Software Requirement Engineering (SE 305)

Project – Online Examination System (2023-2024)

Submitted By

Sanoj (2K21/SE/159)

Shivam Kumar (2K21/SE/167)

Submitted to

Ms. Vaibhavi Rajesh Mishra

Mr. Gaurav Sharma

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

Index – Blog Website

S.No	Experiment Name	Date	Teacher Sign	Remarks
1.	Problem Statement of Case Study – Online Examination System	17 Aug 2023		
2.	Use Case Diagram – Online Examination System	24 Aug 2023		
3.	Use Case Description – Online Examination System	31 Aug 2023		
4.	Entity Relationship Diagram – Online Examination System	14 Sep 2023		
5.	Data Flow Diagram – Online Examination System	21 Sep 2023		
6.	Different Types of COCOMO Model in C language	5 Oct 2023		
7.	Software Requirement Specification (SRS) – Online Examination System	12 Oct 2023		
8.	Some Software Requirement Engineering Tool	19 Oct 2023		
9.				
10.				

Experiment 1: Problem Statement – Online Examination System

An online examination system is a modern way of conducting tests or exams using the internet. It's like taking a test in a virtual classroom. Instead of sitting in a physical exam room, you log in to a computer or device to answer questions. Here's how it works: Questions are displayed on the screen, and you select your answers by clicking or typing. Once you've finished, the system automatically grades your exam, giving you instant results. This not only saves time but also reduces the chances of human errors in grading.

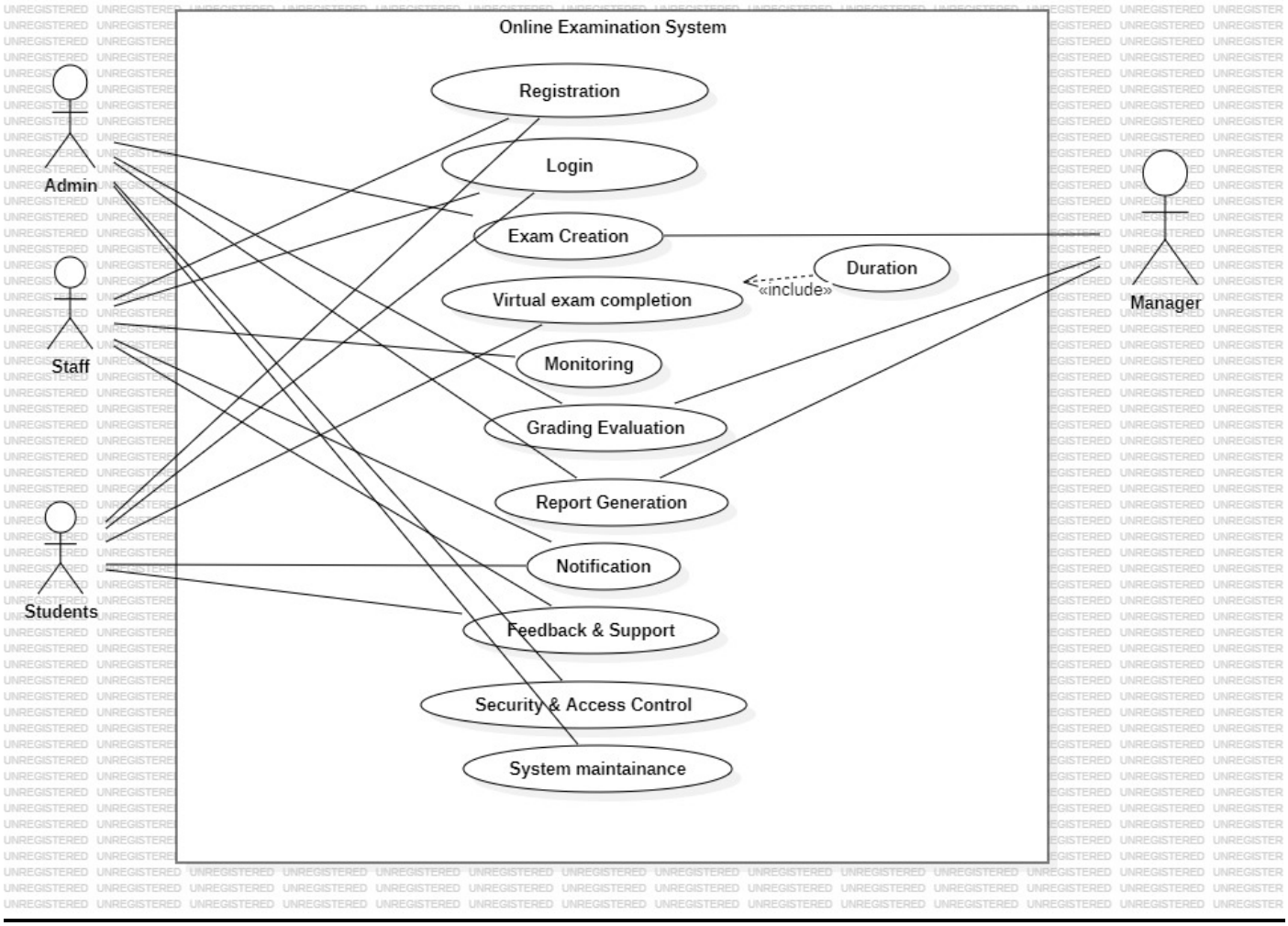
Online exams can be used for various purposes, from school and college tests to job interviews and certification exams. Some systems even have security features to prevent cheating, such as remote monitoring or limiting the time to answer each question.. The existing system has the following limitations:

- Outdated Graphical User Interface
- Limited security and privacy
- Limited features
- Outdated Remote Servers
- Limited Monitoring Features
- Limited Creativity in Assessment
- Having more Technical Issues
- Less Human Interaction
- Outdated Build and features.

So, to overcome the above limitations we are proposing a solution which has the following features:

- Less Technical Issues
- Enhanced Security and Privacy Features
- Up to date Remote Servers
- More Monitoring Features such as screen sharing etc.
- Interacting User Interface
- Latest technologies used for building the system.
- Creative assessment.

Experiment 2: Use Case Diagram – Online Examination System



Experiment 3: Use Case Description – Online Examination System

1	**Use case - Login**
2	**Actors** - Student, Manager, Administrator
3	**Preconditions** The user must have valid login credentials (username and password).
4	**Postconditions** Upon successful login, the user is redirected to their respective dashboard.
5	**Flows** **Basic Flow:** 1. The user accesses the login page. 2. The system presents a login form. 3. The user enters their username and password. 4. The system validates the credentials. 5. If the credentials are valid, the user is granted access to their respective roles. 6. If the credentials are invalid, an error message is displayed. **Alternate Flow**: If the user enters incorrect credentials three times consecutively, the system locks the user's account and prompts them to reset their password.
6	**Special Requirements**
7	**Associated Use Case**

1	**Use case – Exam Creation**
2	**Actors** - Manager, Administrator
3	**Preconditions** The actor must be logged into the system and have the necessary privileges.
4	**Postconditions** • The user is logged in to the blog website. • The user has the necessary permissions to add a blog.
5	**Basic Flow:** 1. The actor selects the option to create a new exam. 2. The system presents a form for creating the exam. 3. The actor enters exam details, such as title, duration, and instructions. 4. The actor defines the exam questions, including type (e.g., multiple-choice, essay) and content. 5. The actor specifies the grading criteria and sets a date for the exam. 6. The system saves the exam for future use. **Alternate Flow:** If the actor attempts to create an exam with a duplicate title, the system prompts them to choose a unique title.
6	**Special Requirements**
7	**Associated Use Case -**

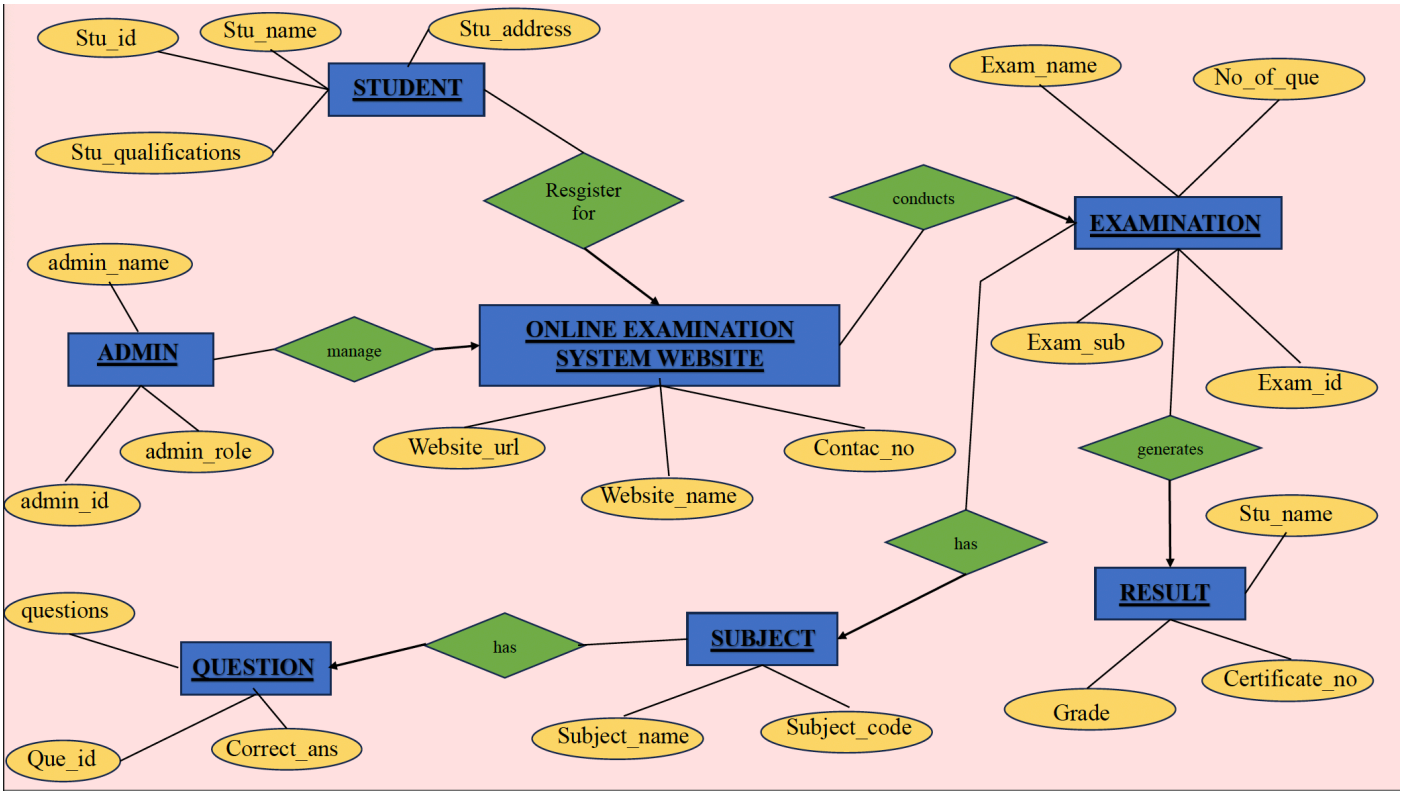
1	**Use case – Virtual Exam Complition**
2	**Actors** - Student
3	**Preconditions** The student must be logged into the system and have an assigned exam.
4	**Postconditions** • The selected blog is deleted from the website and is no longer visible to other users.
5	**Basic Flow:** 1. The student selects the exam they want to take. 2. The system presents the exam questions one by one. 3. The student answers each question within the allotted time. 4. The system records the student's responses. 5. After completing the exam, the student submits their answers. 6. The system stores the submitted answers and calculates the results. **Alternate Flow:** If the student loses internet connectivity during the exam, they can reconnect and continue from where they left off, and the system logs the time spent.
6	**Special Requirements**
7	**Associated Use Case -**

1	**Use case – Report Generation**
2	**Actors** -Administrator, Manager
3	**Preconditions** The actor must be logged into the system and have access to exam data.
4	**Postconditions** • The selected blog is deleted from the website and is no longer visible to other users.
5	**Basic Flow:** • The actor selects the exam for which they want to generate a report. • The system retrieves the exam results and statistics. • The actor can apply filters or select specific criteria for the report. • The system generates a report with information like average scores, individual student performance, and overall exam analysis. • The report is displayed or can be downloaded in a suitable format (e.g., PDF, CSV). **Alternate Flow:** If there are technical issues during report generation, the system logs the error and prompts the user to try again later.
6	**Special Requirements**
7	**Associated Use Case -**

1	**Use case – System Maintenance**
2	**Actors** - Administrator
3	**Preconditions** The administrator must be logged into the system with the necessary privileges.
4	**Postconditions** • The user's profile information is updated on the website with the new information provided by the user
5	**Basic Flow:** • The administrator can access a maintenance panel. • The system allows the administrator to perform various maintenance tasks, such as: • Adding or removing users. • Updating system configurations. • Backing up and restoring data.

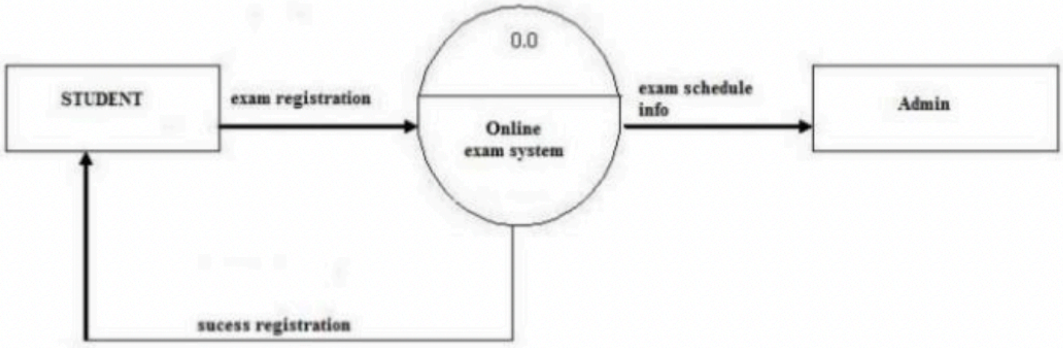
	• Monitoring system performance. • Handling user issues and support requests. • The system logs all maintenance activities for auditing purposes. **Alternate Flow:** If the system encounters an unexpected error during maintenance, it rolls back changes to ensure system integrity.
6	**Special Requirements**
7	**Associated Use Case**

Experiment 4: Entity Relationship Diagram– Online Examination System

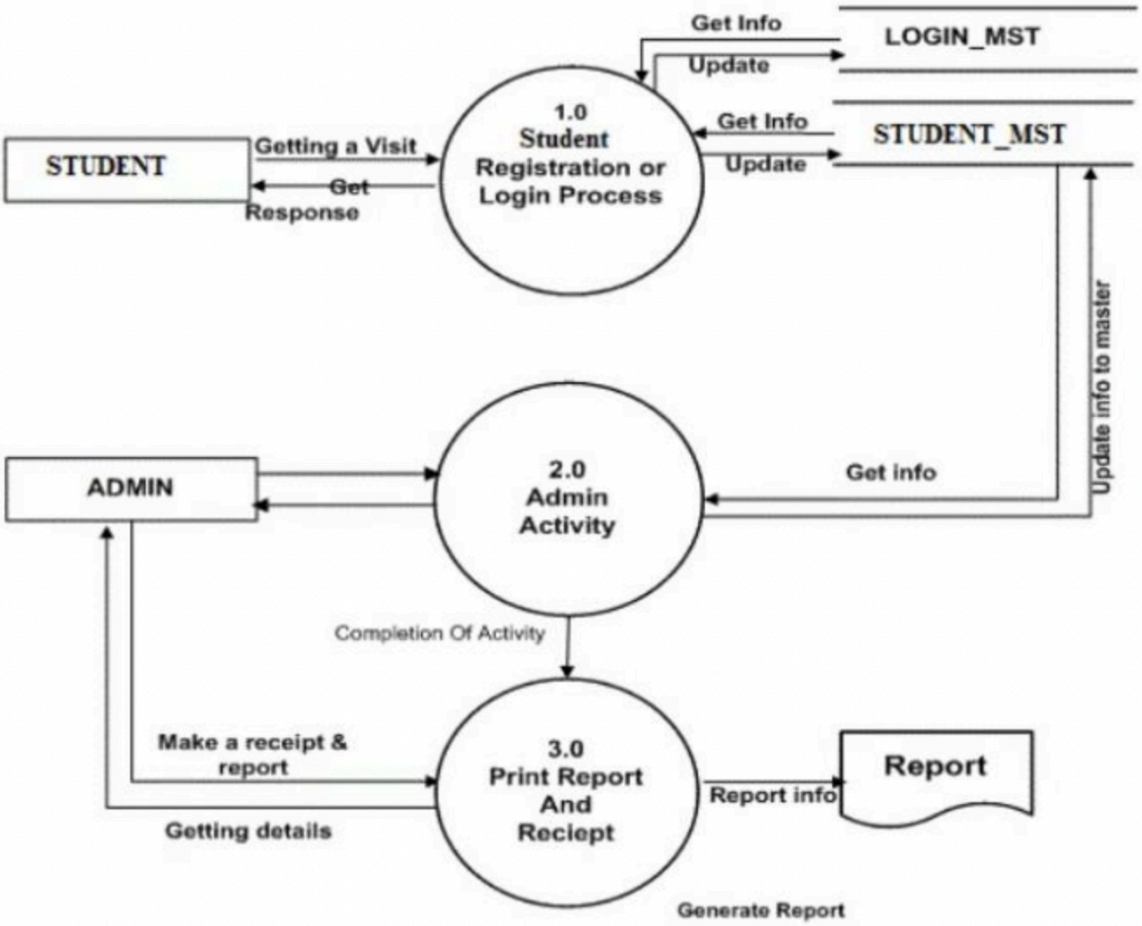


Experiment 5: Data Flow Diagram– Online Examination System

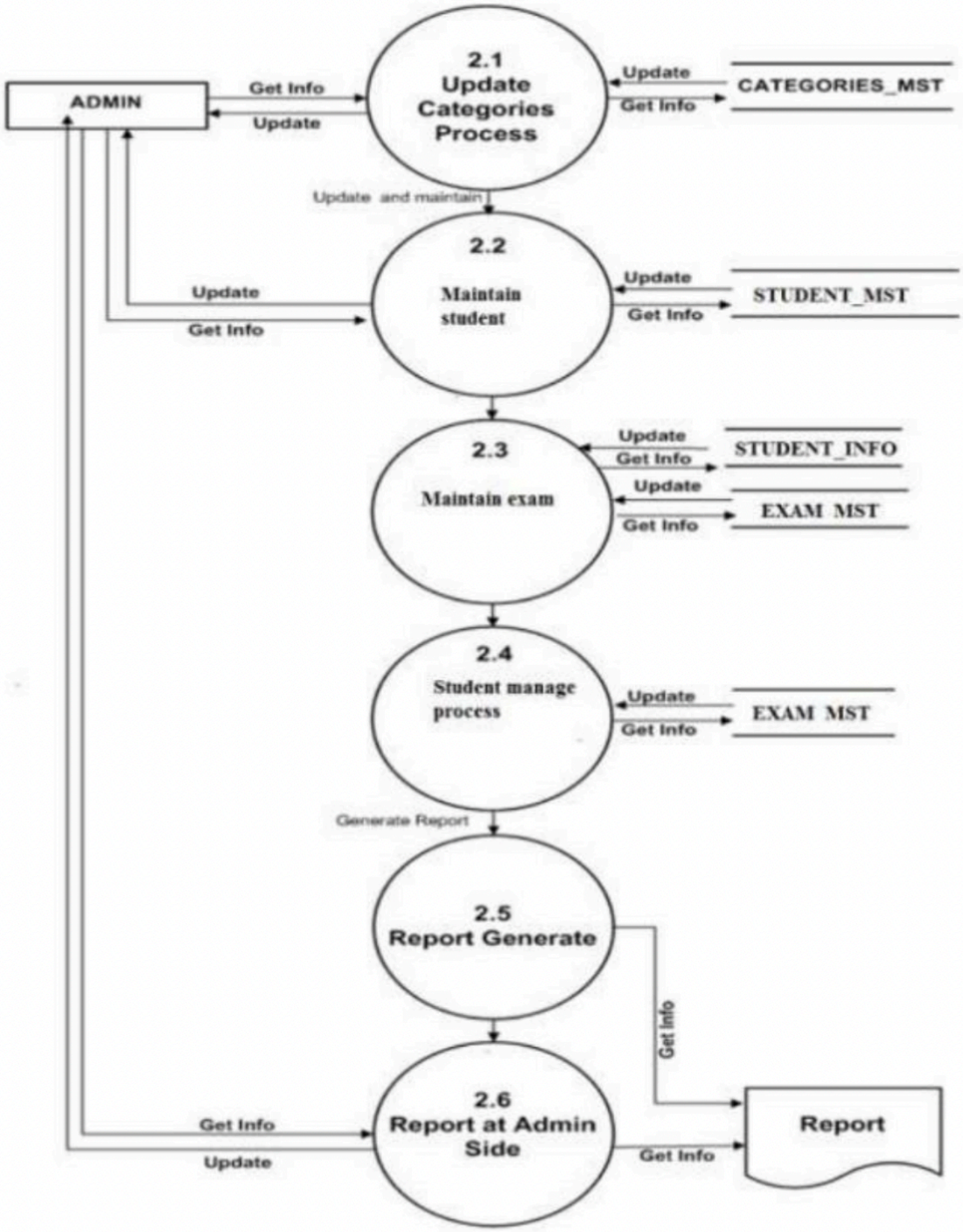
Figure:- Level 0



Level 0



Level 1


ADMIN
Get Info
Update
2.1
Update
Categories
Process
Update
Get Info
CATEGORIES_MST
Update and maintain
2.2
Maintain
student
Update
Get Info
Update
Get Info
STUDENT_MST
2.3
Maintain exam
Update
Get Info
STUDENT_INFO
Update
Get Info
EXAM MST
2.4
Student manage
process
Update
Get Info
EXAM MST
Generate Report
2.5
Report Generate
Get Info
2.6
Report at Admin
Side
Get Info
Update
Get Info
Report

Level 2

Experiment 6: Calculate Time and Effort using COCOMO Models

1. Basic COCOMO Model

```
#include <stdio.h>
#include <math.h>

int main() {
   double KLOC; // Size of the software in Kilo Lines of Code
   double EAF;  // Effort Adjustment Factor
   double PM;   // Effort in Person-Months
   double a, b; // Model-specific parameters

   // Prompt the user for input
   printf("Enter the size of the software in KLOC: ");
   scanf("%lf", &KLOC);

   printf("Enter the Effort Adjustment Factor (EAF): ");
   scanf("%lf", &EAF);

   // Basic COCOMO parameters
   a = 2.4; // Exponent for size
   b = 1.05; // Exponent for EAF

   // Calculate Effort (PM) using the Basic COCOMO formula
   PM = a * pow(KLOC, b) * EAF;

   // Display the result
   printf("Effort required: %.2f Person-Months\n", PM);

   return 0;
}
```

2. Intermediate COCOMO Model

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int fround(float x)
{
  int a;
  x=x+0.5;
  a=x;
  return(a);
}

void main()
{
  float table[3][4]={3.2,1.05,2.5,0.38,3.0,1.12,2.5,0.35,2.8,1.20,2.5,0.32};
  int i,j,size,model,rating;
  char mode[][15]={"Organic","Semi-Detached","Embedded"};
  char
driver[15][6]={"RELY","DATA","CPLX","TIME","STOR","VIRT","TURN","ACAP
","AEXP","PCAP",
  "VEXP","LEXP","MODP","TOOL","SCED"};

  float effort,EAF,a,time,staff,productivity;
  float costdrivers[15][6]={

                {0.75,0.88,1,1.15,1.40,-1},

                {-1,0.94,1,1.08,1.16,-1},

                {0.70,0.85,1,15,1.30,1.65},

                {-1,-1,1,1.11,1.30,1.66},

                {-1,-1,1,1.06,1.21,1.56},

                {-1,0.87,1,1.15,1.30,-1},

                {-1,0.87,1,1.07,1.15,-1},

                {1.46,1.19,1,0.86,0.71,-1},

                {1.29,1.13,1.00,0.91,0.82,-1},
```

```
            {1.42,1.17,1,0.86,0.70,-1},
            {1.21,1.10,1,0.90,-1,-1},
            {1.14,1.07,1,0.95,-1,-1},
            {1.24,1.10,1.00,0.91,0.82,-1},
            {1.24,1.10,1,0.91,0.83,-1},
            {1.23,1.08,1,1.04,1.10,-1}
            };
clrscr();
printf("\nEnter the size of project : ");
scanf("%d",&size);
if(size>=2 && size<=50)
  model=0;
else if(size>50 && size<=300)
  model=1;
else if(size>300)
  model=2;
printf("\nMode = %s",mode[model]);
EAF=1;
for(i=0;i<15;i++)
{
  do
  {
```

```
    printf("\nRate cost driver %s on scale of 0-5 : ",driver[i]);
    printf("\n0-Very Low\t1-Low\t2-Nominal\t3-High\t4-Very High\t5-Extra High\n");
    scanf("%d",&rating);
    a=costdrivers[i][rating];
    if(a==-1)
    {
      printf("\nNo value exist for this rating..Enter another rating...\n");
    }
    }while(a==-1);
    EAF=EAF*a;
  }
  printf("\nEAF = %f",EAF);
  effort=(table[model][0]*pow(size,table[model][1])) * EAF;
  time=table[model][2]*pow(effort,table[model][3]);
  staff=effort/time;
  productivity=size/effort;
  printf("\n\nEffort = %f Person-Month",effort);
  printf("\nDevelopment Time = %f Months",time);
  printf("\nAverage Staff Required = %d Persons",fround(staff));
  printf("\nProductivity = %f KLOC/Person-Month",productivity);
getch();
}
```

3. Detailed COCOMO Model

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int fround(float x)
{
 int a;
 x=x+0.5;
 a=x;
 return(a);
}

void main()
{
 float table[3][4]={3.2,1.05,2.5,0.38,3.0,1.12,2.5,0.35,2.8,1.20,2.5,0.32};
 int i,j,size,model,rating,S;
 char mode[][15]={"Organic","Semi-Detached","Embedded"};
 char
driver[15][6]={"RELY","DATA","CPLX","TIME","STOR","VIRT","TURN","ACAP
","AEXP","PCAP",
 "VEXP","LEXP","MODP","TOOL","SCED"};
 float effort,EAF,a,time,staff,productivity;
 float costdrivers[15][6]={
            {0.75,0.88,1,1.15,1.40,-1},
            {-1,0.94,1,1.08,1.16,-1},
            {0.70,0.85,1,1.15,1.30,1.65},

            {-1,-1,1,1.11,1.30,1.66},
            {-1,-1,1,1.06,1.21,1.56},
            {-1,0.87,1,1.15,1.30,-1},
            {-1,0.87,1,1.07,1.15,-1},

            {1.46,1.19,1,0.86,0.71,-1},
            {1.29,1.13,1.00,0.91,0.82,-1},
            {1.42,1.17,1,0.86,0.70,-1},
            {1.21,1.10,1,0.90,-1,-1},
            {1.14,1.07,1,0.95,-1,-1},

            {1.24,1.10,1.00,0.91,0.82,-1},
            {1.24,1.10,1,0.91,0.83,-1},
            {1.23,1.08,1,1.04,1.10,-1}
            };
```

```
  float mp[6][5]={
          {0.06,0.16,0.26,0.42,0.16},
          {0.06,0.16,0.24,0.38,0.22},
          {0.07,0.17,0.25,0.33,0.25},
          {0.07,0.17,0.24,0.31,0.28},
          {0.08,0.18,0.25,0.26,0.31},
          {0.08,0.18,0.24,0.24,0.34}
        };
  float tp[6][5]={
          {0.10,0.19,0.24,0.39,0.18},
          {0.12,0.19,0.21,0.34,0.26},
          {0.20,0.26,0.21,0.27,0.26},
          {0.22,0.27,0.19,0.25,0.29},
          {0.36,0.36,0.18,0.18,0.28},
          {0.40,0.38,0.16,0.16,0.30}
        };

  char phases[5][30]={"Planning and Requirements","System Design","Detail
Design","Module Code and Test","Integration and Test"};
  clrscr();
  printf("\nEnter the size of project : ");
  scanf("%d",&size);
  if(size>=2 && size<=50)
    model=0;
  else if(size>50 && size<=300)
    model=1;
  else if(size>300)
    model=2;
  printf("\nMode = %s",mode[model]);
  EAF=1;
  for(i=0;i<15;i++)
  {
    do
    {
    printf("\nRate cost driver %s on scale of 0-5 : ",driver[i]);
    printf("\n0-Very Low\t1-Low\t2-Nominal\t3-High\t4-Very High\t5-Extra High\n");
    scanf("%d",&rating);
    a=costdrivers[i][rating];
    if(a==-1)
    {
      printf("\nNo value exist for this rating..Enter another rating...\n");
    }
    }while(a==-1);
```

```
  EAF=EAF*a;
 }
 printf("\nEAF = %f",EAF);

 effort=(table[model][0]*pow(size,table[model][1])) * EAF;
 time=table[model][2]*pow(effort,table[model][3]);
 staff=effort/time;
 productivity=size/effort;

 printf("\n\nEffort = %f Person-Month",effort);
 printf("\nDevelopment Time = %f Months",time);
 printf("\nAverage Staff Required = %d Persons",fround(staff));
 printf("\nProductivity = %f KLOC/Person-Month",productivity);
 printf("\n\n");

 if(model==0)
 {
  printf("\nEnter Ogranic - Small(0) or Medium(1) : ");
  scanf("%d",&S);
 }
 else if(model==1)
 {
  printf("\nEnter Semi-Detached - Medium(2) or Large(3) : ");
  scanf("%d",&S);
 }
 else if(model==2)
 {
  printf("\nEnter Embedded - Large(4) or Extra Large(5) : ");
  scanf("%d",&S);
 }
 printf("\n\nPhase-wise Distribution of Effort is :\n\n");
 for(i=0;i<5;i++)
 {
   printf("\n%s Phase  =  ",phases[i]);
   printf("%f",effort*mp[S][i]);
 }
 printf("\n\nPhase-wise Distribution of Development Time is :\n\n");
 for(i=0;i<5;i++)
 {
   printf("\n%s Phase  =  ",phases[i]);
   printf("%f",time*tp[S][i]);
 }
getch();
}
```

Experiment 7: Software Requirements Specifications (SRS)

1. Introduction:

1.1 Purpose

The purpose of this Software Requirements Specifications (SRS) document is to provide an overall description of the functional and non-functional requirements of the Online Examination System. This document serves as a comprehensive guide for web developers, offering insights into the system's operations, appearance, and available features.

1.2 Scope

The Online Examination System is designed to facilitate the efficient conduct of online exams, enabling users to create, manage, and participate in digital assessments. The system incorporates various features, including user registration, question management, exam administration, and result processing. It encompasses the following functionalities:

a) User registration and authentication.
b) Management of exam questions and answers.
c) Generation and storage of exam papers.
d) Exam scheduling and administration.
e) Automatic scoring and result generation.
f) Reporting and analytics.
g) Integration with social media platforms.
h) Accessibility for both administrators and participants.
i) Comprehensive security measures for data protection.

1.3 Definition, Abbreviation and Acronyms

SRS – Software Requirement Specifications
HTML – Hypertext Markup Language.
JS – JavaScript.
CSS – Cascading Style Sheet.
Website – Online Examination System.
DBMS – Database Management System.
SEO – Search Engine Optimizations.

1.4 References

Books:

i) Object-Oriented Software Engineering by Ruchika Malhotra

ii) IEEE Std 830 - 1984
iii) Database System Concepts (6th ed.) by Avi Silberschatz, Henry F. Korth
iv) Use Case Modeling by Kurt Bittner
v) Java in a Nutshell by David Flanagan
vi) O'Reilly's Software Development
vii) IBM Red Books

1.5 Overview

The Online Examination System is a robust platform designed to streamline the process of creating and conducting online exams. It offers a range of features for user registration, question management, exam administration, and result processing. This SRS document comprises use cases, use case descriptions, Entity Relationship diagrams, UML Diagrams, Functional Requirements, and non-functional requirements that are essential for the successful development and implementation of the system. This system aims to provide a user-friendly and secure environment for online assessments, benefiting both administrators and participants.

2. Overall Description:

The Online Examination System manages the storage of examination-related data on backend servers and databases. It also maintains information about users, administrators, and the examinations themselves. The system administrator is responsible for authorizing users and managing authors and editors. This system enables users to create and customize examinations, with various options for customization.
Authors/users will be able to write the blogs, publish the blogs. Viewers will be able only to view the blogs and comment on others blogs.
Authors and users can create and publish exams, while viewers can only access and comment on existing exams. The administrator is tasked with maintaining the following data:

- Examination details
- User details
- Administrator details

The administrator's functions include:

- Authorizing exams
- Validating users
- Generating reports:

a) List of published exams by authors
b) Details of published exams
c) Total number of exams in the system

d) List of authors
e) List of all users

The administrator requires information about available exams sorted by:

a) Author
b) Exam Title
c) Publisher
d) Subject

2.1 Product Perspective

The Online Examination System will be developed using a client/server architecture and will be compatible with the Microsoft Windows Operating System. The front end of the system will be created using Visual Basic 6.0, and the back end will be powered by MS SQL Server 2005.

2.1.1 System Interfaces

Users Interface

Our website will feature user-friendly, menu-driven interfaces for the following actions:

1. Login - To grant entry only to authorized users with valid login IDs and passwords.
2. Exam Details - To manage examination details.
3. User Membership Details - To keep track of users with active subscriptions.
4. Author Membership Details - To maintain author membership information.
5. Publish an Exam - Allowing users to publish examinations.
6. Delete an Exam - Allowing users and administrators to remove examinations.

software should generate the following information:

1. Details of exams published and deleted daily.
2. Examination data in the system.
3. Examination details organized by author.
4. List of authorized users.

Hardware Interface

1. Screen resolution of at least 640 × 480 or above.
2. Support for printer (dot matrix, desk jet, LaserJet).
3. Computer systems will be in the networked environment as it is a multi-user system

Software Interface

1. MS-Windows Operating System (NT/XP/Vista)
2. Microsoft Visual Basic 6.0 for designing front-end
3. MS SQL Server 2005 for back-end

2.2 Product Functions

The Online Examination System will grant access only to authorized users with specific roles (System administrator, Viewers, Authors, etc.). Depending on their role, users will have access to specific modules of the system. Key functions of the system include:

• A login facility to allow authorized access to the system.
• The system administrator can add, modify, delete, or view exams, users, authors, and login information.
• The system administrator can delete or blacklist an exam.
• The system administrator can search for exams in the exams catalog.
• The system administrator can generate various reports from the website.

2.3 User Characteristics

User characteristics to consider when designing the website include:

• Age: The website should be user-friendly for individuals of all age groups.
• Education Level: The website should be designed to cater to users with varying levels of technical expertise.
• Technical Expertise: The website should accommodate both technically proficient and less technically proficient users.
• Language: The website should support multiple languages to cater to a global user base.

2.4 Constraints

The development of the Online Examination System is subject to several constraints:

• Time Constraint: The website must be developed within a specific timeframe, such as six months.
• Budget Constraint: The development must adhere to a predetermined budget, e.g., $50,000.
• Technology Constraint: The website must be developed using specified technologies, such as open-source software to minimize costs.
• Resource Constraint: The development team may be limited in terms of available resources, such as the number of developers.

2.5 Assumptions and Dependencies

Assumptions
• Users will possess basic knowledge of using an online examination system.
• The website will be developed using PHP and MySQL.
• The website will be hosted on a cloud-based server.
Dependencies

• The website's functionality relies on the availability of internet connectivity.
• Integration with social media platforms depends on the availability and functionality of these platforms.

3. Specific requirements

3.1 External Interfaces

3.1.1 USER INTERFACES

- Front-end software: Vb.net version
- Back-end software: SQL

3.1.2 HARDWARE INTERFACES

- Windows.
- Browser which supports CGI, HTML & Javascript.

3.1.3 SOFTWARE INTERFACES

Operating system: The system will be compatible with Windows operating systems for user-friendliness.
Database: The system will utilize SQL+ database for storing examination records and user data.

3.1.4 COMMUNICATION INTERFACES

The system will be accessible through various web browsers, and electronic forms will be employed for functions like exam reservation and ticket booking.

3.2 System Features

3.2.1 Examination Creation Feature

• The system should enable authorized users to create examinations, including various question types and formatting options.
• Users should be able to categorize and tag examinations for easy organization and retrieval.
• Examinations should support different media types, such as text, images, and multimedia.
• The system should allow for question and answer randomization to prevent cheating.
• Real-time monitoring of exams to detect and prevent irregularities.

3.2.2 User Management Feature

• Scheduling and administration of examinations with date, time, and duration settings.
• Automatic scoring and result generation.
• Support for multiple-choice, true/false, and essay questions.
• Exam reports and analytics for administrators.

• Data backup and recovery.

3.3 Performance Requirement:

- **Response Time**: The system must respond within 2 seconds for all user actions, irrespective of device or network speed.

- **Throughput**: The system should accommodate at least 100 concurrent users without significant slowdowns.

- **Scalability**: The system must be designed to scale to handle increased user traffic and exam data.

- **Availability**: The system should be available 99.99% of the time, excluding planned maintenance.

- **Security**: The system must be protected against common vulnerabilities, such as SQL injection, XSS, and CSRF.

3.4. Design Constraint

Responsive Design:
• The system must have a responsive design that ensures usability on various devices, including desktops, laptops, tablets, and smartphones.
• Optimization for different screen sizes for enhanced usability.
Cross-Browser Compatibility:

• The system must be compatible with major web browsers like Chrome, Firefox, Safari, and Edge.
• Extensive browser compatibility testing.

Accessibility:
• The system must adhere to accessibility guidelines, ensuring access for users with disabilities.
• Compliance with WCAG 2.1 accessibility standards.
Performance:
• The system must provide fast load times and responsive interactions for a smooth user experience.
• Optimization for quick page load times and minimized server requests.
Security:
• The system must employ security best practices, such as encryption and robust authentication mechanisms.

• Protection against unauthorized access and hacking attempts.

3.5 Software System Attributes

Usability:

- The website must be easy to use and navigate for users of all skill levels.
- The website design should be intuitive, with clear navigation menus and user-friendly interfaces.

Reliability:

- The website must be reliable and available to users at all times.
- The website design should include features that ensure uptime and availability, such as redundant servers and automated failover systems.

Availability:

- The website must be available to users at all times, with minimal downtime and interruptions.
- The website design should include features that ensure uptime and availability, such as automated monitoring and alerting systems.

Scalability:

• The system must be scalable and capable of handling increased user activity over time.

• Development with scalability in mind, including efficient code and architecture.

Security:

- The website must be secure and protect user data from unauthorized access and hacking attempts.
- The website design should follow security best practices, such as using SSL encryption and implementing user authentication and authorization.

Maintainability:

- The website must be maintainable and easy to update and modify as needed.
- The website design should include features that make it easy to maintain and update, such as well-documented code and modular architecture.

Performance:

- The website must have fast load times and quick response times to ensure a smooth user experience.
- The website design should be optimized for speed, with a focus on reducing page load times and minimizing server requests.

Experiment 8: Software Requirement Engineering Tool

Software Requirement Engineering (SRE) is a crucial phase in software development where the requirements for a software system are gathered, documented, and analysed. Various tools and techniques can be used to facilitate this process. Here, I'll describe a few important tools for software requirement engineering in detail:

1. **Microsoft Word and Excel**:

    - **Description**: These are widely used general-purpose tools for documenting and managing requirements. Microsoft Word can be used for writing textual descriptions of requirements, while Microsoft Excel can be employed for creating tables, matrices, and lists.

    - **Features**:

        - **Rich Text Formatting**: Word allows you to format text for clear and organized documentation.

        - **Tables and Spreadsheets**: Excel is useful for creating traceability matrices, tables, and lists of requirements.

        - **Collaboration**: These tools can be used in conjunction with cloud-based collaboration platforms for real-time collaboration.

2. **IBM Engineering Requirements Management DOORS**:

    - **Description**: DOORS is a specialized, industry-standard tool for managing and tracing requirements. It's widely used in sectors like aerospace, automotive, and healthcare.

    - **Features**:

        - **Requirements Traceability**: DOORS allows you to establish and maintain traceability links between requirements.

        - **Version Control**: It offers robust version control features to track changes in requirements.

        - **Customization**: You can tailor DOORS to suit the specific needs of your project or industry.

        - **Collaboration**: Multiple stakeholders can work together on a single repository.

3. **JIRA**:

   - **Description**: JIRA is an agile project management and issue tracking tool. While primarily designed for project management, it can be adapted for requirement management.

   - **Features**:

     - **Custom Workflows**: JIRA allows you to create custom workflows to manage requirement documents through various stages.

     - **Integration**: It can be integrated with other tools, like Confluence for better documentation.

     - **Reporting**: JIRA provides various reporting and visualization options for requirements.

4. **Confluence**:

   - **Description**: Confluence is a collaboration tool by Atlassian. While not a dedicated requirement management tool, it can be used for documenting and collaborating on requirements.

   - **Features**:

     - **Wiki-Style Documentation**: Confluence offers a flexible, wiki-style environment for documenting requirements.

     - **Integration**: It integrates seamlessly with JIRA and other Atlassian tools for end-to-end project management.

     - **Permissions and Access Control**: You can control who can access and edit documents, ensuring data security.

5. **Lucidchart**:

   - **Description**: Lucidchart is a web-based diagramming and visual communication tool. It's useful for creating flowcharts, data flow diagrams, and other visual representations of requirements.

   - **Features**:

     - **Diagram Creation**: Lucidchart provides a drag-and-drop interface for creating visual representations of requirements.

     - **Collaboration**: Multiple users can collaborate in real-time on the same diagrams.

- **Integrations**: It can integrate with other project management and requirement management tools for a comprehensive solution.

6. **Tuleap**:

    - **Description**: Tuleap is an open-source software development and project management platform. It includes tools for requirement management, issue tracking, and agile project management.

    - **Features**:

        - **Agile Planning**: Tuleap supports agile methodologies, making it suitable for projects with evolving requirements.

        - **Integration**: It can be integrated with various version control systems and other tools.

        - **Customization**: Tuleap allows for customization to fit specific project needs.

Choosing the right tool depends on the specific needs of your project, your team's preferences, and industry standards. It's important to evaluate these tools to determine which one aligns best with your organization's requirements engineering process.