E - LEARNING PORTAL

Software Engineering Lab {KCS - 651}

BACHELOR OF TECHNOLOGY

{INFORMATION TECHNOLOGY}

SESSION: (2021-2022)

VEER BAHADUR SINGH PURVANCHAL UNIVERSITY
JAUNPUR U.P.

SUBMITTED BY:

Mohd Amaan Hashmi (195453)
Aman Singh (195454)
Saurabh Singh (195434)

SUBMITTED TO:

Mrs. Deepti Pandey (Assistant Professor)
Mr. Dileep Kr. Yadav (Assistant professor)

DEPARTMENT OF INFORMATION TECHNOLOGY
UNSIET VBSPU JAUNPUR

(AUGUST – 2022)

CANDIDATE'S DECLARATION

WE, **Mohd Amaan Hashmi (195453)**, & **Aman Singh (195454)** , and **Saurabh Singh (195434)**, a student of B.Tech. (Information Technology) at **Uma Nath Singh Institute Of Engineering and Technology**, **VBS Purvanchal University**, Jaunpur ,declare that the work presented in this lab file.

Title "**E - Learning Portal**", Submitted to **Department of information Technology** for the award of Bachelor of Technology degree in Information Technology. All the work done in this project is entirely my own expect for the reference quoted. To the best of my knowledge, this work has not been submitted to any other university or institutions for award of any degree.

Date:____/__/_____

Place: UNSITE , VBSPU Jaunpur

Student Name:

Mohd Amaan Hashmi (195453)

Aman Singh (195454)

Saurabh Singh (195434)

Class & Branch:

Bachelor Of Technology

(Information Technology)

{ 3rd Year 6th Semester }

CERTIFICATE

It is certified that, this project entitled **"E - LEARNING PORTAL"** , submitted by (Mohd Amaan Hashmi, Aman Singh and Saurabh Singh) in partial fulfillmentof the requirement for the award of Bachelor Of Technology in Information Technology degree for VBS Purvanchal University, jaunpur is record of students own study carried under my/our supervision. This lab file report has not been submitted to any other university of institution for the award of any degree.

Lab File Guide:

Mrs. Deepti Pandey (Assistance professor)

Mr. Dileep Kr. Yadav (Assistance Professor)

External Examiner

_____

Date:____/___/_____

Place: UNSITE , VBSPU Jaunpur U.P

Information Technology & Engineering Department

B.TECH VI SEMESTER

SESSION (2021-22)
Software Engineering Lab (KCS-651)

S. No	LIST OF THE EXPERIMENTS	DATE	PAGE	REMARK
1	Prepare an SRS document in line with the IEEErecommended standards.			
2	Draw the Entity relationship diagram of a project.			
3	Draw the data flow diagrams at level 0 and level 1.			
4	Draw use case diagram in argo UML.			
5	Draw activity diagram in argo UML.			
6	Draw class diagram in argo UML.			
7	Draw the component diagram in argo UML.			
8	Draw sequence diagram in argo UML.			
9	Draw collaboration diagram in argo UML.			
10	Draw the COCOMO Model.			
11	Draw the Software Testing.			

Experiment No-1

Experiment Name: Prepare an SRS document in line with the IEEE recommended standards.

Outcome: Can produce the requirements in an SRS document.

Objective: To prepare an SRS document in line with the IEEE recommended standards.

Description:

An SRS is basically an organization's understanding (in writing) of a customer or potential client's system requirements and dependencies at a particular point in time (usually) prior toany actual design or development work. It's a two-way insurance policy that assures that both the client andthe organization understand the other's requirements from that perspective at a given point in time.

The SRS document itself states in precise and explicit language those functions and capabilities a software system (i.e., a software application, an eCommerce Web site, and so on) must provide, as well as states any required constraints by which the system must abide. The SRS also functions as a blueprint for completing a project with as little cost growth as possible. The SRS is often referred to as the "parent" document because all subsequent project management documents, such as design specifications, statements of work, software architecture specifications, testing and validation plans, and documentation plans, are related to it.

It's important to note that an SRS contains functional and non-functional requirements only; it doesn't offer design suggestions, possible solutions to technology or business issues, or any other information other than what the development team understands the customer's system requirements to be.

A well-designed, well-written SRS accomplishes four major goals:

- It provides feedback to the customer. An SRS is the customer's assurance that the development organization understands the issues or problems to be solved and the software behavior necessary to address those problems. Therefore, the SRS should be written in natural language (versus a formal language, explained later in this article), in an unambiguous manner that may also include charts, tables, data flow diagrams, decisiontables, and so on.
- It decomposes the problem into component parts. The simple act of writing down software requirements in a well-designed format organizes information, places borders around theproblem, solidifies ideas, and helps break down the problem into its component parts in an orderly fashion.
- It serves as an input to the design specification. As mentioned previously, the SRS serves as the parent document to subsequent documents, such as the software design specification and statement of work. Therefore, the SRS must contain sufficient detail in the functional system requirements so that a design solution can be devised.
- It serves as a product validation check. The SRS also serves as the parent document for testing and validation strategies that will be applied to the requirements for verification.

SRSs are typically developed during the first stages of "Requirements Development," which is the initial product development phase in which information is gathered about what requirements are needed--and not. This information-gathering stage can include onsite visits,

questionnaires, surveys, interviews, and perhaps a return-on-investment (ROI) analysis or needs analysis of the customer or client's current business environment. The actual specification, then,is written after the requirements have been gathered and analysed.

SRS should address the following

The basic issues that the SRS shall address are the following:

● Functionality. What is the software supposed to do?

● External interfaces. How does the software interact with people, the system's hardware, otherhardware, and other software?

● Performance. What is the speed, availability, response time, recovery time of various softwarefunctions, etc.?

● Attributes. What is the portability, correctness, maintainability, security, etc. considerations?

● Design constraints imposed on an implementation. Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.

● **Correct** - This is like motherhood and apple pie. Of course, you want the specification to be correct. No one writes a specification that they know is incorrect. We like to say - "Correct and Ever Correcting." The discipline is keeping the specification up to date when you find things thatare not correct.

● **Unambiguous** - An SRS is unambiguous if, and only if, every requirement stated therein has only one interpretation. Again, easier said than done. Spending time on this area prior to releasingthe SRS can be a waste of time. But as you find ambiguities - fix them.

● **Complete** - A simple judge of this is that is should be all that is needed by the software designers to create the software.

● **Consistent** - The SRS should be consistent within itself and consistent to its reference documents. If you call an input "Start and Stop" in one place, don't call it "Start/Stop" in another.

● **Ranked for Importance** - Very often a new system has requirements that are really marketingwish lists. Some may not be achievable. It is useful provide this information in the SRS.

Table of Contents

CHAPTER NO. 5

Other Non-Functional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

5.3 Security Requirements

5.4 Software Quality Attributes

5.4.1 Reliability

5.4.2 Availability

5.4.3 Security

5.4.4 Maintainability

CHAPTER NO. 6

Other Requirements

6.1 None

CHAPTER NO. 7

Conclusion

1. Introduction

The website E-Learning Portal version1.0 is to be developed. The purpose of E-Learning Portal is to automate the existing manual system with the help of computerized equipment and full- fledged computer software, fulfilling their requirements, so that their valuable data/information can be stored for a longer period with easy accessing and manipulation of the same.

1.1 Purpose

This SRS defines External Interface, Performance and Software System Attributes requirements of E-Learning Portal. This document is intended for the following group of people: -

- Developers for the purpose of maintenance and new releases of the software.
- Administrators of the website.
- Documentation writers.
- Testers.

1.2 Scope

This document applies to E-Learning Portal. The primary goal of "E-Learning Portal" is to override the problems prevailing in the practicing manual system. This Portal is supported to eliminate and, in some cases, reduce the hardships faced by this existing system. Moreover, this system is designed for the particular need of the administrator to carry out operations in a smoothand effective manner and the user to make selections smoothly.

The software is expected to complete in duration of three months.

1.3 Definitions, Acronyms, and Abbreviations

HTML	Hypertext Markup Language
CSS	Cascading Style Sheets
PHP	Hypertext Preprocessor
WAMP	WampServer refers to a solution stack for the Microsoft Windows operating system, created by Romain Bourdon and consisting of the Apache web server, OpenSSL for SSL support, MySQL database and PHP programming language.
OTP	One time password
SRS	Software Requirements Specification.
INTERNET	An interconnected system of networks that connects computers around the world via the TCP/IP protocol.

1.4 References

The references for the above software are as follows: -

- IEEE. Software Requirements Specification Std. 830-1993. Chevy Chase Bank, UMBCBranch.
- Elias M. Ewad, "System Analysis & Design".
- Pankaj Jalote, "An Integrated Approach to Software Engineering".
- Roger S. Pressman, "Software Engineering".
- Korth, "Database System Concepts".

1.5 Overview

- Section 1.0 discusses the purpose and scope of the software.
- Section 2.0 describes the overall functionalities and constraints of the software and usercharacteristics.
- Section 3.0 details all the requirements needed to design the software.

2. The Overall Description

2.1 Product Perspective

E-learning Portal provides education via the Internet, network, or standalone computer. It is basically the network- enabled convey of skills and knowledge. It uses electronic applications and processes to learn. E-learning includes all forms of electronically supported learning and teaching. The information and communication systems, whether networked learning or not, serve as specific media to implement the learning process. This often involves both out-of- classroom and in-classroom educational experiences via technology, even as advances continue concerning devices and curriculum. E-learning is the computer and network-enabled transfer of skills and knowledge. E-learning applications and processes include Web-based learning, computer-based learning, virtual education opportunities and digital collaboration.

2.2 Product Functions

The major functions that E-Learning Portal performs are described as follows: -

1- **Removes the regional restrictions**- The Portal will provide a chance to students living in remote areas having lesser knowledge resources to learn anything from their homes by using internet and laptops or Smartphone.

2- **Removes the financial restrictions**- The Portal will provide a chance to economically less- sound students to enhance their knowledge without paying any fees for doing so.

3- **Removes the need to buy and carry books**- Since all the study materials will be provided onthe portal students don't need to buy or carry any book.

4- **Removes the paperwork involved**- Since the Portal is totally internet-based it will not requireany paperwork whatsoever at any stage like registration.

5- **Removes the time-restriction**- Since the Portal can be accessed using the internet, the students can learn at any time of their choice. They need not to adhere to any schedule as such.

6- **Provides more security to the data**- The Portal will store all the records and information about students in a secured way such that only the administrator can access it.

2.3 User Characteristics

There are different kind of users that will be interacting with the system. The intended user of thesoftware are as follows: -

Admin- An Admin has all the privilege of the portal. It has privileges of a User as well as a manager. The roles granted by the admin cannot be changed by the user. The major tasks done by Admin are as follows:

- Maintaining Login Details
- Update Questions for assignments
- Account Updating
- Maintain Data Security
- Feedback Details

User- The user must provide accurate details in the registration form. The Information provided will be helpful in providing better services. Once registered, the user can select any given courseto start learning.

2.4 Constraints

2.4.1 Technology Constraints

Proposed web portal will be implemented with HTML/CSS for front end design purpose & for the back end (database purpose), we can opt for PHP/MYSQL (WAMP SERVER).

2.4.2 Interface Constraints

Since this is a Web based application so it should work on major browsers like Internet explorer,Mozilla Firefox, Google Chrome, Opera etc.

2.4.3 Safety and Security Constraints

Since, application is intended for the authenticated users only, so anonymous person should not be able to access and operate over the user data.

2.5 Assumptions and dependencies

The requirements stated in the SRS could be affected by the following factors:

One major dependency that the project might face is the changes that need to be incorporated with the updation in the study materials. A delay in doing the same will result to tremendous loss to the students. So, this should be changed as and when required by the developer.

At this stage no quantities measures are imposed on the software in terms of speed and memory although it is implied that all functions will be optimized with respect to speed and memory.

It is furthermore assumed that the scope of the package will increase considerably in the future.

3. External Interface Requirements

3.1 User interfaces

The interface provided to the user should be a very user-friendly one. The interface providedis a menu driven one and the following screens will be provided: -

1. A registration screen is provided in the beginning for entering the required details for gettingregistered.

2. If the user is already registered, he can click on login and enter his credentials.

3. If login credentials do not match, error is displayed and an option to reset password isprovided.

4. Once the login is successful, list of courses is displayed.

5. User can click on any course to start learning.

6. User can logout anytime by using logout option provided on the top right corner of thescreen.

3.2 Hardware interfaces

There are various hardware components with which the machine is required to interact. Various hardware interface requirements that need to be fulfilled for successful functioning of the software are as follows: -

- Processor: Pentium(R) Dual-Core CPU.
- 20 GB hard disk space.
- 512 MB RAM for windows XP/windows7/windows8/windows10

3.3 Software interfaces

In order to perform various functions, this software needs to interact with various other softwares. So, there are certain software interface requirements that need to be fulfilled which are listed as follows: -

- WAMP (Windows, Apache, MySQL and PHP) server should be installed to run thewebsite using localhost.
- Web Browser: Google Chrome/Mozilla Firefox/Internet Explorer 9 etc.

4. System Features

4.1 Modules Description

1] **Admin Module**: An Admin has all the privilege of the portal. It has privileges of a user aswell as a manager. A user cannot change the roles granted by the admin.

This module enables the admin to perform all necessary functions required for smooth functioning of the portal. The functions under this module are as follows: -

1. Maintaining Login Details

2. Uploading Questions for assignments

3. Add new course-options and their contents as and when required.

4. Account Updating

5. Maintain Data Security

This module also contains sub-modules like – Registration, Login and Logout, which are also present in user module, so these sub-modules are discussed in user module.

2] **User module**: User module incorporates all the features provided to the users. This is the mostcomplex modules of the portal; almost whole of the front-end is based on this module only.

Sub-modules under User module are: -

Registration: Provides a registration form to the user to fill his/her details. And provides the user-"user-id" and "password" on successful completion of the registration.

Login: Validates the user-id and password (entered by the user) from the databases.

Course-option: Provides the user different course options to choose from.

Course: Provides the user all the necessary study materials of the course

selected.**Forget Password**: Enables the users to retrieve new password via

email.

Change Password: Enables the user to change his/her password.

Logout: Provides the log-out facility to the user.

4.2 Functional requirements

The features of the e-learning portal will be:

1] It will store all data in a single centralized database. The database will contain the following information:

a] Student details

b] Administrator

detailsc] Courses

Information

2] The site will provide course study material to the students in electronic form.

3] All the information including courses offered and notices for students will also be displayed on the site.

4] The site will be used by two different types of users: Students and the administrators having different interface for each type of user.

5] Each user will register him/her by filling a registration form, so one can choose username andpassword of his/her choice.

6] Each user should be able to: -

- Change his/her password, if required after logging in.
- If the user forgets his/her password, an OTP must be sent on the user's registered e-mail id with a new password.

5. Other Non- functional Requirements

5.1 Performance requirements

The application should be able to operate on all major web-browsers with all of its fundamental functions. It should not slow-down the system even at peak hours without affecting the quality ofservice of the system.

5.2 Safety requirements

In order to make the portal safe from various kinds of attacks, several validity checks have been applied in the inputs. These validation mechanism check for proper format of email, mobile number etc.

5.3 Security requirements

- The server on which the E-learning portal will be present should have its own securityto prevent unauthorized write/delete access.

- The system should provide a secure login to the users by using advanced secure login algorithms and provide access only to the authorized users as security is the key requirement of this system.
- The user ID and the password should not be shared with anyone (students/faculty/or anyone else).

5.4 Software quality attributes

5.4.1 Reliability

The Internet must be always available to avail the services of the portal. The memory system shall be of non-volatile type.

5.4.2 Availability

The product will have a backup power supply in case of power failures. Any abnormal operationsshall result in the shutting down of the system.

After abnormal shutdown of the portal, the system shall have to be manually restarted by a maintenance personnel.

5.4.3 Security

The system shall be compatible with AIMS security standards.

- The system shall have security for user as well as admin.
- The password will be stored in database using md5() of PHP. The md5() function calculates the MD5 hash of a string. The md5() function uses the RSA Data Security, Inc. MD5 Message-Digest Algorithm. From RFC 1321 - The MD5 Message-Digest Algorithm: ***"The MD5 message-digest algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input. The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA."***
- The password shall be 6-14 characters long.
- Passwords shall not contain name of customers as they are easy to be hacked. Passwordscan contain digit, hyphen and underscore.

5.4.4 Maintainability

The system components i.e., modem, memory, disk, drives shall be easily serviceable without requiring access to the portal.

The system should have the mechanism of self-monitoring periodically in order to detect any fault. The system should inform the main branch automatically as soon as it detects any error. The kind of fault and the problem being encountered should also be mentioned by the system automatically.

6. Other Requirements

None

Experiment No-2

Experiment Name: Draw the Entity relationship diagram of a project.

Hardware Requirements: Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard n mouse, coloured monitor.

Software Requirements: SmartDraw, MS Word.

Outcome: Can produce the Entity Relationship diagram of the project.

Objective: Prepare a Requirement document in by using Entity Relationship Diagram.

Description: An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering,business information systems, education and research.

Components: An ER diagram is a means of visualizing how the information a system producesare related. There are five main components of an ERD:

Entities, which are represented by rectangles. An entity is an object or concept about which you want to store information. A weak entity is an entity that must defined by a foreign key relationship with another entity as it cannot be uniquely identified by its own attributes alone. Actions, which are represented by diamond shapes, show how two entities share information in the database.

Attributes, which are represented by ovals. A key attribute is the unique, distinguishing characteristic of the entity. For example, an employee's social security number might be the

employee's key attribute.

A multivalued attribute can have more than one value. For example, an employee entity can

havemultiple skill values.

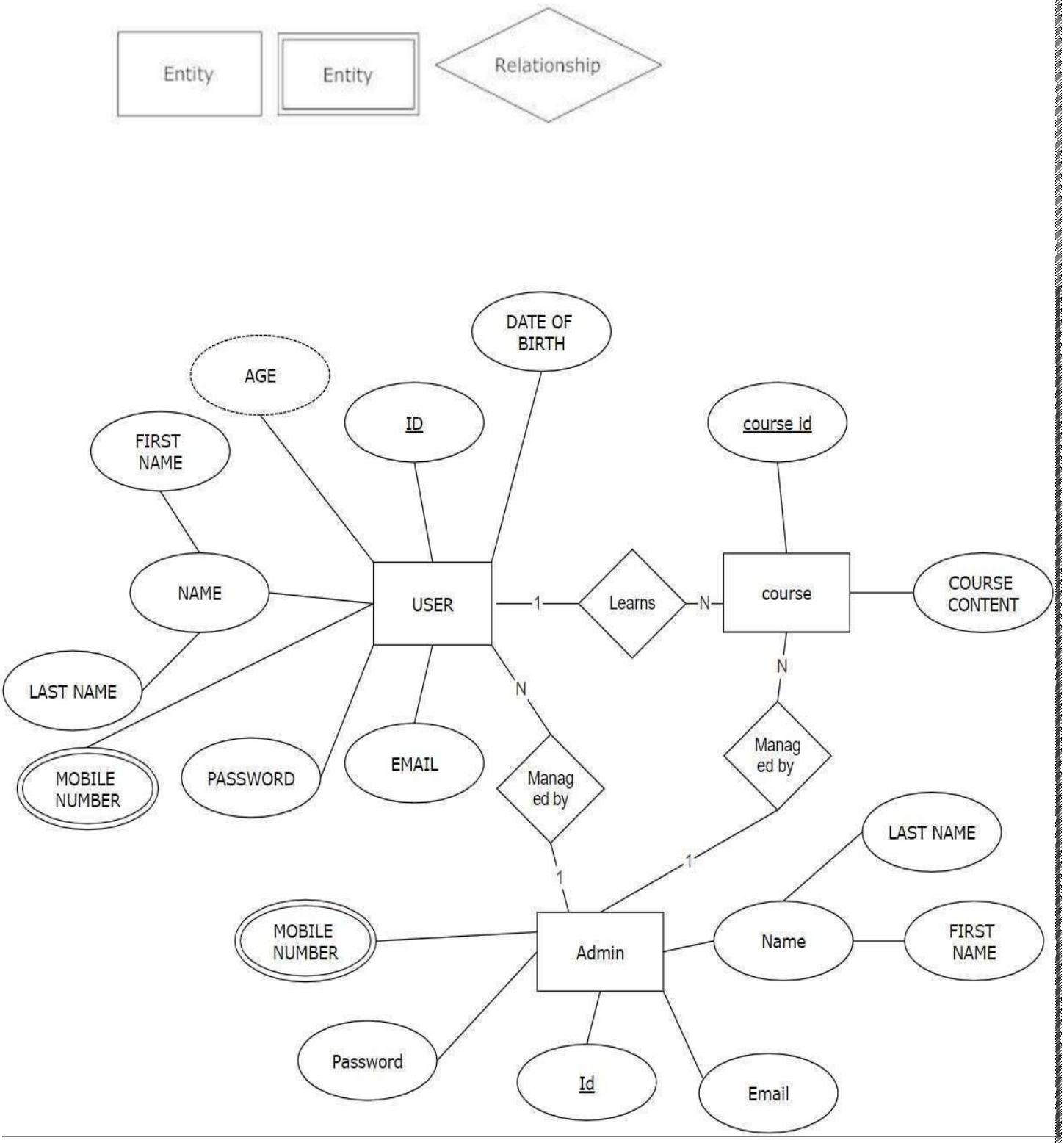
A derived attribute is based on another attribute. For example, an employee's monthly

salary is based on the employee's annual salary.

Connecting lines, solid lines that connect attributes to show the relationships of entities in the diagram.

Cardinality specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other

words, cardinality specifies the maximum number of relationships and ordinality specifies theabsolute minimum number of relationships.



E-R diagram for E-Learning Portal

Experiment No-3

Experiment Name: Draw the data flow diagrams at level 0 and level 1.

Hardware Requirements: Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard n mouse, coloured monitor.

Software Requirements: SmartDraw/MS Word, Windows XP. **Outcome:** Can produce the requirements in Data Flow diagram. **Objective:** Prepare a Data Flow Diagram for the project.

Description: Data flow diagram is graphical representation of flow of data in an information system. It can depict incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD doesnot contain any control or branch elements.

Components:

External Entity an outside system that sends or receives data, communicating with the system being diagrammed. They are the sources and destinations of information entering or leaving the system. They might bean outside organization or person, a computer system or a business system. They are also known as terminators, sources and sinks or actors. They are typically drawn on theedges of the diagram.

Process any process that changes the data, producing an output. It might perform computations, or sort data based on logic, or direct the data flow based on business rules. A short label is used to describe the process, such as "Submit payment."

Data store files or repositories that hold information for later use, such as a database table or a membership form. Each data store receives a simple label, such as "Orders."

Data flow the route that data takes between the external entities, processes and data stores. It portrays the interface between the other components and is shown with arrows, typicallylabelledwith a short data name, like "Billing details."

DFD rules

- Each process should have at least one input and an output.
- Each data store should have at least one data flow in and one data flow out.
- Data stored in a system must go through a process.

● All processes in a DFD go to another process or a data store.

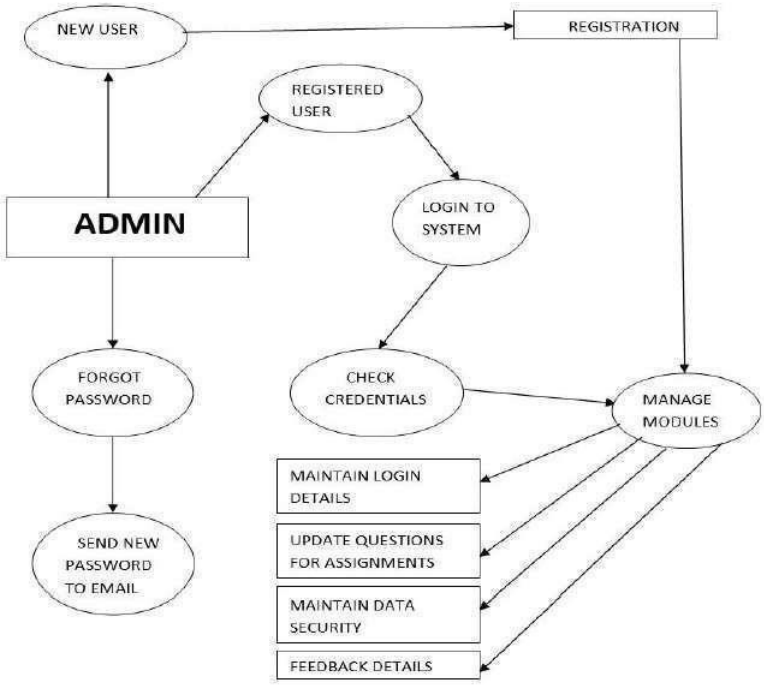
DFD Levels:

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0, 1 or 2, and occasionally go to even Level3 or beyond.

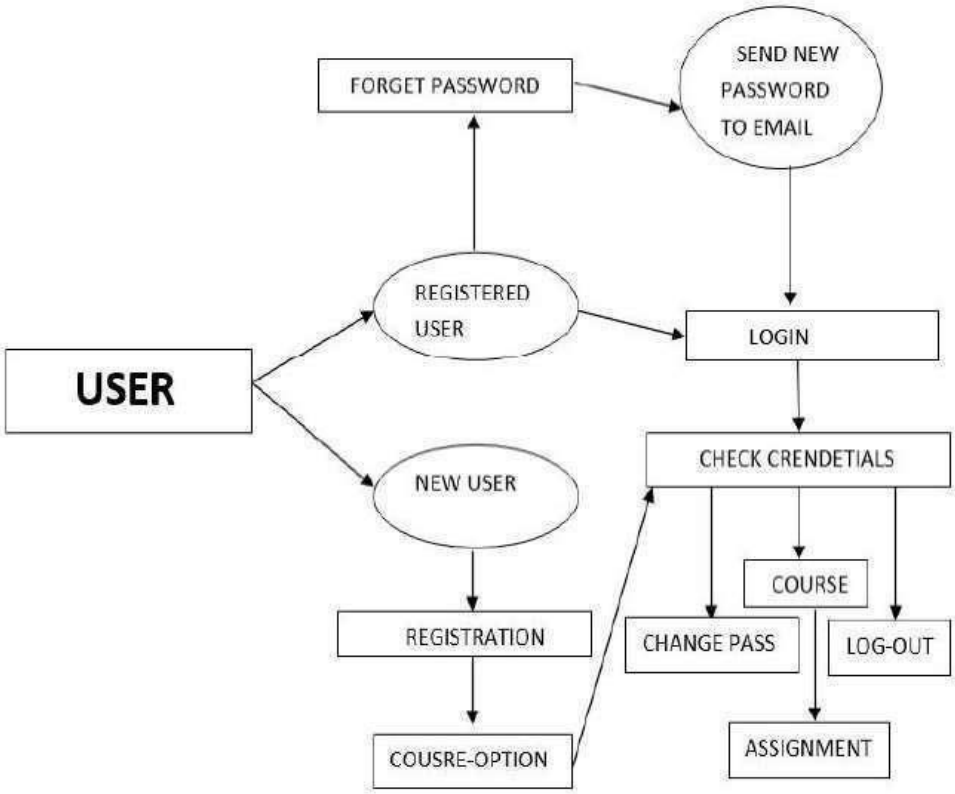
1. DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analysed or modelled. It's designed to be an at-a-glance view, showing thesystem as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.

0-Level DFD

2. DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high- level process of the Context Diagram into its subprocesses.



Level 1 DFD FOR *ADMIN*



Level 1 DFD FOR *USER*

Experiment No-4

Experiment Name: Draw use case diagram in argo UML.

Hardware Requirements: Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboardn mouse, coloured monitor.

Software Requirements: Argo UML, Windows XP.

Outcome: Can produce the requirements in Use Case

diagram.**Objective:** Prepare a Use Case Diagram for E-

Learning Portal.

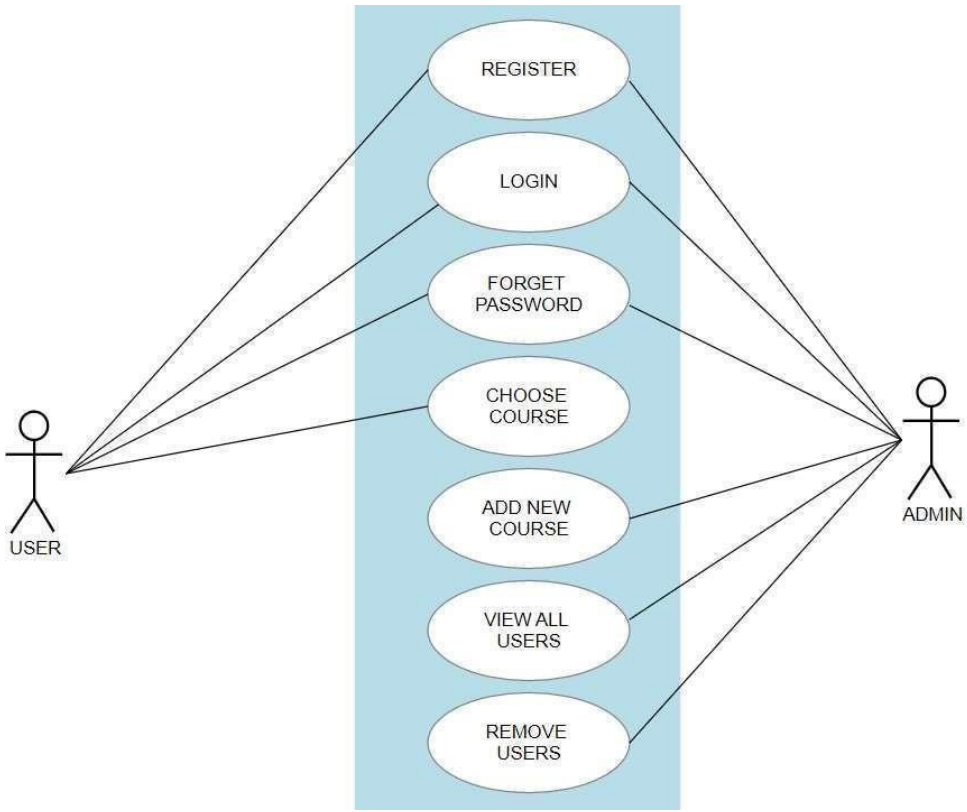
Description: According to the UML specification a use case diagram is "a diagram that shows the relationships among actors and use cases within a system." Use case diagrams are often usedto:

1. Provide an overview of all or part of the usage requirements for a system or organization inthe form of an essential model or a business model.

2. Communicate the scope of a development project.

3. Model your analysis of your usage requirements in the form of a system use case model

Use case models should be developed from the point of view of your project stakeholders and notfrom the (often technical) point of view of developers.



Use Case Diagram for E-Learning Portal

Experiment No-5

Experiment Name: Draw activity diagram in argo UML.

Outcome: Can produce the activity diagram for requirements modelling.

Objective: To Draw a sample activity diagram for real project or system.

Hardware Requirements: Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard n mouse, colored monitor.

Software Requirements: Argo UML, Windows XP,

Theory:

Activity diagrams are typically used for business process modelling, for modelling the logic captured by a single usecase or usage scenario, or for modelling the detailed logic of a business rule. Although UML activity diagrams could potentially model the internal logic of a complex operation it would be far better to simply rewrite the operation so that it is simple enough that you don't require an activity diagram. In many ways UML activity diagrams are the object- oriented equivalent of flow charts and data flow diagrams (DFDs) from structureddevelopment.

Let's start by describing the basic notation:

● Initial node. The filled in circle is the starting point of the diagram. An initial node isn't required although it does make it significantly easier to read the diagram.

● Activity final node. The filled circle with a border is the ending point. An activity diagram can have zero or more activity final nodes.

● Activity. The rounded rectangles represent activities that occur. An activity may be physical, such as Inspect Forms, or electronic, such as Display Create Student Screen.

● Flow/edge. The arrows on the diagram. Although there is a subtle difference between flows and edges, never a practical purpose for the difference although.

● Fork. A black bar with one flow going into it and several leaving it. This denotes the beginningof parallel activity.

● Join. A black bar with several flows entering it and one leaving it. All flows going into the join must reach it before processing may continue. This denotes the end of parallel processing.

● Condition. Text such as [Incorrect Form] on a flow, defining a guard which must evaluate to true in order to traverse the node.

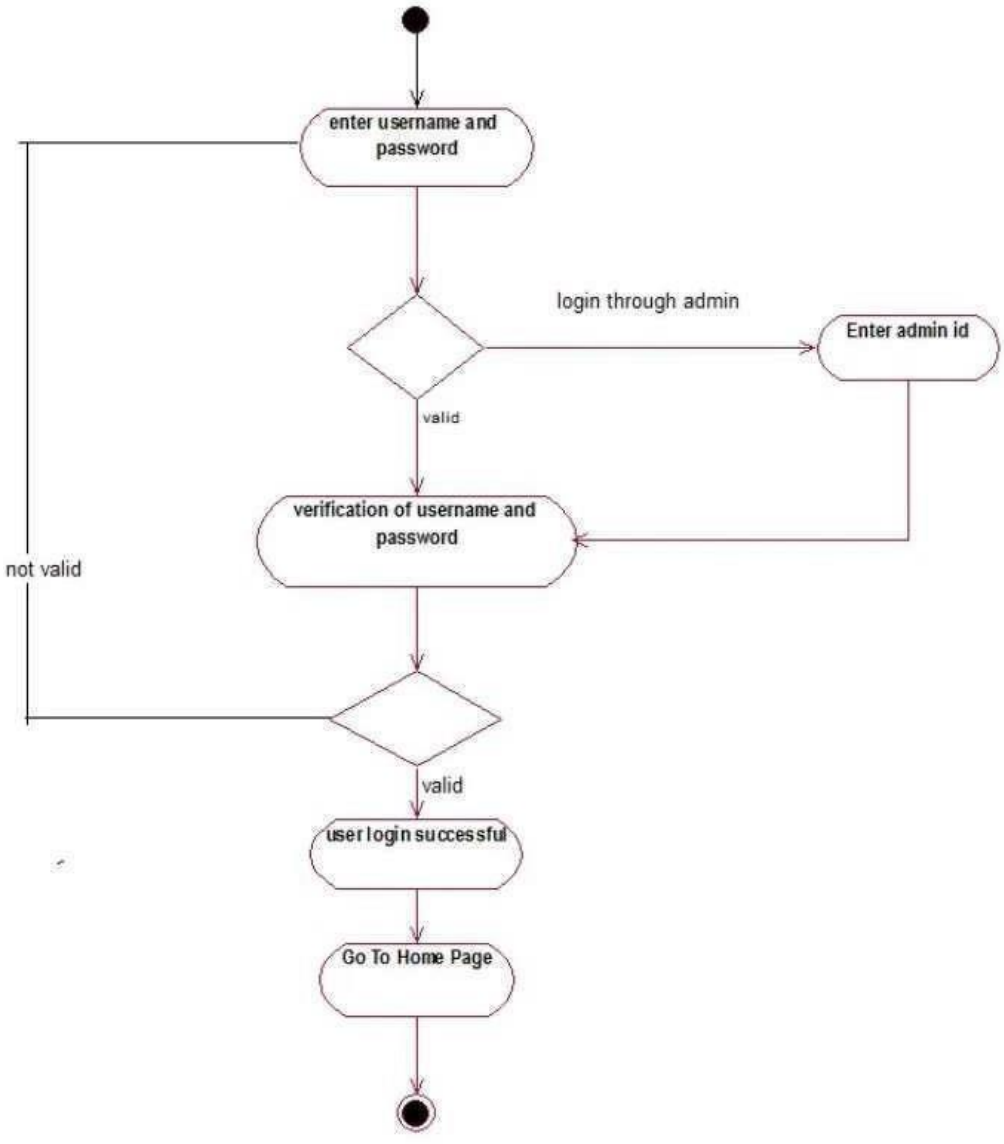
● Decision. A diamond with one flow entering and several leaving. The flows leaving include conditions although some modelers will not indicate the conditions if it is obvious.

● Merge. A diamond with several flows entering and one leaving. The implication is that one or more incoming flows must reach this point until processing continues, based on any guards onthe outgoing flow.

● Partition. If figure is organized into three partitions, it is also called swim lanes, indicating who/what is performing the activities (either the Applicant, Registrar, or System).

● Sub-activity indicator. The rake in the bottom corner of an activity, such as in the Apply to University activity, indicates that the activity is described by a more finely detailed activity diagram.

● Flow final. The circle with the X through it. This indicates that the process stops at this point.



Login Activity Diagram

Experiment No-6

Experiment Name: Draw class diagram in argo UML.

Outcome: Class diagram helps to understand the static structure of a system. It shows relationships between classes, objects, attributes, and operations.

Objective: Identify the classes. Classify them as weak and strong classes and draw the class diagram.

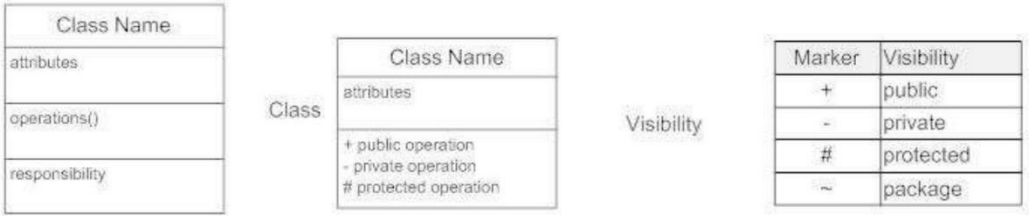
Description:

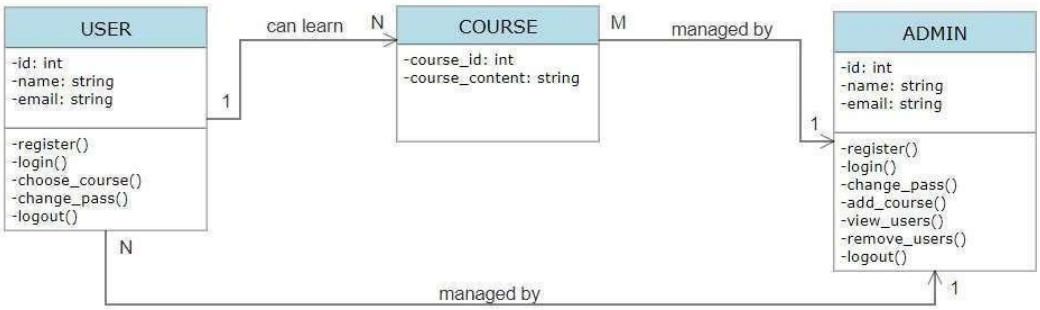
A class diagram models the static structure of a system. It shows relationships between classes, objects, attributes, and operations.

Basic Class Diagram Symbols and Notations

Classes- Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (centred, bolded, and capitalized), list the attributes in the second partition (left- aligned, not bolded, and lowercase), and write operations into the third.





Class diagram for E-Learning Portal

Experiment No-7

Experiment Name: Draw the component diagram in argo UML.

Outcome: Can draw the Component diagram.

Objective: Drawing the component diagram

Description: A component is something required to execute a stereotype function. Examples of stereotypes in components include executables, documents, database tables, files, and library files.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

An assembly connector is a "connector between two components that defines that one componentprovides the services that another component requires. An assembly connector is a connector thatis defined from a required interface or port to a provided interface or port."

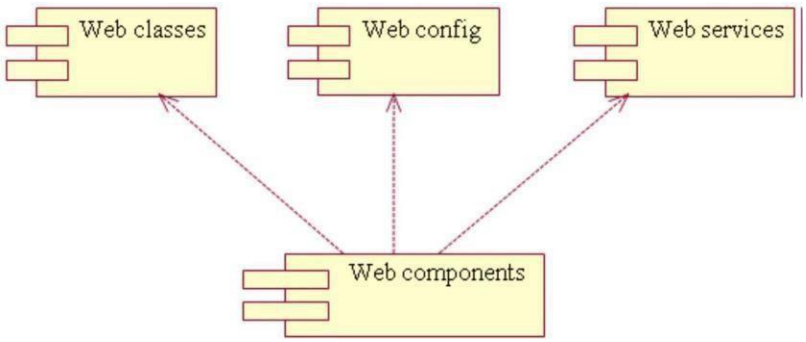
When using a component diagram to show the internal structure of a component, the provided and required interfaces of the encompassing component can delegate to the corresponding interfaces of the contained components.

A delegation connector is a "connector that links the external contract of a component (as specified by its ports) to the internal realization of that behaviour by the components parts."

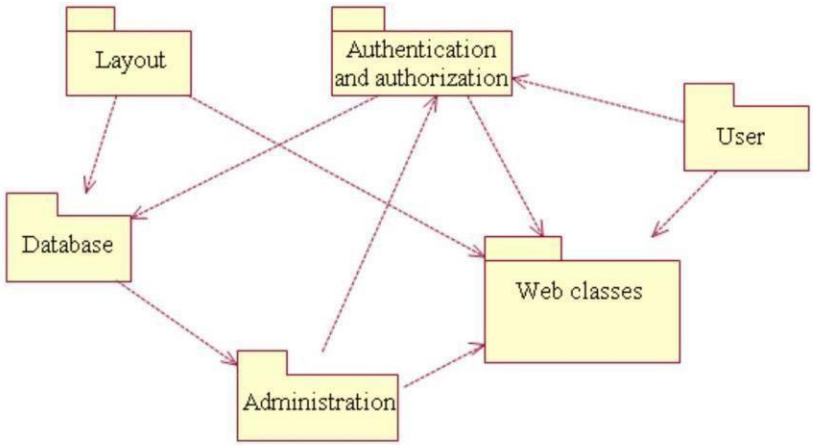
Component is represented by a rectangle with either the keyword "component" or a stereotype in the top right corner: a small rectangle with two even smaller rectangles jutting out on the left.

The lollipop, a small circle on a stick, represents an implemented or provided interface. The socket symbol is a semicircle on a stick that can fit around the lollipop. This socket is a dependency or needed interface.

The component diagram notation set now makes it one of the easiest UML diagrams to draw.



Component diagram-system components



Package diagram-system packages

Experiment No-8

Experiment Name: Draw sequence diagram in argo UML.

Outcome: Can draw the sequence diagram.

Objective: Drawing the sequence diagram

Hardware Requirements: Pentium 4 processor (2.4 GHz), 128 Mb RAM, Standard keyboard and mouse, coloured monitor.

Software Requirements: Argo UML, Windows XP

Theory:

UML sequence diagrams model the flow of logic within the system in a visual manner,enabling the user both to document and validate the logic, and are commonly used for both analysis and design purposes. Sequence diagrams are the most popular UML artifact for dynamic modelling, which focuses on identifying the behaviour within your system. Sequence diagrams, along with class diagrams and physical data models are the most important design- level models for modernapplication development.

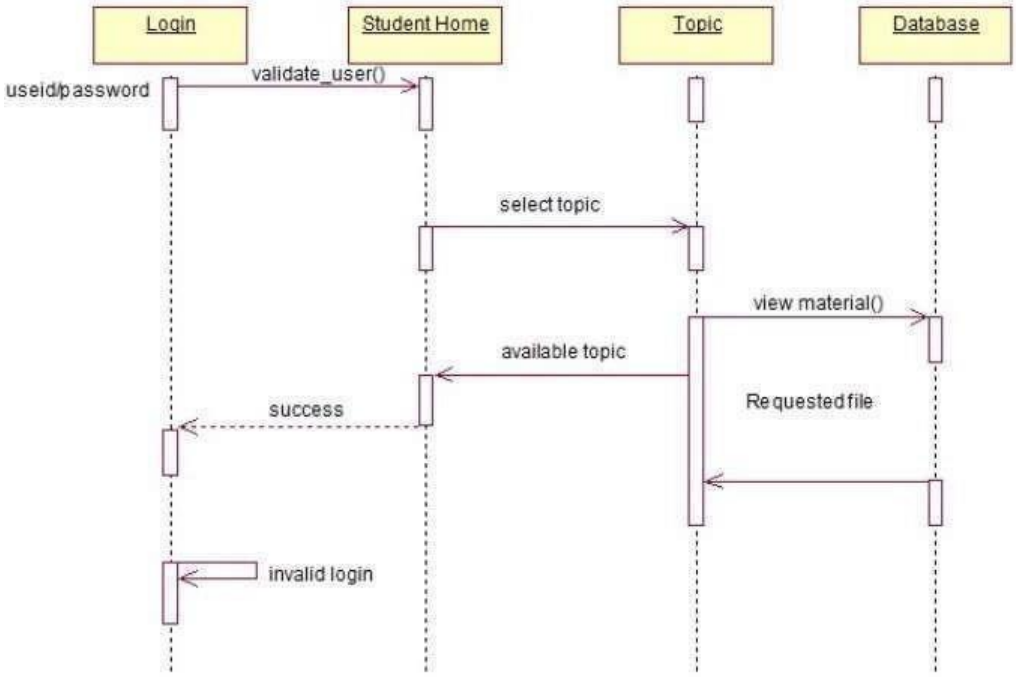
Sequence diagrams are typically used to model:

1. Usage scenarios. A usage scenario is a description of a potential way the system is used. The logic of a usage scenario may be part of a use case, perhaps an alternate course. It may also be one entire pass through a use case, such as the logic described by the basic course of action or a portion of the basic course of action, plus one or more alternate scenarios. The logic of a usage scenario may also be a pass through the logic contained in several usecases. For example,a student enrols in the university, and then immediately enrols in three seminars.

2. The logic of methods. Sequence diagrams can be used to explore the logic of a complex operation, function, or procedure. One way to think of sequence diagrams, particularly highly detailed diagrams, is as visual object code.

3. The logic of services. A service is effectively a high-level method, often one that can be invoked by a wide variety of clients. This includes web-services as well as business transactions implemented by a variety of technologies such as CICS/COBOL or CORBA-compliant object request brokers (ORBs).

The dashed lines hanging from the boxes are called object lifelines, representing the life spanof the object during the scenario being modelled. The long, thin boxes on the lifelines are activation boxes, also called method-invocation boxes, which indicate processing is being performed by the target object/class to fulfil a message.



Sequence Diagram for user

Experiment No-9

Experiment Name: Draw collaboration diagram in argo uml.

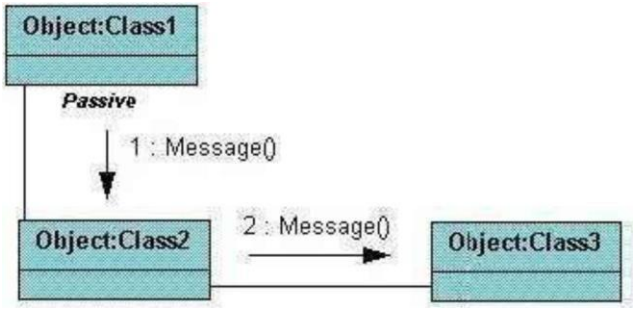
Outcome: able to draw the collaboration diagram

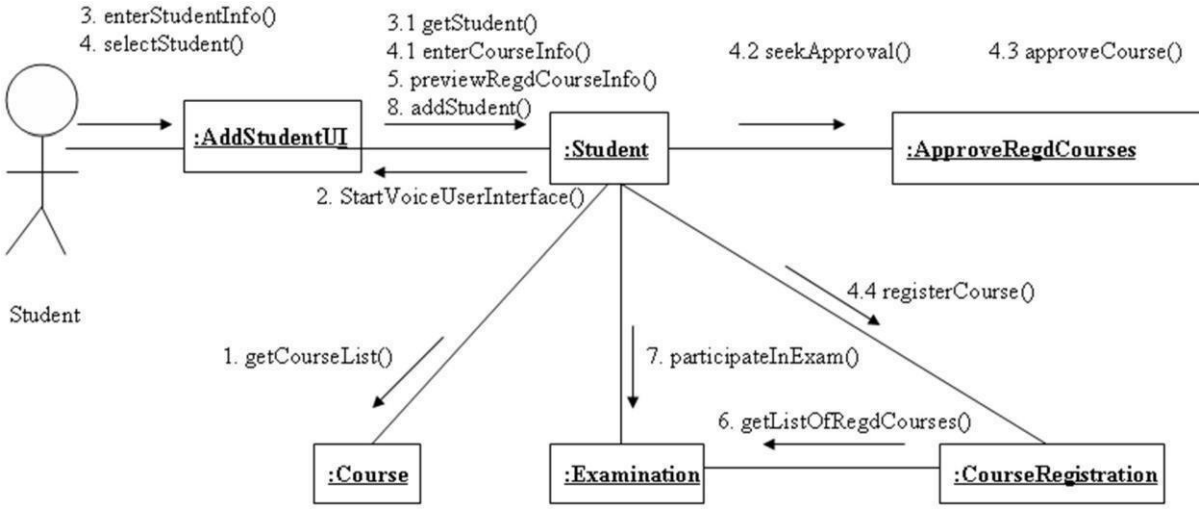
Objective: draw the collaboration diagram using

argoUML.**Description:**

Collaboration diagrams ar relatively easy to draw. They show the relationship between objects and the order of messages passed between them. The objects are listed as icons and arrows indicate the messages being passed

between them. The numbers next to the messages are called sequence numbers. As the name suggests, they show the sequence of the messages as they are passed between the objects. There are many acceptable sequence numbering schemes in UML. A simple 1, 2, 3... format can be used, as the example below shows, or for more detailed and complex diagrams a 1, 1.1 ,1.2, 1.2.1... scheme can be used.





Collaboration diagram for Course Registration

Experiment No -10

Experiment Name: Draw the COCOMO Model.

Cost Estimation (COCOMO II MODEL)

The original COCOMO model became one of the most widely used and discussed software cost estimation models in the industry. It has evolved into a more comprehensive estimation model, called COCOMO II.

COCOMO II models require sizing information. Three different sizing options are available as part of the model hierarchy: -

o Object Points

o Function Points

o Lines Of Source Code

The COCOMO II application composition model uses object points.

Like function point, the **object point is an indirect software measure** that is computed using counts of the number of

(1) screens (at the user interface),

(2) reports,

(3) components likely to be required to build the application.

Each object instance (e.g., a screen or report) is classified into one of three complexity levels (i.e. ,simple ,medium, or difficult).

Once complexity is determined, the number of screens, reports, and components are weighted according to the table illustrated in

TABLE 5.4 COCOMO II Complexity Weights

OBJECT TYPE COMPLEXITIY WEIGHT

SIMPLE MEDIUM DIFFICULT

Computing of cyclomatic Complexity

Let G be a given CFG. Let E donate the number of edges, and N denote the number of nides.

Let V(G) denote the cyclomatic Complexity for the CFG, V(G) can be obtained in either of the following three ways:

- Method #1: V(G) = E – N + 2P
- Method #2: V(G) could be directly computed by visual inspection of the CPG:V(G) = Total number of bounded areas + 1It may be noted here structured programming would always in lead to a planar CFG.
- Method #3: If LN be the total number of loops and decision statement in a program then V(G) = LN + 1

Optimum value of Cyclomatic Complexity

A set of threshold value for cyclomatic complexity has been presented in (VII), Which we reproduce below.

V(G)	Module Category	Risk
1 – 10	Simple	Low
11 – 20	More Complex	Moderate
21 – 50	Complex	High
> 50	Unstable	Very High

Merits

McCabe's Cyclomatic Complexity has certain advantages:

- Independent of programming language
- Helps in risk analysis during development of maintenance phase

Demerits

Cyclomatic Complexity doesn't reflect on cohesion and coupling of modules.

For diagram (A) => Cyclomatic Complexity =>

=> E = 12

=> N = 11

=> P = 3

So formula as,

V (G) = E – N + 2P

= 12 – 11 + 2*3

= 12 – 11 + 6

= 7

V (G) = 7 Ans.

Experiment No -11

Experiment Name: Draw the Software Testing.

Testing:

Need for Software Testing:

Test Cases and Test Suite:

A test case describes and input descriptions and an excepted output descriptions. Input are of two type: preconditions the set of test case is called a test suite be may have suite of all test possible cases.

Types of software testing:

Testing is done in every stage of software development lifecycle, but the testing done at each level of software development if is different in nature and has different objectives.

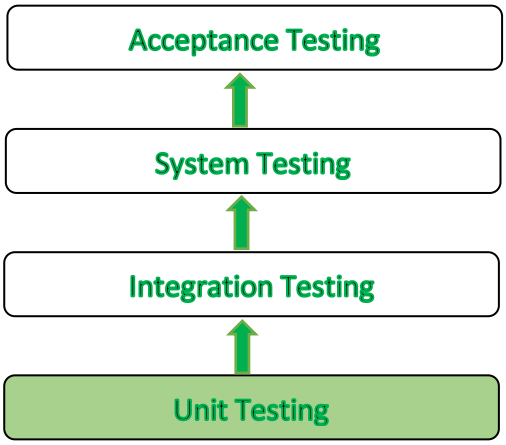
1. Unit Testing
2. Integration Testing
3. System Testing

Unit Testing:

Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures, and operating procedures are tested to determine whether they are suitable for use or not. It is a testing method using which every independent module is tested to determine if there is an issue by the developer himself. It is correlated with the functional correctness of the independent modules. Unit Testing is defined as a type of software testing where individual components of a software are tested. Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer. In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is such a type of testing technique that is usually performed by developers.

Objective of Unit Testing:

- To isolate a section of code.
- To verify the correctness of the code.
- To test every function and procedure.
- To fix bugs early in the development cycle and to save costs.
- To help the developers to understand the code base and enable them to make changes quickly.
- To help with code reuse.



Unit Testing Techniques:

There are 3 types of Unit Testing Techniques. They are,

Black Box Testing:

Black Box Testing is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioural Testing.

White Box Testing: This technique is used in testing the functional behaviour of the system by giving the input and checking the functionality output including the internal design structure and code of the modules.

Gray Box Testing: This technique is used in executing the relevant test cases, test methods, test functions, and analysing the code performance for the modules.

Unit Testing Tools:

Here are some commonly used Unit Testing tools:

- Jtest
- Junit
- N Unit
- EMMA
- PHP Unit

Integration testing:

Integration testing is the process of testing the interface between two software units or modules. It focuses on determining the correctness of the interface. The purpose of integration testing is to expose faults in the interaction between integrated units. Once all the modules have been unit tested, integration testing is performed.

Integration test approaches – There are four types of integration testing approaches. Those approaches are the following:

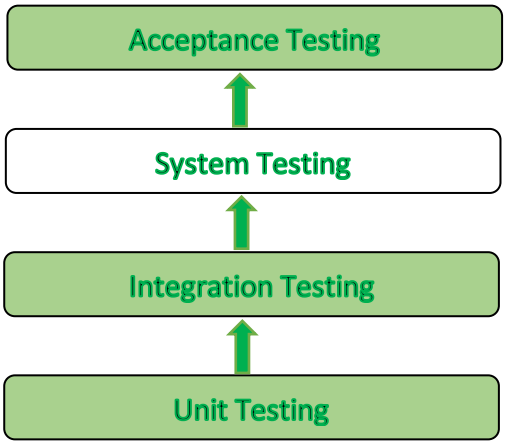
1. Big-Bang Integration Testing – It is the simplest integration testing approach, where all the modules are combined and the functionality is verified after the completion of individual module testing. In simple words, all the modules of the system are simply put together and tested. This approach is practicable only for very small systems. If an error is found during the integration testing, it is very difficult to localize the error as the error may potentially belong to any of the modules being integrated. So, debugging errors reported during big bang integration testing is very expensive to fix.

2. Bottom-Up Integration Testing – In bottom-up testing, each module at lower levels is tested with higher modules until all modules are tested. The primary purpose of this integration testing is that each subsystem tests the interfaces among various modules making up the subsystem. This integration testing uses test drivers to drive and pass appropriate data to the lower level modules.

3. Top-Down Integration Testing – Top-down integration testing technique is used in order to simulate the behaviour of the lower-level modules that are not yet integrated. In this integration testing, testing takes place from top to bottom. First, high-level modules are tested and then low-level modules and finally integrating the low-level modules to a high level to ensure the system is working as intended.

System Testing:

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements. In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together. System testing detects defects within both the integrated units and the whole system. The result of system testing is the observed behaviour of a component or a system when it is tested. System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both. System testing tests the design and behaviour of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS). System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing. System Testing is a black-box testing. System Testing is performed after the integration testing and before the acceptance testing.



Regression Testing:

Regression Testing is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made. Regression means return of something and in the software field, it refers to the return of a bug.

CHAPTER - 7
CONCLUSION

Working on the project was an excellent experience. It helped us to understand the importance of planning, designing and implementation so for we have learnt in our theory books. It helped us unleashing our creativity while working in a team. It also realized the importance of team working, communication as a part of this project.

The project was successfully completed after a lot of efforts and work hours. This project underwent number of compiling, debugging, removing errors, making it bug free, adding more facilities in **E – Learning Portal** making it more reliable and useful.

This project focused that scheduling a project and adhering to that schedule creates a hard sense of time- management. It has also let us known that co-operative teamwork always produce effective results.

The entire project has been developed and deployed as per the requirements stated by the user. It is found to be bug free as per the testing standards that are implemented.

The estimated cost of the project is (efforts) 12 and the estimated size of the project is (FP) 209.72.

There are also few features which can be integrated with this system to make it more flexible. Below list shows the future points to be consider :

• Through this project we can read any book or other document on any online portal.

Finally, with this project we can upload any book on any online portal and it can be read anywhere and can be sent to anyone through that portal.