CASE STUDY: ILEARN – A DIGITAL LEARNING ENVIRONMENT

The iLearn system is a digital learning environment used to support learning in schools with students from age 4 to 18. It is intended to replace an existing system (Glow) that was specially built for the purpose and which includes its own applications for e-mail, etc. It became less and less used as the facilities in freely available systems were far superior to those offered in the closed system.

One of the most important requirements for the iLearn system was that it should be an open system that could easily accommodate new features and existing services. We aimed to achieve this by designing the system so that everything was a service and that, with appropriate permissions, users could replace pre-specified services with their own service version. This approach also allowed us to deal with the complexity of integrating with existing network management systems (local areas had different policies on which web sites could be visited by school students, depending on age and content) and school administration systems. By creating a service interface to these systems, different underlying systems could be accommodated.

- Independent Learning, Environments, And Responsible Networking
- It is the vision for teaching and learning supported by technology
- An opportunity to enhance instructional strategies and resources for continued growth of lifelong learners in our educational community
- Supports 21st Century and STEM skills and concepts: Creativity, Communication, Collaboration, Citizenship, and Critical thinking.

Software Requirements Specification

Aim:

To write the Software Specification Document for ILEARN -A DIGITAL LEARNING ENVIRONMENT.

Table of Contents:

1. Introduction

The iLearn system is a digital learning environment used to support learning in schools with students from age 4 to 18.

1.1 Purpose

This Software Requirement Specification (SRS) specifies the requirements of the Digital learning which will be used by the students and teachers.

1.2 Intended Audience and Reading Suggestions

Students: The remote students will be able to get access to their teachers with this new teaching pedagogy.

Developers: Project developers have an advantage of quickly understanding the methodology enabled and personalizing the product.

Policy Makers: The senior policy makers can view this document as an analysis tool to develop further e-learning products.

The Developers would suggest clients to go through the requirement section thoroughly before installing the software. Policy Makers and Developers can utilize the documentation as a resource in developing the project to a new product.

1.3 Project Scope

It is intended to replace an existing system that was specially built for the purpose and which includes its own applications for e-mail, etc.

Facilitating education for students, teachers and even parents. Provide an integrated educational environment so that students can interact with the system in an easy and flexible

1.4 References

Websites:

-web.cs.dal.ca>~hawkey>srs.template_ieee

-iansommerville.com>casestudies>ilearn

-prezi.com/p/zyikq4yqg3ek/ilearn-a-digital-learning-environment/

-thehighereducationreview.com/magazine/digital-learning-in-higher-education-the-indian-perspective-NQHZ619177297.html

2. Overall Description

2.1 Product Perspective

Due to far reaching impact of digital learning at global as well as at local level, it has become a way of learning. The learner's response on the use of different digital learning platforms shows that the use of digital learning technology increases student engagement in course, barrier free access to learning materials and the use of adaptive technology in digital content shows definite improvement in their performance.

2.2 Product Features

Different groups of students learn best in different ways and progress at different rates. Teaching inclusively enables all students, whatever their circumstances, to enjoy the fullest possible learning experience. It benefits all students because it values their individual strengths and contributions and makes the learning experience richer and more diverse for everyone. Having a wider range of views and experiences in the classroom can lead to a more critical understanding of a subject. As teachers, it can challenge us to rethink what and how we teach, and to widen the materials we include on any given subject.

2.3 User Classes and Characteristics

Independent: An independent learner takes responsibility for their own learning.

Creative thinker: Creative thinking can be stimulated both by an unstructured process such as brainstorming, and by a structured process such as lateral thinking.

Problem solver: Ability to handle difficult or unexpected situations in the classroom as well as complex study challenges.

Social learners: They are so used to getting information at a moment's notice that you have to grab their attention and manage time effectively.

Empowered learner: Being motivated to perform tasks, and more specifically an empowered learner finds the tasks meaningful, feels competent to perform them, and feels his/her efforts have an impact.

2.4 Operating Environment

A digital learning platform is a piece of software designed to heavily assist during the educational process. There are a range of options available depending on the specific needs of the institution. They include: learning management systems (LMS), learning content management systems (LCMS), as well as virtual classroom tools and virtual learning environments (VLE).

2.5 Design and Implementation Constraints

System have students and doctors. We may find lecture notes , readings , quizzes, other learning resources and activities for your units depending on what your lecture has chosen to make available . we can submit assignments and grads. we can make group chats in ilearn. we can add some lectures, quizzes and other in ilearn, student can login ilearn, submit assignment, find lecture and quizzes, communicate with doctor. doctor can add lecture, add assignment for student, add quizzes, communicate with group of student, add grade of student

2.6 Assumptions and Dependencies

1. The Productivity Assumption: The assumption is that we need to find ways to simultaneously raise quality while bending the educational cost curve.

2. The Technology is a Lever Assumption: The assumption here is technology will a positive force, and that if only higher education can utilize technology as effectively as other industries that we will see improvements in postsecondary productivity.

3. The Disruption Assumption: The assumption that educational technology will also follow the innovators dilemma narrative. That the path to higher quality learning technologies will inevitably first go through lower cost and inferior platforms and services. That the system of higher education is not immune from the forces facing other industries.

4. The Active Learning Assumption: This is assumption that active learning is good and passive learning is bad.

5. The Blended Learning Assumption: This is the assumption that the best courses will usually be blended courses. Partly face-to-face and partly online.

6. The Consumerization Assumption: The assumption is that higher education needs to in some sense keep up with the consumer world of technology if we are to remain relevant to our students.

4. External Interface Requirements

4.1 User Interfaces

The students should be able to learn easily and timely the features

provided.

The home page interface should be simple and memorable design

and guide the students directly to the content area where the menu for course selection is available. The

simple and memorable design can help users to easy accessing the interface. The important content area was highlighted to the students by displaying the course menu in a table with different background colour. Besides that, all the important announcements that need to be given to the students were posted just above of the course menu.

Personal page: The first page user sees after entering and is very important in terms of design.

Virtual class: In fact, is the main interactive tool between professor and student.

Communicative part: Allow participants to discuss and speak with each other simultaneously.

Library: Available immediately books, articles, magazines and etc to users.

4.2 Hardware Interfaces

The teacher can use the laptop to do presentations to the students and even for parents; they can take it to meetings or on field trips for student-based research. Palms or some type of hand-held computer can be used by students for field research.

A webcam allows for access to professionals with similar interests. That interaction might not happen without face-to-face voice and image access. The interactive video software is getting better with faster Internet connections and that's more reliable medium.

4.3 Software Interfaces

As for software, the biggest concern is that the computer must have some type of good anti-virus protection -- viruses are becoming worse and worse. A good security system, such as a firewall, also is a must.

The most important software tool that teachers should have, and become expert in using is Microsoft Office 2007. The reason I say that is because Microsoft Office, in general, incorporates essential tools for teaching and learning. Microsoft Office 2007, in particular -- besides being a very nice user interface, which makes the functions of the software more readily available -- has added a considerably larger set of drawing and diagramming tools that are excellent for both teachers and students when brainstorming or in the preparation of visual learning materials."

4.4 Communications Interfaces

For interaction between the teacher and technology, 64% of the students agreed that the professor's knowledge

and use of the technology during the synchronous sessions was critical to the overall success of the class.

5. Other Non-functional Requirements

5.1 Performance Requirements

The important aspects of Pathshala software is time constrain. Pathshala software system is real time and hence should be performed in minimum requirements.

The accountability is a vital feature and this could only be assured if the system is working in full capability.

5.2 Safety Requirements

Reliable Internet is the backbone of the software so for the live broadcasting of the video needs sufficient and uninterrupted internet connection.

Power is a significant feature and the power supply should be always taken care of. An uninterrupted power supply is always recommended.

5.3 Security Requirements

The security system features from having a login for all the users to access the software. The login details will be used in the system also. So, the chances of the software getting intruded are very less.

5.4 Software Quality Attributes

The Java Virtual Machine helps the Pathshala to achieve platform independence. Hence, it can run on any environment that is available in the client computer.

6. Other Requirements

Legal Requirements:

Illegal duplication of the reports will be strictly dealt with. This is not an open source software hence source code of the product won’t be open. Further modifications and improvements rights will be with the developer team.

Appendix A: Glossary

1) SRS: Software Requirement Specification
2) Client/User: Students
3) Server: Teacher
4) RAM: Random Access Memory
5) MB: Megabyte (Unit of Memory Storage)
6) SQL: Structured Query Language
7) HTTP: Hyper Text Transfer Protocol
8) User ID: Unique username issued to each user on login
9) Password: Unique word given to each user as a secret code
10) JVM: Java Virtual Machine
11) JMF: Java Media Framework (v 2.1.5)
12) JRE: Java Runtime Environment.

UML Diagrams

UML stands for “Unified Modelling Language”. The UML diagrams visually represents a system along with it’s main actors, roles, actions, artifacts in order to better understand, alter, maintain or document information about the system. The UML diagrams are based on the diagrammatic representation of the software components.

The types of UML diagrams are:

- Class Diagram
- Object Diagram
- Use case Diagram
- Sequence Diagram
- Collaboration Diagram
- Activity Diagram
- State Diagram
- Component Diagram
- Deployment Diagram

**Design the UML Diagrams for the ILEARN A DIGITAL LEARNING ENVIRONMENT.

Aim: To write the UML code that generates the Usecase Diagram for the ILEARN A DIGITAL LEARNING ENVIRONMENT.

Description:

A **use case diagram** at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses.

Template:

A template is a pre-formatted way that describes the format and the way of creating files or documents. The sample template for Use case diagram is given by:

USE CASE TEMPLATE

- **Name** – A clear verb/noun or actor/verb/noun descriptor that communicates the scope of the use case.
- **Brief Description** – A brief paragraph of text describing the scope of the use case.
- **Actors** – A list of the types of users who can engage in the activities described in the use case. Actor names should not correspond to job titles.
- **Preconditions** – Anything the solution can assume to be true when the use case begins.
- **Basic Flow** – The set of steps the actors take to accomplish the goal of the use case. A clear description of what the system does in response to each user action.
- **Alternate Flows** – Capture the less common user/system interactions, such as being on a new computer and answering a security question.
- **Exception Flows** – The things that can happen that prevent the user from achieving their goal, such as providing an incorrect username and password.
- **Post Conditions** – Anything that must be true when the use case is complete.

Template for ILEARN A DIGITAL LEARNING ENVIRONMENT

Use case:

Use Case Name: **ILEARN A DIGITAL LEARNING ENVIRONMENT**

Summary: The iLearn system is a digital learning environment used to support learning in schools with students

Actors: 1.Admin and User: It collects the data and transfers it to the Database.

2. DB and Archiving system : Takes the data from Admin and User and processes it and stores in the database and retrieves the data for further usage.

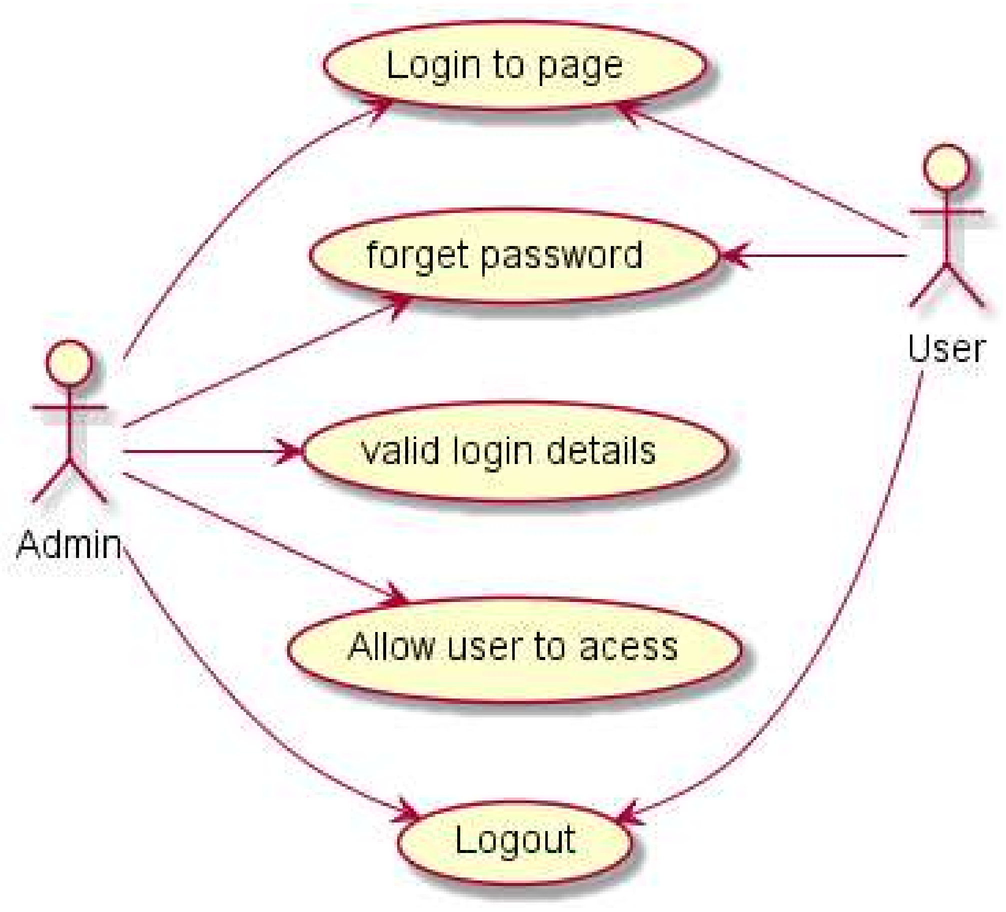
Description ("Scenario"):

- 1. Independent Learning, Environments, And Responsible Networking.
- It is the vision for teaching and learning supported by technology
- An opportunity to enhance instructional strategies and resources for continued growth of lifelong learners in our educational community
- Supports 21st Century and STEM skills and concepts: Creativity, Communication, Collaboration, Citizenship, and Critical thinking.

Program:

```
@startuml
left to right direction
Admin --> (Login to page)
Admin --> (forget password)
Admin --> (valid login details)
Admin --> (Allow user to acess)
Admin --> (Logout)
(Login to page) <-- User
(forget password) <-- User
(Logout) <-- User
@enduml
```

Output:


Login to page
forget password
valid login details
Allow user to acess
Logout
Admin
User

AIM:

To write the UML code that generates the State Diagram for the ILEARN -A DIGITAL LEARNING ENVIRONMENT.

Description:

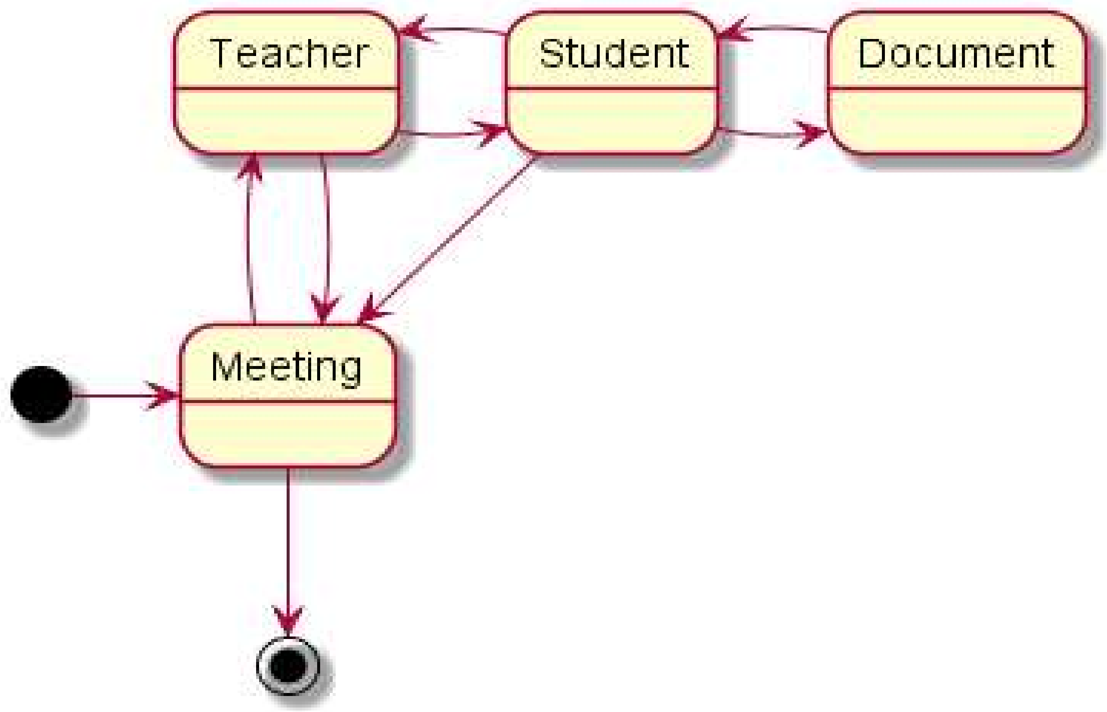
UML state machine, also known as **UML state chart**, is an extension of the mathematical concept of a finite automaton in computer science applications as expressed in the Unified Modelling Language (UML) notation.

The concepts behind it are about organizing the way a device, computer program, or other (often technical) process works such that an entity or each of its sub-entities is always in exactly one of a number of possible states and where there are well-defined conditional transitions between these states.

PROGRAM:

```
@startuml
[*] -right-> Meeting
Meeting -up-> Teacher
Teacher -right-> Student
Student -left-> Teacher
Student -right-> Document
Document -left-> Student
Student --> Meeting
Teacher --> Meeting
Meeting -down-> [*]
@enduml
```

OUTPUT:



Aim:

To write the UML code that generates the Sequence Diagram for the ILEARN-A DIGITAL LEARNING ENVIRONMENT.

Description:

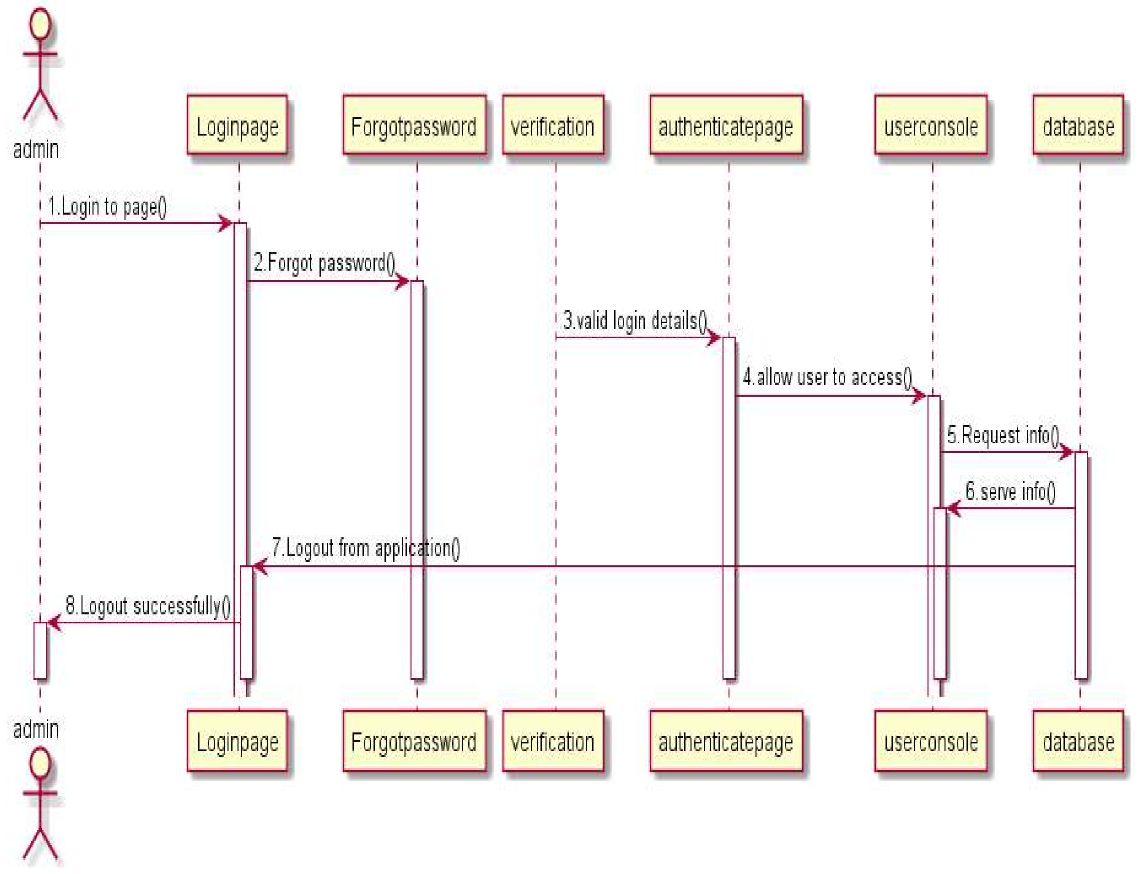
A **sequence diagram** shows object interactions arranged in time sequence. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called **event diagrams** or **event scenarios**.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner.

PROGRAM:

```
@startuml
actor admin
admin->Loginpage++:1.Login to page()
Loginpage->Forgotpassword++:2.Forgot password()
verification->authenticatepage++:3.valid login details()
authenticatepage->userconsole++:4.allow user to access()
userconsole->database++:5.Request info()
database->userconsole++:6.serve info()
database->Loginpage++:7.Logout from application()
Loginpage->admin++:8.Logout successfully()
deactivate admin
deactivate Loginpage
deactivate Forgotpassword
deactivate verification
deactivate authenticatepage
deactivate userconsole
deactivate database
@enduml
```

OUTPUT:


admin
Loginpage
Forgotpassword
verification
authenticatepage
userconsole
database
1.Login to page()
2.Forgot password()
3.valid login details()
4.allow user to access()
5.Request info()
6.serve info()
7.Logout from application()
8.Logout successfully()

Aim :

To write the UML code that generates the Object Diagram for the ILEARN-A DIGITAL LEARNING ENVIRONMENT.

Description:

An **object diagram** in the Unified Modelling Language (UML), is a diagram that shows a complete or partial view of the structure of a modelled system at a specific time.

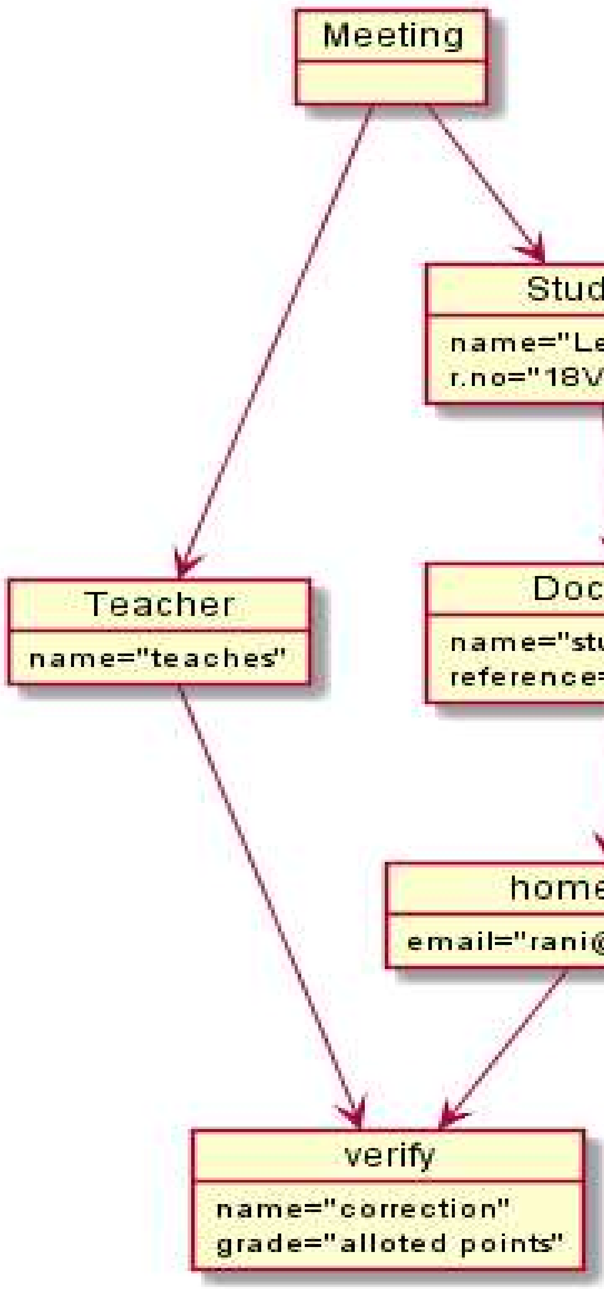
An object diagram is a graph of instances, including objects and data values. A static object diagram is an instance of a class diagram; it shows a snapshot of the detailed state of a system at a point in time. The use of object diagrams is fairly limited, namely to show examples of data structure

Program:

```
@startuml
object Meeting
object Teacher
{
	name="teaches"
}

object Student
{
	name="Learner"
	r.no="18VV1A12--"
}

object Document
{
	name="study material"
	reference="google"
}
```

```
object homework
{
        email="rani@email.com"
}

object verify
{
        name="correction"
        grade="alloted points"
}

Meeting --> Teacher
Meeting --> Student
Teacher -->verify
Student --> Document
Document --> homework
homework --> verify
@enduml
```

OUTPUT:


Meeting
Student
name="Learner"
r.no="18VV1A12--"
Teacher
name="teaches"
Document
name="study material"
reference="google"
homework
email="rani@email.com"
verify
name="correction"
grade="alloted points"

Aim:

To write the UML code that generates the Component Diagram for the ILEARN-A DIGITAL LEARNING ENVIRONMENT.

Description:

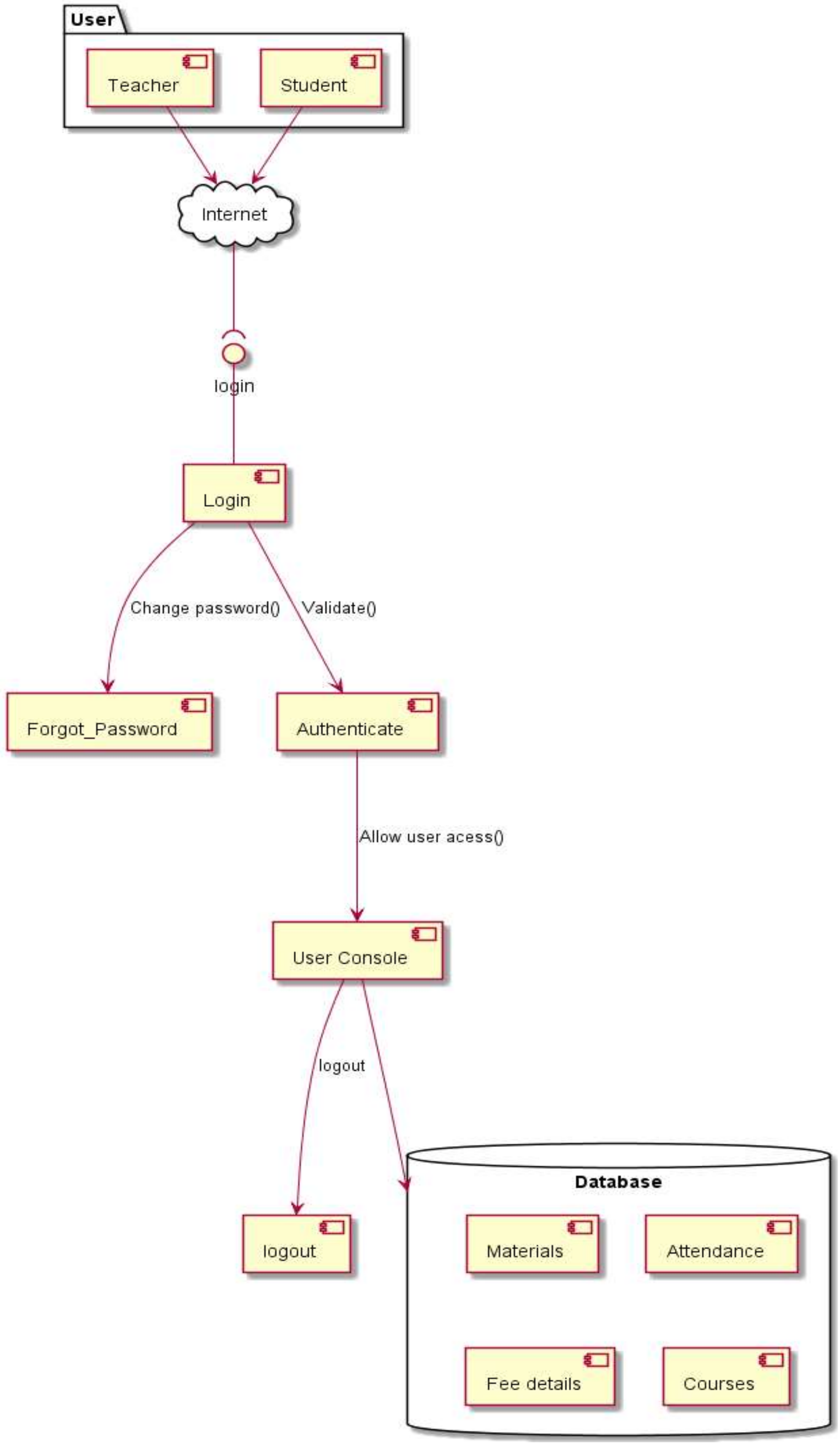
A component diagram allows verification that a system's required functionality is acceptable. These diagrams are also used as a communication tool between the developer and stakeholders of the system. Programmers and developers use the diagrams to formalize a roadmap for the implementation, allowing for better decision-making about task assignment or needed skill improvements. System administrators can use component diagrams to plan ahead, using the view of the logical software components and their relationships on the system

Program:

```
@startuml
package "User"
 {
   component [Teacher] as teacher
   component [Student] as student
}
component [Login] as l1
component [Forgot_Password] as fp
component [Authenticate] as auth
component [User Console] as uc
component [logout] as lout
cloud Internet {
}
database "Database"
{
   [Materials]
   [Attendance]
   [Fee details]
   [Courses]
}
```

```
student -down-> Internet
teacher -down-> Internet
Internet -down-( login
[l1] -up- login
l1-down-->auth: Validate()
l1-down-->fp: Change password()
auth-down-->uc: Allow user acess()
uc-down-->Database
uc-down-->lout:logout
@enduml
```

OUTPUT:


User
Teacher
Student
Internet
login
Login
Change password()
Validate()
Forgot_Password
Authenticate
Allow user acess()
User Console
logout
logout
Database
Materials
Attendance
Fee details
Courses

Aim:

To write the UML code that generates the Collaboration Diagram for the ILEARN-A DIGITAL LEARNING ENVIRONMENT.

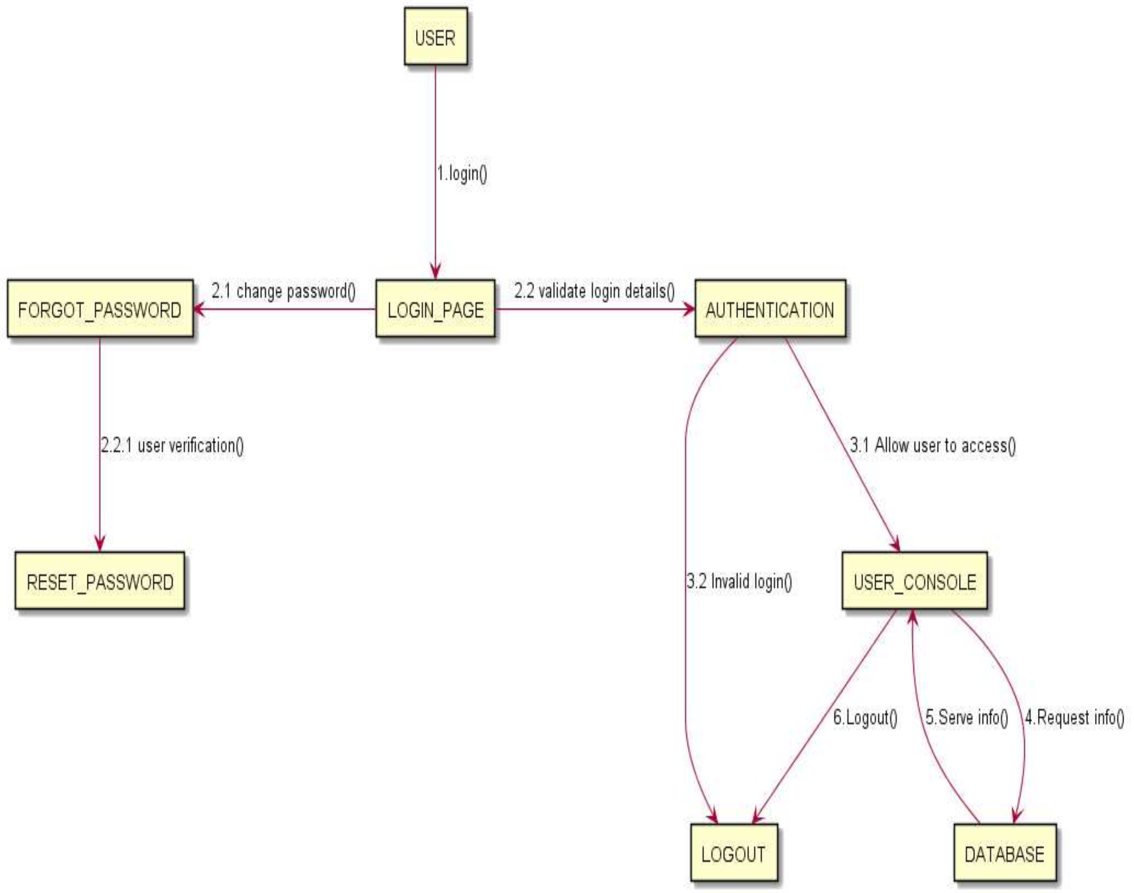
Description:

A Communication diagram models the interactions between objects or parts in terms of sequenced messages. Communication diagrams represent a combination of information taken from Class, Sequence, and Use Case Diagrams describing both the static structure and dynamic behaviour of a system.

Program:

```
@startuml
rectangle USER as R1
rectangle LOGIN_PAGE as R2
rectangle AUTHENTICATION as R3
rectangle FORGOT_PASSWORD as R4
rectangle USER_CONSOLE as R5
rectangle RESET_PASSWORD as R6
rectangle DATABASE as R7
rectangle LOGOUT as R8
R1-down-->R2:1.login()
R2-left-->R4:2.1 change password()
R2-right-->R3:2.2 validate login details()
R4-down-->R6:2.2.1 user verification()
R3-down-->R5:3.1 Allow user to access()
R3-down-->R8:3.2 Invalid login()
R5-down-->R7:4.Request info()
R7-up-->R5:5.Serve info()
R5-down-->R8:6.Logout()
@enduml
```

OUTPUT:


USER
1.login()
FORGOT_PASSWORD
2.1 change password()
LOGIN_PAGE
2.2 validate login details()
AUTHENTICATION
2.2.1 user verification()
3.1 Allow user to access()
RESET_PASSWORD
3.2 Invalid login()
USER_CONSOLE
6.Logout()
5.Serve info()
4.Request info()
LOGOUT
DATABASE

Aim:

To write the UML code that generates the Class Diagram for the ILEARN-A DIGITAL LEARNING ENVIRONMENT.

Description:

In software engineering, a **class diagram** in the Unified Modelling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of object-oriented modelling. It is used for general conceptual modelling of the structure of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

In the diagram, classes are represented with boxes that contain three compartments:

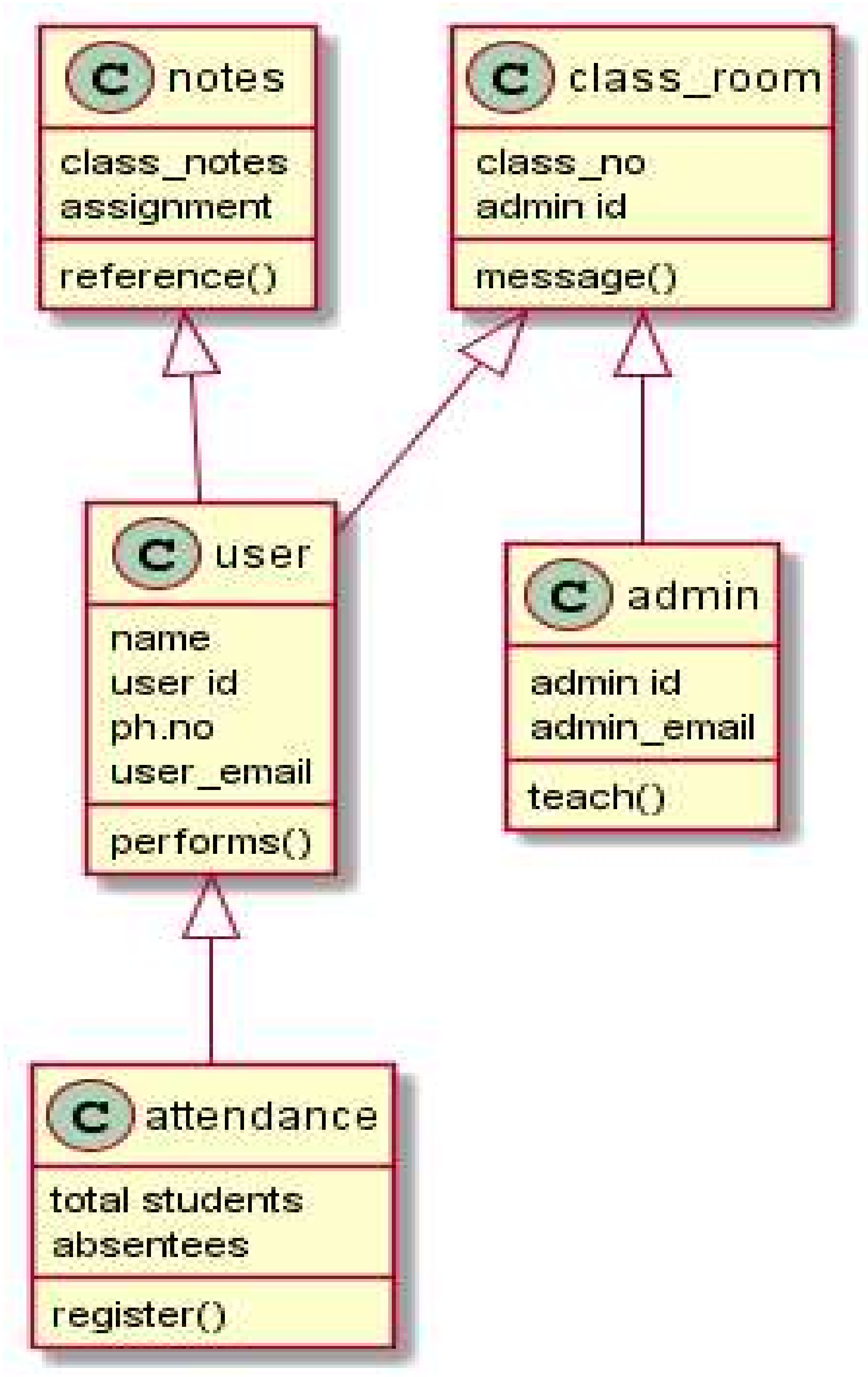
- The top compartment contains the name of the class. It is printed in bold and centered, and the first letter is capitalized.
- The middle compartment contains the attributes of the class. They are left-aligned and the first letter is lowercase.
- The bottom compartment contains the operations the class can execute. They are also left-aligned and the first letter is lowercase.

Program:

```
@startuml
class user
{
        name
        user id
        ph.no
        user_email
        performs()
}
class admin
{
        admin id
        admin_email
        teach()
}
class class_room
{
        class_no
        admin id
        message()
}
```

```
class notes
{
        class_notes
        assignment
        reference()
}
class attendance
{
        total students
        absentees
        register()
}
class_room <|-- user
class_room <|-- admin
notes <|-- user
user <|-- attendance
@enduml
```

OUTPUT


C notes
class_notes
assignment
reference()
C class_room
class_no
admin id
message()
C user
name
user id
ph.no
user_email
performs()
C admin
admin id
admin_email
teach()
C attendance
total students
absentees
register()

Aim:

To write the UML code that generates the Activity Diagram for the ILEARN-A DIGITAL LEARNING ENVIRONMENT.

Description:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Model Language, activity diagrams are intended to model both computational and organizational processes (i.e., workflows), as well as the data flows intersecting with the related activities. Although activity diagrams primarily show the overall flow of control, they can also include elements showing the flow of data between activities through one or more data stores.

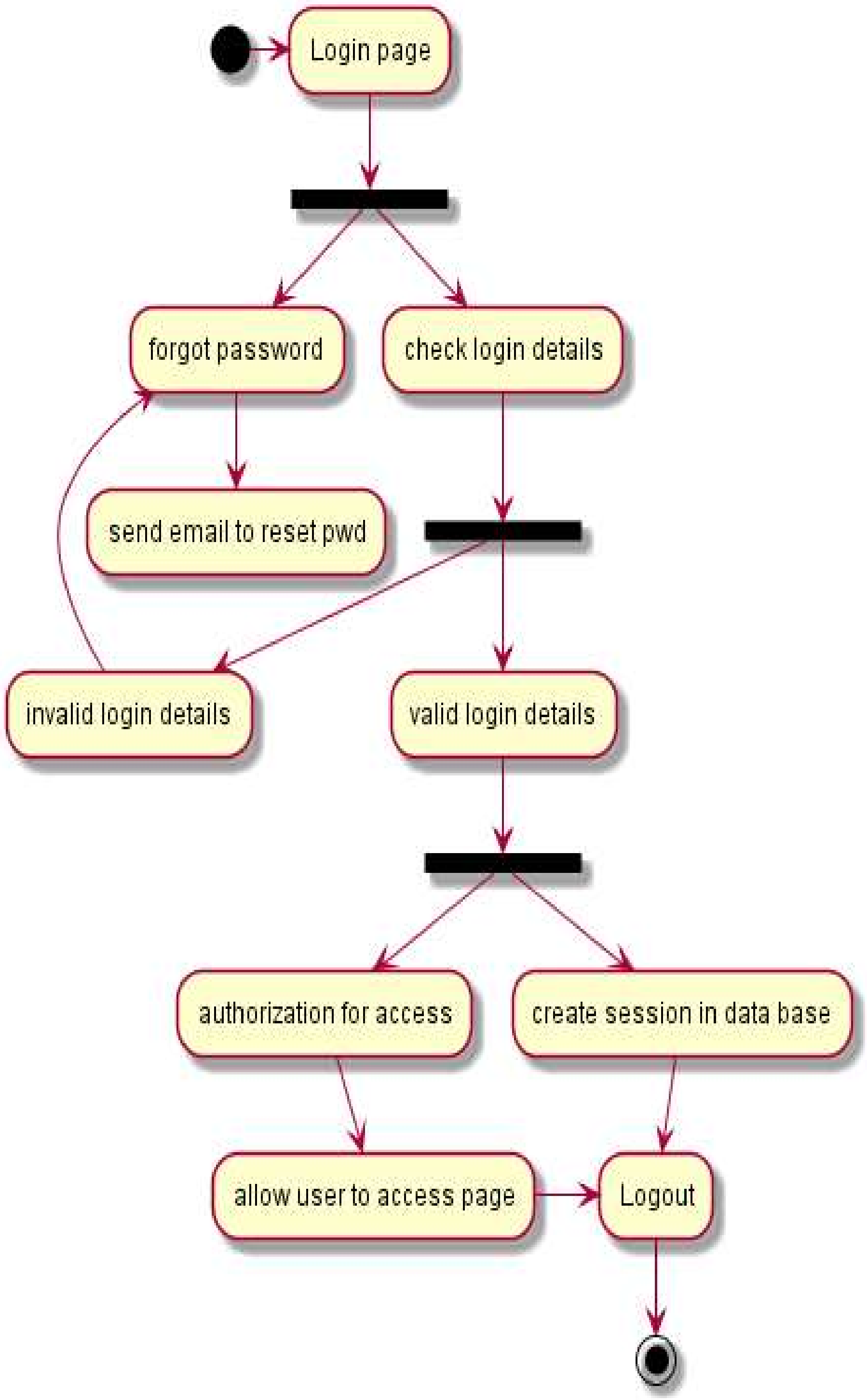
Activity diagrams are constructed from a limited number of shapes, connected with arrows. The most important shape types:

- ellipses represent actions;
- diamonds represent decisions;
- bars represent the start (split) or end (join) of concurrent activities;
- a black circle represents the start (initial node) of the workflow;
- an encircled black circle represents the end (final node).

Program:

```
@startuml
(*) -right->"Login page"
--> ===B1===
-->"forgot password" as F1
===B1=== --> "check login details" as C1
--> ===B3===
-->"invalid login details" as I1
===B3=== -->"valid login details"
--> ===B2===
-->"authorization for access" as A1
===B2=== --> "create session in data base" as C2
F1 -down->"send email to reset pwd"
I1 -up->F1
A1 -down->"allow user to access page" as A2
A2 -right->"Logout" as L1
C2 -down->L1
L1 -down->(*)
@enduml
```

OUTPUT


Login page
forgot password
check login details
send email to reset pwd
invalid login details
valid login details
authorization for access
create session in data base
allow user to access page
Logout

Aim:

To write the UML code that generates the Deployment Diagram for the ILEARN-A DIGITAL LEARNING ENVIRONMENT.

Description:

A **deployment diagram** in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI).

The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers.

There are two types of Nodes:

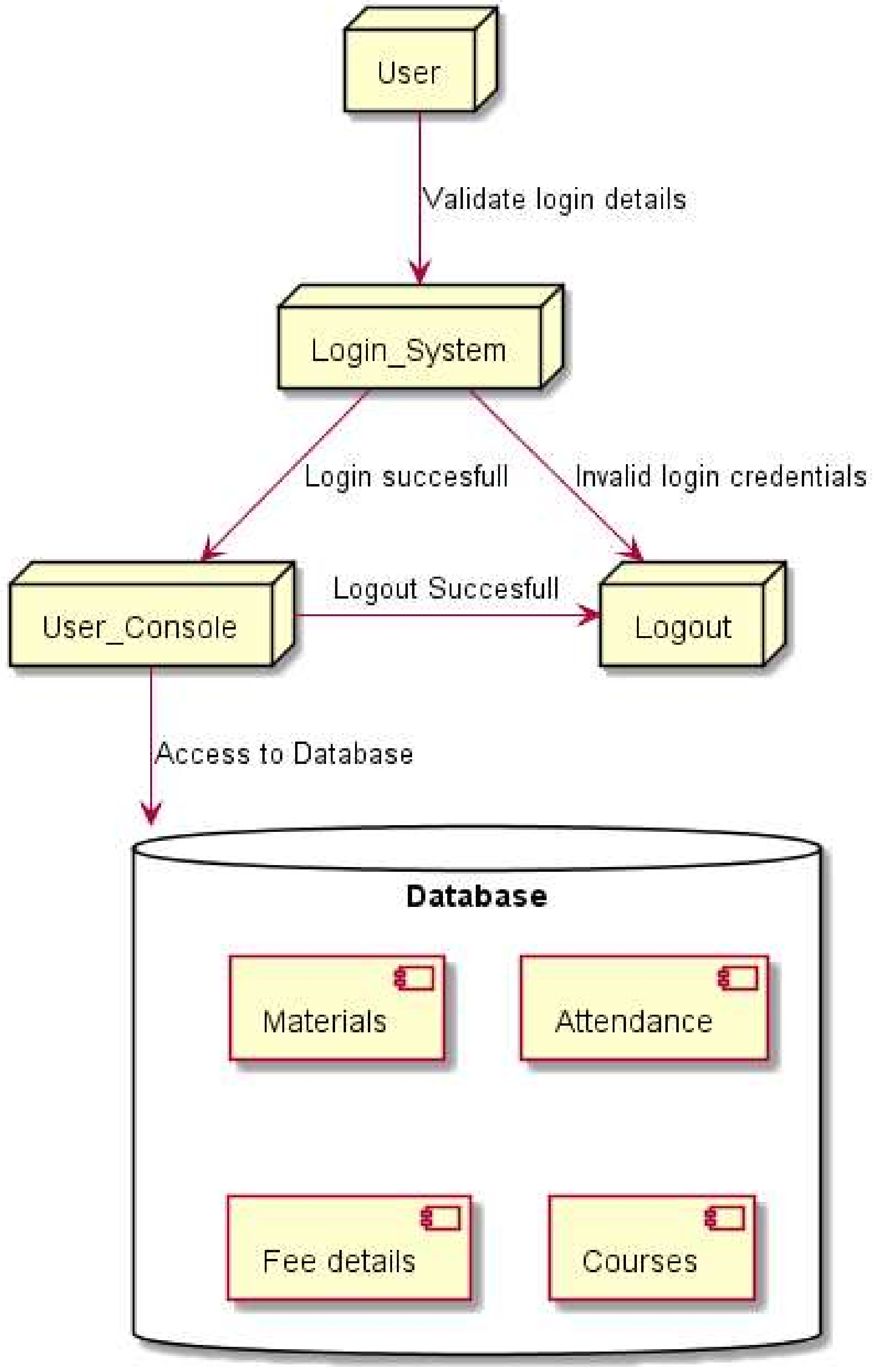
Device Node

Execution Environment Node

Program:

```
@startuml
database "Database"
{
   [Materials]
   [Attendance]
   [Fee details]
   [Courses]
}
node Login_System as ls
node User as user
node Logout as lo
node User_Console as uc

user-->ls : Validate login details
ls-->uc : Login succesfull
uc-->Database : Access to Database
ls-down->lo : Invalid login credentials
uc->lo: Logout Succesfull
@enduml
```

OUTPUT


User
Validate login details
Login_System
Login succesfull
Invalid login credentials
User_Console
Logout Succesfull
Logout
Access to Database
Database
Materials
Attendance
Fee details
Courses