iLearn:The Digital Learning Environment

Submitted by: Chitturi Pavani satya sai-21VV1A1218

May 2, 2023

DEPT. of Information technology
JNTU-GV
Dwarapudi,Vizianagaram, Andhra pradesh 535003

Abstract

1 Introduction

1.1 Purpose

A digital learning environment is a framework in which a set of general-purpose and specially designed tools for learning may beembedded, plus a set of applications that are geared to the needs of the learners using the system. The framework provides general services such as an authentication service, synchronous and asynchronous communication services, and a storage service. The tools included in each version of the environment are chosen by teachers and learners to suit their specific needs. These can be general applications such as spreadsheets, learning management applications such as a Virtual Learning Environment (VLE) to manage homework submission and assessment, games, and simulations.

1.2 Scope

A digital learning environment is a framework in which it is a general-purpose and specifically created learning tools, as well as several applications tailored to the needs of the system's users, may be integrated. The framework offers a variety of general services, including storage, synchronous and asynchronous communication, and authentication. Teachers and students select the resources that are present in each environment version to best meet their individual needs. They can be generic software programmes like spreadsheets or learning management software like a Virtual Learning Environment (VLE) to control assignments, tests, games, and simulations.

1.3 Problem definition

Utility services that provide basic application-independent functionality and that may be used by other services in the system. Utility services are usually developed or adapted specifically for this system. Application services that provide specific applications such as email, conferencing, photo sharing, etc., and access to specific educational content such as scientific films or historical resources. Application services are external services that are either specifically purchased for the system or are available freely over the Internet. Configuration services that are used to adapt to the environment with a specific set of application services and to define how services are shared between students, teachers, and their parents.

1.4 System Overview

The document is intended to read by project managers, developers, tester, users and document writers. The document is organized into 5 parts: Introduction Overall Description Specific Requirements Other Non-Functional Requirements Other Requirements

1.4.1 Definitions, Acronyms and Abbreviations:

Author:
Person submitting an article to be reviewed. In case of multiple authors, this term refers to the principal author, with whom all communication is made.
Database:
Collection of all the information monitored by this system.
Editor:
Person who receives articles, sends articles for review, and makes final judgments for publications.
Field:
A cell within a form.
Historical Society Database:
The existing membership database (also HS database).
Member:
A member of the Historical Society listed in the HS database.
Reader:
Anyone visiting the site to read articles.
CSV:
Comma Separated Values
DNF:
Disjunctive Normal Form
ID :
Identifier
GUI:
Graphtifierical User Interface
HTML:
Hypertext Markup Language

1.5 References

This casestudy is reference from the book written by sommerville.The link of the website is:“ https://iansommerville.com/so
engineering-book/case-studies/ ”
The documentation of the case study is done using OverLeaf .The link of the website is : “https://www.overleaf.com/
”
The source for the UML diagrams is plantUML. The link of the website is : “ https://www.plantuml.com/
”
The website used to design the Usecase diagrams is plantText. It is a tool which designs daigrams of different models .The link of the website is : “ https://www.planttext.com/ ”

2 Overall description

2.1 Product perspective

Ilearning is a digital learning environment which supports the learning of the students . 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.

2.2 Product functions

Utility services that provide basic application-independent functionality and that may be used by other services in the system. Utility services are usually developed or adapted specifically for this system. Application services that provide specific applications such as email, conferencing, photo sharing, etc., and access to specific educational content such as scientific films or historical resources. Application services are external services that are either specifically purchased for the system or are available freely

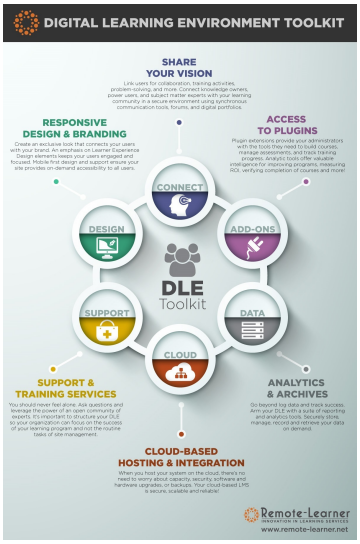

Figure 1: Digital Learning Environment.

over the Internet. Configuration services that are used to adapt the environment with a specific set of application services and to define how services are shared between students, teachers, and their parents

2.3 Uses and Characteristics

A digital learning environment is a framework in which a set of general-purpose and specially designed tools for learning may beembedded, plus a set of applications that are geared to the needs of the learners using the system. The framework provides general services such as an authentication service, synchronous and asynchronous communication services, and a storage service. The tools included in each version of the environment are chosen by teachers and learners to suit their specific needs. These can be general applications such as spreadsheets, learning management applications such as a Virtual Learning Environment (VLE) to manage homework submission and assessment, games, and simulations.

2.4 Constraints

Software design and implementations activities are invariably interleaved .Software design is a creative activity in which you identify software components and their relationships, based on a customer's requirements. Implementation is the process of realizing the design as a program. Sometimes there is a separate design stage, and this design is modeled and documented. Design and Implementations are closely linked and you should normally take implementations issue into the account when developing a design. One of the most important implementations decisions that has to be made at an early stage of software project is whether to build or to buy the application soft-ware.

2.5 Assumption

1. The development of a clear and coherent conceptual framework for e-learning research is hampered by the multitude of different terms that are used to describe the use of digital technologies to support teaching and learning.

2. It is naïve and unrealistic to assume that the use of e-learning, however it is defined, in and of itself will transform students into autonomous and self-directed learners.

3. There are substantial gaps in e-learning research, particularly at the institutional and system-wide level.

4. Both e-learning research and practice face inherent challenges. We need to fully understand the benefits and limitations of implementing e-learning, in relation to costs and learning effectiveness, and the potential impact on access and the ability to improve or worsen the digital divide.

2.6 Dependencies

Digital learning created fruitful opportunities for educational institutions; however, there were some challenges relating to technology, courses, instructors and learners (Händel et al., Citation2020; Shehzadi et al., Citation2020). The limitations of technology platforms, the quality of the internet, learner-teacher interaction, and the limited training of teachers and learners regarding the online learning system influenced the effectiveness of learning in a digital environment. In addition, the ability to adapt to immediate changes in the new situation would affect the future of online learning (Dinh Nguyen, Citation2020).

3 Specific requirements

3.1 External interfaces

3.1.1 Interfaces

The logical characteristics of each interface between the software product and its users. This includes the required screen formats, page or window layouts, the content of any reports or menus, or availability of programmable function keys necessary to accomplish the software requirements. Interface optimization methods and the people responsible for them. A simple list of dos and don'ts will do the trick.

3.1.2 Software interfaces

The application allows import a structured MS Word document via HTML data format. The application allows populating a MS Word document with project data via HTML data format. The application allows import / export a list of requirements from / to MS Excel sheet via CSV data format.

3.2 Functions

1. Planning and management of lessons and other activities.
2. Access to digital learning materials: texts, videos, images, podcasts, etc.
3. Group discussions and one-on-one chats with a teacher.
4. Submitting homework and other tasks; Grading, tracking students' performance, providing feedback.
5. Holding live lessons.

Such a computer-aided approach reveals new opportunities for students, teachers, and administration as well as brings some challenges.

3.3 Performance requirements

Startup Time
The application should display the opened document within 5s after it is started.
Edit Response Time
The application should display updated values within 1s after user triggers the edit oper- ation.
Document Size
The application shall allow users to open documents up to 10000 objects and 100 file attachments with total size up to 100MB.

3.4 Logical database requirements

A requirements list for your learning management system database model might look like the one detailed below. This list must have the approval of the project stakeholders, so you can work safely. You're less likely to hear that phrase that causes us software developers terror: "This is not what I asked for". Remember that, among the many things that A DATABASE DESIGNER DOES, an important one is to correctly interpret what users and stakeholders really need.
The requirements list sets out the conditions, constraints, and criteria on which you will base your design. For a learning management system, these might include:

- Course information includes a course code (unique identifier), a course name or title, start/end dates, course category, course abstract, and bibliography. The course category must belong to a list of predetermined categories.
- Students' registrations must contain their names and email addresses. Other data, such as date of birth and telephone number, can optionally be registered.
- Teachers' registration must contain their names and email addresses. Optionally, their telephone numbers can also be included.
- A teacher can teach more than one course, and more than one teacher can teach one course. Each student can enroll in one or more courses. When a student enrolls in a course, the date of enrollment is registered.
- A student's enrollment in a course can be canceled. In that case, the reason for cancellation is registered.
- The database should keep a record of each time a student takes a test, storing the score obtained.

3.5 Design constraints

Software design and implementations activities are invariably interleaved .Software design is a creative activity in which you identify software components and their relationships, based on a customer's requirements. Implementation is the process of realizing the design as a program. Sometimes there is a separate design stage, and this design is modeled and documented. Design and Implementations are closely linked and you should normally take implementations issue into the account when developing a design. One of the most important implementations decisions that has to be made at an early stage of software project is whether to build or to buy the application soft-ware.

3.6 Key features

Replicate, Scale and Sustain (RSS)
Firstly, begin building the exosystem with core features that can be tested on a smaller scale. The next step is to gradually add more functions and components, so it is immediately upgradeable as and when required and manageable for architects and users as it develops. Of course, an exosystem is not a static organism. It needs to be continually evaluated and the different components scaled up to an organizational level as they are tested and found dependable. Finally, ensure growth is sustainable by regularly reviewing and refining the components of the exosystem.

Learn and Relearn
A major advantage of this component model is that each of these components are in active use in other exosystems. Every component may have already gone through its own 'trial-by-fire' at cloud-scale by serving many thousands or millions of concurrent users in other systems.

Adhere to Open Technology Conventions
All components or applications are products of different IT organizations, communities technologies. Ensure those selected and used adhere to a set of well-tested, robust, and scalable software service patterns.

Keeping It Cost Efficient
Education is expensive to deliver. An advantage of the motherboard concept is the ability to select cost effective components from different sources. Being tied into a single large vendor removes this flexibility and choice. Choosing components for different but interoperable sources, creates an agile, up-to-date system that remains economically advantageous.

Being Data Smart
Ensure your exosystem motherboard is able to effectively and securely collate, standardize and secure your data at all points and at all times. Data that is generated by different members of your exosystem, should be able to be standardized and stored by the exosystem, but at the same time be accessible to different levels of management, so analytics can be used effectively to drive improvements and efficiency.

4 Validation check

4.1 Validity check

Just like with traditional training program, it is essential for learners to be rewarded with a valid qualification/certificate that is going to be recognized by both industrial and academic bodies. If an online course leads to a University degree or an industry qualification that is an equivalent of its face-to-face delivered counterpart, its validity is unlikely to be questioned by anyone. For example, in Australia we have many online providers of training courses. The ones offering accredited degrees and certifications are by far the most popular ones. Just like in a traditional learning environment, an online MBA from a country's leading University is likely to attract a lot of interest among the learners but websites that offer to teach "business administration skills", but have no recognized awards to offer are usually doomed to fail. Obviously, there are exceptions from this "rule"... but they only confirm the rule!

4.2 Consistency check

- Create and post 3 simple classroom rules and follow them at all times. If a rule lends itself to be broken at any point, it doesn't need to be a rule.
- Brainstorm and post reasonable consequences to your rules. The operative word here is reasonable. Threatening to paddle a student for disrespect or assign suspension for eating in class is too harsh. The punishment needs to fit the crime.
- Follow the rules for ALL students in ALL classes. Your 6th period seniors will give you weeks of grief if they find out you allowed the 1st period freshmen eat in class but you strictly forbid them to eat in class.
- Adhere to the consequences for ALL students in ALL classes. Word will spread like wildfire if a student learns you called his mother for an infraction, but did not call another student's parent for the same infraction. Failing to do this will only weaken a teacher's credibility, which will only hurt that teacher as he tries to gain the trust of his students.

4.3 Realism check

Realism is the doctrine that is associated with the study of the world we live in. It is a philosophy away from the world of ideas or spiritual things. In Realism the word 'real' denotes actual or the existing. It indicates those things or events which exist in the world in its own right. It opposes the thing or event which is imaginary or fictitious. It holds the view that knowledge acquired through senses is true and what we observe and perceive through our own senses is real and the true entity of the world. It says that physical world is objective and factual whereas personal feelings and desires are subjective and secondary. That is why this philosophy is also known as objectivism. Aristotle is generally regarded as the father of Realism. John Locke, Erasmas, Rabelias, Comenius, Bertrand Russell, Francis Bacon, Milton are the chief protagonists of Realism.

4.4 Verifiability check

Unarguably, technology has become an irreversible force driving the transformation of teaching and learning practices. Cloud computing, learning analytics, big data, and artificial intelligence are being adopted in today's teaching and learning, though to different extents. For over a decade, educational researchers have been exploring how different innovative means could be integrated into traditional learning in order to enrich learning experience and enhance learning effectiveness. Enabled by various pedagogical and technological innovations, brand new learning environments can be created to optimize learners' ability to learn. They are collectively referred as the commonly known "smart learning environments" which can best delineate the future learning environments. Embracing a variety of concepts, including but not limited to flexible learning, personalized learning, mobile learning, adaptive learning, and blended learning, for obvious reasons, there are no one single form of smart learning environments. The concepts, and even definitions, of smart learning environments have continuously emerging.

5 Validation techniques

5.1 Requirement reviews

- Interoperability and integration: shares data across the platform.
- Personalization: personalized learning activities for individual users.
- Analytics, advising, and learning assessment: tools more measuring student learning outcomes Collaboration: programming that makes it easy to share and communicate with others.
- Accessibility and universal design: optimized to be used and accessed by students across various devices.

5.2 Prototyping techniques

A prototype is a mini version of the eLearning course. When selecting slides for a prototype, it is best to select a few that are representative of the various types used in the course. For instance, a functional prototype would include:

1. Instructional design strategies with interactions.
2. Classroom interactions adapted to eLearning Attention grabbers.
3. Different ways of presenting the learning objectives.
4. Assessments that are mapped to the content.

5.3 Testcase generation

Specification-based Test Case Generation Techniques
The specification requirement document can be used as a basis for output checking, significantly reducing one of the major costs of testing. Specifications can also be analyzed with respect to their testability (Memon et al., 1999). The process of generating tests from the specifications will often help the test engineer discover problems with the specifications themselves. If this step is done early, the problems can be eliminated early, saving time and resources. Generating tests during development also allows testing activities to be shifted to an earlier part of the development process, allowing for more effective planning and utilization of resources. Test generation can be independent of any particular implementation of the specifications (Offutt et al., 1999).

Sketch Diagram-based Test Case Generation Techniques
Heumann (2001) presented how using use cases to generate test cases can help launch the testing process early in the development lifecycle and also help with testing methodology. In a software development project, use cases define system software requirements. Use case development begins early

on, so real use cases for key product functionality are available in early iterations. According to the Rational Unified Process (RUP), a use case is used to fully describe a sequence of actions performed by a system to provide an observable result of value to a person or another system using the product under development. Use cases tell the customer what to expect, the developer what to code, the technical writer what to document and the tester what to test. He proposed three-step process to generate test cases from a fully detailed use case: (1) for each use case, generate a full set of use-case scenarios (2) for each scenario, identify at least one test case and the conditions that will make it execute and (3) for each test case, identify the data values with which to test.

Experiment-2
To design all UML Diagrams

Aim:
To design Class UML diagram for Digital Learning Environment.

Description:

A digital learning environment is a system that provides online learning resources and tools for students and teachers. Here is a description of a class diagram for a digital learning environment:

User: This class represents the users of the system, which can include students, teachers, and administrators.

Course: This class represents a course in the digital learning environment. It includes information such as the course name, description, and course materials.

Lesson: This class represents a lesson within a course. It includes information such as the lesson name, description, and lesson materials.

Assignment: This class represents an assignment that is given to students as part of a lesson. It includes information such as the assignment name, description, and due date.

Grade: This class represents the grades given to students for their assignments and overall performance in the course.

Discussion Forum: This class represents the discussion forum where students can interact with each other and with their teachers to discuss course materials and assignments.

Test module: This class represents the assessment of student learning and includes information such as tests, exams, and quizzes.

Learning Resource: This class represents the learning resources available to students, such as videos, articles, and textbooks.

Analytics: This class represents the analytics tools used to track student progress and performance, as well as the overall effectiveness of the digital learning environment.

Permission: This class represents the authentication system used to verify the identity of users and protect the system from unauthorized access.

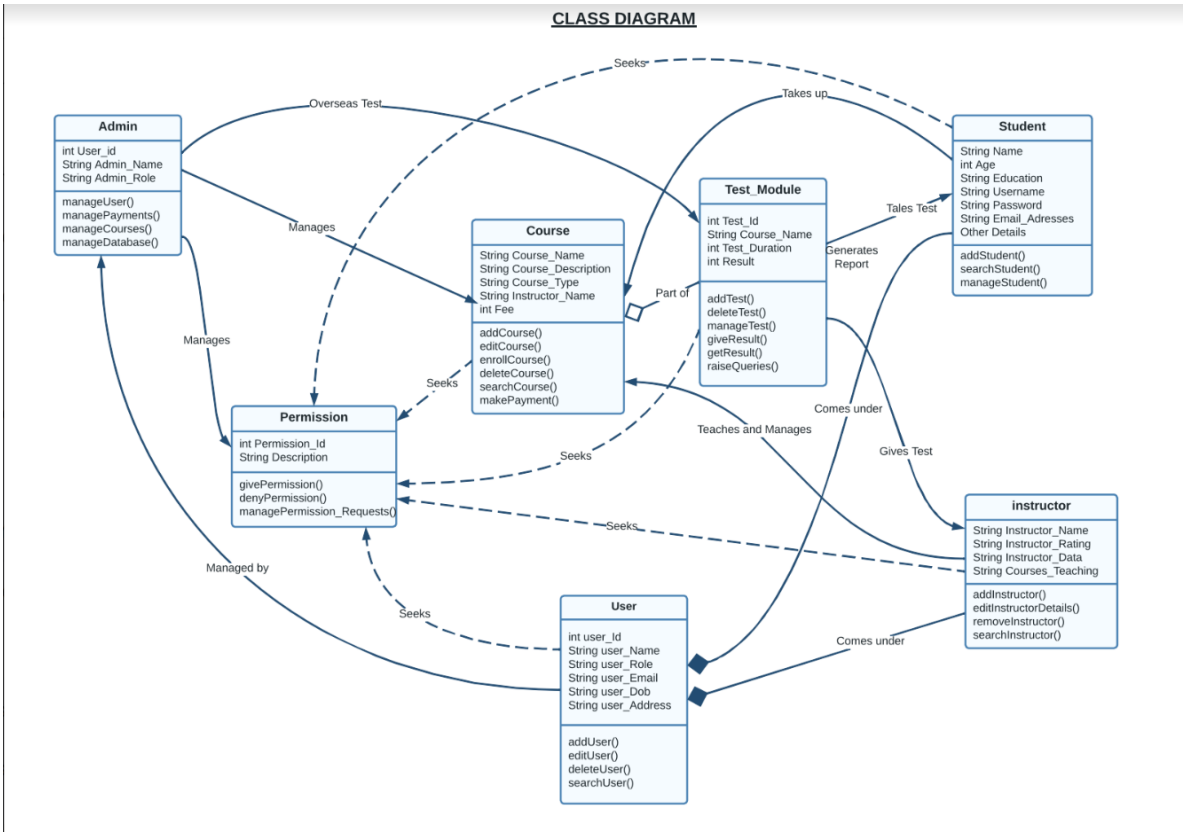
CLASS DIAGRAM
Admin
int User_id
String Admin_Name
String Admin_Role
manageUser()
managePayments()
manageCourses()
manageDatabase()
Student
String Name
int Age
String Education
String Username
String Password
String Email_Adresses
Other Details
addStudent()
searchStudent()
manageStudent()
Course
String Course_Name
String Course_Description
String Course_Type
String Instructor_Name
int Fee
addCourse()
editCourse()
enrollCourse()
deleteCourse()
searchCourse()
makePayment()
Test_Module
int Test_Id
String Course_Name
int Test_Duration
int Result
addTest()
deleteTest()
manageTest()
giveResult()
getResult()
raiseQueries()
Permission
int Permission_Id
String Description
givePermission()
denyPermission()
managePermission_Requests()
instructor
String Instructor_Name
String Instructor_Rating
String Instructor_Data
String Courses_Teaching
addInstructor()
editInstructorDetails()
removeInstructor()
searchInstructor()
User
int user_Id
String user_Name
String user_Role
String user_Email
String user_Dob
String user_Address
addUser()
editUser()
deleteUser()
searchUser()
Seeks
Takes up
Overseas Test
Manages
Tales Test
Generates Report
Part of
Manages
Seeks
Comes under
Teaches and Manages
Seeks
Gives Test
Seeks
Managed by
Seeks
Comes under

Figure 2:

.

Aim:

To Design Usecase UML diagram for Digital Learning Environment.

Description:

A use case diagram for a digital learning environment would describe the different ways that users interact with the system to achieve their goals. Here is a description of a use case diagram for a digital learning environment:

Login: Users can log into the system using their username and password.

View Courses: Users can view the list of courses available in the digital learning environment.

Enroll in Course: Students can enroll in a course by selecting it from the list of courses and clicking on the enroll button.

View Lessons: Users can view the list of lessons within a course.

View Learning Resources: Users can view the learning resources associated with a lesson.

Submit Assignments: Students can submit assignments for a particular lesson by uploading them to the system.

Take Quizzes: Students can take quizzes associated with a lesson.

Participate in Discussions: Users can participate in discussions related to a particular lesson or course.

View Grades: Students can view their grades for assignments, quizzes, and overall performance in the course.

Create Course: Teachers can create a new course by providing the course name, description, and course materials.

Create Lesson: Teachers can create a new lesson within a course by providing the lesson name, description, and lesson materials.

Manage Grades: Teachers can manage the grades of students in their course.

Analyze Performance: Administrators can analyze the performance of students and courses using analytics tools.

Manage User Accounts: Administrators can manage user accounts by creating, modifying, or deleting them.

Send Notifications: The system can send notifications to users to inform them of new course materials, assignment due dates, and other important information.

.

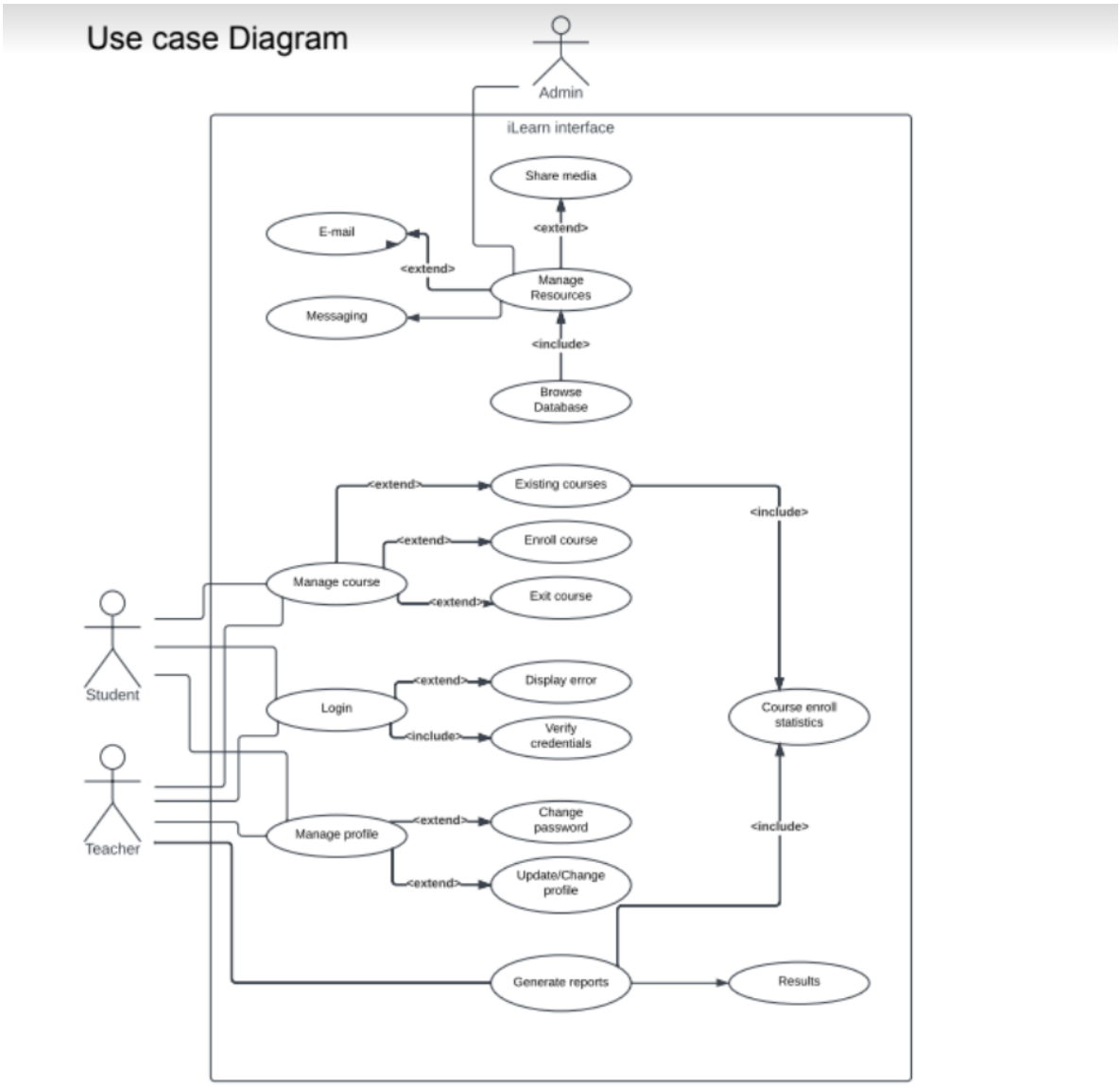
Use case Diagram
Admin
iLearn interface
Share media
<extend>
E-mail
<extend>
Manage Resources
Messaging
<include>
Browse Database
<extend>
Existing courses
<include>
<extend>
Enroll course
Manage course
<extend>
Exit course
Student
<extend>
Display error
Login
Course enroll statistics
<include>
Verify credentials
Teacher
<extend>
Change password
Manage profile
<include>
<extend>
Update/Change profile
Generate reports
Results

Figure 3:

Aim:

To design Object UML diagram for Digital Learning Environment

Description:

An object UML diagram for a digital learning environment would provide a snapshot of the system at a particular point in time, showing the objects and their relationships. Here is a description of an object UML diagram for a digital learning environment:

User Object: Represents a user of the system, with attributes such as name, email, and password.

Course Object: Represents a course in the system, with attributes such as name, description, and course materials.

Lesson Object: Represents a lesson within a course, with attributes such as name, description, and lesson materials.

Assignment Object: Represents an assignment for a lesson, with attributes such as name, description, and due date.

Discussion Forum Object: Represents a discussion forum for a course, with attributes such as topic, posts, and comments.

Quiz Object: Represents a quiz for a lesson, with attributes such as name, description, and questions.

Assessment Object: Represents an assessment of student learning, such as a test or exam, with attributes such as name, description, and questions.

Learning Resource Object: Represents a learning resource available to students, such as a video or article, with attributes such as name, description, and URL.

Authentication Object: Represents the authentication system used to verify the identity of users and protect the system from unauthorized access.

Analytics Object: Represents the analytics tools used to track student progress and performance, with attributes such as data, reports, and charts. .

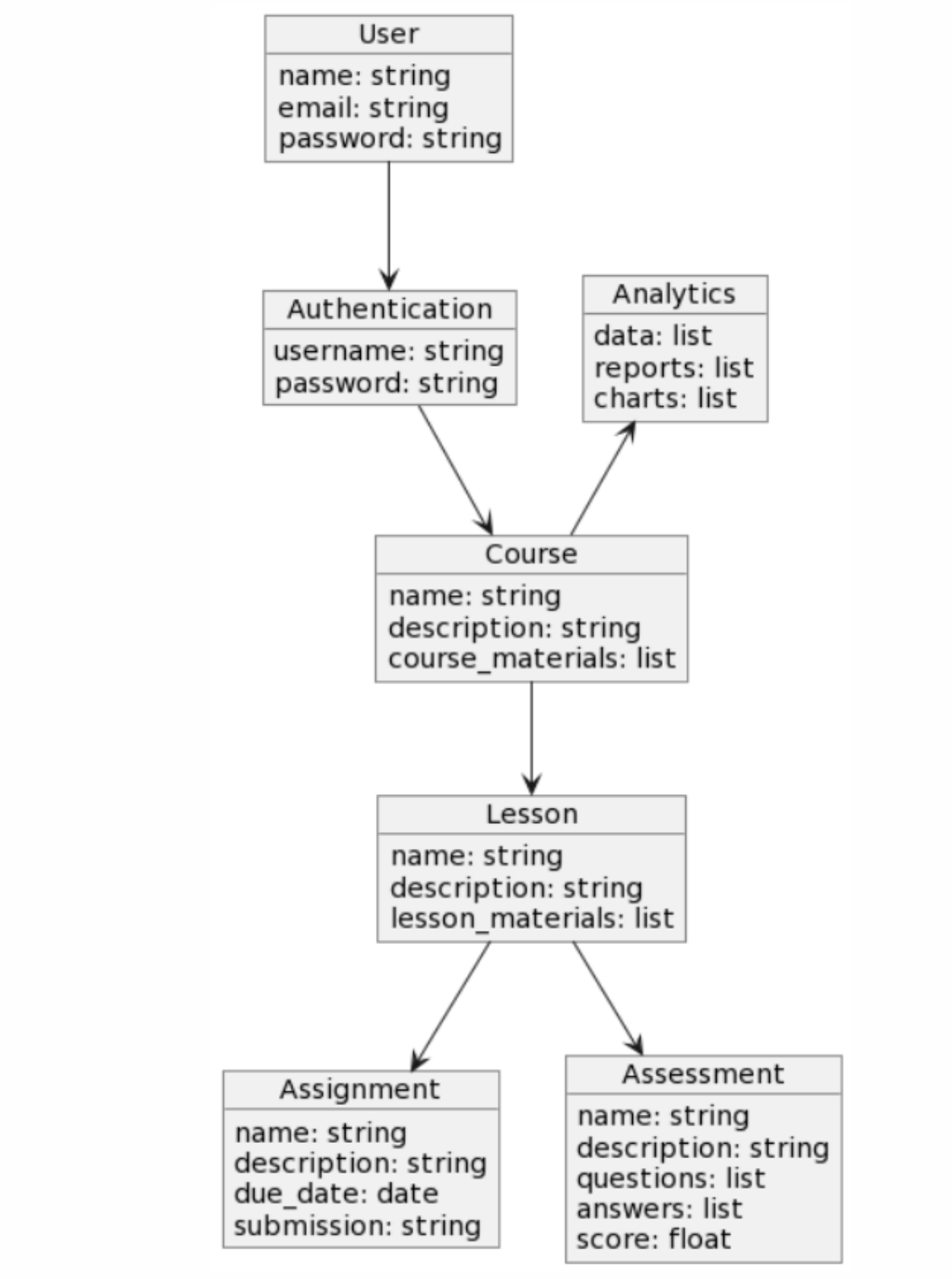
User
name: string
email: string
password: string
Authentication
username: string
password: string
Analytics
data: list
reports: list
charts: list
Course
name: string
description: string
course_materials: list
Lesson
name: string
description: string
lesson_materials: list
Assignment
name: string
description: string
due_date: date
submission: string
Assessment
name: string
description: string
questions: list
answers: list
score: float

Figure 4:

Aim:

To design Sequence UML diagram for Digital Learning Environment.

Description:

A sequence UML diagram for a digital learning environment would show the sequence of events that occur between objects over time, providing a visual representation of how different objects interact with each other. Here is a description of a sequence UML diagram for a digital learning environment:

User Login Sequence: The user logs in by entering their username and password, which are authenticated by the system.

View Courses Sequence: The user views the list of available courses, which are retrieved from the system's database.

Enroll in Course Sequence: The user enrolls in a course by selecting it from the list of courses and clicking on the enroll button. This action is recorded in the system's database.

View Lessons Sequence: The user views the list of lessons within a course, which are retrieved from the system's database.

View Learning Resources Sequence: The user views the learning resources associated with a lesson, which are retrieved from the system's database.

Submit Assignment Sequence: The user submits an assignment for a particular lesson by uploading it to the system, which is stored in the system's database.

Take Quiz Sequence: The user takes a quiz associated with a lesson, answering questions and receiving feedback, which is stored in the system's database.

Participate in Discussion Sequence: The user participates in discussions related to a particular lesson or course by posting comments, which are stored in the system's database.

View Grades Sequence: The user views their grades for assignments, quizzes, and overall performance in the course, which are retrieved from the system's database.

Create Course Sequence: The teacher creates a new course by providing the course name, description, and course materials. This action is recorded in the system's database.

Create Lesson Sequence: The teacher creates a new lesson within a course by providing the lesson name, description, and lesson materials. This action is recorded in the system's database.

Manage Grades Sequence: The teacher manages the grades of students in their course by entering scores, which are recorded in the system's database.

Analyze Performance Sequence: The administrator analyzes the performance of students and courses using analytics tools, retrieving data from the system's database.

Manage User Accounts Sequence: The administrator manages user accounts by creating, modifying, or deleting them, which is recorded in the system's database. .

Sequence Diagrams

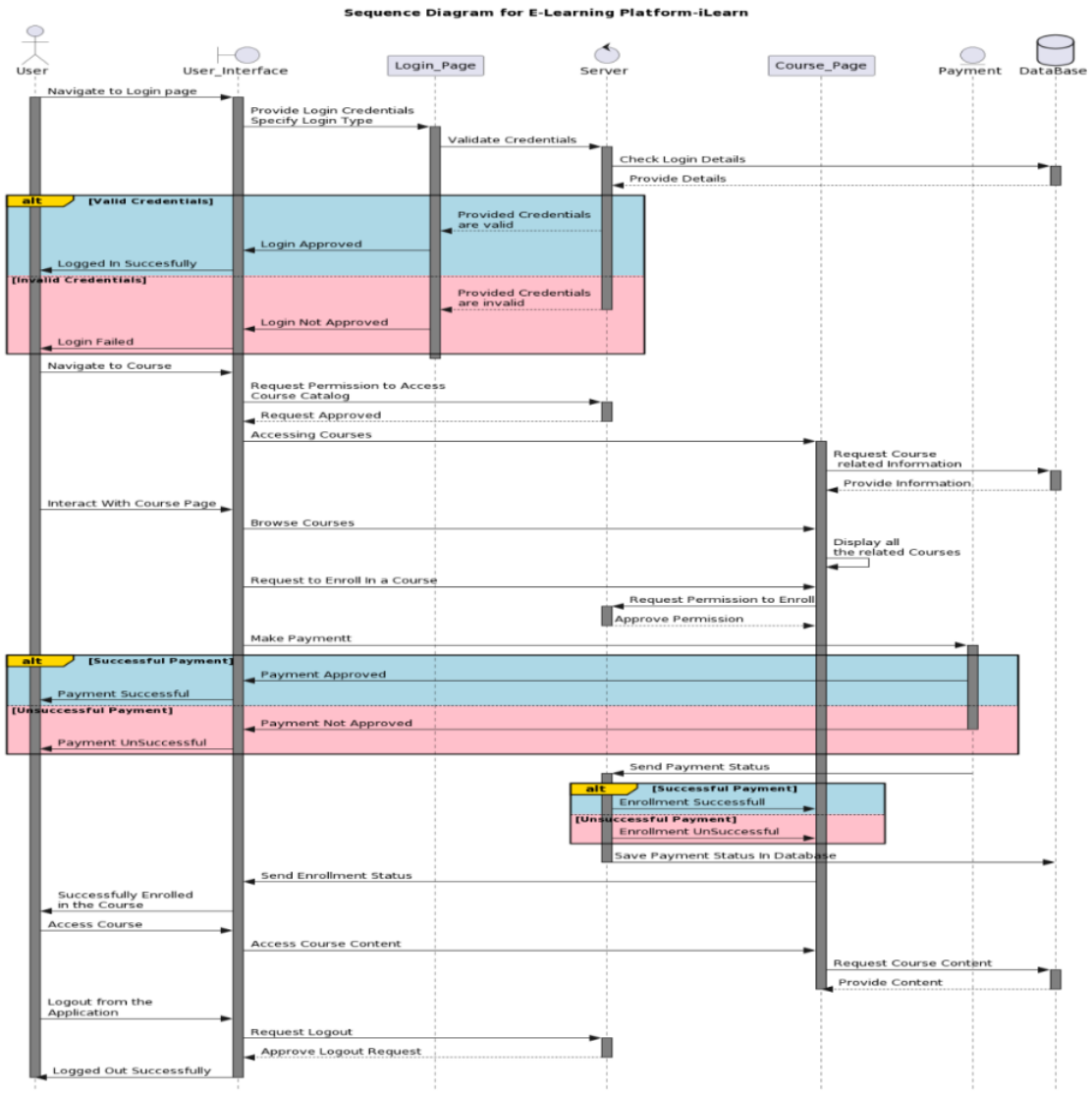

Figure 5:

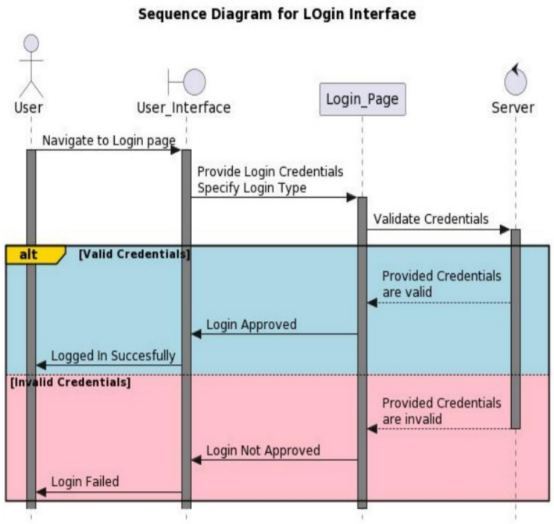

Figure 6:

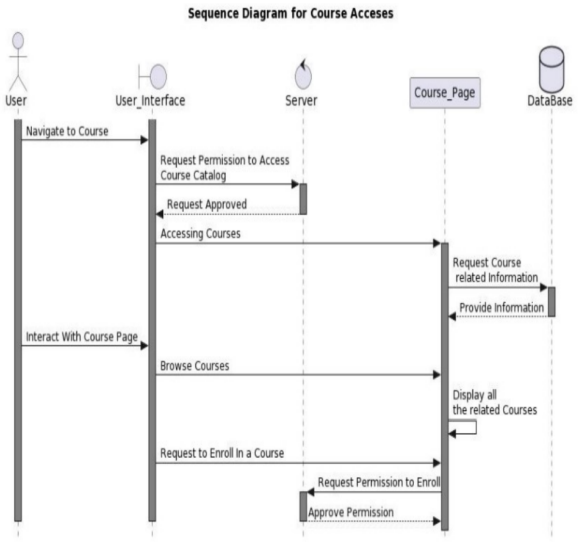

Figure 7:

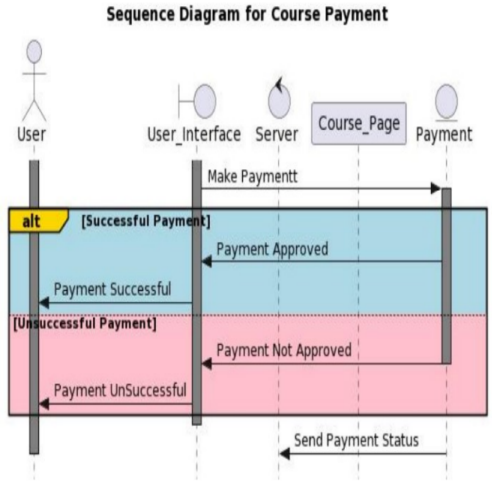

Figure 8:

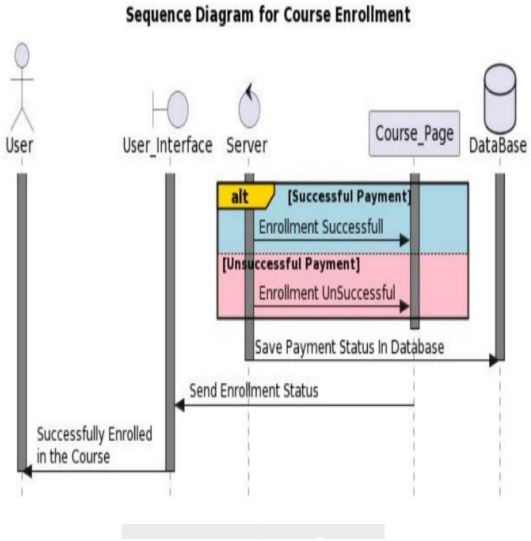

Figure 9:

Aim:

To design Collaboration UML diagram for Digital Learning Environment.

Description:

A collaboration UML diagram for a digital learning environment would show how different objects work together to achieve a common goal. Here is a description of a collaboration UML diagram for a digital learning environment:

User-Course Collaboration: The user enrolls in a course by selecting it from the list of available

courses. The course object is then created, and the user object is associated with the course object. The course object retrieves course materials from the system's database and provides them to the user.

Lesson-Course Collaboration: The user selects a lesson from the list of lessons within a course. The lesson object is then created, and the lesson object is associated with the course object. The lesson object retrieves lesson materials from the system's database and provides them to the user.

Assignment-Lesson Collaboration: The user submits an assignment for a particular lesson by uploading it to the system. The assignment object is then created, and the assignment object is associated with the lesson object. The assignment object is then stored in the system's database.

Quiz-Lesson Collaboration: The user takes a quiz associated with a lesson, answering questions and receiving feedback. The quiz object is then created, and the quiz object is associated with the lesson object. The quiz object is then stored in the system's database.

Discussion-Lesson Collaboration: The user participates in discussions related to a particular lesson or course by posting comments. The discussion object is then created, and the discussion object is associated with the lesson or course object. The discussion object is then stored in the system's database.

Analytics Collaboration: The administrator analyzes the performance of students and courses using analytics tools. The analytics object retrieves data from the system's database and provides reports and charts to the administrator.

Authentication Collaboration: The user logs in by entering their username and password, which are authenticated by the authentication object. If the user's credentials are valid, the authentication object creates a session object and associates it with the user object.

The sequence UML diagram provides a clear and visual representation of the sequence of events that occur within a digital learning environment, showing how different objects interact with each other over time.

Collaboration Diagrams

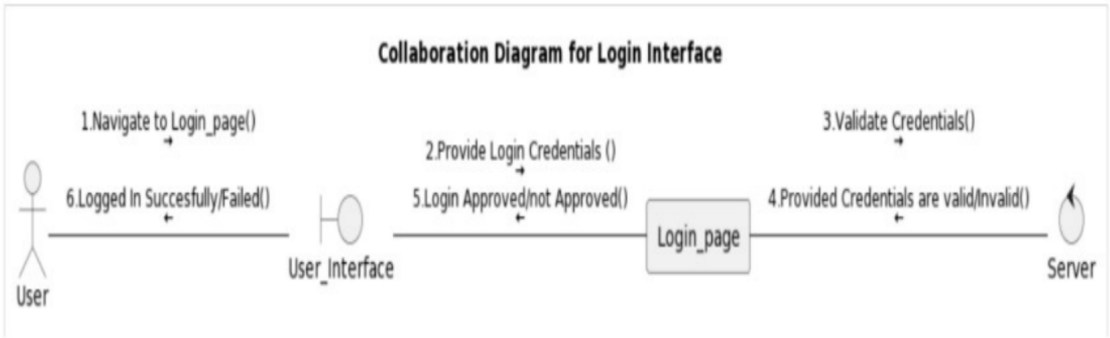

Figure 10:

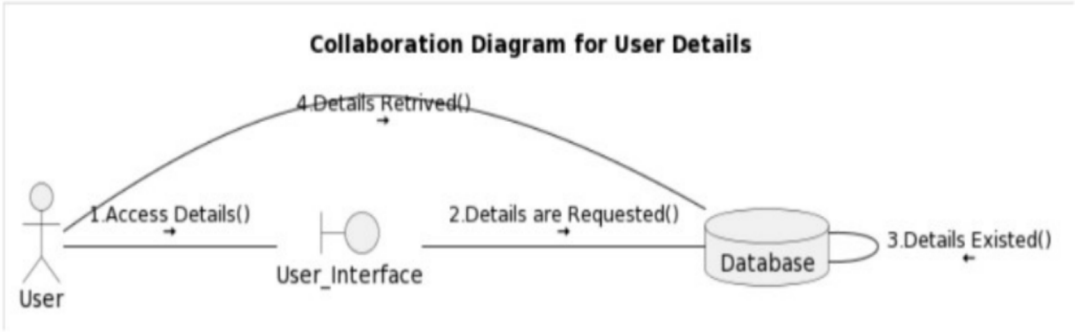

Figure 11:

.

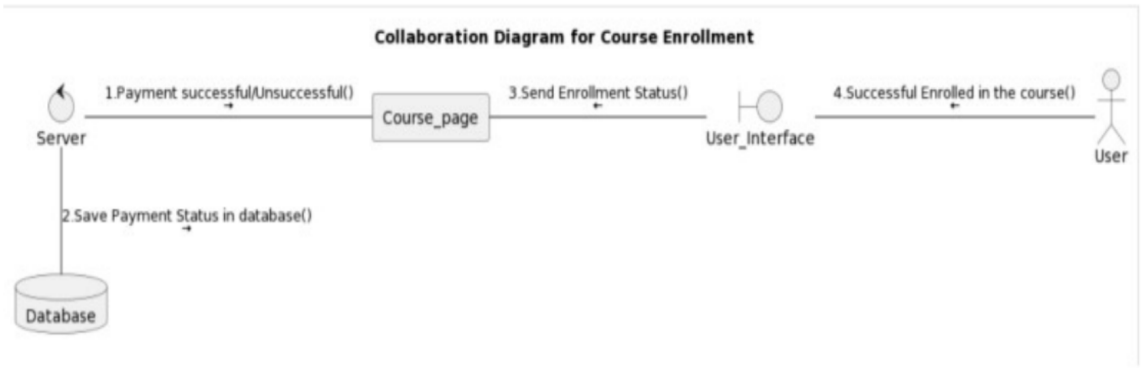

Figure 12:

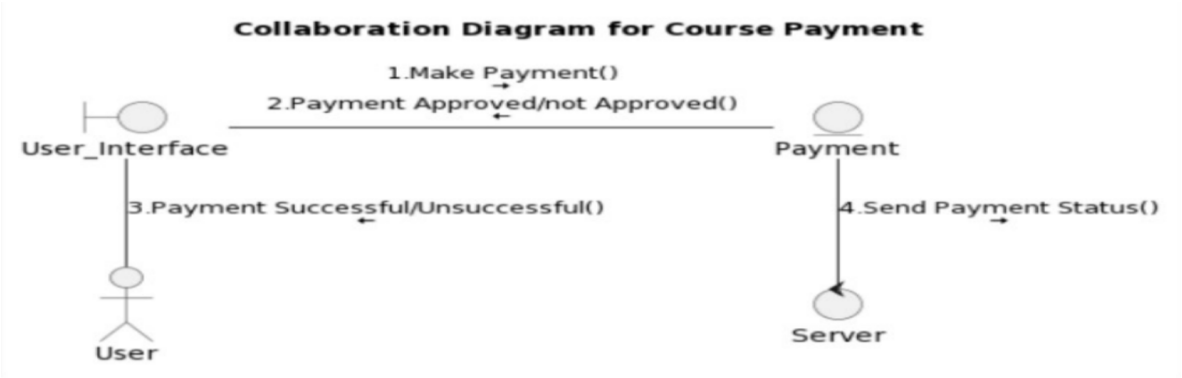

Figure 13:

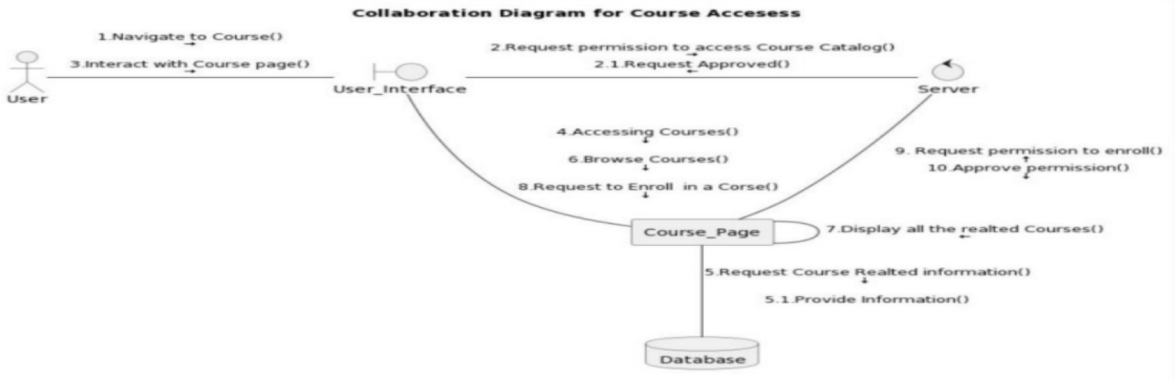

Figure 14:

Aim:

To design Activity UML diagram for Digital Learning Environment.

Description:

An activity UML diagram for a digital learning environment can provide a visual representation of the different states and activities involved in the learning process. Here is a description of an activity/state UML diagram for a digital learning environment:

User Login Activity: The user initiates the login process by entering their username and password. The system checks the user's credentials and either allows them to proceed or denies access.

Course Enrollment Activity: The user selects a course from a list of available courses and submits an enrollment request. The system processes the request and either approves or denies the enrollment.

Lesson Viewing Activity: The user selects a lesson from a course and views the lesson material. The system provides the lesson material and records the user's progress.

Assignment Submission Activity: The user completes an assignment and submits it to the system. The system checks the assignment for completeness and records the submission.

Quiz Taking Activity: The user takes a quiz associated with a lesson and answers the questions. The system provides feedback on the user's answers and records the quiz results.

Discussion Participation Activity: The user participates in a discussion related to a lesson or course by posting comments. The system records the user's comments and displays them to other users.

Assessment Activity: The system evaluates the user's performance based on their participation, assignment submissions, quiz results, and other factors. The system generates a report and provides feedback to the user.

The activity UML diagram provides a clear and visual representation of the different activities and states involved in the learning process. It shows the user's interaction with the system and the system's response to user actions. This UML diagram can help identify potential issues and improve the system's functionality and usability.

.

Activity diagram:

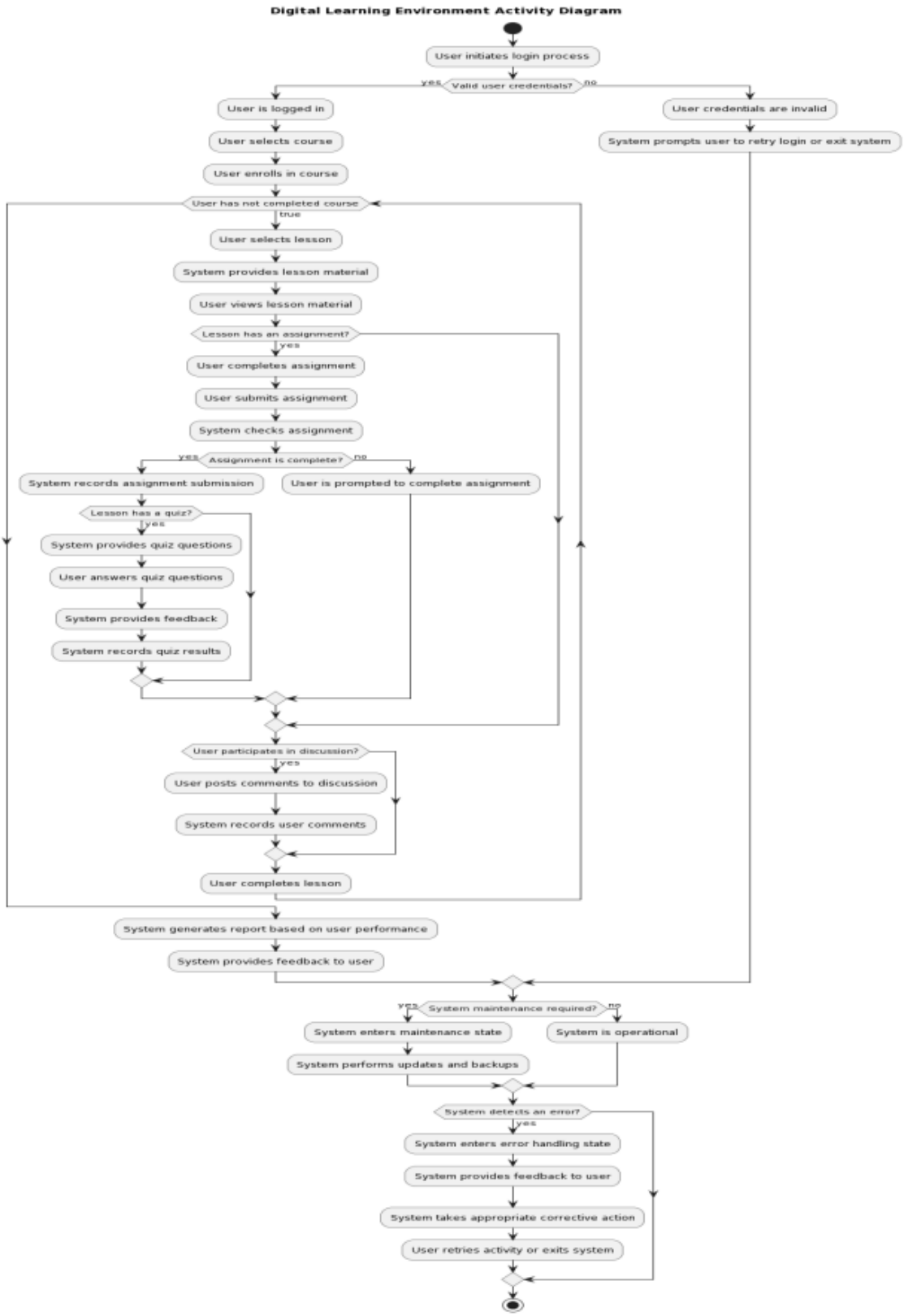

Figure 15:

Aim:

To design Component UML diagram for Digital Learning Environment.

Description:

The component UML diagram for a Digital Learning Environment represents the various software components that make up the system, and how they interact with each other. These components could include the database, web server, user interface, authentication module, analytics module, etc. The diagram shows the relationships between these components and how they work together to provide the functionality of the system. Here are the components and their brief description for a Digital Learning Environment:

User Interface: This component represents the user interface that the user interacts with to access the digital learning environment. It could include web pages, mobile apps, or desktop software.

Web Server: This component is responsible for receiving user requests and sending responses to the user interface. It could include Apache, IIS, or any other web server software.

Application Server: This component is responsible for managing the business logic of the digital learning environment. It could include Node.js, Ruby on Rails, or any other server-side software.

Database: This component is responsible for storing and managing the data related to the digital learning environment. It could include MySQL, MongoDB, or any other database software.

Authentication Module: This component is responsible for authenticating the user and managing their access to the system. It could include OAuth, LDAP, or any other authentication software.

Analytics Module: This component is responsible for collecting and analyzing data related to the user's performance in the digital learning environment. It could include Google Analytics, Mixpanel, or any other analytics software.

Content Management System: This component is responsible for managing the digital content used in the learning environment. It could include WordPress, Joomla, or any other CMS software.

Payment Gateway: This component is responsible for handling payments made by the user for accessing premium content in the digital learning environment. It could include PayPal, Stripe, or any other payment gateway software.

The component UML diagram for a Digital Learning Environment shows how these components are connected to each other, and how they work together to provide the functionality of the system. It helps in understanding the architecture of the system and how the different components interact with each other. .

Component diagram:

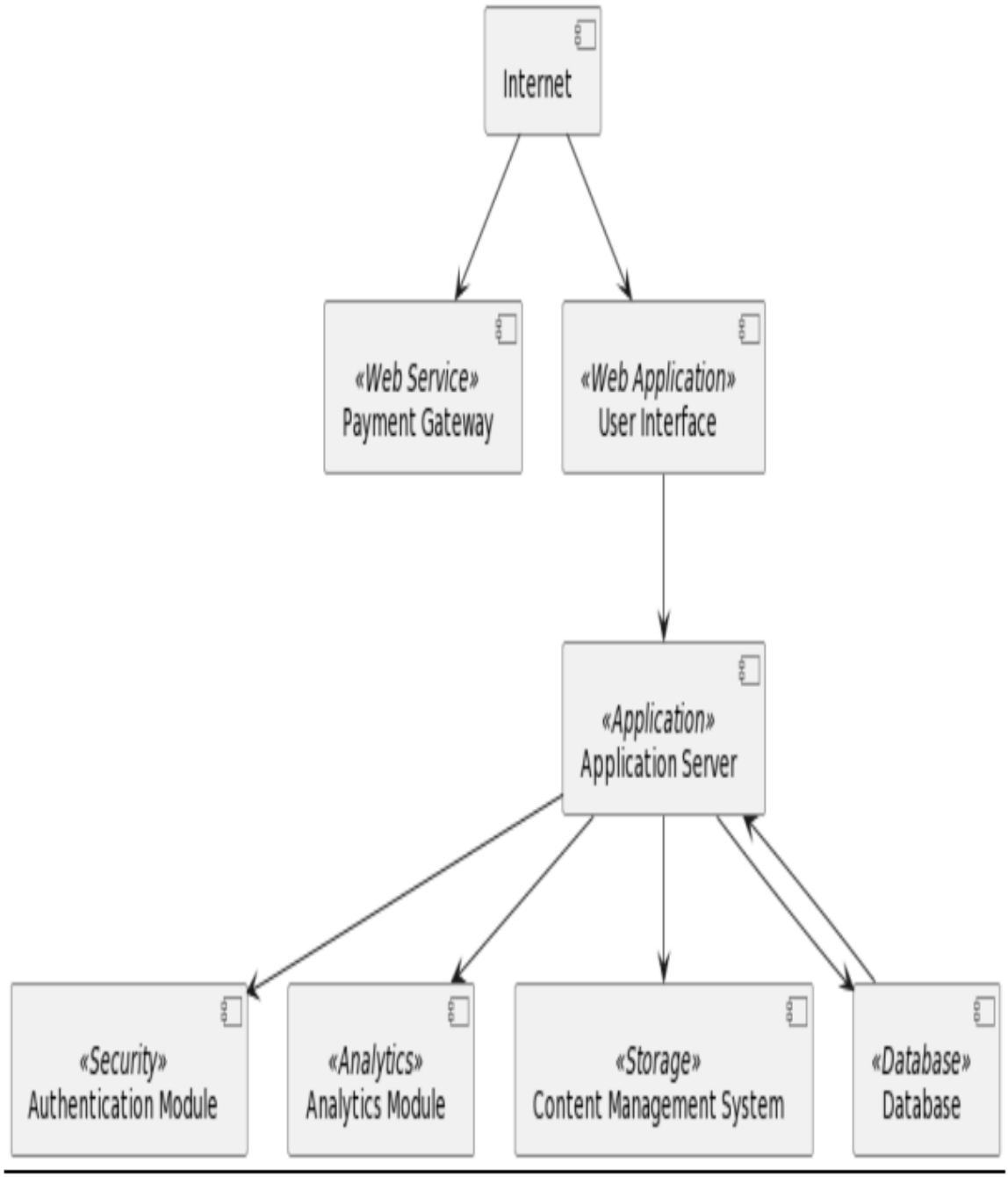

Figure 16:

Aim:

To design Deployment UML diagram for Digital Learning Environment.

Description:

A Deployment UML diagram for a Digital Learning Environment shows the physical deployment of the software components and hardware infrastructure required to run the system. It depicts how the software components are deployed on the hardware infrastructure, and how they interact with each other.

The diagram may also include information about the operating system, middleware, and network topology used in the deployment.

The main purpose of a Deployment UML diagram is to illustrate the deployment architecture of a system, and to identify the hardware and software components required to deploy the system in a production environment.

It can be used as a reference during the deployment process, and can help to ensure that the system is deployed correctly and efficiently. It helps to identify potential performance issues, bottlenecks, and security vulnerabilities that may arise during deployment.

The Deployment UML diagram can also be used to communicate the deployment architecture to developers, system administrators, and other stakeholders involved in the deployment process. It helps to identify potential performance issues, bottlenecks, and security vulnerabilities that may arise during deployment.

It can be used as a reference during the deployment process, and can help to ensure that the system is deployed correctly and efficiently.

In the context of a Digital Learning Environment, the Deployment UML diagram would typically include components such as web servers, application servers, databases, load balancers, firewalls, and other infrastructure components required to run the system. It may also include cloud services, virtualization, and other technologies used to deploy the system in a scalable and flexible manner. .

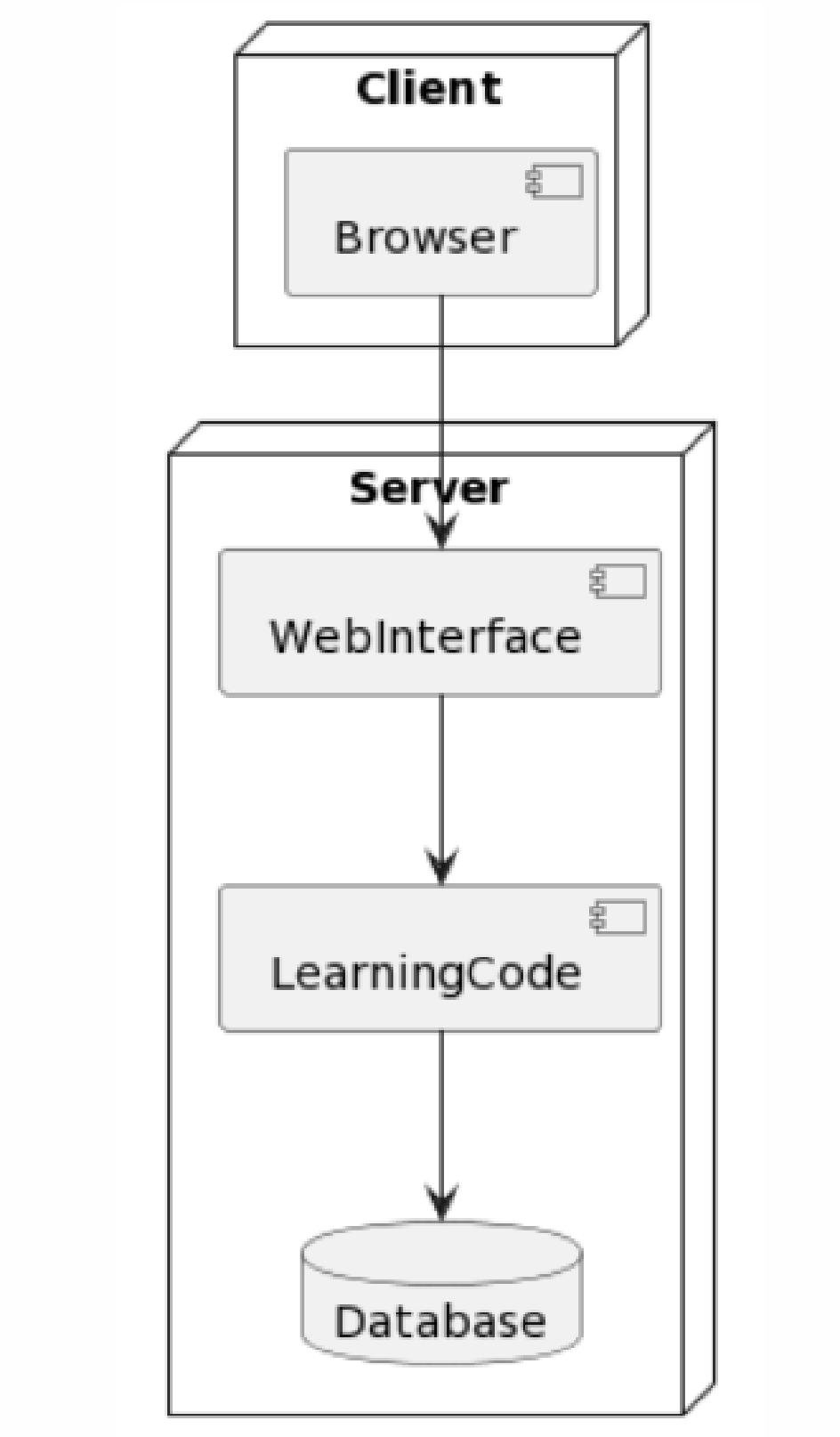
Client
Browser
Server
WebInterface
LearningCode
Database

Figure 17:

Aim:

To design Package UML diagram for digital learning environment.

Description:

A package UML diagram for a digital learning environment can be used to represent the high-level organization and structure of the various software components that make up the system.

The diagram shows how the components are organized into different packages and how they interact with each other to provide the required functionality.

In this diagram, the digital learning environment is represented as a top-level package that contains four sub-packages: the web application, database, video streaming, and learning tools.

The web application package contains the software components that are responsible for providing the user interface and application logic to the end-users. This package includes components such as the web server, application server, and web application. The database package contains the software components that are responsible for storing and retrieving data such as user accounts, course information, and progress tracking. This package includes components such as the database server and database API.

The video streaming package contains the software components that are responsible for providing the video content to the users. This package includes components such as the streaming server and streaming service.

The learning tools package contains the software components that provide various digital resources such as video lectures, e-books, and quizzes to the users. Note that this is just an example, and the actual package UML diagram for a digital learning environment may differ based on the specific requirements of the system.

The package diagram provides a high-level view of the system, which can be used to understand the overall structure of the software components and their interactions.

The database package contains the software components that are responsible for storing and retrieving data such as user accounts, course information, and progress tracking.

This package includes components such as the database server and database API. .

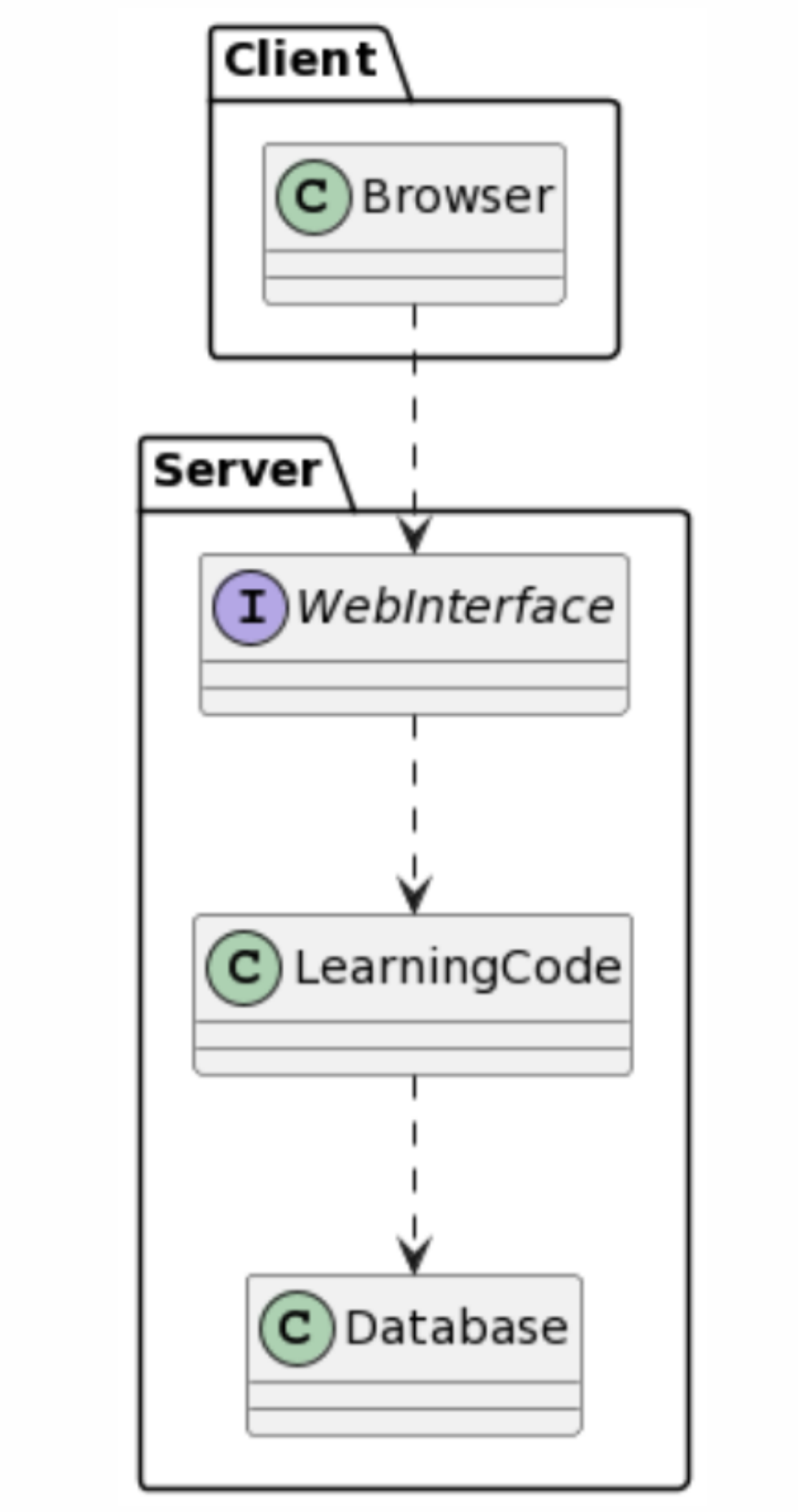
Client
Browser
Server
WebInterface
LearningCode
Database

Figure 18:

Experiment-3

Aim:
To estimate the cost of the iLearn The Digital Learning Environment by using COCOMO Model.

Description:

COCOMO(Constructive Cost Model) is a regression model based on LOC, i.e number of Lines of Code. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality. It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models. The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort Schedule.

Effort:
Amount of labor that will be required to complete a task. It is measured in person months units.

Schedule:
Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, months. Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required.

All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

Boehm's definition of organic, semidetached, and embedded systems: Organic – A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience xxix regarding the problem.

Semi-detached –
A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, knowledge of the various programming environment.lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Required software reliability extent xxxi Size of the application database The complexity of the product

(ii) Hardware attributes – Run-time performance constraints Memory constraints The volatility of the virtual machine environment Required turnabout time

(iii) Personnel attributes Analyst capability Software engineering capability Applications experience Virtual machine experience Programming language experience

(iv) Project attributes – Use of software tools Application of software engineering methods Required development schedule

Detailed Model –
Detailed COCOMO incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step of the software engineering process. The detailed model uses different effort multipliers for each cost driver attribute. In detailed cocomo, the whole software is divided into different modules and then we apply COCOMO in different modules to estimate effort and then sum the effort.

The Six phases of detailed COCOMO are:

YOUR BASIC COCOMO RESULTS!!

MODE	"A" variable	"B" variable	"C" variable	"D" variable	KLOC	EFFORT, (in person/months)	DURATION, (in months)	STAFFING, (recommended)
organic	2.4	1.05	2.5	0.38	14	38.33943241241644	9.993678334925779	3.8363684648952816

Explanation: The coefficients are set according to the project mode selected on the previous page, (as per Boehm,81). The final estimates are determined in the following manner:

effort =a*KLOC^{b}, in person/months, with KLOC = lines of code, (in the thousands), and:

duration =c*effort^{d}, finally:

staffing =effort/duration

For further reading, see Boehm, "Software Engineering Econimics",(81)

WARNING: If you see "NaN" in any field above, you have entered an **INVALID** value for KLOC!! Hit the "BACK" button on your browser, hit the "RESET" button, and enter a **DECIMAL NUMBER** in the KLOC input text box!

Thank you, and happy software engineering!

Figure 19:

Planning and requirements xxxii System design Detailed design Module code and test Integration and test Cost Constructive model The effort is calculated as a function of program size and a set of cost drivers are given according to each phase of the software lifecycle

.

Experiment-6

Aim:
To draw E-R diagrams for Digital Learning Environment.

Description:

An entity relationship diagram (ERD), also known as an entity relationship model, is a graphical representation that depicts relationships among people, objects, places, concepts or events within an information technology (IT) system. An ERD uses data modeling techniques that can help define business processes and serve as the foundation for a relational database.

Purpose of ERD:
The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD. oThe ERD serves as a documentation tool. oFinally, the ERD is used to connect the logical structure of the database to users. In particular, the ERD effectively communicates the logic of the database to users.An entity can be a real-world object, either animate or inanimate, that can be merely identifiable. An entity is denoted as a rectangle in an ER diagram.

Importance of ERDs and their uses:
Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a reference point, should any debugging or business process re-engineering be needed later Components of an ER Diagrams

1. Entity An entity can be a real-world object, either animate or inanimate, that can be merely identifiable. An entity is denoted as a rectangle in an ER diagram. For example, in a school database, students, teachers, classes, and courses offered can be treated as entities. All these entities have some attributes or properties that give them their identity.The database analyst gains a better understanding of the data to be contained in the database through the step of constructing the ERD. oThe ERD serves as a documentation tool. oFinally, the ERD is used to connect the logical structure of the database to users.

Entity Set
An entity set is a collection of related types of entities. An entity set may include entities with attribute sharing similar values.Entity relationship diagrams provide a visual starting point for database design that can also be used to help determine information system requirements throughout an organization. After a relational database is rolled out, an ERD can still serve as a reference point, For example, a Student set may contain all the students of a school; likewise, a Teacher set may include all the teachers of a school from all faculties. Entity set need not be disjoint.

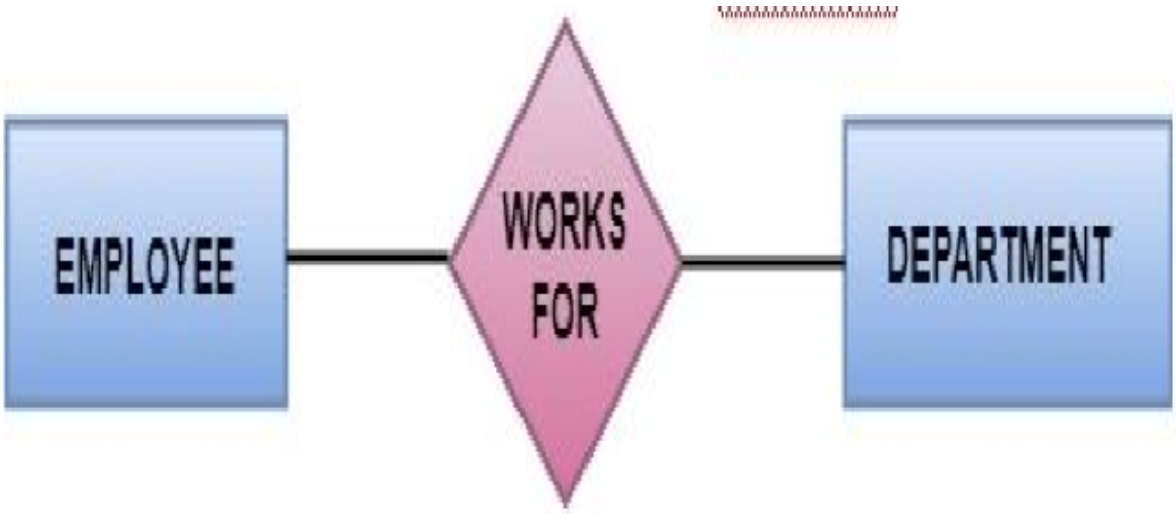

Figure 20:

2. Attributes Entities are denoted utilizing their properties, known as attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes. There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.There exists a domain or range of values that can be assigned to attributes. For example, a student entity may have name, class, and age as attributes.Entities are denoted utilizing their properties, known as attributes. All attributes have values.

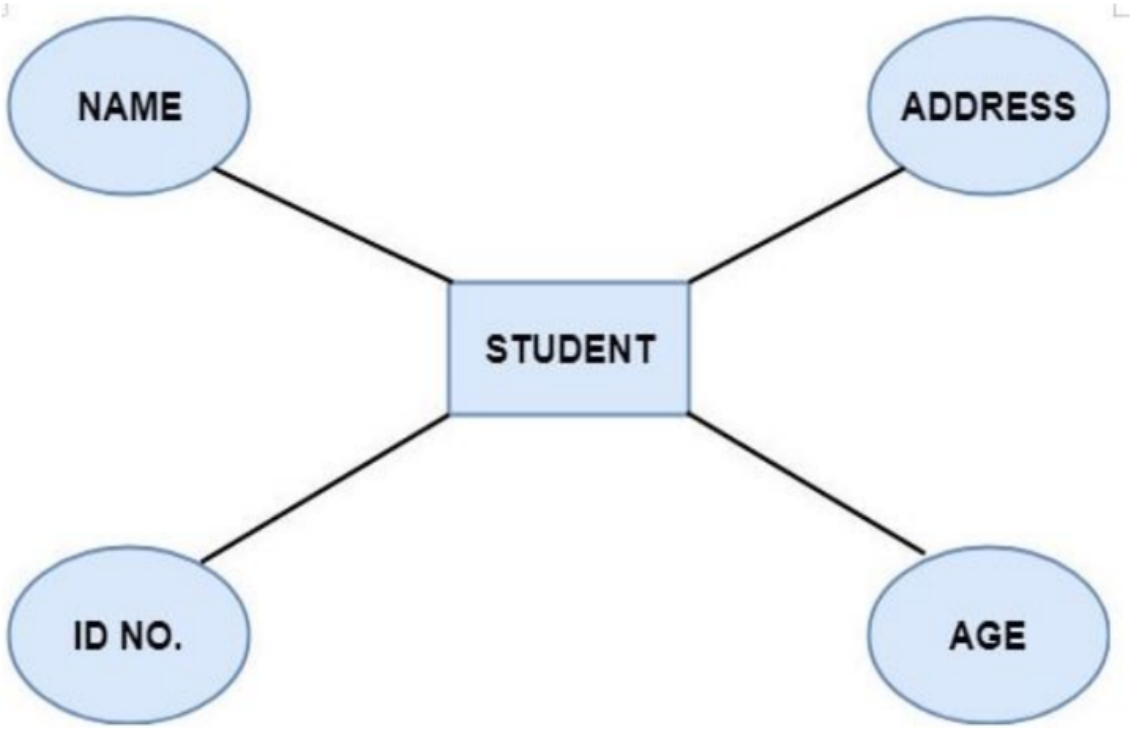

Figure 21:

There are four types of Attributes: 1.Key attribute 2.Composite attribute 3.Single-valued attribute 4.Multi-valued attribute 5.Derived attribute

1. Key attribute: Key is an attribute or collection of attributes that uniquely identifies an entity among the entity set. For example, the rollnumber of a student makes him identifiable among students.

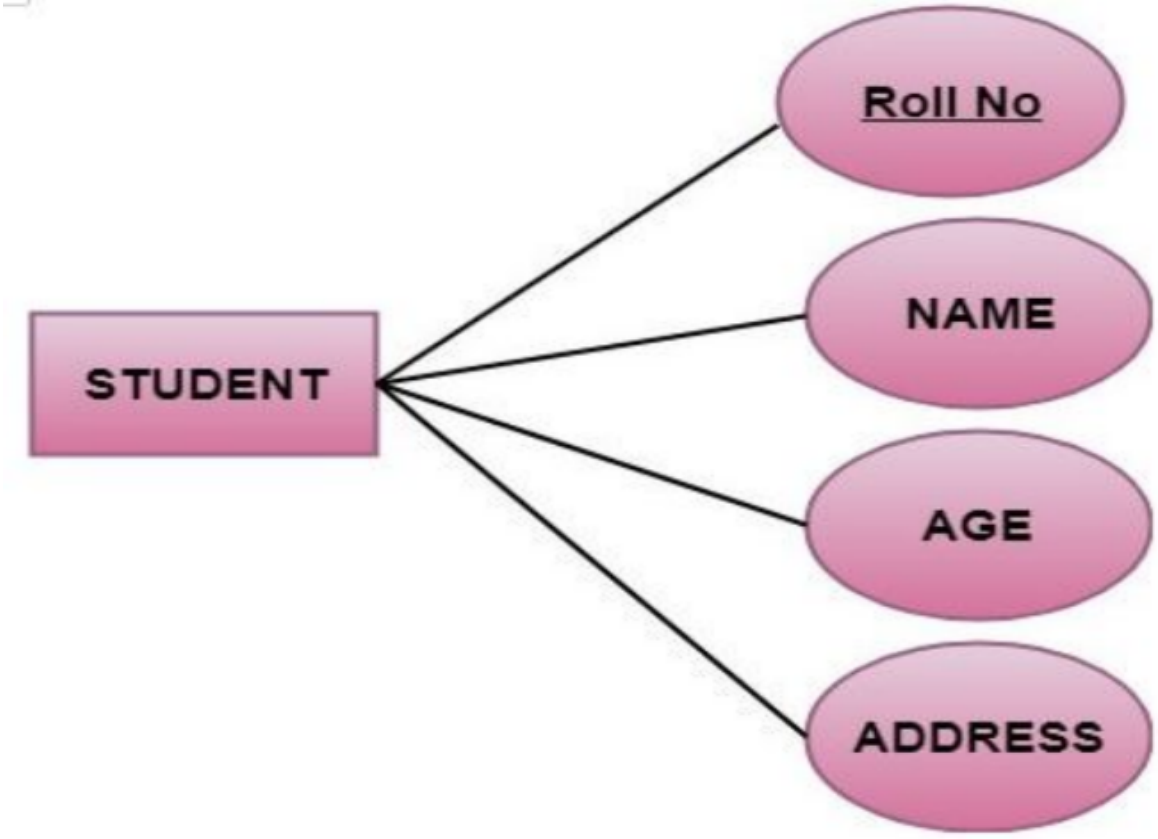

Figure 22:

There are mainly three types of keys:

1.Super key:A set of attributes that collectively identifies an entity in the entity set.

2.Candidate key:A minimal super key is known as a candidate key. An entity set may have more than one candidate key.

3.Primary key:A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

2. Composite attribute:An attribute that is a combination of other attributes is called a composite attribute. For example, In student entity, the student address is a composite attribute as an address is composed of other characteristics such as pin code, state, country.

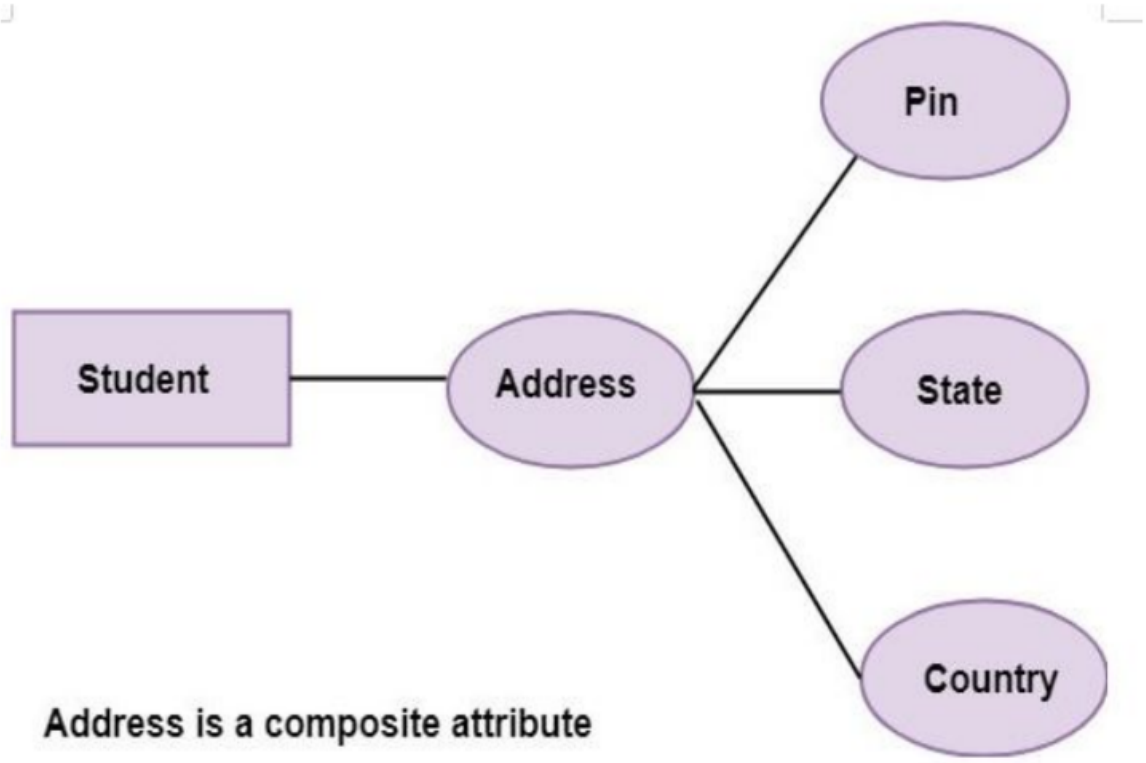

Figure 23:

3. Single-valued attribute:Single-valued attribute contain a single value. For example, SocialSecurityNumber.

4. Multi-valued Attribute:If an attribute can have more than one value, it is known as a multi-valued attribute. Multi-valued attributes are depicted by the double ellipse. For example, a person can have more than one phone number, email-address, etc.

5. Derived attribute:Derived attributes are the attribute that does not exist in the physical database, but their values are derived from other attributes present in the database. For example, age can be derived from dateofbirth. In the ER diagram, Derived attributes are depicted by the dashed ellipse.

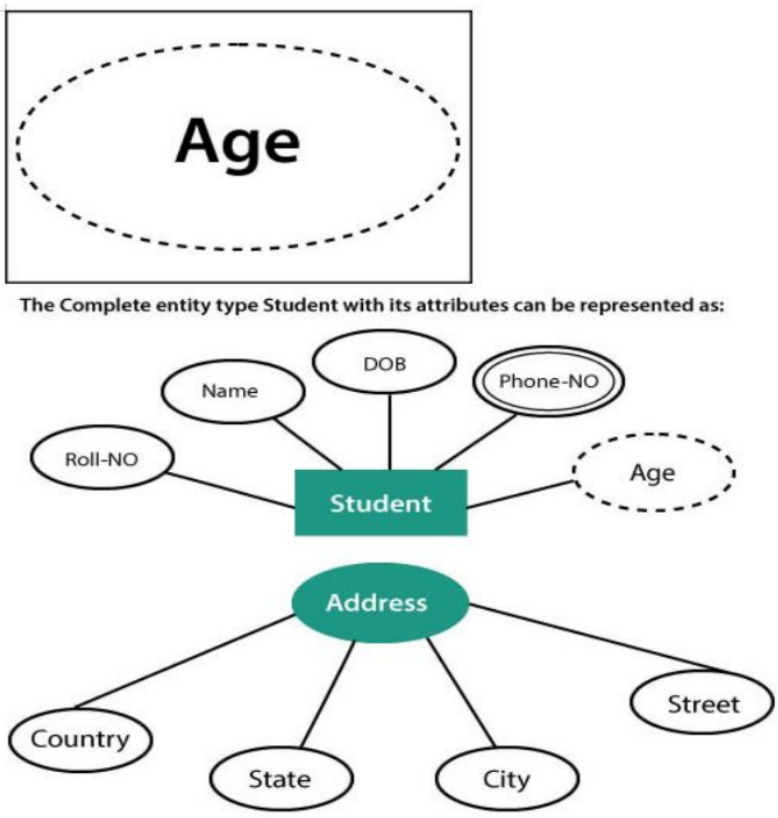

Figure 24:

3. Relationships: The association among entities is known as relationship. Relationships are represented by the diamond-shaped box.

For example, an employee worksat a department, a student enrolls in a course. Here, Worksat and Enrolls are called relationships.

Fig: Relationships in ERD

Figure 25:

E-R diagram:


Login_role_id
Login_username
#login_id
User_password
Login
#user_id
user_name
user_mobile
User
user_email
user_address
Has
#role_id
role_name
Roles
role_desc
#per_id
per_role_id
Permission
per_module
per_name
stu_name
#stu_id
stu_mobile
student
stu_email
stu_add
stu_pass
Manage
sch_desc
sch_date
schedule
sch_time
sch_name
#sch_id
Has
fees
#crs_fee_id
crs_fee_desc
crs_fee_type
cor_stu_id
#cor_id
course
cor_name
cor_type
cor_desc

Figure 26:

.

Aim:

To draw DFD diagram for Digital Learning Environment.

Description:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

Levels in Data Flow Diagrams (DFD):

The DFD may be used to perform a system or software at any level of abstraction. Infact, DFDs may be partitioned into levels that represent increasing information flow and functional detail. Levels in DFD are numbered 0, 1, 2 or beyond. Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

0-level DFDM

It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows.

Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood.

It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by DeMacro. Thus, if bubble "A" has two inputs x1 and x2 and one output y, then the expanded DFD, that represents "A" should have exactly two external inputs and one external output as shown in fig:

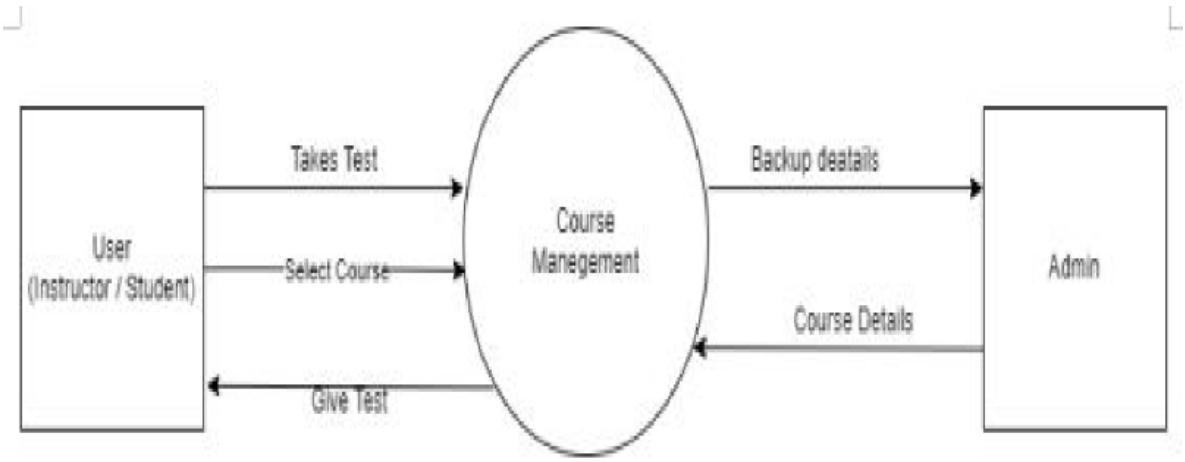

Figure 27:

The Level-0 DFD, also called context diagram of the result management system is shown in fig. As the bubbles are decomposed into less and less abstract bubbles, the corresponding data flow may also be needed to be decomposed.

1-level DFD

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and breakdown the high-level process of 0-level DFD into subprocesses.

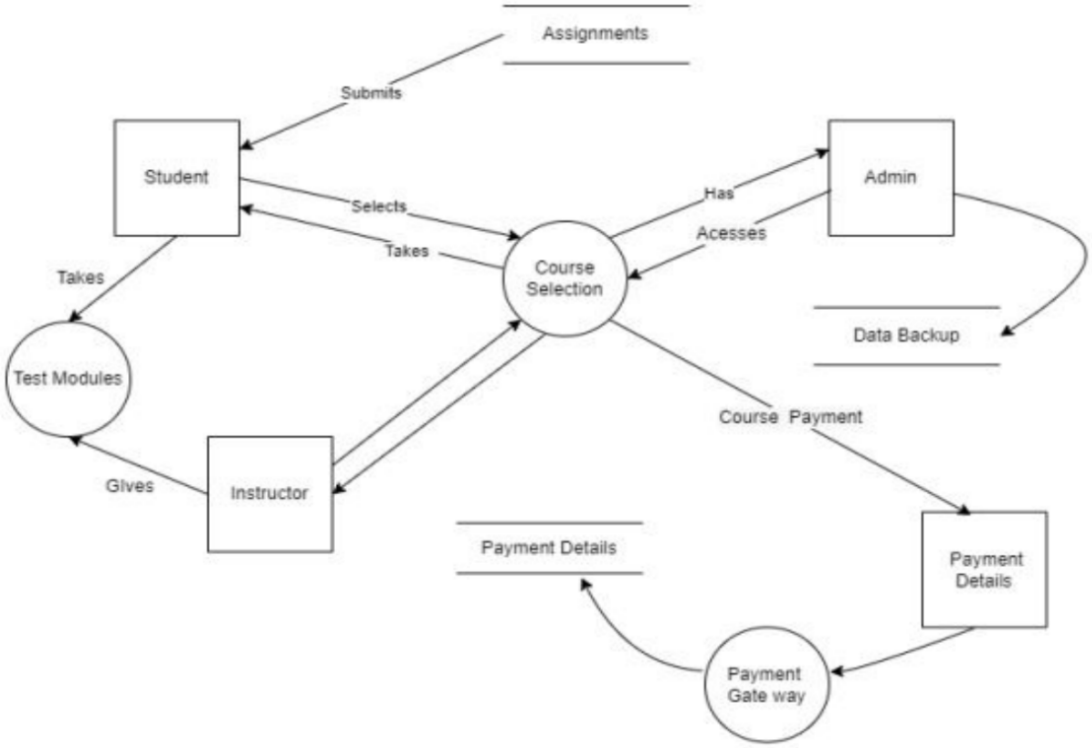
Assignments
Submits
Student
Selects
Takes
Has
Admin
Acesses
Takes
Course
Selection
Data Backup
Test Modules
Course Payment
Gives
Instructor
Payment Details
Payment
Details
Payment
Gate way

Figure 28:

.

Experiment-7

Aim:
To Design of Test cases based on requirements and design for " I Learn Digital Learning Environment" using Unit Testing.

Description:
Unit testing is a critical part of software development for digital learning environments. Unit tests are designed to test individual components of the software, such as classes, functions, or methods, in isolation from the rest of the system. By testing each component in isolation, developers can identify and fix bugs and ensure that the software functions as expected.

In a digital learning environment, unit tests can be used to test a variety of components, such as user interfaces, database connections, and business logic. Here are some examples of unit test cases that can be used in a digital learning environment

A unit test is a way of testing a unit - the smallest piece of code that can be logically isolated in a system. In most programming languages, that is a function, a subroutine, a method or property. The isolated part of the definition is important. In his book "Working Effectively with Legacy Code", author Michael Feathers states that such tests are not unit tests when they rely on external systems: "If it talks to the database, it talks across the network, it touches the file system, it requires system configuration, or it can't be run at the same time as any other test."

Software developers have been testing software for as long as they have been writing code, but the ability to automate software testing appeared around the 1980s. (Testing references provides a handy timeline of software testing.) Suddenly, instead of having to run a program manually against lists of test values, or setting breakpoints in the code and tracing the program's logic step by step, developers could do what they do best — turn testing into another program. **Objective of Unit Testing:** The objective of Unit Testing is:

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

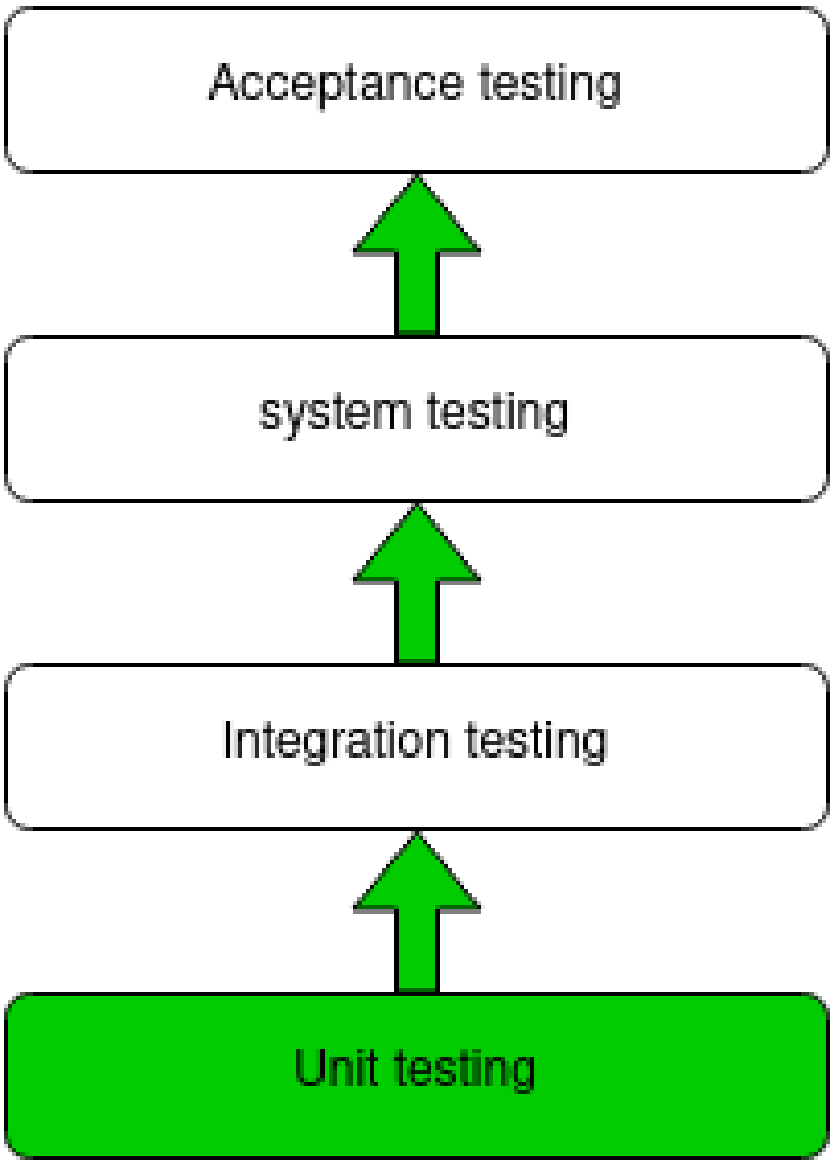

Figure 29: Digital Learning Environment.

Types of Unit Testing:
There are 2 types of Unit Testing:
1.Manual
2.Automated.
Workflow of Unit Testing:

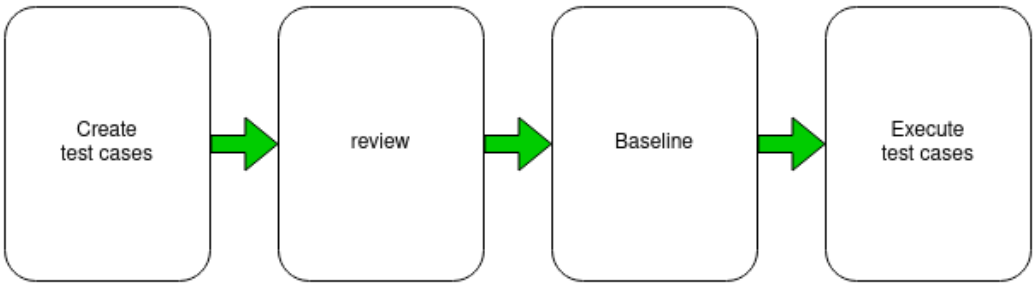

Figure 30: Digital Learning Environment.

Unit Testing Techniques:
There are 3 types of Unit Testing Techniques. They are 1.Black Box Testing:This testing technique is used in covering the unit tests for input, user interface, and output parts.
2.White Box Testing:This technique is used in testing the functional behavior of the system by giving the input and checking the functionality output including the internal design structure and code of the modules.
3.Gray Box Testing:This technique is used in executing the relevant test cases, test methods, test functions, and analyzing the code performance for the modules.

Unit Testing Tools:
Here are some commonly used Unit Testing tools:
1.Jtest
2.Junit
3.NUnit
4.EMMA
5.PHPUnit

Advantages of Unit Testing:
1.Unit Testing allows developers to learn what functionality is provided by a unit and how to use it to gain a basic understanding of the unit API.
2.Unit testing allows the programmer to refine code and make sure the module works properly.
3.Unit testing enables testing parts of the project without waiting for others to be completed.

Disadvantages of Unit Testing:
1.The process is time-consuming for writing the unit test cases.
2.Unit Testing will not cover all the errors in the module because there is a chance of having errors in the modules while doing integration testing.
3.Unit Testing is not efficient for checking the errors in the UI(User Interface) part of the module.
4.It requires more time for maintenance when the source code is changed frequently.
5.It cannot cover the non-functional testing parameters such as scalability, the performance of the system, etc.

Program:

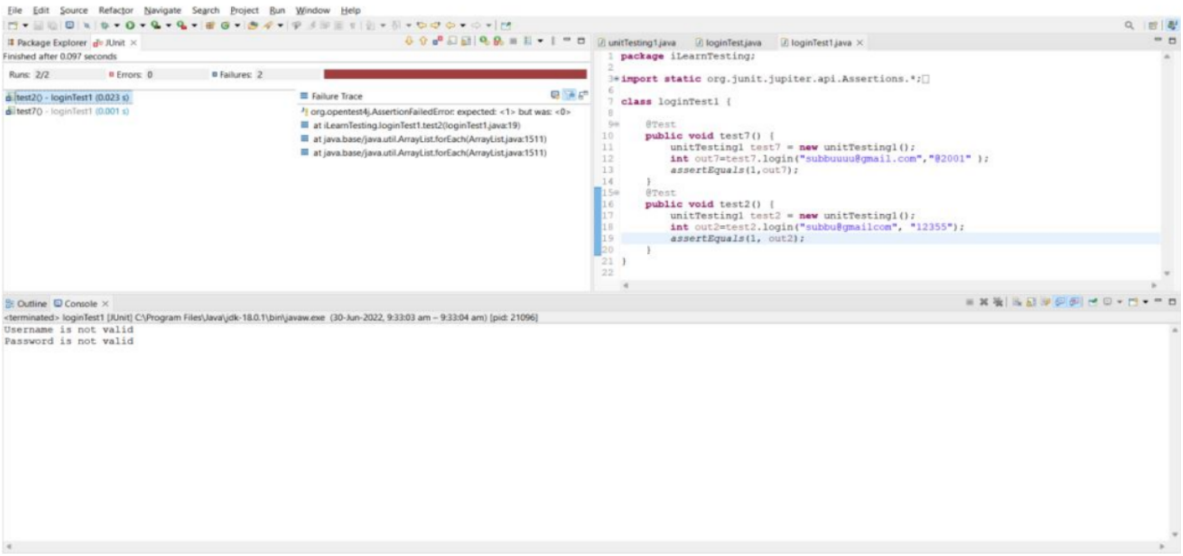

Figure 31: Digital Learning Environment.

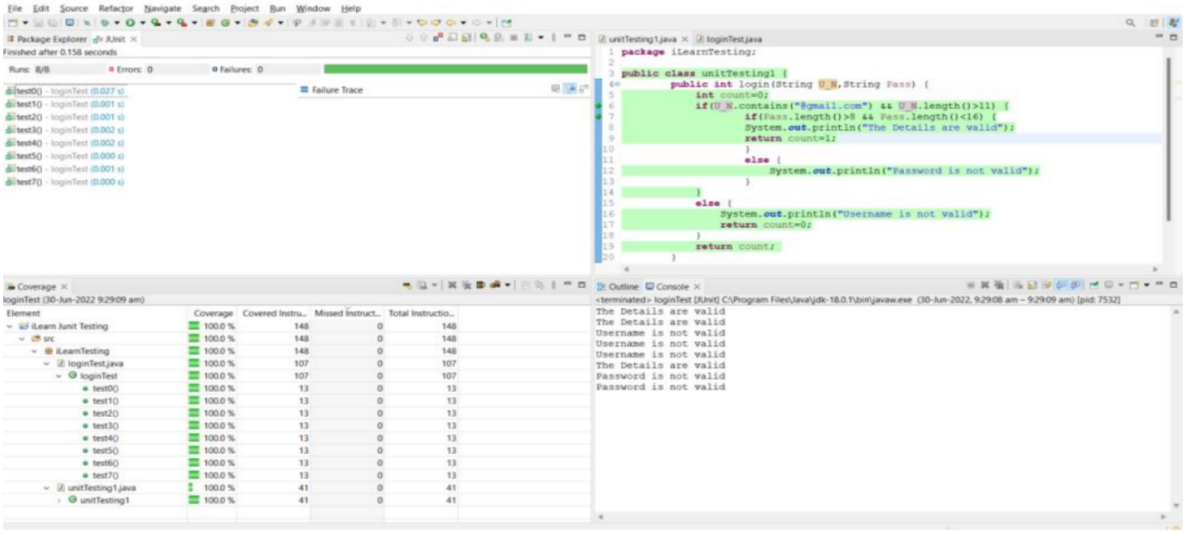

Figure 32: Digital Learning Environment.

.

Experiment-8

Aim:
Prepare version control and change control for software configuration items for Digital Learning Environment System.

Description:
Version control and change control are critical for managing software configuration items in a digital learning environment. Here are some steps you can take to set up a version control and change control process:

Version Control:

- Version control is an essential practice for managing software configuration items in a digital learning environment.
- Version control enables developers to track changes made to software over time, including who made the change, when the change was made, and why it was made. This information is critical for maintaining a stable and reliable digital learning environment.
- To implement version control for software configuration items, a version control system such as Git, SVN, or Mercurial can be used.
- These systems provide a central repository to store all software configuration items that will be version controlled. The use of a version control system helps developers to work on the same codebase without interfering with each other's work.
- It helps avoid conflicts and ensures that all code changes are properly integrated into the main codebase.
- Version control systems allow developers to create branches for different features or bug fixes, and then merge the changes back into the main codebase when the work is complete. This enables multiple developers to work on the same codebase at the same time without interfering with each other's work.
- Additionally, version control enables code reviews, which allows other developers to review changes made by their colleagues, providing feedback and catching any errors or issues before the code is merged into the main codebase.
- Version control also helps with quality assurance. Each change made to the software can be tested before it is merged into the main codebase.
- This helps ensure that the software remains in a working state and that all changes are properly integrated into the main codebase.
- Version control is also useful for maintaining a history of all modifications made to the software. This information is critical for identifying bugs and troubleshooting issues that arise. By having a complete history of all changes made to the software, developers can easily roll back to a previous version of the software if necessary.

  In summary, version control is an essential practice for managing software configuration items in a digital learning environment. It enables developers to track changes made to software over time, allows for collaboration between developers, enables code reviews and quality assurance, and helps maintain a history of all modifications made to the software. By implementing version control, digital learning environments can be maintained in a stable and reliable state.

Change Control:

- Change control is a process for managing updates and modifications to software configuration items in a digital learning environment.
- Change control helps ensure that changes are made in a controlled and systematic manner to minimize risks and avoid negative impacts on the system.
- In a digital learning environment, change control includes a change request form, a review process, and a change log.
- When a change request is made, it is evaluated by a change control board to determine its impact on the digital learning environment. If the change is approved, it is implemented by developers following established procedures.
- After the change is made, it is tested and reviewed to ensure that it is functioning correctly. This helps ensure that the change does not negatively impact the digital learning environment or introduce new bugs.
- Change control is critical for managing software configuration items in a digital learning environment.
- It helps prevent unauthorized changes to the software, ensures that changes are properly evaluated and tested, and provides a record of all modifications made to the system.
- This information is essential for identifying issues and troubleshooting problems that arise.
- In addition to the change request form, review process, and change log, change control may also include a rollback plan.
- A rollback plan outlines the steps that will be taken if a change negatively impacts the digital learning environment. This plan helps minimize downtime and ensures that the system can be restored to a working state quickly.
- Change control helps ensure that changes are made in a controlled and systematic manner to minimize risks and avoid negative impacts on the system.
- It helps prevent unauthorized changes to the software, ensures that changes are properly evaluated and tested, and provides a record of all modifications made to the system.

In conclusion, version control and change control are critical for managing software configuration items in a digital learning environment.

Version control provides a history of all modifications made to the software, helps avoid conflicts between developers, and ensures that the software is always in a working state.

Change control helps ensure that changes are made in a controlled and systematic manner to minimize risks and avoid negative impacts on the system. Together, these processes help ensure the stability and reliability of the digital learning environment.

To facilitate change control, you can also implement a pull request workflow, where changes are reviewed and approved by a designated reviewer before being merged into the main branch. This can help ensure that changes are thoroughly tested and reviewed before being deployed to production.

It's also a good idea to use descriptive commit messages that clearly explain the changes made, so that anyone reviewing the code later can easily understand the purpose and context of each change. Additionally, you can use tags to mark significant releases or milestones in the development process, and use branches to maintain separate codebases for different versions of the DLE.