editor has accessed the Article Manager main screen. |
| **Basic Path** | 1. The system creates and presents an alphabetical list of the active articles that are flagged as having their copyright form returned. 2. The Editor selects an article to publish. 3. The system accesses the Online Database and transfers the article and its accompanying information to the Online Journal database. 4. The article is removed from the active article database. |
| **Alternative Paths** | None. |
| **Postcondition** | The article is properly transferred. |
| **Exception Paths** | The Editor may abandon the operation at any time. |
| **Other** | Find out from the Editor to see if the article information should be archived somewhere. |

### 3.2.12 Remove Article

|  |  |
| --- | --- |
| **Use Case Name** | Remove Article |
| **XRef** | Section 2.2.4, Remove Article  SDD, Section 7.12 |
| **Trigger** | The Editor selects to remove an article from the active article database. |
| **Precondition** | The Editor has accessed the Article Manager main screen. |
| **Basic Path** | 1. The system provides an alphabetized list of all active articles. 2. The editor selects an article. 3. The system displays the information about the article and requires that the Editor confirm the deletion. 4. The Editor confirms the deletion. |
| **Alternative Paths** | None. |
| **Postcondition** | The article is removed from the database. |
| **Exception Paths** | The Editor may abandon the operation at any time. |
| **Other** | Find out from the Editor to see if the article and its information information should be archived somewhere. |

## 3.3 Detailed Non-Functional Requirements

### 3.3.1 Logical Structure of the Data

The logical structure of the data to be stored in the internal Article Manager database is given below.

Review

Reviewer

Article

Author

writes

sent to

writes

has

Figure - Logical Structure of the Article Manager Data

The data descriptions of each of these data entities is as follows:

**Author Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of principle author |  |
| Email Address | Text | Internet address |  |
| Article | Pointer | Article entity | May be several |

**Reviewer Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of principle author |  |
| ID | Integer | ID number of Historical Society member | Used as key in Historical Society Database |
| Email Address | Text | Internet address |  |
| Article | Pointer | Article entity of | May be several |
| Num Review | Integer | Review entity | Number of not returned reviews |
| History | Text | Comments on past performance |  |
| Specialty | Category | Area of expertise | May be several |

**Review Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Article | Pointer | Article entity |  |
| Reviewer | Pointer | Reviewer entity | Single reviewer |
| Date Sent | Date | Date sent to reviewer |  |
| Returned | Date | Date returned; null if not returned |  |
| Contents | Text | Text of review |  |

**Article Data Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of Article |  |
| Author | Pointer | Author entity | Name of principle author |
| Other Authors | Text | Other authors is any; else null | Not a pointer to an Author entity |
| Reviewer | Pointer | Reviewer entity | Will be several |
| Review | Pointer | Review entity | Set up when reviewer is set up |
| Contents | Text | Body of article | Contains Abstract as first paragraph. |
| Category | Text | Area of content | May be several |
| Accepted | Boolean | Article has been accepted for publication | Needs Copyright form returned |
| Copyright | Boolean | Copyright form has been returned | Not relevant unless Accepted is True. |
| Published | Boolean | Sent to Online Journal | Not relevant unless Accepted is True. Article is no longer active and does not appear in status checks. |

The Logical Structure of the data to be stored in the Online Journal database on the server is as follows:

**Published Article Entity**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data Item** | **Type** | **Description** | **Comment** |
| Name | Text | Name of Article |  |
| Author | Text | Name of one Author | May be several |
| Abstract | Text | Abstract of article | Used for keyword search |
| Content | Text | Body of article |  |
| Category | Text | Area of content | May be several |

### 3.3.2 Security

The server on which the Online Journal resides will have its own security to prevent unauthorized *write*/*delete* access. There is no restriction on *read* access. The use of email by an Author or Reviewer is on the client systems and thus is external to the system.

The PC on which the Article Manager resides will have its own security. Only the Editor will have physical access to the machine and the program on it. There is no special protection built into this system other than to provide the editor with *write* access to the Online Journal to publish an article.

# Index

Abstract, 6, 17, 27

add, 9, 11, 19, 20, 21

Add, 8, 9, 19

Article, 1, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28

Article Manager, 5, 8, 9, 10, 11, 12, 13, 14, 15, 16, 19, 20, 21, 22, 23, 24, 25, 28

Author, 1, 4, 5, 6, 7, 8, 9, 13, 14, 16, 17, 19, 20, 22, 23, 25, 26, 27

Category, 5, 14, 17, 18, 20, 21, 23, 26, 27

Database, 2, 9, 11, 14, 15, 16, 17, 18, 19, 20, 21, 22, 24, 25, 26, 27

Editor, 1, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20, 21, 22, 23, 24, 25, 28

Field, 17, 19, 20

Form, 1, 6, 9, 10, 11, 12, 14, 19, 20, 21, 23, 24, 27

Grid, 9, 11, 12, 19, 20, 21

Historical Society, 1, 5, 9, 11, 16, 17, 19, 20, 26

Online Journal, 4, 5, 6, 7, 15, 16, 17, 18, 24, 27, 28

Reader, 4, 5, 6, 16, 17, 18

Review, 1, 7, 11, 12, 18, 21, 23, 26, 27

Reviewer, 1, 4, 5, 6, 7, 9, 11, 16, 17, 19, 20, 21, 22, 23, 26, 27

Security, 27, 28

Status, 11, 12, 13, 14, 17, 21, 22, 23, 27

update, 9, 11, 20, 21

Update, 8, 9, 10, 11, 12, 13, 14, 15, 17, 19, 20, 21, 22

User, 7, 16, 18

Web Publishing System, 1, 4, 5, 17