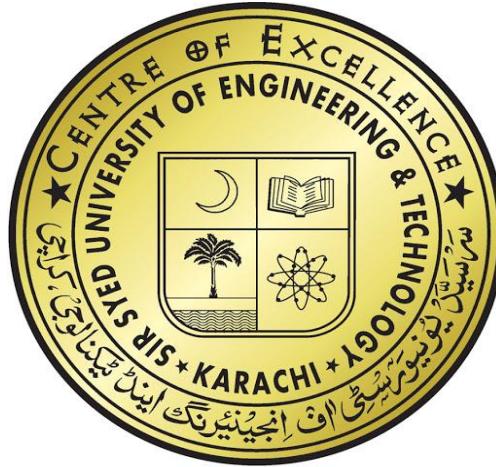


Software Laboratory Manual  
**Introduction to Software Engineering (SWE-106)**  
2nd Semester

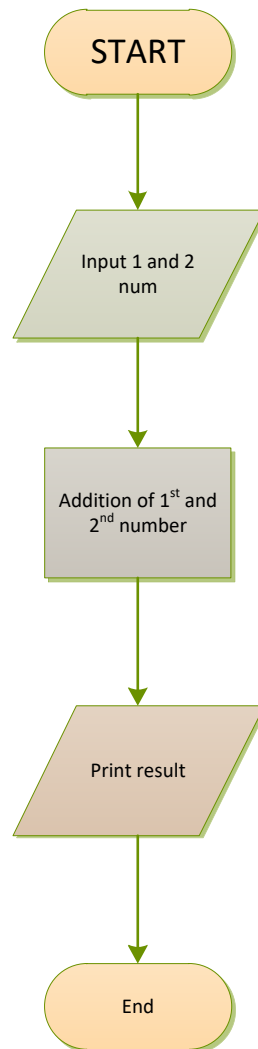


Name	:	Abdul Moiz Chishti Abdul Moiz Khan
Academic Session	:	2020F-2023
Roll No.	:	2020F-SE-022 2020F-SE-021
Discipline	:	Software Engineering
Section	:	A
Course Teacher	:	Sir Yusuf Ali Khan
Lab Teacher	:	Sir Yusuf Ali Khan

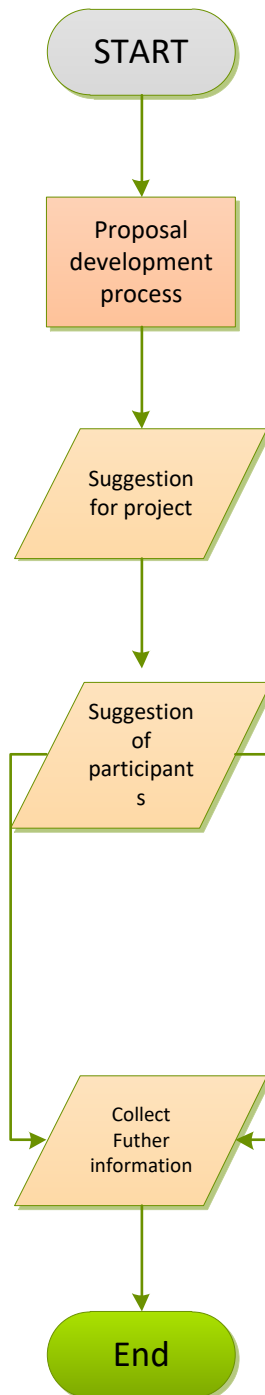
Software Engineering Department  
**Sir Syed University of Engineering & Technology**

# Lab 1

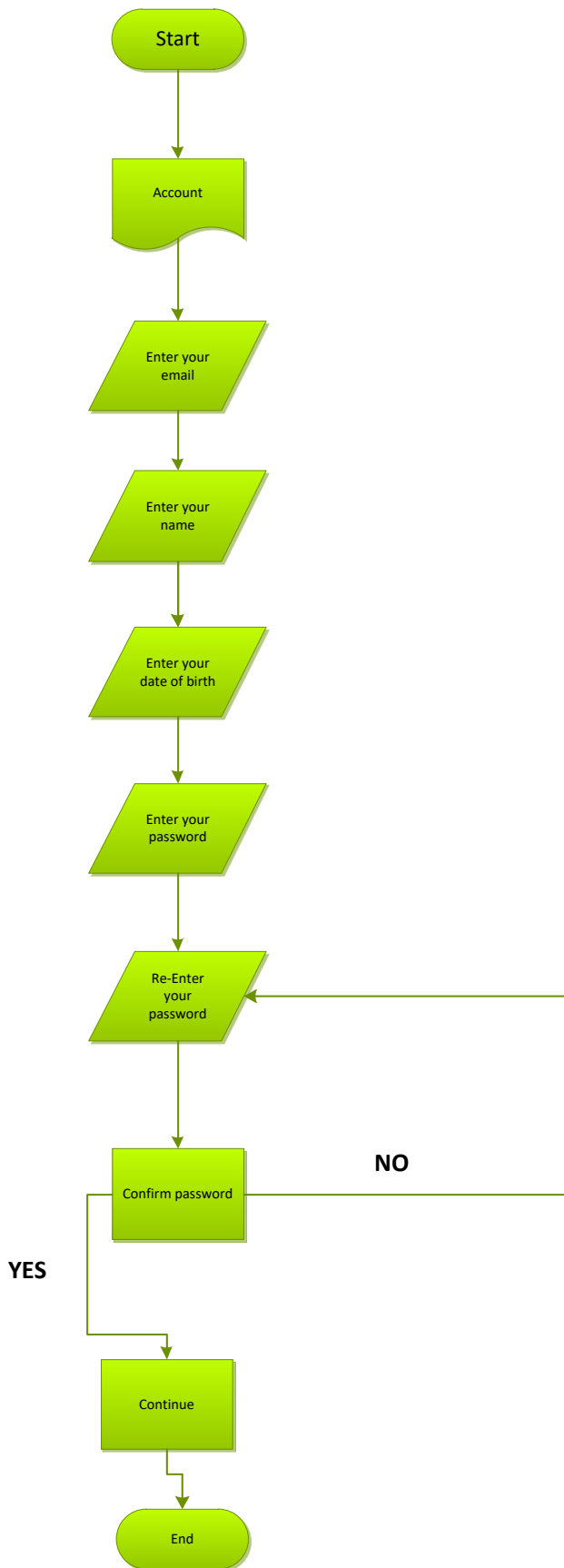
1. Draw a flowchart to add two numbers entered by user with the help of MS Visio tool.



2. Draw a flowchart to show basic business process like the proposal development process. Firstly, team members suggest a project then participants decide that if it's a good idea or not, if team rejects the idea then create further important information and if agrees then create further important information.

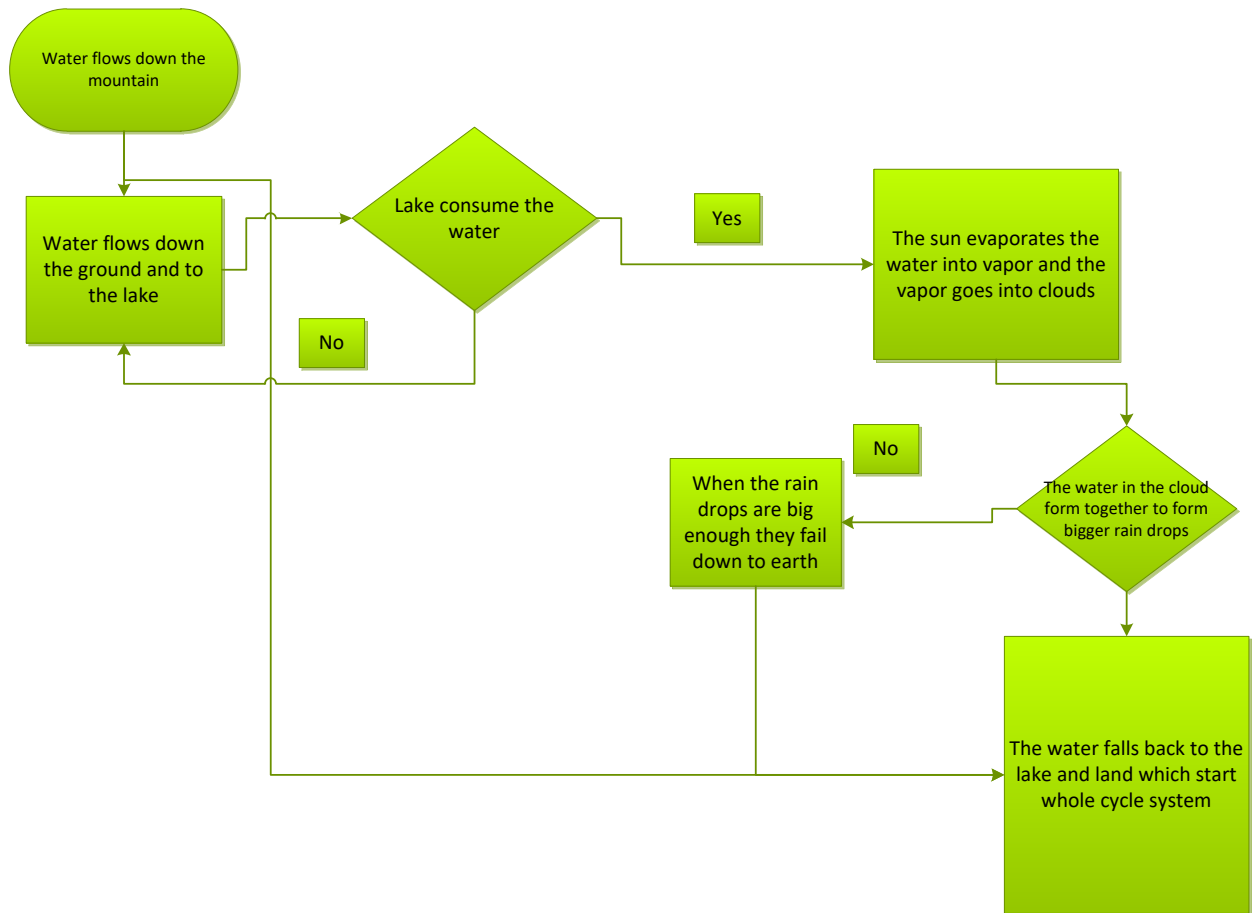


3. Draw a flowchart for Registration Form (or signup page). Mention all necessary fields to complete your registration

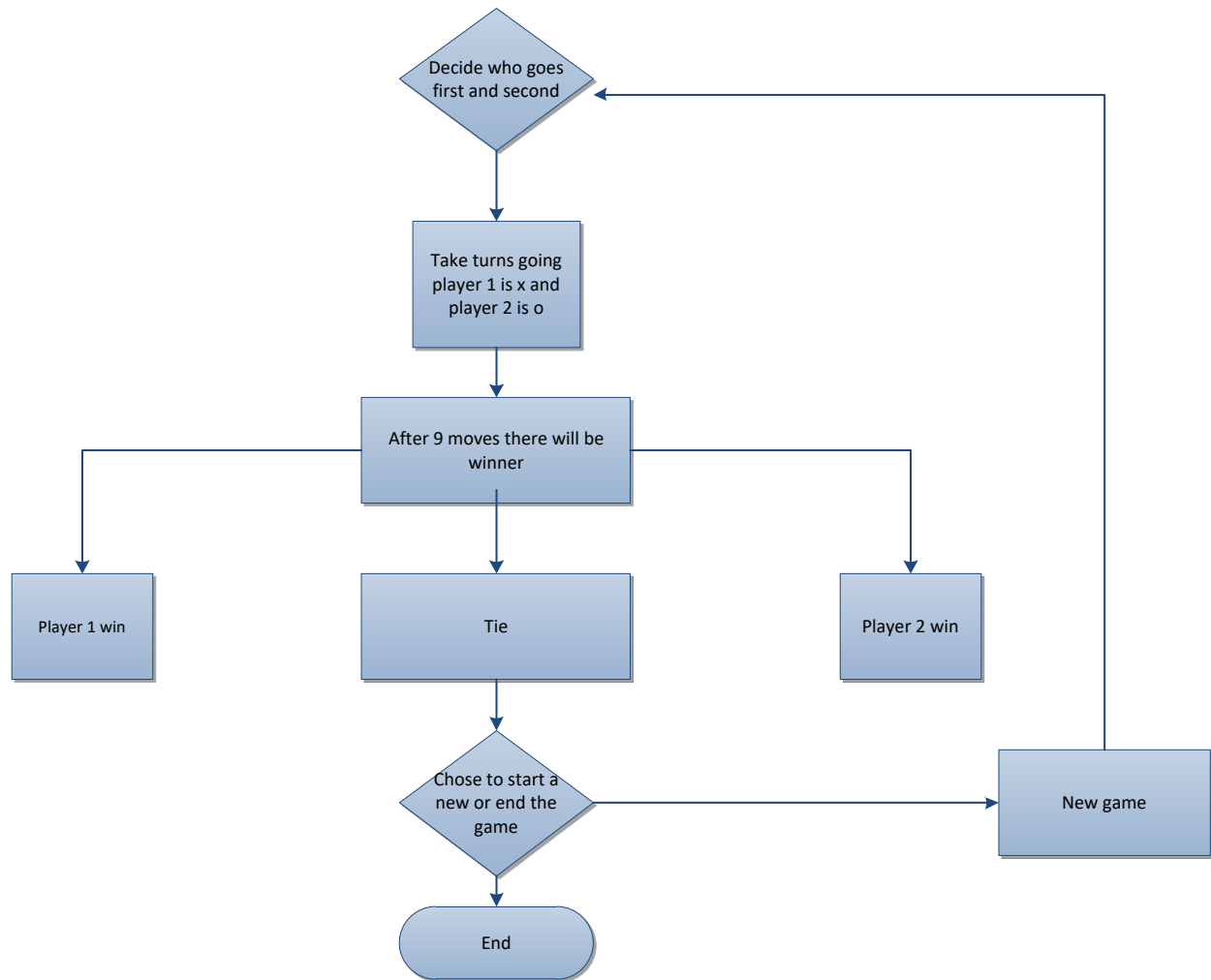


## Lab 2

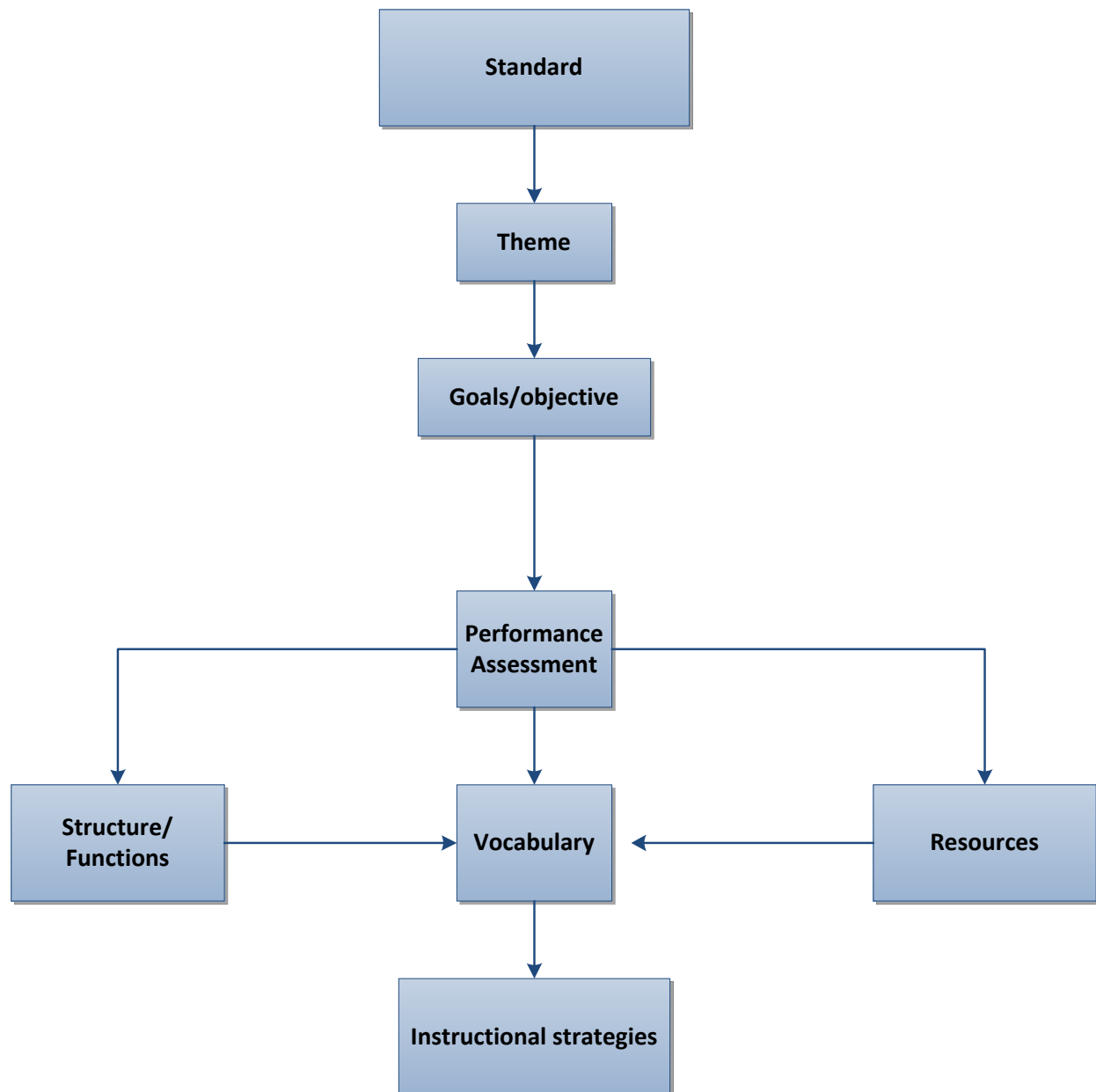
### Water Cycle Flowchart:



Tic tac toe flowchart:



Educational flowchart:



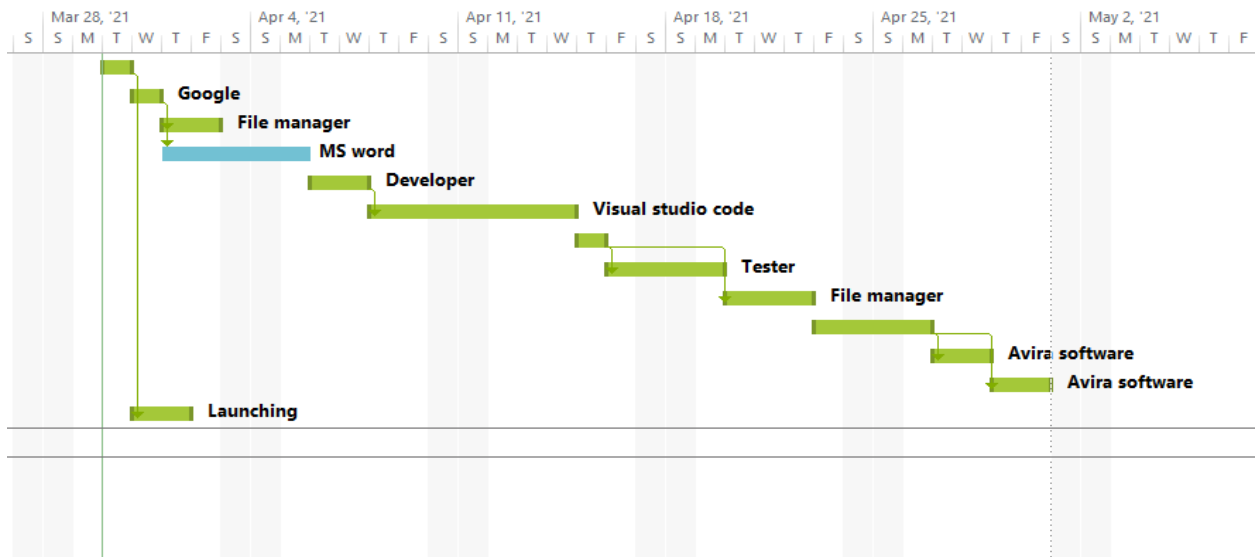
## Lab # 03

### Creating Gantt Chart & PERT Chart Using MS Project

#### Exercise:

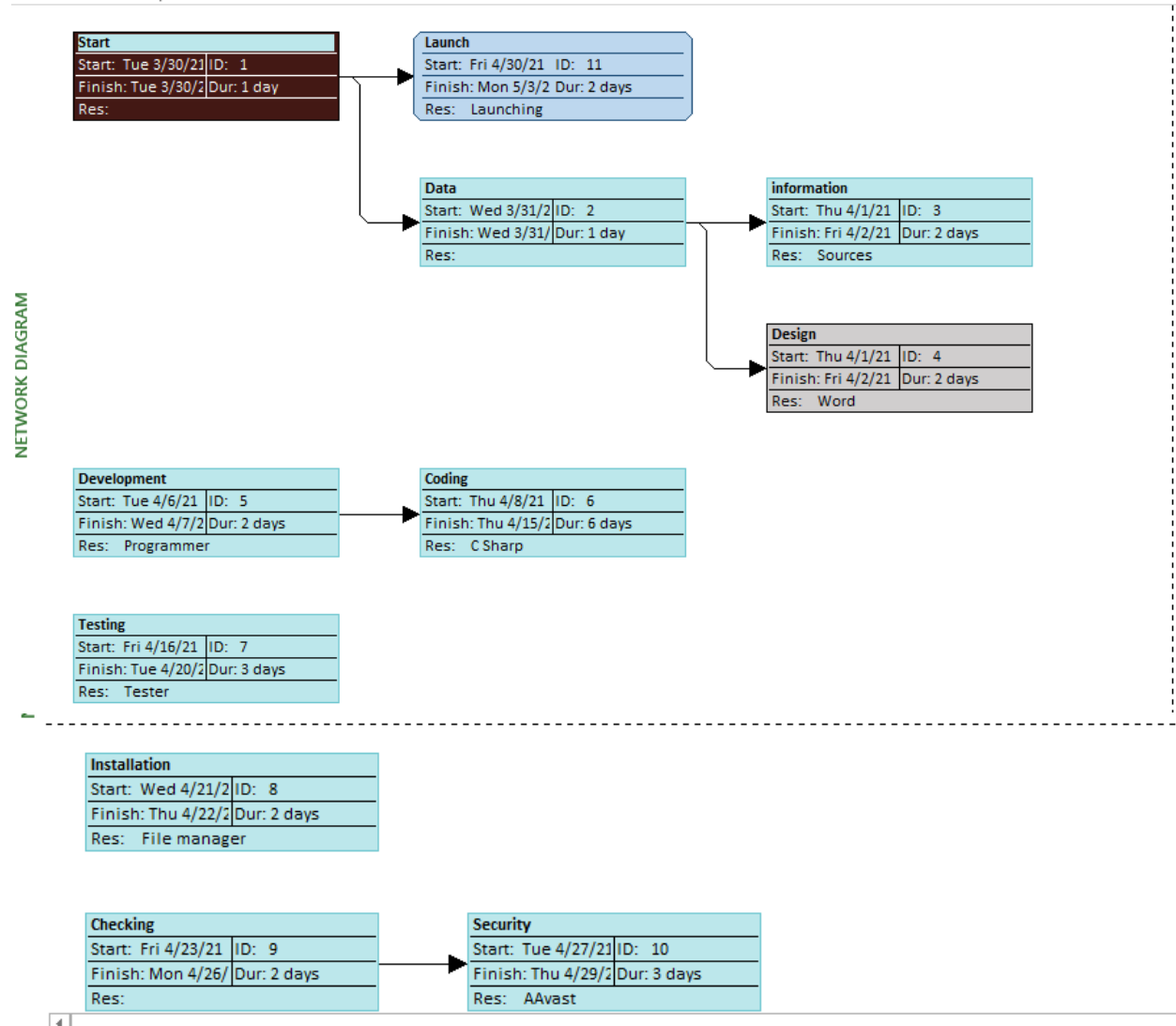
Create Gantt Chart and Network Diagram in Microsoft Project with above mention projects.  
Social website like (Facebook, Twitteretc.)

✦	Start	1 day	Tue 3/30/21	Tue 3/30/21		
✦	Data	1 day	Wed 3/31/21	Wed 3/31/21	1	
✦	information	2 days	Thu 4/1/21	Fri 4/2/21	2	Sources
✦	Design	2 days	Thu 4/1/21	Fri 4/2/21	2	Word
✦	Development	2 days	Tue 4/6/21	Wed 4/7/21		Programmer
✦	Coding	6 days	Thu 4/8/21	Thu 4/15/21	5	C Sharp
✦	Testing	3 days	Fri 4/16/21	Tue 4/20/21		Tester
✦	Installation	2 days	Wed 4/21/21	Thu 4/22/21		File manager
✦	Checking	2 days	Fri 4/23/21	Mon 4/26/21		
✦	Security	3 days	Tue 4/27/21	Thu 4/29/21	9	AAvast
✦	Launch	2 days	Fri 4/30/21	Mon 5/3/21	1	Launching



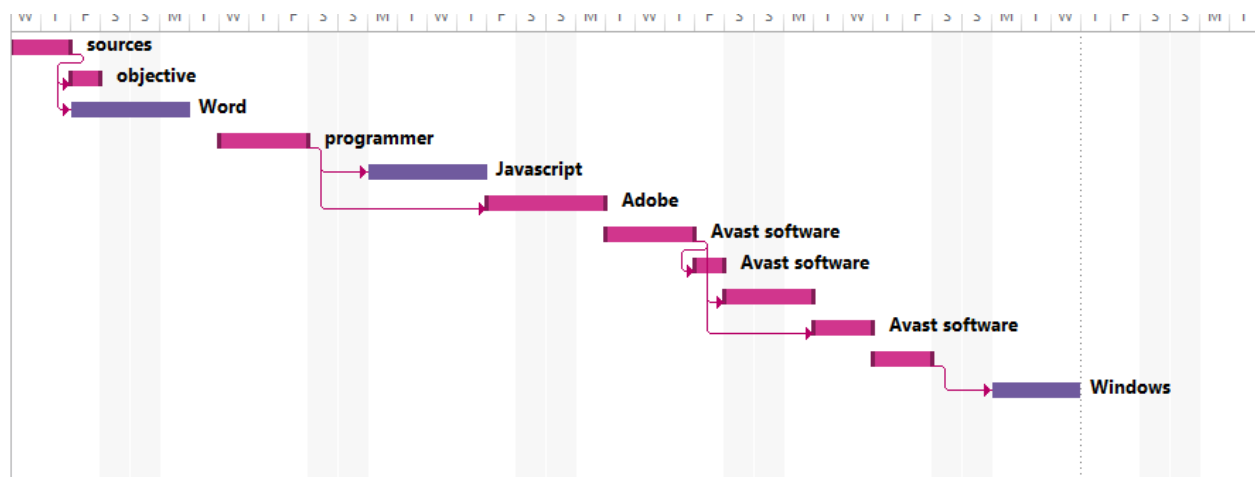


## Network Diagram:



# Hospital Management System

🚀	Research	2 days	Wed 3/31/21	Thu 4/1/21		sources	
🚀	Objects	1 day	Fri 4/2/21	Fri 4/2/21	1	objective	
🔧	Design	2 days	Fri 4/2/21	Mon 4/5/21	1	Word	
🚀	Establishment	3 days	Wed 4/7/21	Fri 4/9/21		programmer	
🔧	Coding	4 days	Mon 4/12/21	Thu 4/15/21	4	Javascript	
🚀	Editing	2 days	Fri 4/16/21	Mon 4/19/21	4	Adobe	
🚀	Testing	3 days	Tue 4/20/21	Thu 4/22/21		Avast software	
🚀	Faults	1 day	Fri 4/23/21	Fri 4/23/21	7	Avast software	
🚀	Quality controls	2 days	Sat 4/24/21	Mon 4/26/21	7		
🚀	Security	2 days	Tue 4/27/21	Wed 4/28/21	7	Avast software	
🚀	Implementation	2 days	Thu 4/29/21	Fri 4/30/21			
🔧	Launching	3 days	Mon 5/3/21	Wed 5/5/21	11	Windows	



## Network Diagram:

NETWORK DIAGRAM

Research
Start: Wed 3/31/21 ID: 1
Finish: Thu 4/1/21 Dur: 2 days
Res: sources

Objects
Start: Fri 4/2/21 ID: 2
Finish: Fri 4/2/21 Dur: 1 day
Res: objective

Design
Start: Fri 4/2/21 ID: 3
Finish: Mon 4/5/21 Dur: 2 days
Res: Word

Establishment
Start: Wed 4/7/21 ID: 4
Finish: Fri 4/9/21 Dur: 3 days
Res: programmer

Coding
Start: Mon 4/12/21 ID: 5
Finish: Thu 4/15/21 Dur: 4 days
Res: Javascript

Editing
Start: Fri 4/16/21 ID: 6
Finish: Mon 4/19/2 Dur: 2 days
Res: Adobe

Testing
Start: Tue 4/20/21 ID: 7
Finish: Thu 4/22/21 Dur: 3 days
Res: Avast software

Faults
Start: Fri 4/23/21 ID: 8
Finish: Fri 4/23/21 Dur: 1 day
Res: Avast software

Quality controls
Start: Sat 4/24/21 ID: 9
Finish: Mon 4/26/2 Dur: 2 days
Res:

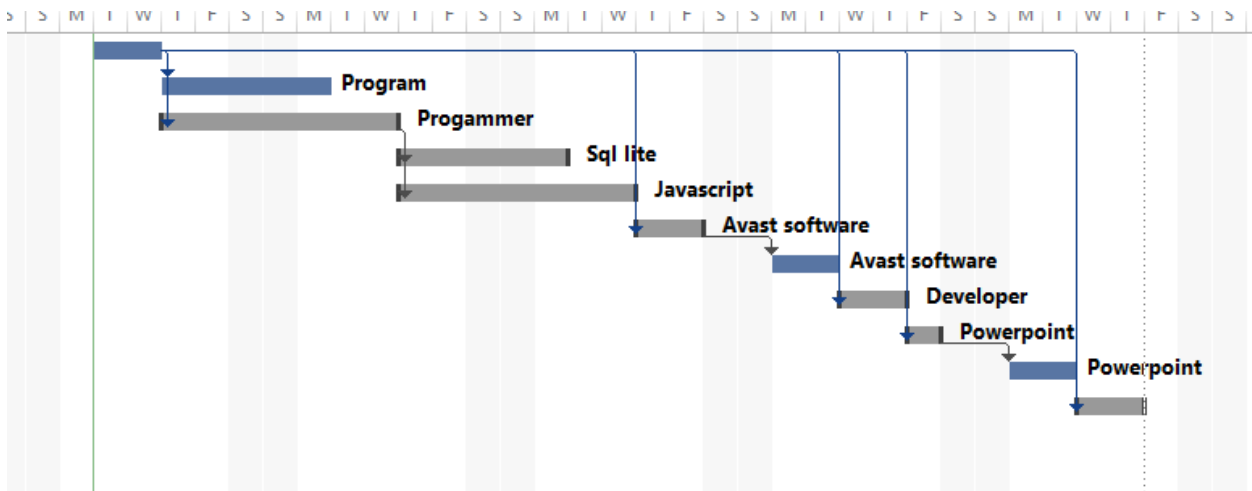
Security
Start: Tue 4/27/21 ID: 10
Finish: Wed 4/28/2 Dur: 2 days
Res: Avast software

Implementation
Start: Thu 4/29/21 ID: 11
Finish: Fri 4/30/21 Dur: 2 days
Res:

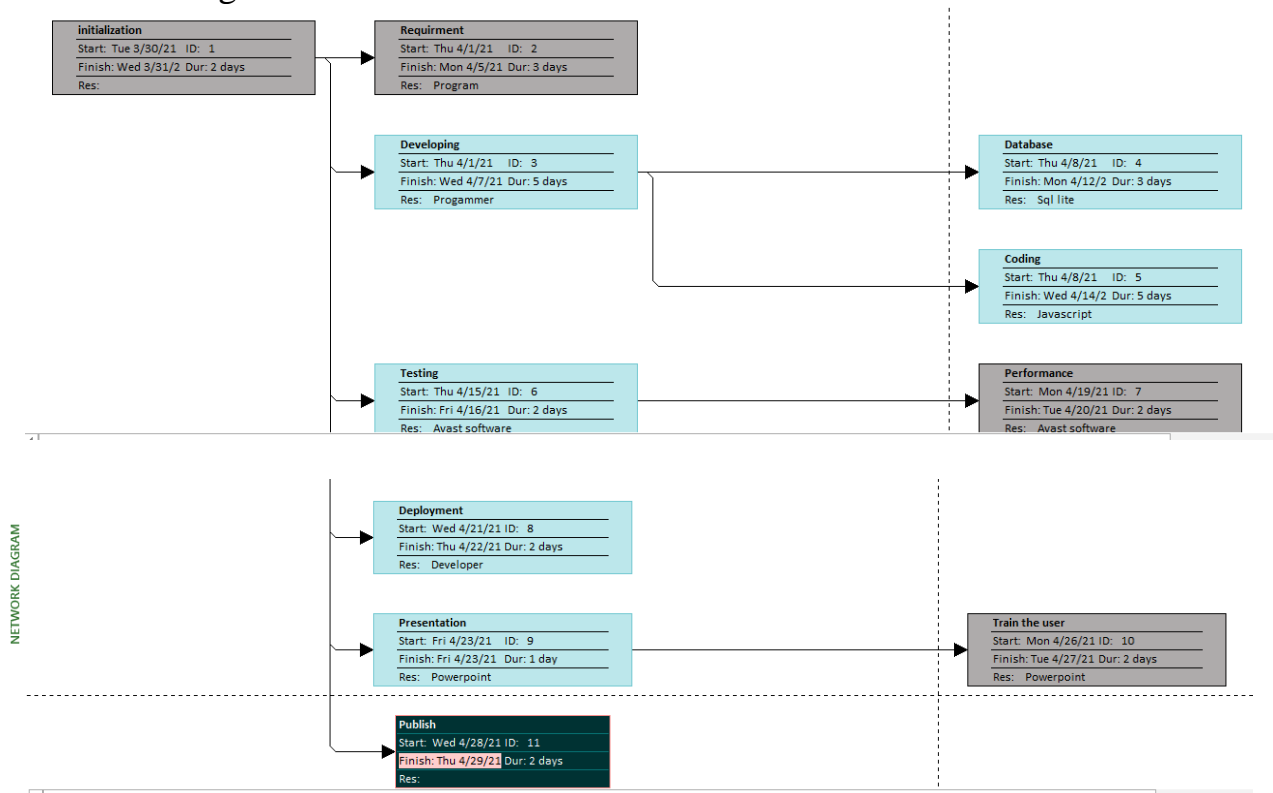
Launching
Start: Mon 5/3/21 ID: 12
Finish: Wed 5/5/21 Dur: 3 days
Res: Windows

## Faculty Management System:

Mode	TASK ID/NAME	DURATION	Start	Finish	Predecessors	Names	ADD /REV
→	initialization	2 days	Tue 3/30/21	Wed 3/31/21			
→	Requirment	3 days	Thu 4/1/21	Mon 4/5/21	1	Program	
★	Developing	5 days	Thu 4/1/21	Wed 4/7/21	1	Progammer	
★	Database	3 days	Thu 4/8/21	Mon 4/12/21	3	Sql lite	
★	Coding	5 days	Thu 4/8/21	Wed 4/14/21	3	Javascript	
★	Testing	2 days	Thu 4/15/21	Fri 4/16/21	1	Avast software	
→	Performance	2 days	Mon 4/19/21	Tue 4/20/21	6	Avast software	
★	Deployment	2 days	Wed 4/21/21	Thu 4/22/21	1	Developer	
★	Presentation	1 day	Fri 4/23/21	Fri 4/23/21	1	Powerpoint	
→	Train the user	2 days	Mon 4/26/21	Tue 4/27/21	9	Powerpoint	
★	Publish	2 days	Wed 4/28/21	Thu 4/29/21	1		



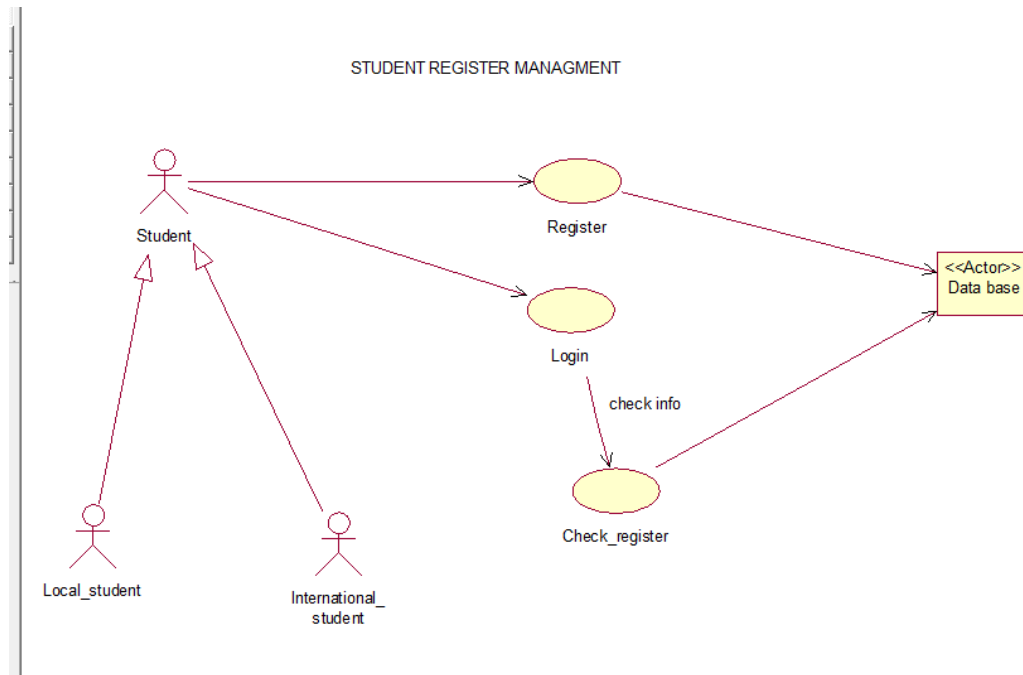
Network Diagram:



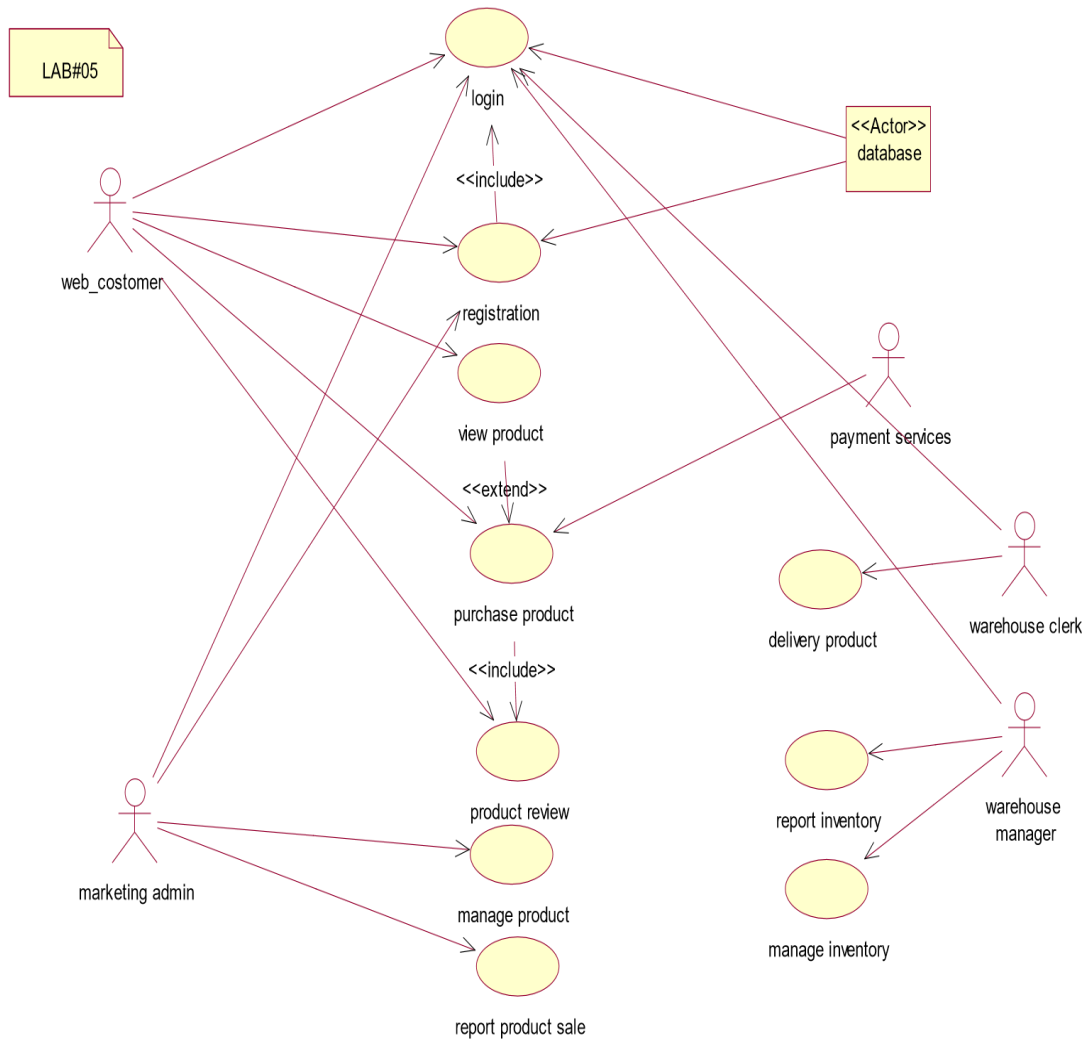
# Lab # 04

## Introduction to Rational Rose Software

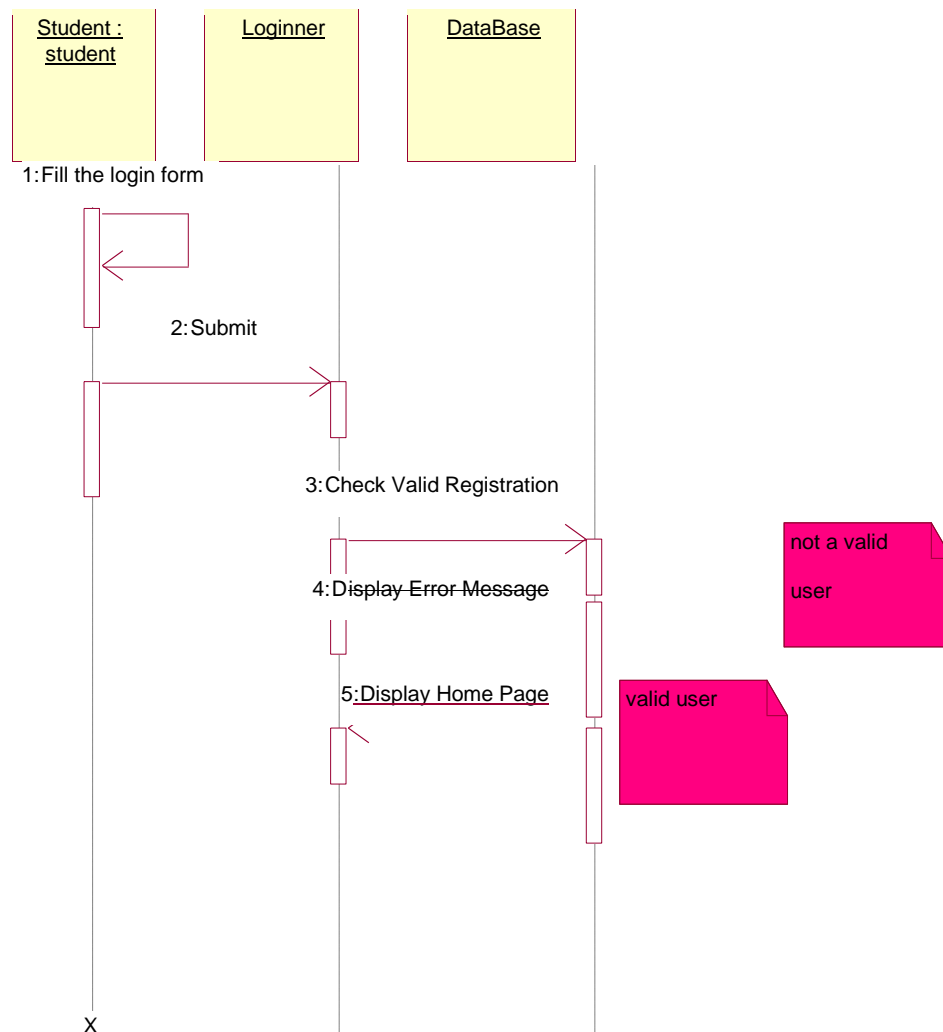
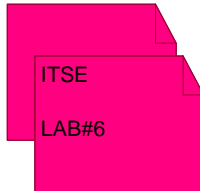
Student registration managment using rational rose



# Lab 5



# Lab- 06





## **Lab - 7 and 8**

### **Software Requirement Specification**

# **Software Requirements Specification**

Version 1.0

<<Annotated Version>>

April 15, 2004

Web Publishing System

Joan Team leader

Paul Adams

Bobbie Baker

Charles Charlie

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.

1 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 Thomas Clay, Dustin Denney, Erjon Dervishaj, Tiffanie Dew, Blake Guice, Jonathan Medders, Marla Medders, Tammie Odom, Amro Shorbatli, Joseph Smith, Jay Snellen, Chase Tinney, and Stefanie Watts. >>

## Table of Contents

Table of Contents .....	2
List of Figures .....	3
1.0. Introduction .....	4
1.1. Purpose .....	4
1.2. Scope of Project.....	4
1.3. Glossary.....	5
1.4. References .....	5
1.5. Overview of Document .....	5
2.0. Overall Description .....	6
2.1 System Environment .....	6
2.2 Functional Requirements Specification .....	7

2.2.1 Reader Use Case.....	7
Use case: Search Article .....	7
2.2.2 Author Use Case.....	8
Use case: Submit Article .....	8
2.2.3 Reviewer Use Case.....	9
Use case: Submit Review .....	9
2.2.4 Editor Use Cases .....	9
Use case: Update Author .....	10
Use case: Update Reviewer .....	11
Use case: Update Article .....	11
Use case: Receive Article .....	12
Use case: Assign Reviewer .....	13
Use case: Receive Review .....	13
Use case: Check Status .....	14
Use case: Send Response.....	14
Use case: Send Copyright.....	15
Use case: Remove Article .....	16
Use case: Publish Article.....	16
2.3 User Characteristics .....	17
2.4 Non-Functional Requirements .....	18
<b>3.0. Requirements Specification .....</b>	<b>18</b>
3.1 External Interface Requirements.....	18
3.2 Functional Requirements .....	18
3.2.1 Search Article .....	19
3.2.2 Communicate.....	19
3.2.3 Add Author.....	20
3.2.4 Add Reviewer.....	20
3.2.5 Update Person.....	21
3.2.6 Update Article Status.....	21
3.2.7 Enter Communication.....	22
3.2.8 Assign Reviewer .....	22
3.2.9 Check Status.....	23
3.2.10 Send Communication .....	24
3.2.11 Publish Article.....	24
3.2.12 Remove Article .....	25
3.3 Detailed Non-Functional Requirements.....	25
3.3.1 Logical Structure of the Data .....	25
3.3.2 Security .....	28
Index.....	28

## List of Figures

Figure 1 - System Environment .....	9
Figure 2 - Article Submission Process .....	11
Figure 3 - Editor Use Cases .....	13
Figure 4 - Logical Structure of the Article Manager Data .....	30

## **1.0. Introduction**

### **1.1. Purpose**

The purpose of this document is to present a detailed description of the Web Publishing 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 and will be proposed to the Regional Historical Society for its approval.

### **1.2. Scope of Project**

This software system will be a Web Publishing System for a local editor of a regional historical society. This system will be designed to maximize the editor's productivity by providing tools to assist in automating the article review and publishing process, which would otherwise have to be performed manually. By maximizing the editor's work efficiency and production the system will meet the editor's needs while remaining easy to understand and use.

More specifically, this system is designed to allow an editor to manage and communicate with a group of reviewers and authors to publish articles to a public website. The software will facilitate communication between authors, reviewers, and the editor via E-Mail. Preformatted reply forms are used in every stage of the articles' progress through the system to provide a uniform review process; the location of these forms is configurable via the application's maintenance options. The system also contains a relational database containing a list of Authors, Reviewers, and Articles.

### 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.
Author	Person submitting an article to be reviewed. In case of multiple authors, this term refers to the principal author, with whom all communication is made.
Database	Collection of all the information monitored by this system.
Editor	Person who receives articles, sends articles for review, and makes final judgments for publications.
Field	A cell within a form.
Historical Society Database	The existing membership database (also HS database).
Member	A member of the Historical Society listed in the HS database.
Reader	Anyone visiting the site to read articles.
Review	A written recommendation about the appropriateness of an article for publication; may include suggestions for improvement.
Reviewer	A person that examines an article and has the ability to recommend approval of the article for publication or to request that changes be made in the article.
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.
Stakeholder	Any person with an interest in the project who is not a developer.
User	Reviewer or Author.

### 1.4. References

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

### 1.5. Overview of Document

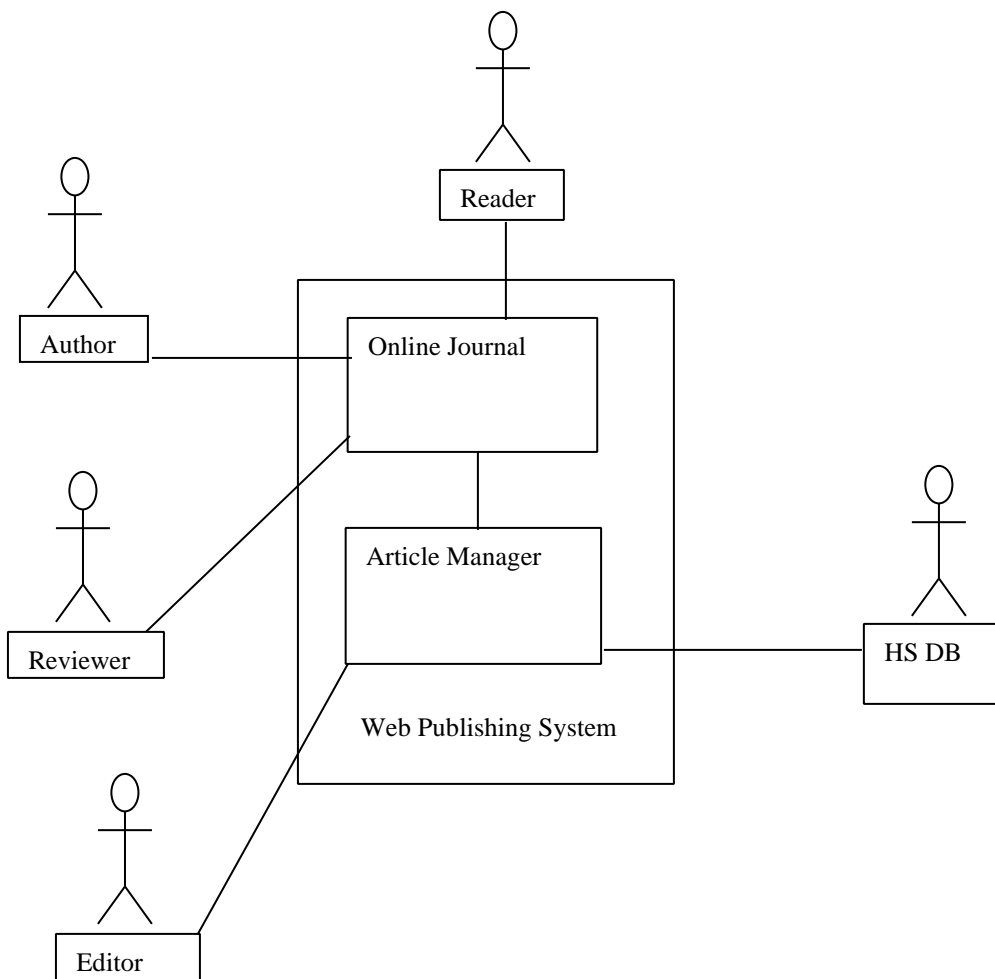
The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.

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.

Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different language.

## 2.0. Overall Description

### 2.1 System Environment



**Figure 1 - System Environment**

The Web Publishing System has four active actors and one cooperating system.

The Author, Reader, or Reviewer accesses the Online Journal through the Internet. Any Author or Reviewer communication with the system is through email. The Editor accesses the entire system directly. There is a link to the (existing) Historical Society. << The division of the Web Publishing System into two component parts, the Online Journal and the Article Manager, is an example of using domain classes to make an explanation clearer. >>

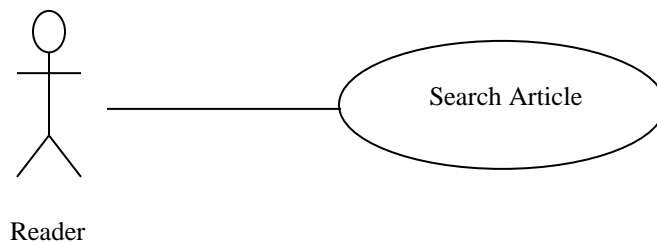
## 2.2 Functional Requirements Specification

This section outlines the use cases for each of the active readers separately. The reader, the author and the reviewer have only one use case apiece while the editor is main actor in this system.

### 2.2.1 Reader Use Case

*Use case: Search Article*

**Diagram:**



#### **Brief Description**

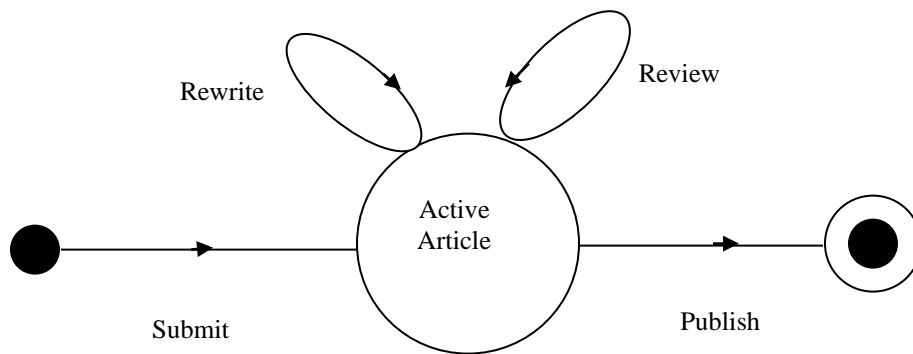
The Reader accesses the Online Journal Website, searches for an article and downloads it to his/her machine.

#### **Initial Step-By-Step Description**

Before this use case can be initiated, the Reader has already accessed the Online Journal Website.

1. The Reader chooses to search by author name, category, or keyword.
2. The system displays the choices to the Reader.
3. The Reader selects the article desired.
4. The system presents the abstract of the article to the reader.
5. The Reader chooses to download the article.
6. The system provides the requested article.

**Xref:** Section 3.2.1, Search Article



**Figure 2 - Article Submission Process**

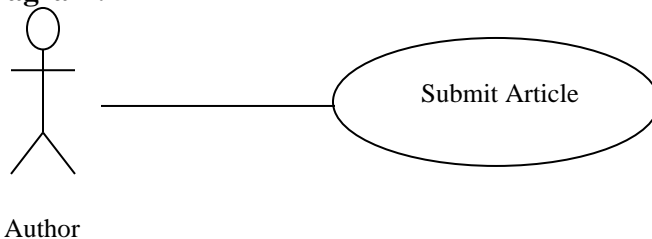
The Article Submission Process state-transition diagram summarizes the use cases listed below. An Author submits an article for consideration. The Editor enters it into the system and assigns it to and sends it to at least three reviewers. The Reviewers return their comments, which are used by the Editor to make a decision on the article. Either the article is accepted as written, declined, or the Author is asked to make some changes based on the reviews. If it is accepted, possibly after a revision, the Editor sends a copyright form to the Author. When that form is returned, the article is published to the Online Journal. Not shown in the above is the removal of a declined article from the system.

### 2.2.2 Author Use Case

In case of multiple authors, this term refers to the principal author, with whom all communication is made.

#### *Use case: Submit Article*

##### **Diagram:**



##### **Brief Description**

The author either submits an original article or resubmits an edited article.



### Initial Step-By-Step Description

Before this use case can be initiated, the Author has already connected to the Online Journal Website.

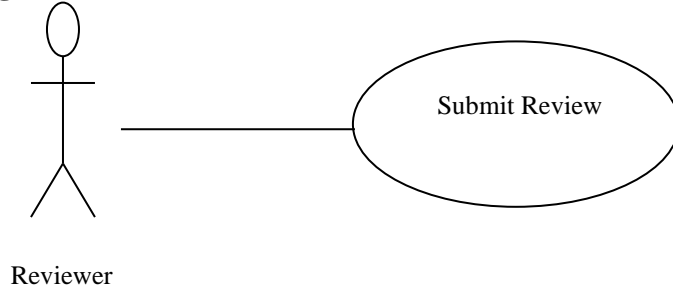
1. The Author chooses the Email Editor button.
2. The System uses the sendto HTML tag to bring up the user's email system.
3. The Author fills in the Subject line and attaches the files as directed and emails them.
4. The System generates and sends an email acknowledgement.

**Xref:** Section 3.2.2, Communicate

### 2.2.3 Reviewer Use Case

*Use case: Submit Review*

**Diagram:**



### Brief Description

The reviewer submits a review of an article.

### Initial Step-By-Step Description

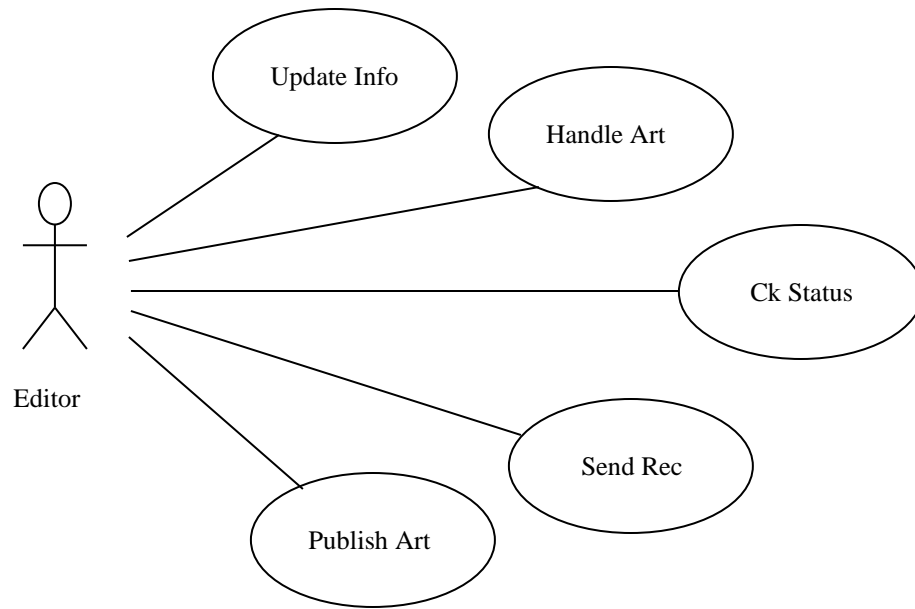
Before this use case can be initiated, the Reviewer has already connected to the Online Journal Website.

1. The Reviewer chooses the Email Editor button.
2. The System uses the sendto HTML tag to bring up the user's email system.
3. The Reviewer fills in the Subject line and attaches the file as directed and emails it.
4. The System generates and sends an email acknowledgement.

**Xref:** Section 3.2.2, Communicate

### 2.2.4 Editor Use Cases

The Editor has the following sets of use cases:

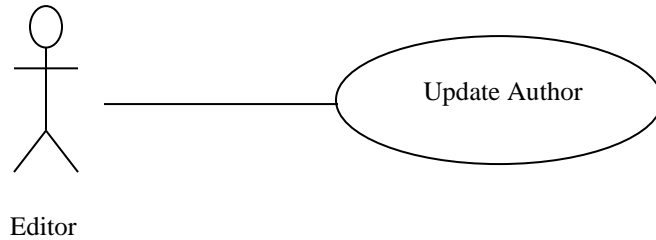


**Figure 3 - Editor Use Cases**

### Update Information use cases

*Use case: Update Author*

#### Diagram:



### Brief Description

The Editor enters a new Author or updates information about a current Author.

### Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager.

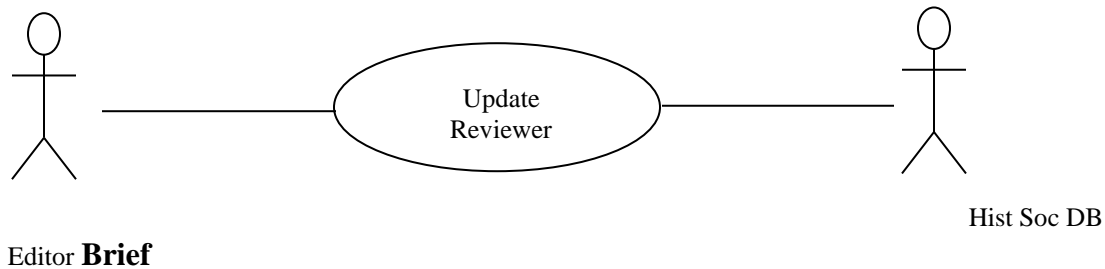
1. The Editor selects to Add/Update Author.
2. The system presents a choice of adding or updating.
3. The Editor chooses to add or to update.
4. If the Editor is updating an Author, the system presents a list of authors to choose from and presents a grid filling in with the information; else the system presents a blank grid.
5. The Editor fills in the information and submits the form.

6. The system verifies the information and returns the Editor to the Article Manager main page.

**Xref:** Section 3.2.3, Add Author; Section 3.2.5 Update Person

*Use case: Update Reviewer*

**Diagram:**



### **Description**

The Editor enters a new Reviewer or updates information about a current Reviewer.

### **Initial Step-By-Step Description**

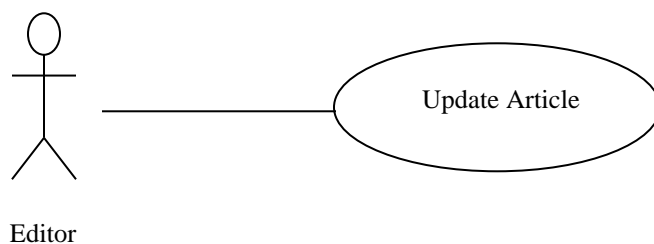
Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager.

1. The Editor selects to Add/Update Reviewer.
2. The system presents a choice of adding or updating.
3. The Editor chooses to add or to update.
4. The system links to the Historical Society Database.
5. If the Editor is updating a Reviewer, the system and presents a grid with the information about the Reviewer; else the system presents list of members for the editor to select a Reviewer and presents a grid for the person selected.
6. The Editor fills in the information and submits the form.
7. The system verifies the information and returns the Editor to the Article Manager main page.

**Xref:** Section 3.2.4, Add Reviewer; Section 3.2.5, Update Person

*Use case: Update Article*

**Diagram:**



### **Brief Description**

The Editor enters information about an existing article.

### **Initial Step-By-Step Description**

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager.

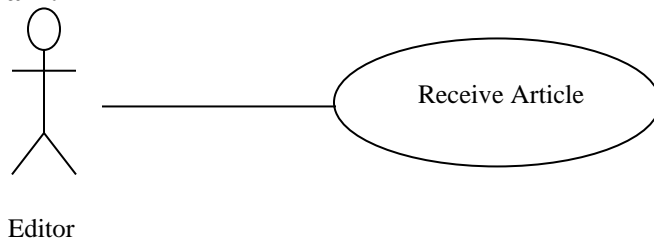
1. The Editor selects to Update Article.
2. The system presents s list of active articles.
3. The system presents the information about the chosen article.
4. The Editor updates and submits the form.
5. The system verifies the information and returns the Editor to the Article Manager main page.

**Xref:**Section 3.2.6, Update Article Status

### **Handle Article use cases**

*Use case: Receive Article*

#### **Diagram:**



### **Brief Description**

The Editor enters a new or revised article into the system.

### **Initial Step-By-Step Description**

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager and has a file containing the article available.

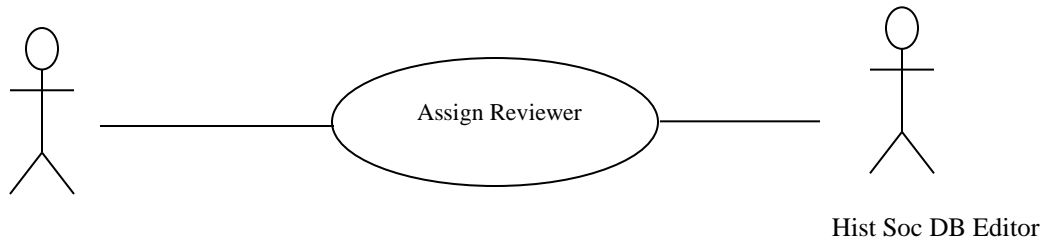
1. The Editor selects to Receive Article.
2. The system presents a choice of entering a new article or updating an existing article.
3. The Editor chooses to add or to update.
4. If the Editor is updating an article, the system presents a list of articles to choose from and presents a grid for filling with the information; else the system presents a blank grid.
5. The Editor fills in the information and submits the form.
6. The system verifies the information and returns the Editor to the Article Manager main page.

**Xref:** Section 3.2.7, Enter Communication

### *Use case: Assign Reviewer*

This use case extends the Update Article use case.

#### **Diagram:**



#### **Brief Description**

The Editor assigns one or more reviewers to an article.

#### **Initial Step-By-Step Description**

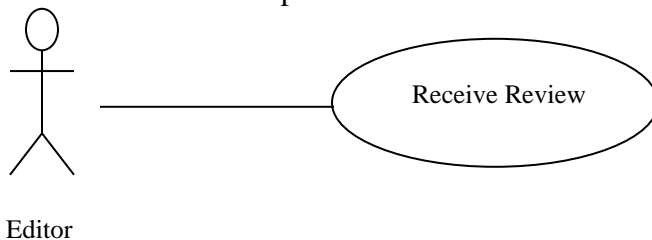
Before this use case can be initiated, the Editor has already accessed the article using the Update Article use case.

1. The Editor selects to Assign Reviewer.
2. The system presents a list of Reviewers with their status (see data description is section 3.3 below).
3. The Editor selects a Reviewer.
4. The system verifies that the person is still an active member using the Historical Society Database.
5. The Editor repeats steps 3 and 4 until sufficient reviewers are assigned.
6. The system emails the Reviewers, attaching the article and requesting that they do the review.
7. The system returns the Editor to the Update Article use case.

**Xref:** Section 3.2.8, Assign Reviewer

### *Use case: Receive Review*

This use case extends the Update Article use case.



#### **Diagram:**

#### **Brief Description**

The Editor enters a review into the system.

### Initial Step-By-Step Description

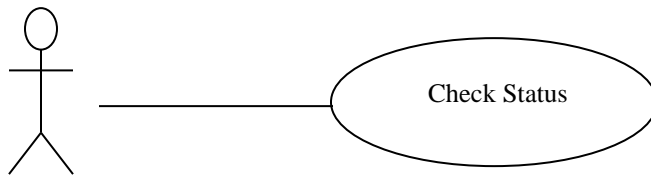
Before this use case can be initiated, the Editor has already accessed the article using the Update Article use case.

1. The Editor selects to Receive Review.
2. The system presents a grid for filling with the information.
3. The Editor fills in the information and submits the form.
4. The system verifies the information and returns the Editor to the Article Manager main page.

**Xref:** Section 3.2.7, Enter Communication

### Check Status use case:

*Use case: Check Status* **Diagram:**



Editor **Brief**

### Description

The Editor checks the status of all active articles.

### Initial Step-By-Step Description

Before this use case can be initiated, the Editor has already accessed the main page of the Article Manager.

1. The Editor selects to Check Status.
2. The system returns a scrollable list of all active articles with their status (see data description in section 3.3 below).
3. The system returns the Editor to the Article Manager main page.

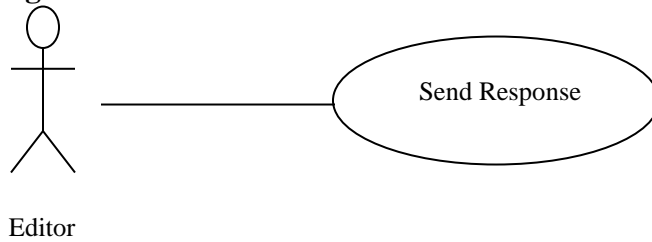
**Xref:**Section 3.2.9, Check Status

### Send Recommendation use cases:

*Use case: Send Response*

This use case extends the Update Article use case.

**Diagram:**



**Brief Description**

The Editor sends a response to an Author.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Editor has already accessed the article using the Update Article use case.

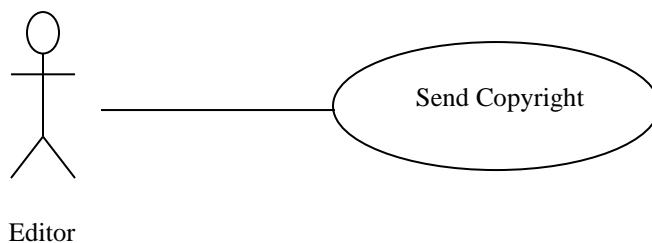
1. The Editor selects to Send Response.
2. The system calls the email system and puts the Author's email address in the Recipient line and the name of the article on the subject line.
3. The Editor fills out the email text and sends the message.
4. The system returns the Editor to the Article Manager main page.

**Xref:**Section 3.210, Send Communication

*Use case: Send Copyright*

This use case extends the Update Article use case.

**Diagram:**



**Brief Description**

The Editor sends a copyright form to an Author.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Editor has already accessed the article using the Update Article use case.

1. The Editor selects to Send Copyright.

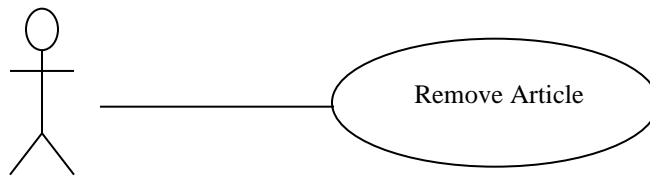
2. The system calls the email system and puts the Author's email address in the Recipient line, the name of the article on the subject line, and attaches the copyright form.
3. The Editor fills out the email text and sends the message.
4. The system returns the Editor to the Article Manager main page.

**Xref:**Section 3.2.10, Send Communication

*Use case: Remove Article*

This use case extends the Update Article use case.

**Diagram:**



Editor **Brief**

**Description**

The Editor removes an article from the active category.

**Initial Step-By-Step Description**

Before this use case can be initiated, the Editor has already accessed the article using the Update Article use case.

1. The Editor selects to remove an article from the active database.
2. The system provides a list of articles with the status of each.
3. The Editor selects an article for removal.
4. The system removes the article from the active article database and returns the Editor to the Article Manager main page.

**Xref:**Section 3.2.12, Remove Article

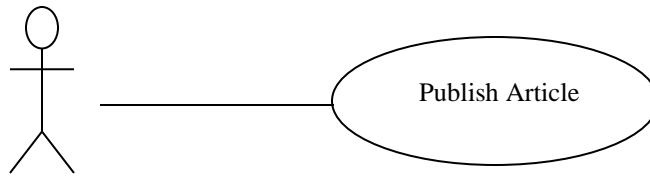
**Publish Article use case:**

*Use case: Publish Article*

This use case extends the Update Article use case.

**Diagram:**





Editor

### **Brief Description**

The Editor transfers an accepted article to the Online Journal.

### **Initial Step-By-Step Description**

Before this use case can be initiated, the Editor has already accessed the article using the Update Article use case.

1. The Editor selects to Publish Article.
2. The system transfers the article to the Online Journal and updates the search information there.
3. The system removes the article from the active article database and returns the Editor to the Article Manager home page.

**Xref:**Section 3.2.11, Publish Article

<< Since three of the actors only have one use case each, the summary diagram only involves the Editor. Adapt the rules to the needs of the document rather than adapt the document to fit the rules.

>>

## **2.3 User Characteristics**

The Reader is expected to be Internet literate and be able to use a search engine.

The main screen of the Online Journal Website will have the search function and a link to

“Author/Reviewer Information.”

The Author and Reviewer are expected to be Internet literate and to be able to use email with attachments.

The Editor is expected to be Windows literate and to be able to use button, pulldown menus, and similar tools.

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

## **2.4 Non-Functional Requirements**

The Online Journal will be on a server with high speed Internet capability. The physical machine to be used will be determined by the Historical Society. The software developed here assumes the use of a tool such as Tomcat for connection between the

Web pages and the database. The speed of the Reader's connection will depend on the hardware used rather than characteristics of this system.

The Article Manager will run on the editor's PC and will contain an Access database. Access is already installed on this computer and is a Windows operating system.

## **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

<b>Use Case Name</b>	Search Article
<b>XRef</b>	Section 2.2.1, Search Article SDD, Section 7.1
<b>Trigger</b>	The Reader assesses the Online Journal Website
<b>Precondition</b>	The Web is displayed with grids for searching
<b>Basic Path</b>	<ol style="list-style-type: none"><li>1. The Reader chooses how to search the Web site. The choices are by Author, by Category, and by Keyword.</li><li>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.</li><li>3. The Reader selects an author.</li><li>4. The system creates and presents a list of all articles by that author in the database.</li><li>5. The Reader selects an article.</li></ol>
	<ol style="list-style-type: none"><li>6. The system displays the Abstract for the article.</li><li>7. The Reader selects to download the article or to return to the article list or to the previous list.</li></ol>
<b>Alternative Paths</b>	<p>In step 2, if the Reader selects to search by category, the system creates and presents a list of all categories in the database.</p> <ol style="list-style-type: none"><li>3. The Reader selects a category.</li><li>4. The system creates and presents a list of all articles in that category in the database. Return to step 5.</li></ol> <p>In step 2, if the Reader selects to search by keyword, the system presents a dialog box to enter the keyword or phrase.</p> <ol style="list-style-type: none"><li>3. The Reader enters a keyword or phrase.</li><li>4. 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.</li></ol>
<b>Postcondition</b>	The selected article is downloaded to the client machine.
<b>Exception Paths</b>	The Reader may abandon the search at any time.
<b>Other</b>	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

<b>Use Case Name</b>	Communicate
<b>XRef</b>	Section 2.2.2, Submit Article; Section 2.2.3, Submit Review SDD, Section 7.2
<b>Trigger</b>	The user selects a mailto link.
<b>Precondition</b>	The user is on the Communicate page linked from the Online Journal Main Page.

<b>Basic Path</b>	This use case uses the mailto HTML tag. This invokes the client email facility.
<b>Alternative Paths</b>	If the user prefers to use his or her own email directly, sufficient information will be contained on the Web page to do so.
<b>Postcondition</b>	The message is sent.
<b>Exception Paths</b>	The attempt may be abandoned at any time.
<b>Other</b>	None

### 3.2.3 Add Author

<b>Use Case Name</b>	Add Author
<b>XRef</b>	Section 2.2.4, Update Author SDD, Section 7.3
<b>Trigger</b>	The Editor selects to add a new author to the database.
<b>Precondition</b>	The Editor has accessed the Article Manager main screen.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system presents a blank grid to enter the author information.</li> <li>2. The Editor enters the information and submits the form.</li> <li>3. The system checks that the name and email address fields are</li> </ol>
	not blank and updates the database.
<b>Alternative Paths</b>	If in step 2, either field is blank, the Editor is instructed to add an entry. No validation for correctness is made.
<b>Postcondition</b>	The Author has been added to the database.
<b>Exception Paths</b>	The Editor may abandon the operation at any time.
<b>Other</b>	The author information includes the name mailing address and email address.

### 3.2.4 Add Reviewer

<b>Use Case Name</b>	Add Reviewer
<b>XRef</b>	Section 2.2.4, Update Reviewer SDD, Section 7.4
<b>Trigger</b>	The Editor selects to add a new reviewer to the database.
<b>Precondition</b>	The Editor has accessed the Article Manager main screen.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system accesses the Historical Society (HS) database and presents an alphabetical list of the society members.</li> <li>2. The Editor selects a person.</li> <li>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.</li> <li>4. The information is entered into the AM database.</li> </ol>

<b>Alternative Paths</b>	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.
<b>Postcondition</b>	The Reviewer has been added to the database.
<b>Exception Paths</b>	The Editor may abandon the operation at any time.
<b>Other</b>	The Reviewer information includes name, membership number, mailing address, categories of interest, and email address.

### 3.2.5 Update Person

<b>Use Case Name</b>	Update Person
<b>XRef</b>	Sec 2.2.4 Update Author; Sec 2.2.4 Update Reviewer SDD, Section 7.5
<b>Trigger</b>	The Editor selects to update an author or reviewer and the person is already in the database.
<b>Precondition</b>	The Editor has accessed the Article Manager main screen.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The Editor selects Author or Reviewer.</li> <li>2. The system creates and presents an alphabetical list of people in the category.</li> <li>3. The Editor selects a person to update.</li> <li>4. The system presents the database information in grid form for modification.</li> <li>5. The Editor updates the information and submits the form.</li> <li>6. The system checks that required fields are not blank.</li> </ol>
<b>Alternative Paths</b>	In step 5, if any required field is blank, the Editor is instructed to add an entry. No validation for correctness is made.
<b>Postcondition</b>	The database has been updated.
<b>Exception Paths</b>	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.
<b>Other</b>	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

<b>Use Case Name</b>	Update Article Status
<b>XRef</b>	Section 2.2.4, Update Article SDD, Section 7.6
<b>Trigger</b>	The Editor selects to update the status of an article in the database.
<b>Precondition</b>	The Editor has accessed the Article Manager main screen and the article is already in the database.

<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system creates and presents an alphabetical list of all active articles.</li> <li>2. The Editor selects the article to update.</li> <li>3. The system presents the information about the article in grid format.</li> <li>4. The Editor updates the information and resubmits the form.</li> </ol>
<b>Alternative Paths</b>	In step 4, the use case Enter Communication may be invoked.
<b>Postcondition</b>	The database has been updated.
<b>Exception Paths</b>	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.
<b>Other</b>	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

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

### 3.2.8 Assign Reviewer

<b>Use Case Name</b>	Assign Reviewer
<b>XRef</b>	Section 2.2.4, Assign Reviewer SDD, Section 7.8
<b>Trigger</b>	The Editor selects to assign a reviewer to an article.

<b>Precondition</b>	The Editor has accessed the Article Manager main screen and the article is already in the database. .
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The Editor selects the article using the 3.2.6, Update Article Status use case.</li> <li>2. The system presents an alphabetical list of reviewers with their information.</li> <li>3. The Editor selects a reviewer for the article.</li> <li>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.</li> <li>5. The Editor has the option of repeating this use case from step 2.</li> </ol>
<b>Alternative Paths</b>	None.
<b>Postcondition</b>	At least one reviewer has been added to the article information and the appropriate communication has been sent.
<b>Exception Paths</b>	The Editor may abandon the operation at any time.
<b>Other</b>	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

<b>Use Case Name</b>	Check Status
<b>XRef</b>	Section 2.2.4, Check Status SDD, Section 7.9
<b>Trigger</b>	The Editor has selected to check status of all active articles.
<b>Precondition</b>	The Editor has accessed the Article Manager main screen.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system creates and presents a list of all active articles organized by their status.</li> <li>2. The Editor may request to see the full information about an article.</li> </ol>
<b>Alternative Paths</b>	None.
<b>Postcondition</b>	The requested information has been displayed.
<b>Exception Paths</b>	The Editor may abandon the operation at any time.

<b>Other</b>	<p>The editor may provide an enhanced list of status later. At present, the following categories must be provided:</p> <ol style="list-style-type: none"> <li>1. Received but no further action taken</li> <li>2. Reviewers have been assigned but not all reviews are returned (include dates that reviewers were assigned and order by this criterion).</li> <li>3. Reviews returned but no further action taken.</li> <li>4. Recommendations for revision sent to Author but no response as of yet.</li> <li>5. Author has revised article but no action has been taken.</li> <li>6. Article has been accepted and copyright form has been sent.</li> <li>7. Copyright form has been returned but article is not yet published.</li> </ol> <p>A published article is automatically removed from the active article list.</p>
--------------	---

### 3.2.10 Send Communication

<b>Use Case Name</b>	Send Communication
<b>XRef</b>	Section 2.2.4, Send Response; Section 2.2.4, Send Copyright SDD, Section 7.10
<b>Trigger</b>	The editor selects to send a communication to an author.
<b>Precondition</b>	The Editor has accessed the Article Manager main screen.
<b>Basic Path</b>	<ol style="list-style-type: none"> <li>1. The system presents an alphabetical list of authors.</li> <li>2. The Editor selects an author.</li> <li>3. The system invokes the Editor's email system entering the author's email address into the To: entry.</li> <li>4. The Editor uses the email facility.</li> </ol>
<b>Alternative Paths</b>	None.
<b>Postcondition</b>	The communication has been sent.
<b>Exception Paths</b>	The Editor may abandon the operation at any time.
<b>Other</b>	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

<b>Use Case Name</b>	Publish Article
<b>XRef</b>	Section 2.2.4, Publish Article SDD, Section 7.11
<b>Trigger</b>	The Editor selects to transfer an approved article to the Online Journal.
<b>Precondition</b>	The Editor has accessed the Article Manager main screen.



<b>Basic Path</b>	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.
<b>Alternative Paths</b>	None.
<b>Postcondition</b>	The article is properly transferred.
<b>Exception Paths</b>	The Editor may abandon the operation at any time.
<b>Other</b>	Find out from the Editor to see if the article information should be archived somewhere.

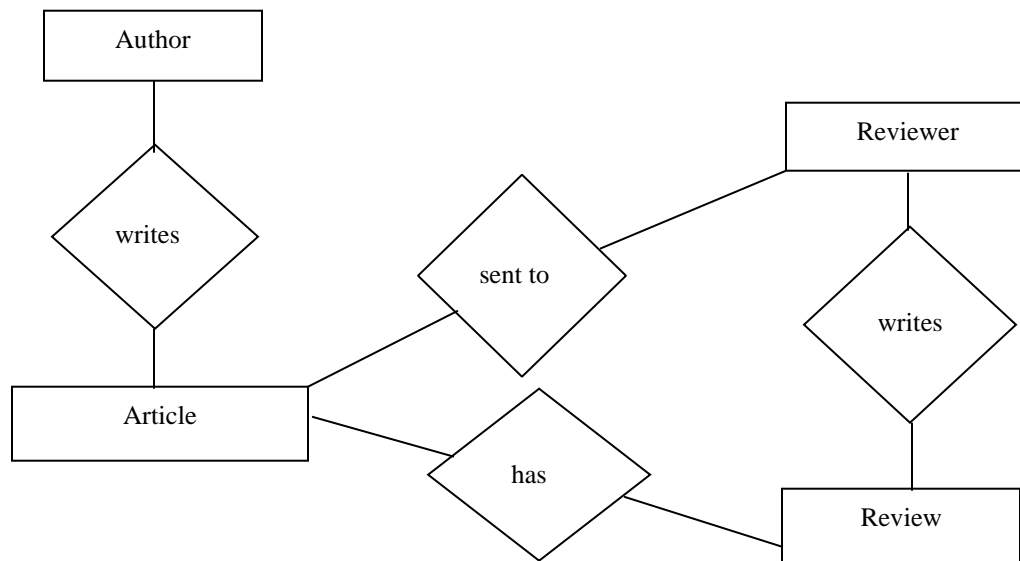
### 3.2.12 Remove Article

<b>Use Case Name</b>	Remove Article
<b>XRef</b>	Section 2.2.4, Remove Article SDD, Section 7.12
<b>Trigger</b>	The Editor selects to remove an article from the active article database.
<b>Precondition</b>	The Editor has accessed the Article Manager main screen.
<b>Basic Path</b>	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.
<b>Alternative Paths</b>	None.
<b>Postcondition</b>	The article is removed from the database.
<b>Exception Paths</b>	The Editor may abandon the operation at any time.
<b>Other</b>	Find out from the Editor to see if the article and its 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.



**Figure 4 - Logical Structure of the Article Manager Data**

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

#### **Author Data Entity**

<b>Data Item</b>	<b>Type</b>	<b>Description</b>	<b>Comment</b>
Name	Text	Name of principle author	
Email Address	Text	Internet address	
Article	Pointer	Article entity	May be several

#### **Reviewer Data Entity**

<b>Data Item</b>	<b>Type</b>	<b>Description</b>	<b>Comment</b>
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**

<b>Data Item</b>	<b>Type</b>	<b>Description</b>	<b>Comment</b>
------------------	-------------	--------------------	----------------

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



# **SIRSYED UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

## **DEPARTMENT OF SOFTWARE ENGINEERING**

### **Lab 9 Introduction to Software Engineering.**

#### **TITLE:**

**Login page(Html Code).**

<b>Abdul Moiz Khan</b>	<b>2020F-BSE-021</b>
<b>Abdul Moiz Chishti</b>	<b>2020F-BSE-022</b>
<b>Shaheer Khan Qureshi</b>	<b>2020F-BSE-003</b>
<b>Muhammad Talal Moin</b>	<b>2020F-BSE-016</b>

#### **GROUP LEADER:**

**Muhammad Talal Moin**

**Task:** To create a login page using Html, CSS and JavaScript.

**Program features:**

- There are two forms login and Register.
- In login form there are two fields Username and password with a check box of Remember Password.
- In Register Form there are three fields user Id , email and password with a checkbox of I agree to Terms & conditions
- If any of the field is empty and you click login an alert will appear that fields cannot be empty.
- You can go to Registration form clicking on Register button next to login button.

**Html Code:**

```
<html>
<head>
  <title> login form </title>
  <link rel="stylesheet" href="Style.css">
</head>
<body>
  <div class="background">
    <div class="namebox">
      <p>Developed By:</p>
      <p> Abdul Moiz Khan (SE20F-021)</p>
      <p> Abdul Moiz Chishti (SE20F-022)</p>
      <p> Muhammad Talal Moin (SE20F-016)</p>
      <p> Shaheer Khan Qureshi (SE20F-003)</p>
    </div>
    <div class="formbox">
      <div class="buttonbox">
        <div id="bt"></div>
        <button type="button" class="toggle-
bt" onclick="login()">Log In</button>
        <button type="button" class="toggle-
bt" onclick="register()">Register</button>
      </div>
      <div class="social-icons">
        
      </div>
    </div>
  </div>
</body>
</html>
```

```

        
        
    </div>
    <form id="login" class="input-group">
        <input type="text" class="input-
field" placeholder="User Id" required>
        <input type="password" class="input-
field" placeholder="Enter Password" required>
        <input type="checkbox" class="check-
box"><span>Remember Password</span>
        <button type="submit" class="submit-bt">Log In</button>
    </form>
    <form id="register" class="input-group">
        <input type="text" class="input-
field" placeholder="User Id" required>
        <input type="email" class="input-
field" placeholder="Email Id" required>
        <input type="password" class="input-
field" placeholder="Enter Password" required>
        <input type="checkbox" class="check-
box"><span>I Agree to the Terms & Conditions</span>
        <button type="submit" class="submit-bt">Register</button>
    </form>
</div>
<script>
    var x = document.getElementById("login");
    var y = document.getElementById("register");
    var z = document.getElementById("bt");

    function register(){
        x.style.left="-400px";
        y.style.left="50px";
        z.style.left="110px";
    }
    function login(){
        x.style.left="50px";
        y.style.left="450px";
        z.style.left="0";
    }
</script>

</body>

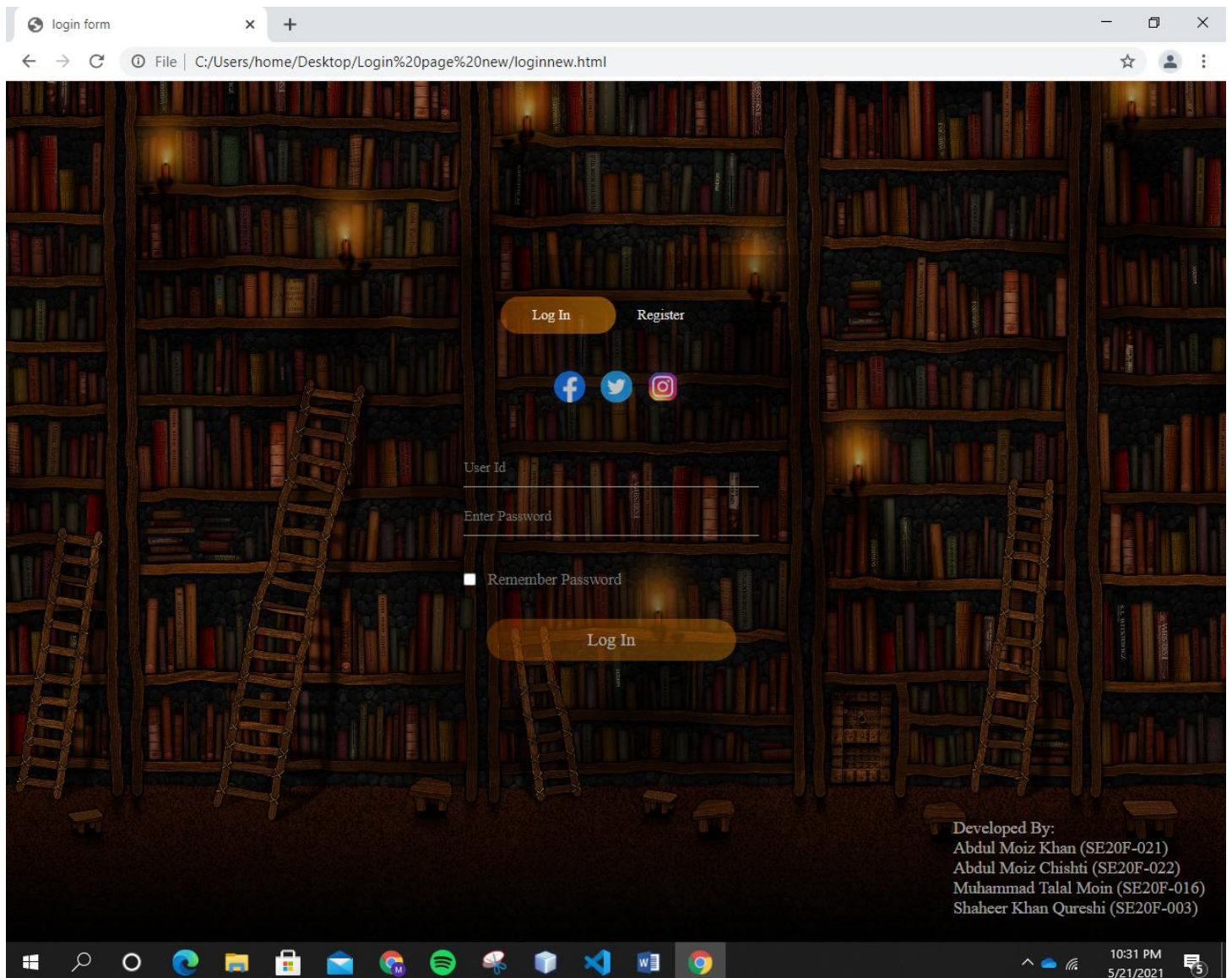
```



```
</html>
```

## Output:

- This is the Snapshot of login form.



- This is the Snapshot of Registration Form.

login form

File | C:/Users/home/Desktop/Login%20page%20new/loginnew.html

Log In Register

Facebook Twitter Instagram

User Id

Email Id

Enter Password

☒ I Agree to the Terms & Conditions

Register

Developed By:  
Abdul Moiz Khan (SE20F-021)  
Abdul Moiz Chishti (SE20F-022)  
Muhammad Talal Moin (SE20F-016)  
Shaheer Khan Qureshi (SE20F-003)

10:33 PM  
5/21/2021



# **SIRSYED UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

## **DEPARTMENT OF SOFTWARE ENGINEERING**

### **Lab 10 Introduction to Software Engineering.**

#### **TITLE:**

**Login page (CSS & JavaScript Code).**

<b>Abdul Moiz Khan</b>	<b>2020F-BSE-021</b>
<b>Abdul Moiz Chishti</b>	<b>2020F-BSE-022</b>
<b>Shaheer Khan Qureshi</b>	<b>2020F-BSE-003</b>
<b>Muhammad Talal Moin</b>	<b>2020F-BSE-016</b>

#### **GROUP LEADER:**

**Muhammad Talal Moin**



**Task :** To create a login page using Html, CSS and JavaScript.

### CSS & JAVASCRIPT Code:

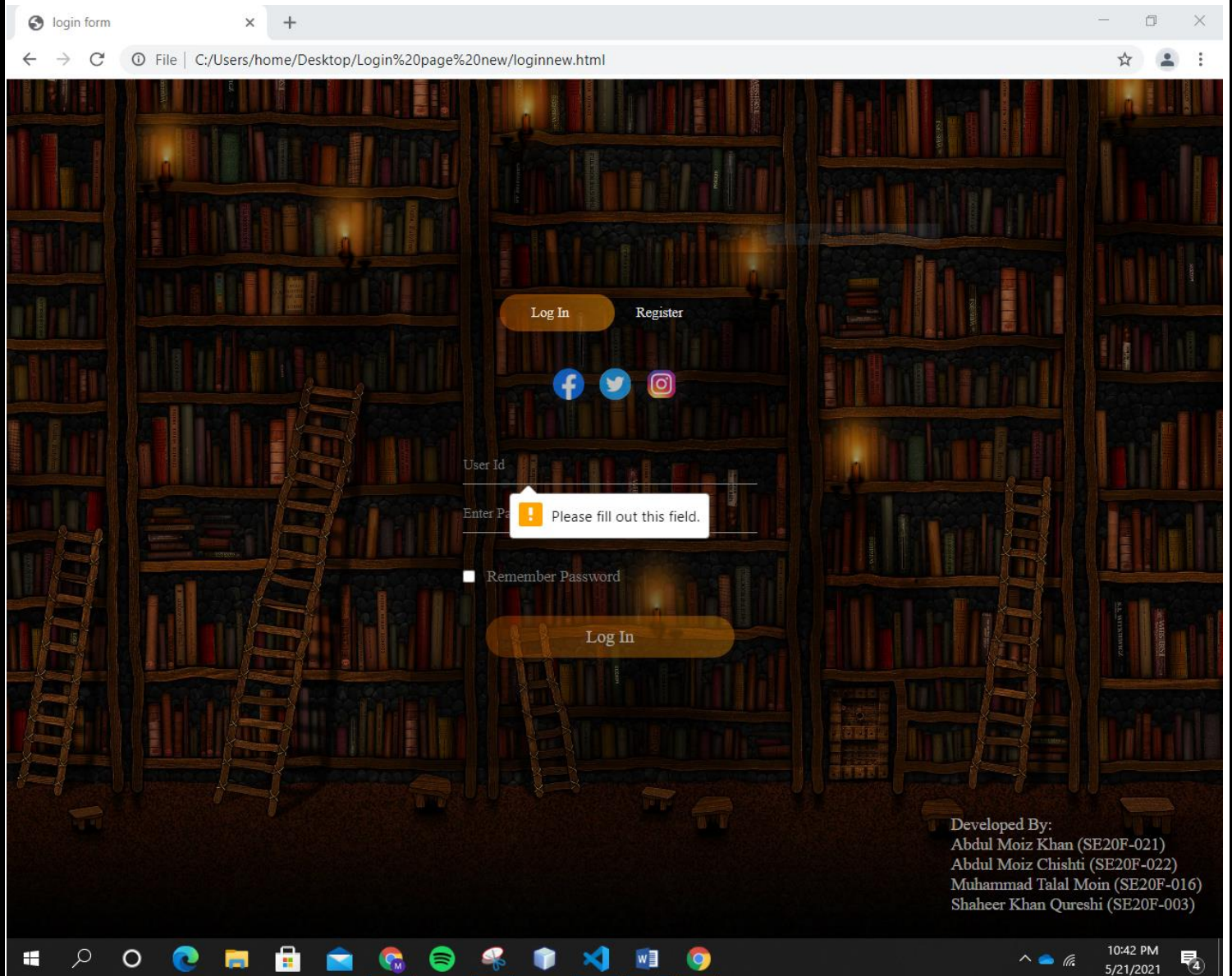
```
*{
  margin: 0;
  padding: 0;
  font-family: 'Times New Roman';
  color: rgba(255, 255, 255, 0.889);
}
body{
  height: 100%;
  width: 100%;
  background-image: linear-
gradient(rgba(0,0,0,0.4),rgba(0,0,0,0.4)),url(bg.jpg);
  background-position: center;
  background-size: cover;
  position: absolute;
}
.formbox{
  width: 380px;
  height: 480px;
  position: relative;
  margin: 6% auto;
  background: rgba(3, 3, 3, 0.257);
  padding: 5px;
  overflow: hidden;
}
.buttonbox{
  width: 220px;
  margin: 35px auto;
  position: relative;
  box-shadow: 0 0 20px 9px #0000002e;
  border-radius: 30px;
}
.toggle-bt{
  padding: 10px 30px;
  cursor: pointer;
  background: transparent;
  border: 0;
  outline: none;
  position: relative;
}
#bt{
  top: 0;
```

```
    left: 0;
    position: absolute;
    width: 110px;
    height: 100%;
    background: linear-gradient(to right, #ff740260, #d9810795);
    border-radius: 30px;
    transition: .5s;
}
.social-icons{
    margin: 30px auto;
    text-align: center;
}
.social-icons img{
    width: 30px ;
    margin: 0 5px;
    box-shadow: 0 0 20px 0 #7f7f7f3d;
    border-radius: 100%;
    opacity: 75%;
}
.social-icons img :hover{
    transform: translate(0, -10px);
}
.input-group{
    top: 180px;
    position: absolute;
    width: 280px;
    transition: .5s;
}
.input-field{
    width: 100%;
    padding: 10px 0;
    margin: 5px 0;
    border-left: 0;
    border-top: 0;
    border-right: 0;
    border-bottom: 1px solid #999;
    outline: none;
    background: transparent;
}
.submit-bt{
    width: 85%;
    padding: 10px 30px;
    cursor: pointer;
    display: block;
    margin: auto;
```

```
font-size: 17;
opacity: 70%;
background: linear-gradient(to right, #ff740260, #d9810795);
border: 0;
outline: none;
border-radius: 30px;
}
.check-box{
margin: 30px 10px 30.5px 0;
}
span{
color: #777;
font-size: 15px;
bottom: 68px;
position: absolute;
}
#login{
left: 50px;
}
#register{
left: 450px;
}
.namebox {

top: 700px;
left: 900px;
width: 250px;
position: relative;
background: rgba(0, 0, 0, 0.257);
font-size: 16;
opacity: 70%;
}
```

## Output:



## **Lab # 11**

<b>Name</b>	<b>Roll no</b>
Abdul Moiz Chishti	<b>2020F-BSE-022</b>
Abdul Moiz Khan	<b>2020F-BSE-021</b>
Shaheer Khan Qureshi	<b>2020F-BSE-003</b>
Muhammad Talal Moin	<b>2020F-BSE-016</b>

### **Web Development Framework Using “Dreamweaver CS 5”**

Web Development Dreamweaver and Html, Web Development (Open/Close Source Technologies), Architecture and Development a Form Based Website.

#### **Objective:**

1. To understand Web Development Technologies i.e. Open and Close Source Technologies.
2. To understand Client - Server Architecture
3. To understand the Software Architecture or Architecture of a Web Based Application.
4. To learn Web Development by making a web page for User Side Scripting using “HTML” for Login purpose.

#### **Exercise:**

#### **Task:**

Create a login form that logs into index page in html.

#### **Source Code:**



```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

```
<style> body {font-family: Arial, Helvetica,  
sans-serif;} form {border: 3px solid #f1f1f1;}  
input[type=text], input[type=password] {  
width: 100%; padding: 12px 20px; margin:  
8px 0; display: inline-block; border: 1px  
solid #ccc; box-sizing: border-box;  
}
```

```
button { background-color:  
#04AA6D;  
color: white;  
padding: 14px 20px;  
margin: 8px 0;  
border: none; cursor:  
pointer; width:  
100%;  
}
```

```
button:hover {  
opacity: 0.8;  
}
```

```
.cancelbtn {  
width: auto;
```

```
padding: 10px
```

```
18px;
```

```
background-color:
```

```
#f44336;
```

```
}
```

```
.imgcontainer {
```

```
text-align: center;
```

```
margin: 24px 0 12px 0;
```

```
}
```

```
img.avatar { width:
```

```
40%; border-radius:
```

```
50%;
```

```
}
```

```
.container {
```

```
padding: 16px;
```

```
}
```

```
span.psw {
```

```
float: right;
```

```
padding-top: 16px;
```

```
}
```

```
/* Change styles for span and cancel button on extra small screens */
```

```
@media screen and (max-width: 300px) {
```

```
span.psw {
```

```
        display: block;

float: none;

    }

    .cancelbtn {
width: 100%;

    }

}

</style>

</head>

<body>


<h2>Login Form</h2>


<form action="/action_page.php" method="post">

    <div class="imgcontainer">

    </div>


    <div class="container">

        <label for="uname"><b>Username</b></label>

        <input type="text" placeholder="Enter Username" name="uname" required>


        <label for="psw"><b>Password</b></label>

        <input type="password" placeholder="Enter Password" name="psw" required>    <button
type="submit">Login</button>

        <label>

            <input type="checkbox" checked="checked" name="remember"> Remember me

        </label>
```

</div>

<div class="container" style="background-color:#f1f1f1">

<button type="button" class="cancelbtn">Cancel</button>

<span class="psw">Forgot <a href="#">password?</a></span>

</div>

</form>


</body>

</html>

## Output:

Result Size: 1010 x 83

### Login Form



**Username**

**Password**

**Login**

☒ Remember me

**Cancel** [Forgot password?](#)

## **Lab # 12**

### **Server Programming using MY SQL**

### **Web Development Server Side using MYSQL**

#### **Objective:**

1. To understand the concept of Database and to get to know about MySQL.
2. To study the relational logical structure and to get an idea of Design and Data Entry Mode.
3. To design a Database in XAMPP (MySQL) and insert a table that holds fields and records of Login information.

#### **Exercise:**

#### **Source Code:**

```
<!DOCTYPE HTML>

<html>

<head>

  <title>Register Form</title>

</head>

<body>

  <form action="insert.php" method="POST">

    <table>

      <tr>

        <td>Name :</td>

        <td><input type="text" name="username" required></td>
```

```
</tr>

<tr>

<td>Password :</td>

<td><input type="password" name="password" required></td>

</tr>

<tr>

<td>Gender :</td>

<td>

<input type="radio" name="gender" value="m" required>Male

<input type="radio" name="gender" value="f" required>Female

</td>

</tr>

<tr>

<td>Email :</td>

<td><input type="email" name="email" required></td>

</tr>

<tr>

<td>Phone no :</td>

<td>

<select name="phoneCode" required>

<option selected hidden value="">Select Code</option>

<option value="977">977</option>

<option value="978">978</option>

<option value="979">979</option>

<option value="973">973</option>

<option value="972">972</option>

<option value="974">974</option>

</select>
```

```
<input type="phone" name="phone" required>

</td>

</tr>

<tr>

<td><input type="submit" value="Submit" name="submit"></td>

</tr>

</table>

</form>

</body>

</html>
```

### **Output:**

Name :	<input type="text"/>
Password :	<input type="password"/>
Gender :	<input type="radio"/> Male <input type="radio"/> Female
Email :	<input type="text"/>
Phone no :	<input type="text" value="977"/> <input type="text"/>
<input type="submit" value="Submit"/>	

**SIR SYED UNIVERSITY OF ENGINEERING & TECHNOLOGY****“GROUP MEMBERS”**

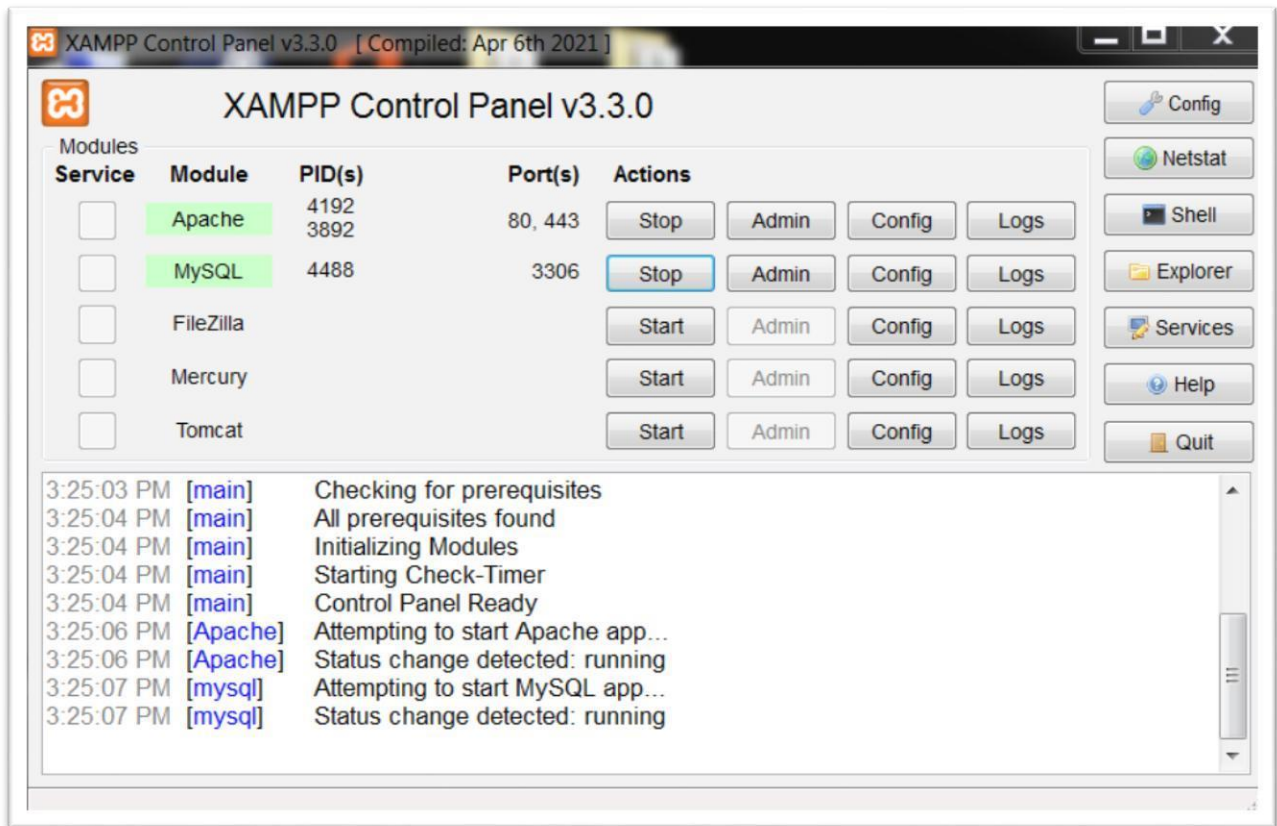
- SHAHEER KHAN QURESHI (BSE-2020F-003)
- MUHAMMAD TALAL MOIN (BSE-2020F-016)
  - ABDUL MOIZ KHAN (BSE-2020F-021)
  - ABDUL MOIZ CHISHTI (BSE-2020F-022)

**LAB 13****Web Development with PHP****Objective:**

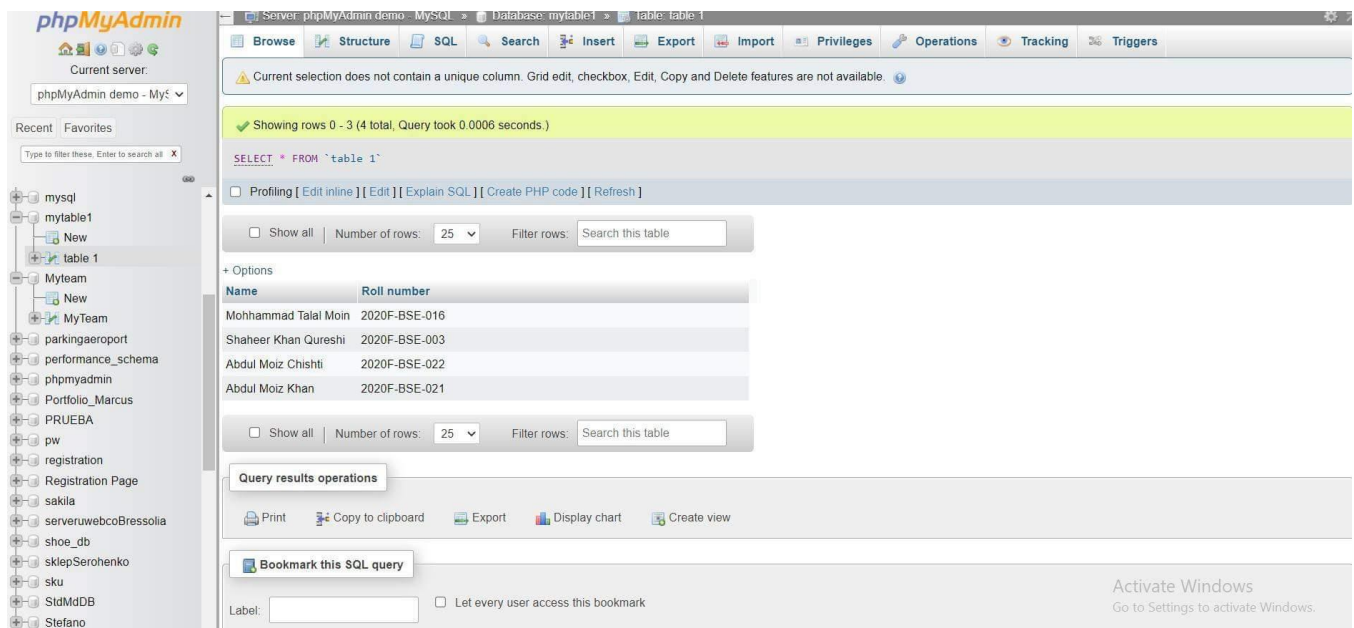
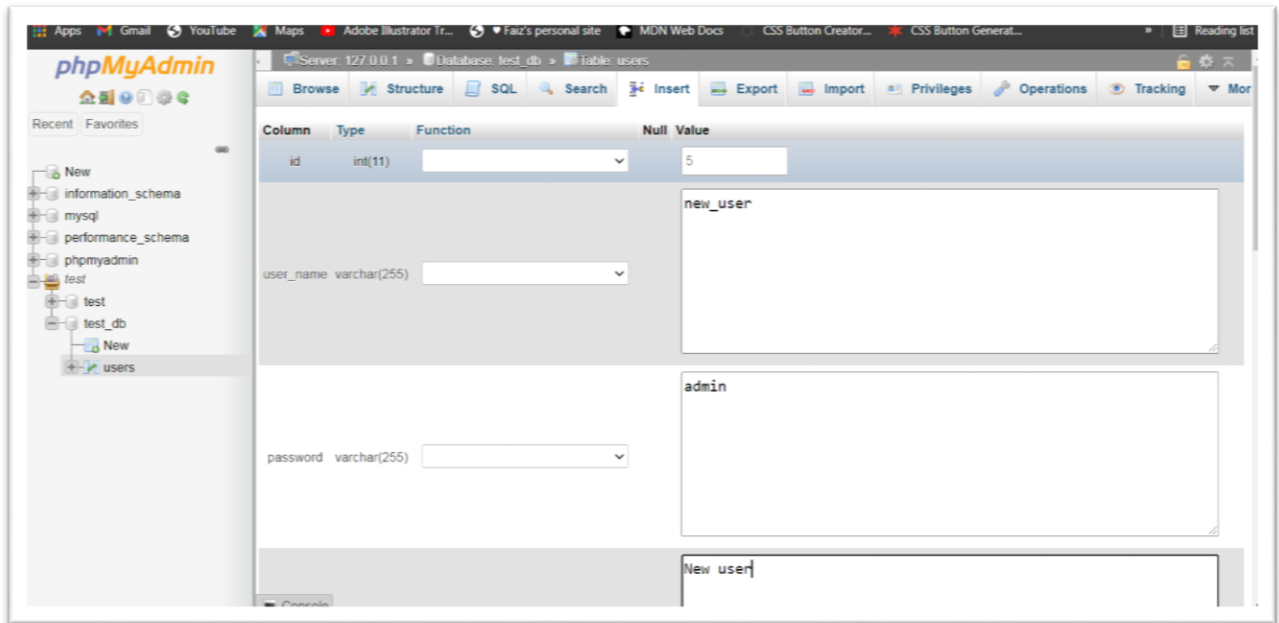
- 1) To understand the concept of inserting in and deleting records from database.
- 2) To insert record in MySQL database using PHP.
- 3) To delete record from MySQL database.

- 
- 1) Firstly we have to download xampp and open xampp control panel and start hostinh locally your website and MySQL as shown in figure.





- 2) Secondly Open phpMyAdmin Panel Where we can easily insert and delete our data. We also make copies, access through browse button, and edit our data through this panel.
- 3) In order to insert data in our Database we have to simply click on Insert button as shown in above snapshot. The dialogue box will appear on the screen of our phpMyAdmin control panel and simply insert new information of Username and Password so the user may access this site.

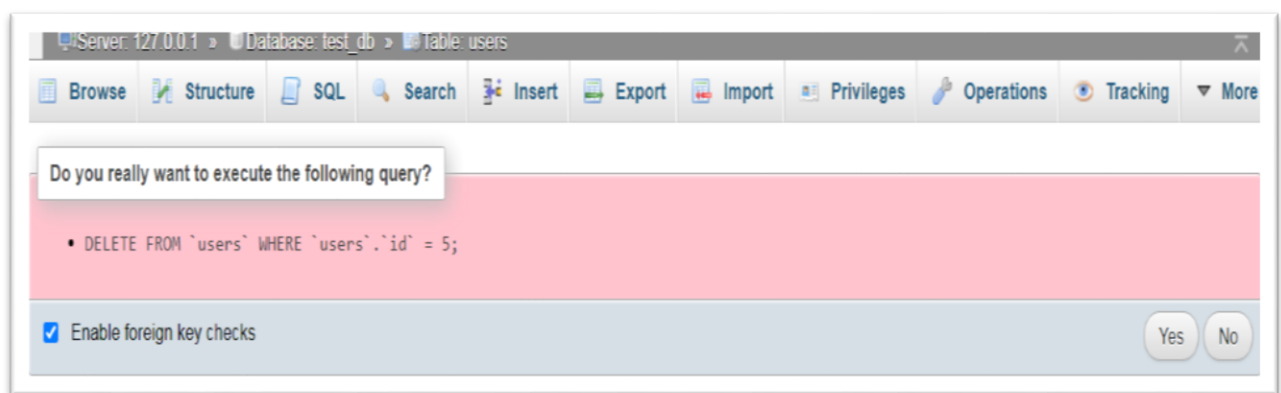


- 4) In order to delete our data from MySQL Database. We should simply select the data cell of Username and Password and click on delete button, which is very simplest way of deleting our data using phpMyAdmin control panel. Snapshots are given below.

- First we click on check button.



- Then click on delete button, then click on yes button.



- Window after deleting data row of Username and Password.

## An Overview of Software Estimation Tool

**INTRODUCTION:** We choose our object oriented programming project –Library Management System and use estimation techniques and process of software and calculate estimated cost and effort.

**PROJECT:** Library Management System.

**DURATION:** 28<sup>th</sup> May – 30<sup>th</sup> May, 48 Credit hours, Implement file parsing for project –Library Management. In which we have to make methods or functions of library system like adding, issuing, returning a book.

**ESTIMATION OF PROJECT –LIBRARY MANAGEMENT SYSTEM:**

There are three different ways of estimation of project given below.

- 1) According to Lines Of Code (LOC).
- 2) According to Function Points (FP).
- 3) According to Process Based Method.

**LAB 14**

#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
<input type="checkbox"/>	1	Moiz	int(25)		No	None			Change Drop More
<input type="checkbox"/>	2	Chishti	int(25)		No	None			Change Drop More
<input type="checkbox"/>	3	Shaheer	int(25)		No	None			Change Drop More
<input type="checkbox"/>	4	Talal	int(25)		No	None			Change Drop More

In this heading we work on estimation of project size, cost per LOC, Labour rate, Total estimated project cost and Estimated effort.

- **Size Of Project:** The size of project is estimated or we can calculated by line of our code as we are using LOC method for estimation of project, That how many lines are in our code or how many lines are in one method.  
**In our project the size of code is 194 lines.**
- **Cost Per LOC:** In order to find estimated or actual cost per line of code we just simply divide the labour rate by average productivity. In which labour rate was set by the labour or developers who are working on this project and the average productivity is set by manager who knows about the developers efforts and work done by them.

**Pre decided labour rate = 200**

**Average Productivity = 50**

**Cost per LOC=labour rate / Average Productivity.**

$$= 200/50$$

$$= 4 \text{ Rs per month}$$

- **Total Estimated Project Cost:** The total estimated project cost should be calculated by multiplying above cost per LOC and total LOC in your project.

**Total estimated project cost = Cost per LOC x Total LOC**

$$= 4 \times 194$$

$$= 776 \text{ Rs per month}$$

- **Estimated Effort:** The total estimated effort should be calculated by dividing total estimated project cost and labour rate.

**Estimated effort = Total estimated project cost / labour rate**

$$= 776 / 200$$

$$= 4 \text{ person per month}$$