

# **Course4U E-Learning Platform**

*Project Report Submitted By*

**SANJU ABRAHAM BINU**

**Reg. No: AJC17MCA-I051**

*In Partial fulfillment for the Award of the Degree Of*

**INTEGRATED MASTER OF COMPUTER APPLICATIONS  
(INMCA)**

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**



**AMAL JYOTHI COLLEGE OF ENGINEERING  
KANJIRAPPALLY**

[Affiliated to APJ Abdul Kalam Technological University, Kerala. Approved by AICTE, Accredited by NAAC with 'A' grade. Koovappally, Kanjirappally, Kottayam, Kerala – 686518]

**2017-2022**

**DEPARTMENT OF COMPUTER APPLICATIONS**  
**AMAL JYOTHI COLLEGE OF ENGINEERING**  
**KANJIRAPPALLY**



**CERTIFICATE**

This is to certify that the Project report, “**COURSE4U E-Learning Platform**” is the bonafided work of **Sanju Abraham Binu (Reg.No: AJC17MCA-I051)** in partial fulfillment of the requirements for award of the Degree of Integrated Master of Computer Applications under APJ Abdul Kalam Technological University during the year 2017-22.

**Mr. Rony Tom**  
**Internal Guide**

**Ms. Meera Rose Mathew**  
**Coordinator**

**Rev. Fr. Dr. Rubin Thottupurathu Jose**  
**Head of the Department**

**External Examiner**

## **DECLARATION**

I hereby declare that the project report “**Course4U E-Learning Platform**” is a bonafided work done at Amal Jyothi College of Engineering, towards the partial fulfilment of the requirements for the award of the Degree of Integrated Master of Computer Applications (INMCA) from APJ Abdul Kalam Technological University, during the academic year 2017-2022.

**Date:**

**KANJIRAPPALLY**

**Sanju Abraham Binu**

**Reg. No: AJC17MCA-I051**

## ACKNOWLEDGEMENT

First and foremost, I thank God almighty for his eternal love and protection throughout the project. I take this opportunity to express my gratitude to all who helped me in completing this project successfully. It has been said that gratitude is the memory of the heart. I wish to express my sincere gratitude to our manager **Rev. Fr. Dr. Mathew Paikatt** and Principal **Dr. Lillykutty Jose** for providing good faculty for guidance.

I owe a great depth of gratitude towards our Head of the Department **Rev. Fr. Dr. Rubin Thottupurathu Jose** for helping us. I extend my whole hearted thanks to the project coordinator **Ms. Meera Rose Mathew** for her valuable suggestions and for overwhelming concern and guidance from the beginning to the end of the project. I would also like to express sincere gratitude to my guide, **Mr. Rony Tom** for his inspiration and helping hand.

I thank our beloved teachers for their cooperation and suggestions that helped me throughout the project. I express my thanks to all my friends and classmates for their interest, dedication, and encouragement shown towards the project. I convey my hearty thanks to my family for the moral support, suggestions, and encouragement to make this venture a success.

SANJU ABRAHAM BINU

## **ABSTRACT**

**“COURSE4U E-Learning Platform”** is a web application system which is simple and easy to use. E-learning fulfils the thirst of knowledge and offers online content that can be delivered for the learner at anywhere, anytime and any age through a wide range of e-learning solution while compared with traditional learning system.

It also provides the rapid access to specific knowledge and information. With the rapid growth of voluminous information sources and the time constraint the learning methodology has changed. Learners obtain knowledge through e-Learning systems rather than manually teaching and learning.

# CONTENT

Sl. No	Topic	Page No
1	INTRODUCTION	1
1.1	PROJECT OVERVIEW	2
1.2	PROJECT SPECIFICATION	2
2	SYSTEM STUDY	4
2.1	INTRODUCTION	5
2.2	EXISTING SYSTEM	6
2.3	DRAWBACKS OF EXISTING SYSTEM	6
2.4	PROPOSED SYSTEM	6
2.5	ADVANTAGES OF PROPOSED SYSTEM	7
3	REQUIREMENT ANALYSIS	8
3.1	FEASIBILITY STUDY	9
3.1.1	ECONOMICAL FEASIBILITY	9
3.1.2	TECHNICAL FEASIBILITY	10
3.1.3	BEHAVIORAL FEASIBILITY	10
3.2	SYSTEM SPECIFICATION	11
3.2.1	HARDWARE SPECIFICATION	11
3.2.2	SOFTWARE SPECIFICATION	11
3.3	SOFTWARE DESCRIPTION	11
3.3.1	REACTJS	11
3.3.2	NODEJS	11
4	SYSTEM DESIGN	13
4.1	INTRODUCTION	14
4.2	UML DIAGRAM	14
4.2.1	USE CASE DIAGRAM	15
4.2.2	SEQUENCE DIAGRAM	18
4.2.3	ACTIVITY DIAGRAM	20
4.2.4	DEPLOYMENT DIAGRAM	22
4.2.5	DATA FLOW DIAGRAM	23
4.2.6	CLASS DIAGRAM	24
4.3	USER INTERFACE DESIGN	25
4.4	DATA BASE DESIGN	27

<b>5</b>	<b>SYSTEM TESTING</b>	<b>33</b>
<b>5.1</b>	<b>INTRODUCTION</b>	<b>34</b>
<b>5.2</b>	<b>TEST PLAN</b>	<b>35</b>
<b>5.2.1</b>	<b>UNIT TESTING</b>	<b>35</b>
<b>5.2.2</b>	<b>INTEGRATION TESTING</b>	<b>36</b>
<b>5.2.3</b>	<b>VALIDATION TESTING</b>	<b>36</b>
<b>5.2.4</b>	<b>USER ACCEPTANCE TESTING</b>	<b>37</b>
<b>5.2.5</b>	<b>SELENIUM TESTING</b>	<b>37</b>
<b>6</b>	<b>IMPLEMENTATION</b>	<b>41</b>
<b>6.1</b>	<b>INTRODUCTION</b>	<b>42</b>
<b>6.2</b>	<b>IMPLEMENTATION PROCEDURE</b>	<b>42</b>
<b>6.2.1</b>	<b>USER TRAINING</b>	<b>43</b>
<b>6.2.2</b>	<b>TRAINING ON APPLICATION SOFTWARE</b>	<b>43</b>
<b>6.2.3</b>	<b>SYSTEM MAINTENANCE</b>	<b>43</b>
<b>7</b>	<b>CONCLUSION &amp; FUTURE SCOPE</b>	<b>44</b>
<b>7.1</b>	<b>CONCLUSION</b>	<b>45</b>
<b>7.2</b>	<b>FUTURE SCOPE</b>	<b>45</b>
<b>8</b>	<b>BIBLIOGRAPHY</b>	<b>46</b>
<b>9</b>	<b>APPENDIX</b>	<b>48</b>
<b>9.1</b>	<b>SAMPLE CODE</b>	<b>49</b>
<b>9.2</b>	<b>SCREEN SHOTS</b>	<b>67</b>

## **List of Abbreviation**

- IDE - Integrated Development Environment
- UML - Unified Modeling Language
- CSS - Cascading Style Sheet
- NoSQL - Not Only Structured Query Language



# **CHAPTER 1**

## **INTRODUCTION**

## 1.1 PROJECT OVERVIEW

**“Course4U E-Learning Platform”** is a web system which will be helpful for students and teachers. The students can use this by registering into the system. The students can find their interested courses in different category. They can enroll to their interested courses and start watching tutorials and related notes. The authorized teachers can add different courses and they can schedule live classes if needed. They can monitor the progress of the enrolled students to their courses. The admin can manage the courses added the teachers. They can manage users and teachers as well. Admin have the complete control over the system.

## 1.2 PROJECT SPECIFICATION

The proposed system is made to help the students for an easy and convenient way of learning things online from anywhere at anytime. They can repeat any content they want. It helps to analyses the progress of the learning which is useful for future purpose.

The system includes 3 modules. They are:

### 1. Admin Module

Admin can modify various courses and categories. Admin can remove or block courses and teachers. Admin can manage the users as well. The admin can view the students enrolled to different courses.

## **2. Teacher Module**

Teachers can add Courses, which are paid and students should buy it. Teachers can also provide resources etc.

## **3. Student Module**

Contractors can register and they can see all the courses that are available. They can search for their interested course. They can enroll to the courses. Various study materials will be available for the student for the enrolled course.

## **CHAPTER 2**

### **SYSTEM STUDY**

## 2.1 INTRODUCTION

System analysis is a process of gathering and interpreting facts, diagnosing problems and the information to recommend improvements on the system. It is a problem solving activity that requires intensive communication between the system users and system developers. System analysis or study is an important phase of any system development process. The system is studied to the minute's detail and analyzed. The system analyst plays the role of the interrogator and dwells deep into the working of the present system. The system is viewed as a whole and the input to the system are identified. The outputs from the organizations are traced to the various processes. System analysis is concerned with becoming aware of the problem, identifying the relevant and decisional variables, analyzing and synthesizing the various factors and determining an optimal or at least a satisfactory solution or program of action.

A detailed study of the process must be made by various techniques like interviews, questionnaires etc. The data collected by these sources must be scrutinized to arrive to a conclusion. The conclusion is an understanding of how the system functions. This system is called the existing system. Now the existing system is subjected to close study and problem areas are identified. The designer now functions as a problem solver and tries to sort out the difficulties that the enterprise faces. The solutions are given as proposals. The proposal is then weighed with the existing system analytically and the best one is selected. The proposal is presented to the user for an endorsement by the user. The proposal is reviewed on user request and suitable changes are made. This is loop that ends as soon as the user is satisfied with proposal.

Preliminary study is the process of gathering and interpreting facts, using the information for further studies on the system. Preliminary study is problem solving activity that requires intensive communication between the system users and system developers. It does various feasibility studies. In these studies, a rough figure of the system activities can be obtained, from which the decision about the strategies to be followed for effective system study and analysis can be taken.

## **2.2 EXISTING SYSTEM**

Existing system here under consideration is Coursera. Coursera is an e-learning platform that enables to join various courses and learn various technologies. Coursera also provides certification after completion of each course. It also provides various measures for progress, deadlines, recommendations etc.

## **2.3 DRAWBACKS OF EXISTING SYSTEM**

- Less convenient in managing courses and live classes.
- They can't keep track of the progress of the student.
- The teacher student interactions are not available.

## **2.4 PROPOSED SYSTEM**

The proposed system is defined to meets all the disadvantages of the existing system. It is necessary to have a system that is more user friendly and user attractive. Proposed system provides the features of existing system. It also enables users to interact more easily via a simple user interface. It provides paid as well as free courses. Teachers can lively interact with students and vice versa. Give various recommendations about courses. The students can interact with the teachers through the system.

## 2.5 ADVANTAGES OF PROPOSED SYSTEM

The system is very simple in design and to implement. The system requires very low system resources and the system will work in almost all configurations. It has got following features:

➤ **All features under one roof:-**

Our customers will get all pet care facilities under one system. In a single window provide all that you need.

➤ **Better security: -**

For data to remain secure measures must be taken to prevent unauthorized access. Security means that data are protected from various forms of destruction. The system security problem can be divided into four related issues: security, integrity, privacy and confidentiality. Username and password requirement to sign in ensures security. It will also provide data security as we are using the secured databases for maintaining the documents.

➤ **Ensure data accuracy: -**

The proposed system eliminates the manual errors while entering the details of the users during the registration.

➤ **Better service: -**

The system will avoid the burden of hard copy storage. We can also conserve the time and human resources for doing the same task. The data can be maintained for longer period with no loss of data.

## **CHAPTER 3**

# **REQUIREMENT ANALYSIS**



### 3.1 FEASIBILITY STUDY

Feasibility study is made to see if the project on completion will serve the purpose of the organization for the amount of work, effort and the time that spend on it. Feasibility study lets the developer foresee the future of the project and the usefulness. A feasibility study of a system proposal is according to its workability, which is the impact on the organization, ability to meet their user needs and effective use of resources. Thus, when a new application is proposed it normally goes through a feasibility study before it is approved for development.

The document provides the feasibility of the project that is being designed and lists various areas that were considered very carefully during the feasibility study of this project such as Technical, Economic and Operational feasibilities. The following are its features: -

#### 3.1.1 Economic Feasibility

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

The proposed system is developed as part of project work, there is no manual cost to spend for the proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

The cost of project, “COURSE4U E-Learning Platform” was divided according to the system used, its development cost and cost for hosting the project. According to all the calculations the project was developed in a low cost. As it is completely developed using open source software.

### 3.1.2 Technical Feasibility

The system must be evaluated from the technical point of view first. The assessment of this feasibility must be based on an outline design of the system requirement in the terms of input, output, programs and procedures. Having identified an outline system, the investigation must go on to suggest the type of equipment, required method developing the system, of running the system once it has been designed.

Technical issues raised during the investigation are:

- Does the existing technology sufficient for the suggested one?
- Can the system expand if developed?

The project should be developed such that the necessary functions and performance are achieved within the constraints. The project requires High Resolution Scanning device and utilizes Cryptographic techniques. Through the technology may become obsolete after some period of time, due to the fact that newer version of same software supports older versions, the system may still be used. So there are minimal constraints involved with this project. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The system has been developed using PHP in front end and MySQL in server in back end, the project is technically feasible for development. The System used was also of good performance of Processor Intel i3 core; RAM 4GB and, Hard disk 1TB

### 3.1.3 Behavioral Feasibility

The proposed system includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

COURSE4U E-LEARNING PLATFORM, GUI is simple so that users can easily use it. COURSE4U E-LEARNING PLATFORM is simple enough so that no training is needed.

## 3.2 SYSTEM SPECIFICATION

### 3.2.1 Hardware Specification

Processor - Intel core i3

RAM - 4 GB

Hard disk - 1 TB

### 3.2.2 Software Specification

Front End - ReactJs, CSS

Backend - NodeJS, MongoDB

Client on PC - Windows 7 and above.

Technologies used - NodeJS, ReactJs, CSS

## 3.3 SOFTWARE DESCRIPTION

### 3.3.1 ReactJs

React (also known as React.js or ReactJS) is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta (formerly Facebook) and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

### 3.3.2 NodeJs

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux. Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent. Asynchronous and Event Driven. All APIs of

Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call Very Fast. Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution. Single Threaded but Highly Scalable Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server. No Buffering. Node.js applications never buffer any data. These applications simply output the data in chunks

### **MongoDB.**

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL).

## **CHAPTER 4**

### **SYSTEM DESIGN**

## 4.1 INTRODUCTION

Design is the first step into the development phase for any engineered product or system. Design is a creative process. A good design is the key to effective system. The term “design” is defined as “the process of applying various techniques and principles for the purpose of defining a process or a system in sufficient detail to permit its physical realization”. It may be defined as a process of applying various techniques and principles for the purpose of defining a device, a process or a system in sufficient detail to permit its physical realization. Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm that is used. The system design develops the architectural detail required to build a system or product. As in the case of any systematic approach, this software too has undergone the best possible design phase fine tuning all efficiency, performance and accuracy levels. The design phase is a transition from a user oriented document to a document to the programmers or database personnel. System design goes through two phases of development: Logical and Physical Design.

## 4.2 UML DIAGRAM

UML is a standard language for specifying, visualizing, constructing, and documenting the artifacts of software systems. UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.

UML stands for **Unified Modeling Language**. UML is different from the other common programming languages such as C++, Java, COBOL, etc. UML is a pictorial language used to make software blueprints. UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system. Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc. UML is not a programming language but tools can be used to generate code in various languages using UML diagrams. UML has a direct relation with object oriented analysis and design.

---

After some standardization, UML has become an OMG standard. All the elements, relationships are used to make a complete UML diagram and the diagram represents a system. The visual effect of the UML diagram is the most important part of the entire process. All the other elements are used to make it complete. UML includes the following nine diagrams.

- Class diagram
- Object diagram
- Use case diagram
- Sequence diagram
- Collaboration diagram
- Activity diagram
- State chart diagram
- Deployment diagram
- Component diagram

### 4.2.1 USE CASE DIAGRAM

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. In this context, the term "system" refers to something being developed or operated, such as a mail-order product sales and service Web site. Use case diagrams are employed in UML (Unified Modeling Language), a standard notation for the modeling of real-world objects and systems.

System objectives can include planning overall requirements, validating a hardware design, testing and debugging a software product under development, creating an online help reference, or performing a consumer-service-oriented task. For example, use cases in a product sales environment would include item ordering, catalog updating, payment processing, and customer relations. A use case diagram contains four components.

- The boundary, which defines the system of interest in relation to the world around it.
- The actors, usually individuals involved with the system defined according to their

roles.

- The use cases, which are the specific roles are played by the actors within and around the system.
- The relationships between and among the actors and the use cases.

Use case diagrams are drawn to capture the functional requirements of a system. After identifying the above items, we have to use the following guidelines to draw an efficient use case diagram

- The name of a use case is very important. The name should be chosen in such a way so that it can identify the functionalities performed.
- Give a suitable name for actors.
- Show relationships and dependencies clearly in the diagram.
- Do not try to include all types of relationships, as the main purpose of the diagram is to identify the requirements.
- Use notes whenever required to clarify some important points.



Fig 1 : Use case diagram for E-Learning Platform



## 4.2.2 SEQUENCE DIAGRAM

A sequence diagram simply depicts interaction between objects in a sequential order i.e. the order in which these interactions take place. We can also use the terms event diagrams or event scenarios to refer to a sequence diagram. Sequence diagrams describe how and in what order the objects in a system function. These diagrams are widely used by businessmen and software developers to document and understand requirements for new and existing systems.

### Sequence Diagram Notations –

- i. **Actors** – An actor in a UML diagram represents a type of role where it interacts with the system and its objects. It is important to note here that an actor is always outside the scope of the system we aim to model using the UML diagram. We use actors to depict various roles including human users and other external subjects. We represent an actor in a UML diagram using a stick person notation. We can have multiple actors in a sequence diagram.
- ii. **Lifelines** – A lifeline is a named element which depicts an individual participant in a sequence diagram. So basically each instance in a sequence diagram is represented by a lifeline. Lifeline elements are located at the top in a sequence diagram.
- iii. **Messages** – Communication between objects is depicted using messages. The messages appear in a sequential order on the lifeline. We represent messages using arrows. Lifelines and messages form the core of a sequence diagram.

Messages can be broadly classified into the following categories:

- Synchronous messages
- Asynchronous Messages
- Create message
- Delete Message
- Self-Message
- Reply Message
- Found Message

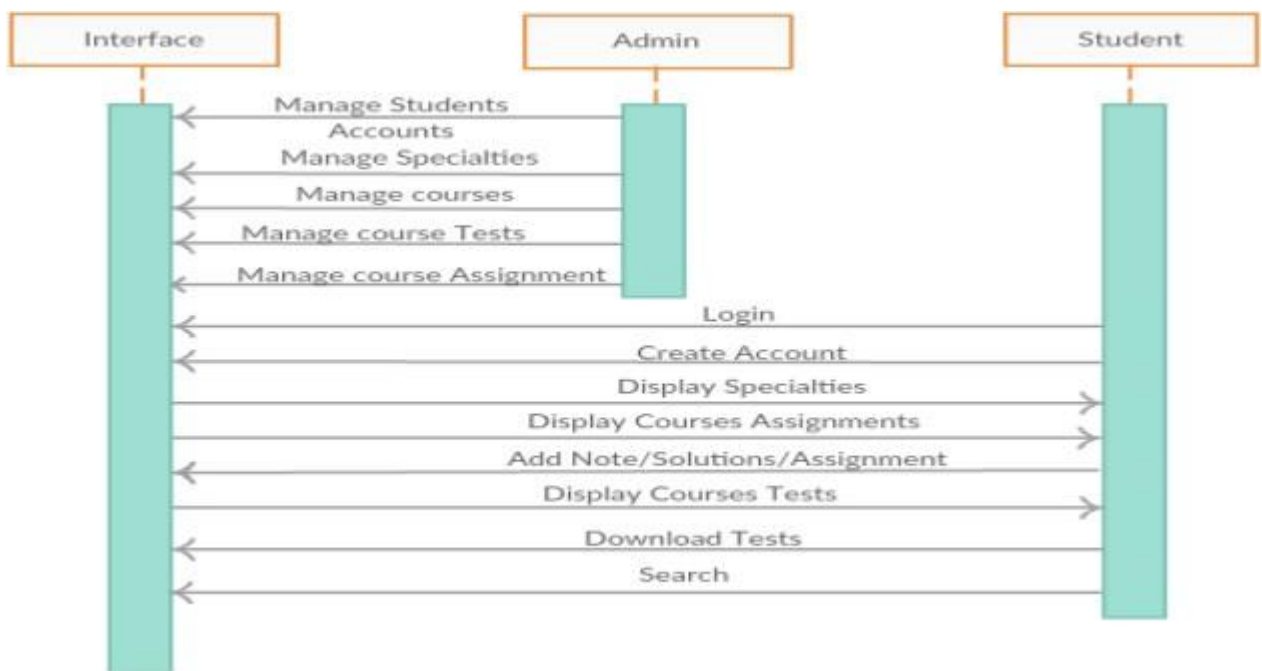
- Lost Message

**iv. Guards** – To model conditions we use guards in UML. They are used when we need to restrict the flow of messages on the pretext of a condition being met. Guards play an important role in letting software developers know the constraints attached to a system or a particular process.

**Uses of sequence diagrams –**

- Used to model and visualize the logic behind a sophisticated function, operation or procedure.
- They are also used to show details of UML use case diagrams.
- Used to understand the detailed functionality of current or future systems.
- Visualize how messages and tasks move between objects or components in a system.

Fig 1 : Sequence diagram for Course4U E-Learning Platform



---

### 4.2.3 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

The basic purposes of activity diagrams is similar to other four diagrams. It captures the dynamic behavior of the system. Other four diagrams are used to show the message flow from one object to another but activity diagram is used to show message flow from one activity to another.

Activity is a particular operation of the system. Activity diagrams are not only used for visualizing the dynamic nature of a system, but they are also used to construct the executable system by using forward and reverse engineering techniques. The only missing thing in the activity diagram is the message part.

It does not show any message flow from one activity to another. Activity diagram is sometimes considered as the flowchart. Although the diagrams look like a flowchart, they are not. It shows different flows such as parallel, branched, concurrent, and single.

The purpose of an activity diagram can be described as –

- Draw the activity flow of a system.
- Describe the sequence from one activity to another.
- Describe the parallel, branched and concurrent flow of the system.

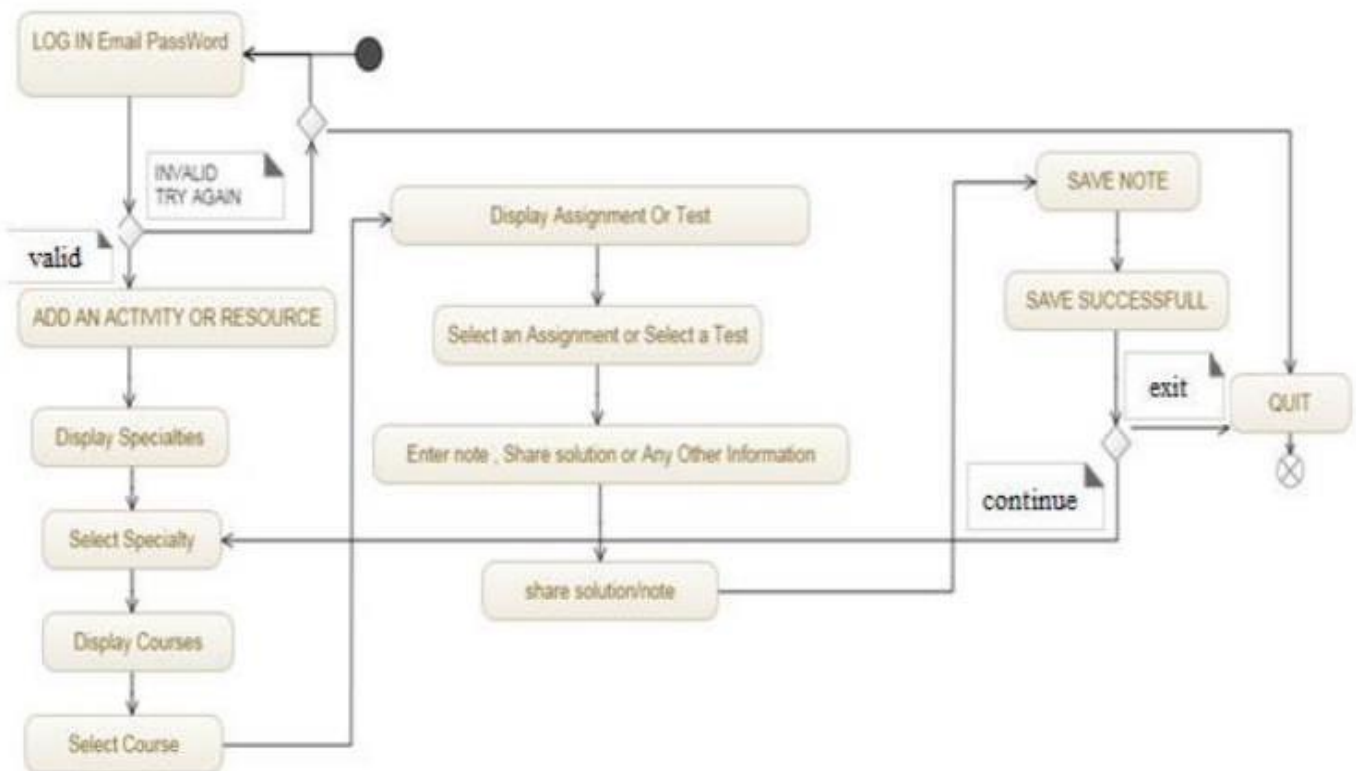
Before drawing an activity diagram, we must have a clear understanding about the elements used in activity diagram. The main element of an activity diagram is the activity itself. An activity is a function performed by the system. After identifying the activities, we need to understand how they are associated with constraints and conditions.

Before drawing an activity diagram, we should identify the following elements –

- Activities
- Association
- Conditions
- Constraints

Once the above-mentioned parameters are identified, we need to make a mental layout of the entire flow. This mental layout is then transformed into an activity diagram.

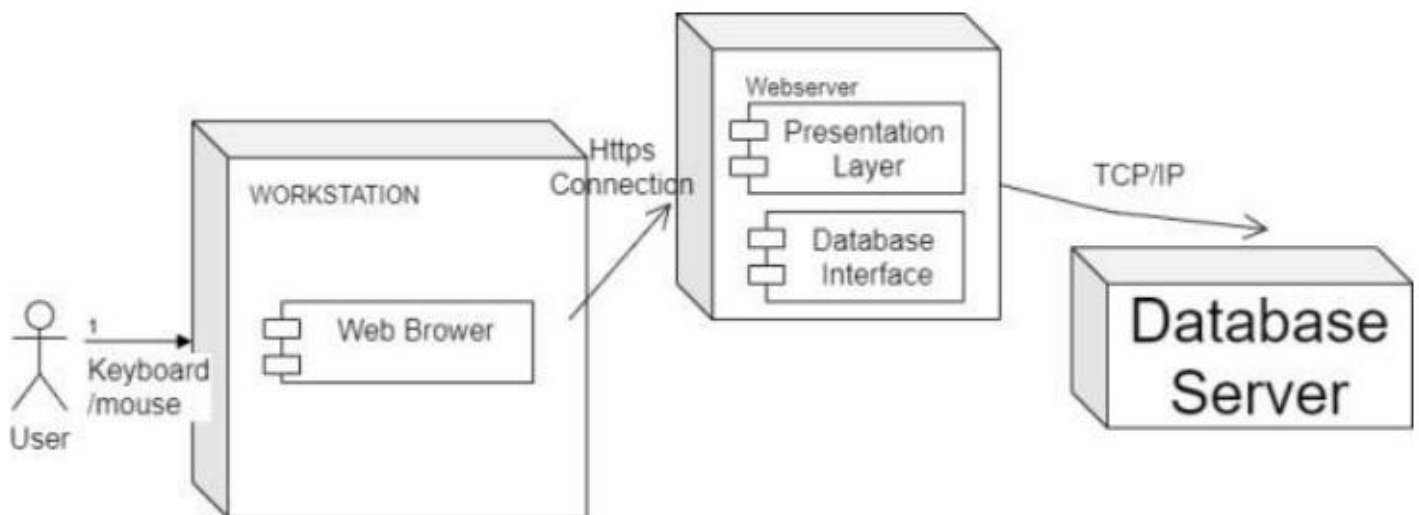
Fig 3: Activity diagram for Course4U E-Learning Platform



### 4.2.4 Deployment Diagram

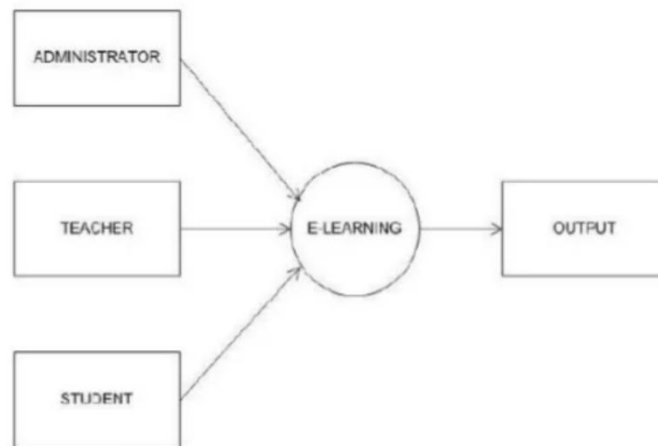
A deployment diagram is a UML diagram type that shows the execution architecture of a system, including nodes such as hardware or software execution environments, and the middleware connecting them. Deployment diagrams are typically used to visualize the physical hardware and software of a system.

Using it you can understand how the system will be physically deployed on the hardware. Deployment diagrams help model the hardware topology of a system compared to other UML diagram types which mostly outline the logical components of a system.



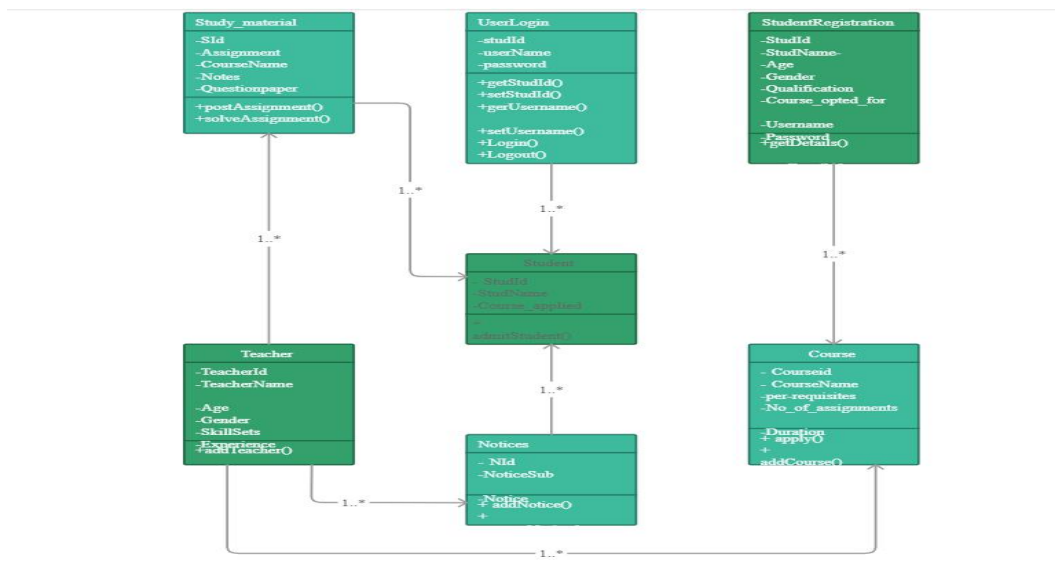
### 4.2.5 Data Flow Diagram

A data-flow diagram is a way of representing a flow of data through a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow — there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart. There are several notations for displaying data-flow diagrams. The notation presented above was described in 1979 by Tom DeMarco as part of structured analysis.



## 4.2.6 Class Diagram

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modeling of object oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

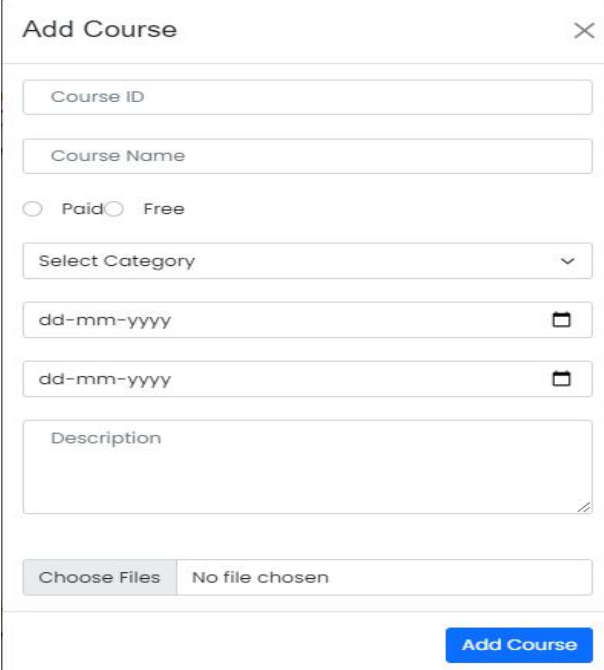




## 4.3 USER INTERFACE DESIGN USING FIGMA

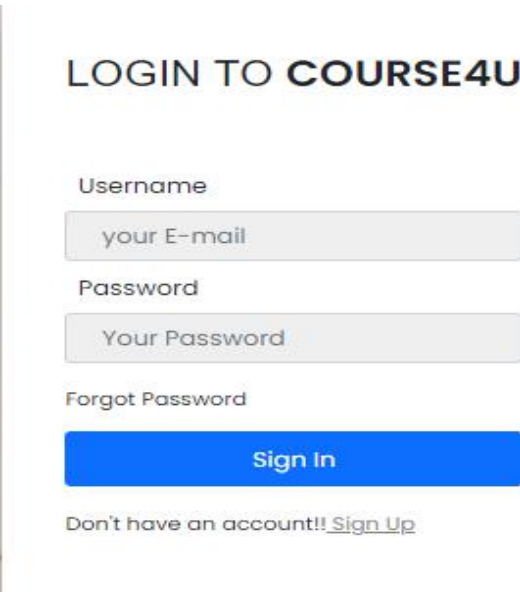
### 4.3.1-INPUT DESIGN

Form Name : Add Course



The 'Add Course' form is a vertical layout with a title bar at the top containing the text 'Add Course' and a close button (X). Below the title bar, the form contains the following elements: a text input field for 'Course ID', a text input field for 'Course Name', a radio button group with 'Paid' and 'Free' options, a dropdown menu for 'Select Category', two date input fields with the placeholder 'dd-mm-yyyy' and calendar icons, a text area for 'Description', a file upload section with a 'Choose Files' button and a 'No file chosen' text, and a blue 'Add Course' button at the bottom right.

Form Name : User Login



The 'User Login' form is a vertical layout with a title 'LOGIN TO COURSE4U' at the top. Below the title, the form contains the following elements: a 'Username' label above a text input field with the placeholder 'your E-mail', a 'Password' label above a text input field with the placeholder 'Your Password', a 'Forgot Password' link, a blue 'Sign In' button, and a text line at the bottom that says 'Don't have an account!!' followed by a 'Sign Up' link.

Form Name : User Registration

REGISTER TO **COURSE4U**

Full Name

your name

E-mail

your E-mail

Password

Your Password

Confirm Password

Your Password

Already have an account!![Sign In](#)

Sign Up

## 4.4 DATABASE DESIGN

A database is an organized mechanism that has the capability of storing information through which a user can retrieve stored information in an effective and efficient manner. The data is the purpose of any database and must be protected.

The database design is a two level process. In the first step, user requirements are gathered together and a database is designed which will meet these requirements as clearly as possible. This step is called Information Level Design and it is taken independent of any individual DBMS.

In the second step, this Information level design is transferred into a design for the specific DBMS that will be used to implement the system in question. This step is called Physical Level Design, concerned with the characteristics of the specific DBMS that will be used. A database design runs parallel with the system design. The organization of the data in the database is aimed to achieve the following two major objectives.

- Data Integrity
- Data independence

### 4.4.1 Database Management System (DBMS)

Non-relational databases (often called NoSQL databases) are different from traditional relational databases in that they store their data in a non-tabular form. Instead, non-relational databases might be based on data structures like documents. A document can be highly detailed while containing a range of different types of information in different formats. This ability to digest and organize various types of information side-by-side makes non-relational databases much more flexible than relational databases.

#### **Relations, Domains & Attributes**

A document data store manages a set of named string fields and object data values in an entity that's referred to as a document. These data stores typically store data in the form of JSON documents. Each field value could be a scalar item, such as a number, or a compound element, such as a list or a parent-child collection. The data in the fields of a document can be encoded in various ways, including XML, YAML, JSON, BSON, or even stored as plain text. The fields within documents are exposed to the storage management system, enabling an application to query and filter data by using the values in these fields.

### Relationships

- The two main keys of prime importance are Primary Key & Foreign Key. Entity Integrity and Referential Integrity Relationships can be established with these keys.
- Entity Integrity enforces that no Primary Key can have null values.
- Referential Integrity enforces that no Primary Key can have null values.
- Referential Integrity for each distinct Foreign Key value, there must exist a matching Primary Key value in the same domain. Other key are Super Key and Candidate Keys.

#### 4.6.2 Normalization

Data are grouped together in the simplest way so that later changes can be made with minimum impact on data structures. Normalization is formal process of data structures in manners that eliminates redundancy and promotes integrity. Normalization is a technique of separating redundant fields and breaking up a large table into a smaller one. It is also used to avoid insertion, deletion, and updating anomalies. Normal form in data modelling use two concepts, keys and relationships. A key uniquely identifies a row in a table. There are two types of keys, primary key and foreign key. A primary key is an element or a combination of elements in a table whose purpose is to identify records from the same table. A foreign key is a column in a table that uniquely identifies record from a different table. All the tables have been normalized up to the third normal form.

As the name implies, it denotes putting things in the normal form. The application developer via normalization tries to achieve a sensible organization of data into proper tables and columns and where names can be easily correlated to the data by the user. Normalization eliminates repeating groups at data and thereby avoids data redundancy which proves to be a great burden on the computer resources. These include:

- ✓ Normalize the data.
- ✓ Choose proper names for the tables and columns.
- ✓ Choose the proper name for the data.

**First Normal Form**

The First Normal Form states that the domain of an attribute must include only atomic values and that the value of any attribute in a tuple must be a single value from the domain of that attribute. In other words 1NF disallows “relations within relations” or “relations as attribute values within tuples”. The only attribute values permitted by 1NF are single atomic or indivisible values. The first step is to put the data into First Normal Form. This can be done by moving data into separate tables where the data is of similar type in each table. Each table is given a Primary Key or Foreign Key as per requirement of the project. In this we form new relations for each non-atomic attribute or nested relation. This eliminated repeating groups of data. A relation is said to be in first normal form if only if it satisfies the constraints that contain the primary key only.

**Second Normal Form**

According to Second Normal Form, for relations where primary key contains multiple attributes, no non-key attribute should be functionally dependent on a part of the primary key. In this we decompose and setup a new relation for each partial key with its dependent attributes. Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it. This step helps in taking out data that is only dependent on a part of the key. A relation is said to be in second normal form if and only if it satisfies all the first normal form conditions for the primary key and every non-primary key attributes of the relation is fully dependent on its primary key alone.

**Third Normal Form**

According to Third Normal Form, Relation should not have a non-key attribute functionally determined by another non-key attribute or by a set of non-key attributes. That is, there should be no transitive dependency on the primary key. In this we decompose and set up relation that includes the non-key attributes that functionally determines other non-key attributes. This step is taken to get rid of anything that does not depend entirely on the Primary Key. A relation is said to be in third normal form if only if it is in second normal form and more over the non key attributes of the relation should not be depend on other non-key attribute.

## TABLE DESIGN

**Table No**           **01**

**Table Name**       **: tbl\_courses**

**Table Description: To store course information**

```
_id: ObjectId("61b90b3d4c24003b08d9c734")
courseStatus: "active"
> courseVideos: Array
courseId: "C001"
courseName: "HTML"
courseType: "Free"
courseCategory: "Computer Science"
courseStartDate: "2021-12-15"
courseEndDate: "2022-01-06"
courseDesc: "You will learn how to add content to your webpage using the basic cont..."
courseLinkVideo: "C:\Users\Sanju\Desktop\MINI_NEW\my-app\src\videos\1. HTML"
__v: 0
```

**Table No**           **02**

**Table Name**       **: tbl\_enrolledcourses**

**Foreign key**       **: userId**

**Table Description: To store enrolled courses information**

```
> _id: ObjectId("61b90c454c24003b08d9c737")
  userId: ObjectId("61b62d8eb2279a3834ea24f7")
  userName: "Jose K"
  courseId: "C001"
  courseName: "HTML"
  __v: 0
```

**Table No            03**

**Table Name : tbl\_users**

**Table Description: To store user information**

```
_id: ObjectId("61b05f70a20ab301d825b1bc")
status: "active"
role: "admin"
fullname: "Admin"
email: "admin@gmail.com"
password: "$2b$10$BPuUBbEj1ceGtf1WNnYVe.KYIng9prjNOMPhz.2HEd2IRF/2vGVwW"
__v: 0
```

```
_id: ObjectId("61b62d8eb2279a3834ea24f7")
status: "active"
role: "student"
fullname: "Jose K"
email: "jose@gmail.com"
password: "$2b$10$QwqBQzJJpreV5Q1RbyWmQ.FjhQ.SZSHfokoUZALw5YEoFBpRpY6RG"
__v: 0
```

**Table No            04**

**Table Name : tbl\_otp**

**Table Description: To store otp**

```
_id: ObjectId('627e85ee8870b014f4e9ab1b')
email: "admin@gmail.com"
code: "2383"
expireIn: 1652459290543
createdAt: 2022-05-13T16:23:10.560+00:00
updatedAt: 2022-05-13T16:23:10.560+00:00
__v: 0
```

```
]
_id: ObjectId('627e86188870b014f4e9ab1c')
email: "admin@gmail.com"
code: "4468"
expireIn: 1652459332756
createdAt: 2022-05-13T16:23:52.757+00:00
updatedAt: 2022-05-13T16:23:52.757+00:00
__v: 0
```

**information**

**Table No        04**

**Table Name : tbl\_paid\_courses**

**Table Description: To store otp information**

```
_id: ObjectId('6289eb028b6db51430e81417')
courseStatus: "active"
> courseVideos: Array
  courseId: "M111"
  courseName: "Maths"
  courseCategory: "Mathematics"
  courseStartDate: "2022-05-22"
  courseEndDate: "2022-05-31"
  courseAmount: 198
  courseDesc: "jvnjzdhvzvznzn"
__v: 0
```



# **CHAPTER 5**

## **SYSTEM TESTING**

## 5.1 INTRODUCTION

Software Testing is the process of executing software in a controlled manner, in order to answer the question - Does the software behave as specified? Software testing is often used in association with the terms verification and validation. Validation is the checking or testing of items, includes software, for conformance and consistency with an associated specification. Software testing is just one kind of verification, which also uses techniques such as reviews, analysis, inspections, and walkthroughs. Validation is the process of checking that what has been specified is what the user actually wanted.

Other activities which are often associated with software testing are static analysis and dynamic analysis. Static analysis investigates the source code of software, looking for problems and gathering metrics without actually executing the code. Dynamic analysis looks at the behavior of software while it is executing, to provide information such as execution traces, timing profiles, and test coverage information.

Testing is a set of activity that can be planned in advanced and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it vital success of the system testing objectives, there are several rules that can serve as testing objectives. They are:

Testing is a process of executing a program with the intent of finding an error.

- A good test case is one that has high possibility of finding an undiscovered error.
- A successful test is one that uncovers an undiscovered error.

If a testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrate that the software function appear to be working according to the specification, that performance requirement appear to have been met.

There are three ways to test program.

- For correctness
- For implementation efficiency
- For computational complexity

Test for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

## 5.2 TEST PLAN

A test plan implies a series of desired course of action to be followed in accomplishing various testing methods. The Test Plan acts as a blue print for the action that is to be followed. The software engineers create a computer program, its documentation and related data structures. The software developers is always responsible for testing the individual units of the programs, ensuring that each performs the function for which it was designed. There is an independent test group (ITG) which is to remove the inherent problems associated with letting the builder to test the thing that has been built. The specific objectives of testing should be stated in measurable terms. So that the mean time to failure, the cost to find and fix the defects, remaining defect density or frequency of occurrence and test work-hours per regression test all should be stated within the test plan.

The levels of testing include:

- ❖ Unit testing
- ❖ Integration Testing
- ❖ Data validation Testing
- ❖ Output Testing

### 5.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design – the software component or module. Using the component level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered scope established for unit testing. The unit testing is white-box oriented, and step can be conducted in parallel for multiple components. The modular interface is tested to ensure that information properly flows into and out of the program unit under test. The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution. Boundary conditions are tested to ensure that all statements in a module have been executed at least once. Finally, all error handling paths are tested.

Tests of data flow across a module interface are required before any other test is initiated. If data do not enter and exit properly, all other tests are moot. Selective testing of execution paths is an essential task during the unit test. Good design dictates that error conditions be anticipated and error handling paths set up to reroute or cleanly terminate processing when an error does occur. Boundary testing is the last task of unit testing step. Software often fails at its boundaries.

Unit testing was done in Sell-Soft System by treating each module as separate entity and testing each one of them with a wide spectrum of test inputs. Some flaws in the internal logic of the modules were found and were rectified. After coding each module is tested and run individually. All unnecessary code were removed and ensured that all modules are working, and gives the expected result.

### **5.2.2 Integration Testing**

Integration testing is systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design. The entire program is tested as whole. Correction is difficult because isolation of causes is complicated by vast expanse of entire program. Once these errors are corrected, new ones appear and the process continues in a seemingly endless loop. After performing unit testing in the System all the modules were integrated to test for any inconsistencies in the interfaces. Moreover differences in program structures were removed and a unique program structure was evolved.

### **5.2.3 Validation Testing or System Testing**

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as BlackBox testing or System tests.

Black Box testing method focuses on the functional requirements of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that willfully exercise all functional requirements for a program.

Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external data access, performance errors and initialization errors and termination errors.

### 5.2.4 Output Testing or User Acceptance Testing

The system considered is tested for user acceptance; here it should satisfy the firm's need. The software should keep in touch with perspective system; user at the time of developing and making changes whenever required. This done with respect to the following points:

- Input Screen Designs,
- Output Screen Designs,

The above testing is done taking various kinds of test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested using that test data. While testing the system by which test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

### 5.2.5 Selenium Testing

Selenium is one of the most widely used open-source Web UI (User Interface) automation testing suite. It was originally developed by Jason Huggins in 2004 as an internal tool at Thought Works. Selenium supports automation across different browsers, platforms and programming languages.

Selenium can be easily deployed on platforms such as Windows, Linux, Solaris and Macintosh. Moreover, it supports OS (Operating System) for mobile applications like iOS, windows mobile and android. Selenium supports a variety of programming languages through the use of drivers specific to each language.

Languages supported by Selenium include C#, Java, Perl, PHP, Python and Ruby. Currently, Selenium Web driver is most popular with Java and C#. Selenium test scripts can be coded in any of the supported programming languages and can be run directly in most modern web browsers.

Browsers supported by Selenium include Internet Explorer, Mozilla Firefox, Google Chrome and Safari. Selenium can be used to automate functional tests and can be integrated with automation test tools such as Maven, Jenkins, & Docker to achieve continuous testing. It can also be integrated with tools such as TestNG, & JUnit for managing test cases and generating reports.

## Login

```
package tests;
```

```
import org.openqa.selenium.By;
```

```
import org.openqa.selenium.WebDriver;
```

```
import browserimplementation.DriverSetup;
```

```
public class Firsts {
```

```
    public static WebDriver driver;
```

```
    public static void main(String[] args) throws InterruptedException {
```

```
        // TODO Auto-generated method stub
```

```
        driver = DriverSetup.getWebDriver("localhost:3000/login");
```

```
        //login-Invalid case
```

```
        driver.findElement(By.id("username")).sendKeys("athul");
```

```
        driver.findElement(By.id("password")).sendKeys("123");
```

```
        driver.findElement(By.name("sbtn")).click();
```

```
        if(driver.findElement(By.xpath("/html/body/center/div")).isDisplayed()) {
```

```
            System.out.println("Error");
```

```
        }
```

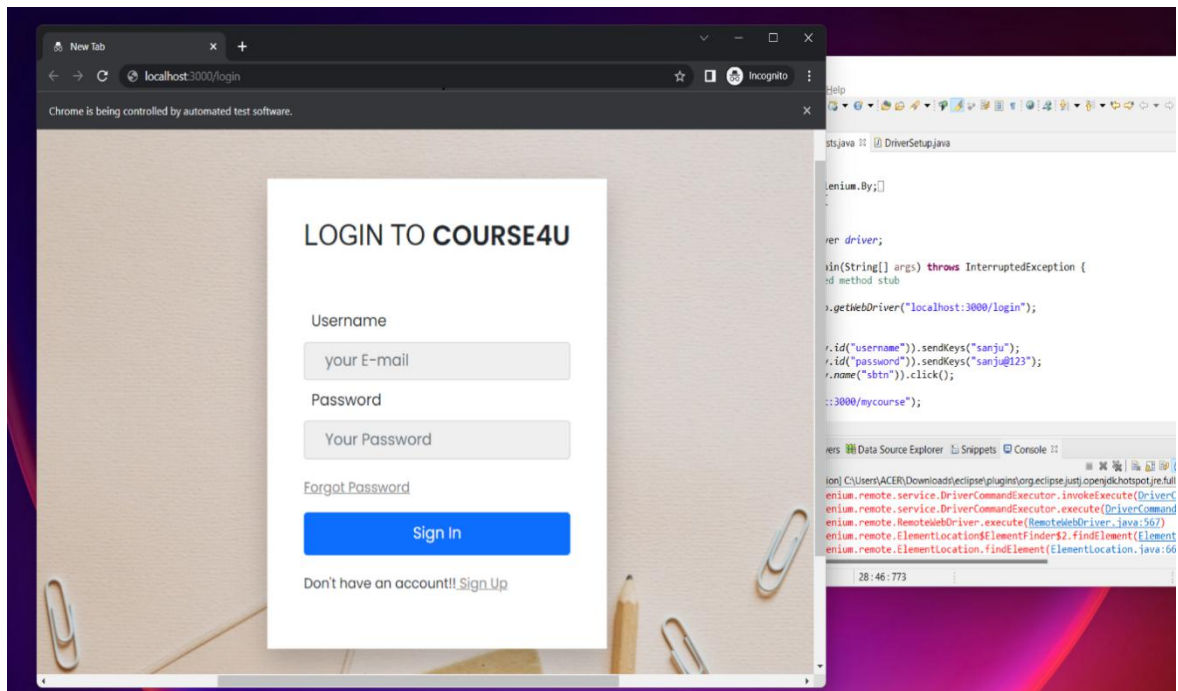
```
        Thread.sleep(2000);
```

```
        driver.findElement(By.name("FNP")).click();
```

```
        driver.quit();
```

```
    }
```

```
}
```



## Mycourse

package tests;

```
import org.openqa.selenium.By;
```

```
import org.openqa.selenium.WebDriver;
```

```
import browserimplementation.DriverSetup;
```

```
public class Firsts {
```

```
    public static WebDriver driver;
```

```
    public static void main(String[] args) throws InterruptedException {  
        // TODO Auto-generated method stub
```

```
        driver = DriverSetup.getWebDriver("localhost:3000/login");
```

```
        //login-Invalid case
```

```
        driver.findElement(By.id("username")).sendKeys("athul");
```

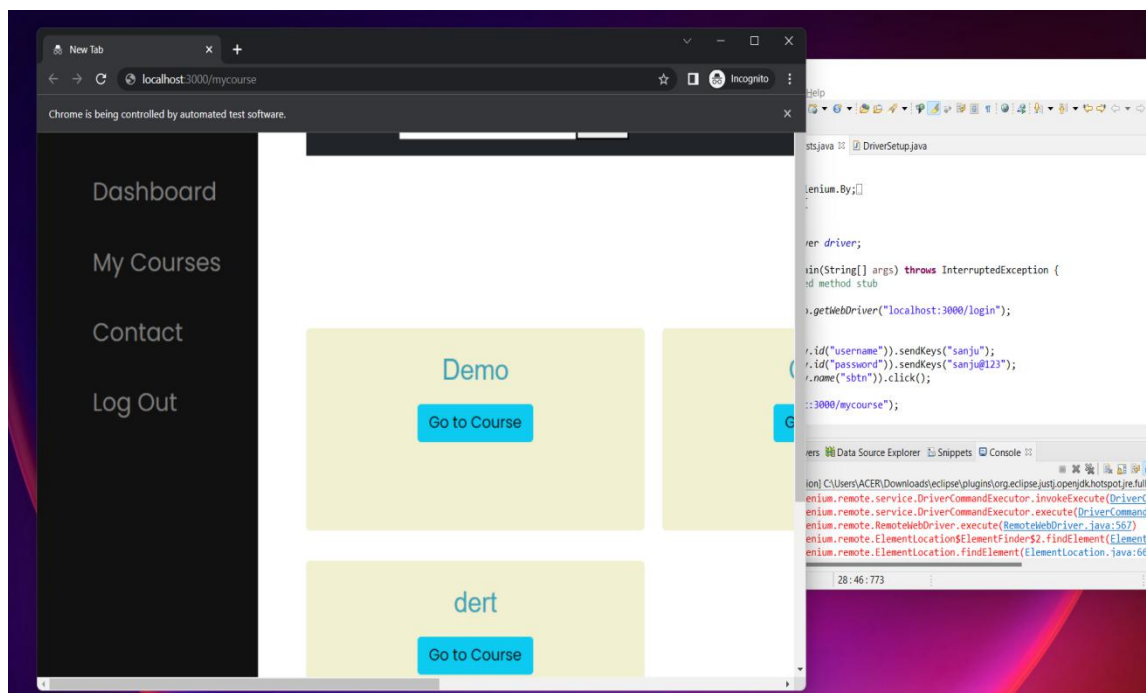
```
        driver.findElement(By.id("password")).sendKeys("123");
```

```
        driver.findElement(By.name("sbtn")).click();
```

```
driver.get("localhost:3000/mycourse");

if(driver.findElement(By.xpath("/html/body/center/div")).isDisplayed()) {
    System.out.println("Error");
}
Thread.sleep(2000);
driver.findElement(By.id("gotobtn")).click();
driver.quit();
}

}
```







Test Case 1					
Project Name: Course4U - E-Learning Platform					
Login Test Case					
Test Case ID: Fun_1			Test Designed By: Sanju Abraham Binu		
Test Priority(Low/Medium/High): High			Test Designed Date: 24-05-2022		
Module Name: Login Screen			Test Executed By : Mr Rony Tom		
Test Title : Verify login with validemail and password			Test Execution Date: 24-05-2022		
Description: Test the Login Page					
Pre-Condition : User has valid email id and password					
Step	Test Step	Test Data	Expected Result	Actual Result	Status(Pass/Fail)
1	Navigation toLogin Page		Login Page should be displayed	Login page displayed	Pass
2	Provide Valid Email	User Name: jose@gmail.com	User should beable to Login	User Logged inand navigated to Student Dashboard with records	Pass
3	Provide Valid Password	Password: Jose@123			
4	Click on Sign In button				
5	Provide Invalid Email Id or password	Username: mine@12 Password: Mine	User shouldnot be able to Login	Message for enter valid email id or password displayed	Pass
6	Provide Null Username or Password	Username: null Password: null			
7	Click on Sign In button				
Post-Condition: User is validated with database and successfully login into account.The Account session details are logged in database					

## **CHAPTER 6**

# **IMPLEMENTATION**

## 6.1 INTRODUCTION

Implementation is the stage of the project where the theoretical design is turned into a working system. It can be considered to be the most crucial stage in achieving a successful new system gaining the users confidence that the new system will work and will be effective and accurate. It is primarily concerned with user training and documentation. Conversion usually takes place about the same time the user is being trained or later. Implementation simply means convening a new system design into operation, which is the process of converting a new revised system design into an operational one.

At this stage the main work load, the greatest upheaval and the major impact on the existing system shifts to the user department. If the implementation is not carefully planned or controlled, it can create chaos and confusion.

Implementation includes all those activities that take place to convert from the existing system to the new system. The new system may be a totally new, replacing an existing manual or automated system or it may be a modification to an existing system. Proper implementation is essential to provide a reliable system to meet organization requirements. The process of putting the developed system in actual use is called system implementation. This includes all those activities that take place to convert from the old system to the new system. The system can be implemented only after through testing is done and if it is found to be working according to the specifications. The system personnel check the feasibility of the system. The more complex the system being implemented, the more involved will be the system analysis and design effort required to implement the three main aspects: education and training, system testing and changeover.

The implementation state involves the following tasks:

- ☐ Careful planning.
- ☐ Investigation of system and constraints.
- ☐ Design of methods to achieve the changeover.

## 6.2 IMPLEMENTATION PROCEDURES

Implementation of software refers to the final installation of the package in its real environment, to the satisfaction of the intended uses and the operation of the system. In many organizations someone who will not be operating it, will commission the software development project. In the initial stage people doubt about the software but we have to

ensure that the resistance does not build up, as one has to make sure that:

- The active user must be aware of the benefits of using the new system.
- Their confidence in the software is built up.
- Proper guidance is imparted to the user so that he is comfortable in using the application.

Before going ahead and viewing the system, the user must know that for viewing the result, the server program should be running in the server. If the server object is not up running on the server, the actual process won't take place.

### **6.2.1 User Training**

User training is designed to prepare the user for testing and converting the system. To achieve the objective and benefits expected from computer based system, it is essential for the people who will be involved to be confident of their role in the new system. As system becomes more complex, the need for training is more important. By user training the user comes to know how to enter data, respond to error messages, interrogate the database and call up routine that will produce reports and perform other necessary functions.

### **6.2.2 Training on the Application Software**

After providing the necessary basic training on computer awareness the user will have to be trained on the new application software. This will give the underlying philosophy of the use of the new system such as the screen flow, screen design type of help on the screen, type of errors while entering the data, the corresponding validation check at each entry and the ways to correct the data entered. It should then cover information needed by the specific user/ group to use the system or part of the system while imparting the training of the program on the application. This training may be different across different user groups and across different levels of hierarchy

### **6.2.3 System Maintenance**

Maintenance is the enigma of system development. The maintenance phase of the software cycle is the time in which a software product performs useful work. After a system is successfully implemented, it should be maintained in a proper manner. System maintenance is an important aspect in the software development life cycle. The need for system maintenance is for it to make adaptable to the changes in the system environment. Software maintenance is of course, far more than "Finding Mistakes".

## **CHAPTER 7**

### **CONCLUSION AND FUTURE SCOPE**

## **7.1 CONCLUSION**

The current system working technology is old fashioned there is no single platform for various features. The proposed system provides all the features under single roof. The students can study any course according to their interest. The teachers can also add courses in which there are expert it. The students can also interact with the teachers personally also. Continues monitoring of the student is also done. For the paid courses the payment can also done online.

## **7.2 FUTURE SCOPE**

- The proposed system is designed in such a way that the payment should be done in online mode.
- Student can able to do advanced search options
- Customers can able to add complaints and feedbacks etc.
- Contractors can able to view daily progress report etc.
- Data security can be enhanced.

## **CHAPTER 8**

## **BIBLIOGRAPHY**



**REFERENCES:**

- Gary B. Shelly, Harry J. Rosenblatt, “*System Analysis and Design*”, 2009.
- Roger S Pressman, “*Software Engineering*”, 1994.
- PankajJalote, “*Software engineering: a precise approach*”, 2006.
- James lee and Brent ware Addison, “Open source web development with LAMP”, 2003
- IEEE Std 1016 Recommended Practice for Software Design Descriptions.

**WEBSITES:**

- [www.w3schools.com](http://www.w3schools.com)
- [www.bootstrap.com](http://www.bootstrap.com)
- <https://reactjs.org/>
- <https://app.diagrams.net>
- <https://nodejs.org/>
- [www.agilemodeling.com/artifacts/useCaseDiagram.html](http://www.agilemodeling.com/artifacts/useCaseDiagram.html)

## **CHAPTER 9**

## **APPENDIX**

## 9.1 Sample Code

### MyCourse.js

```
import React, { useState, useEffect } from "react";
import { Link, useHistory } from "react-router-dom";
import Navbar from "../UserDashboard/navbar/navbar";
import "../userhome.css";
import { Modal, Button, Form } from "react-bootstrap";
import GooglePayButton from '@google-pay/button-react'
const axios = require("axios");

const UserHome = () => {
  const history = useHistory();
  let search=window.localStorage.getItem("search");
  const [courses, setCourses] = useState(null);
  const [paidcourses, setPaidCourses] = useState(null);
  const [showmodal, setShowModal] = useState(false);
  const userId = window.localStorage.getItem("id");
  const userName = window.localStorage.getItem("username");
  const [courseName, setCourseName] = useState();
  const [courseCategory, setCourseCategory] = useState();
  const [courseId, setCourseId] = useState();
  const [courseDesc, setCourseDesc] = useState();
  const [courseAmount, setCourseAmount] = useState();
  const [unid, setCourseUId] = useState();
  let button="";

  async function getCourses() {
    const data={
      search:search,
    }
    let response = await axios.get("http://localhost:5000/getcourses",data);
    if (response.status === 200) {
      setCourses(response.data.courses);
    }
  }

  async function getPaidCourses() {
    let response = await axios.get("http://localhost:5000/getpaidcourses");
    if (response.status === 200) {
      setPaidCourses(response.data.tcourses);
    }
  }

  const handleClose = () => setShowModal(false);
  const handleShow = (unid,cid, cname, ccategory, cdesc, camt) => {
    setCourseUId(unid);
    setCourseId(cid);
    setCourseName(cname);
    setCourseCategory(ccategory);
    setCourseDesc(cdesc);
    setCourseAmount(camt);
    setShowModal(true);
  }
}
```

---

```

    };

    const logoutHandler = () => {
      window.localStorage.removeItem("id");
      history.push("/login");
    };

    async function handleEnrol(unid,cid, cname) {
      let data = {
        unid:unid,
        uid: userId,
        uname: userName,
        cid: cid,
        cname: cname,
      };
      let response = await axios.post(
        "http://localhost:5000/addenrolledcourses",
        data
      );
      if (response.status === 200) {
        alert("Enrolled");
      } else {
        console.log("failed");
      }
    }
    if(courseAmount===null){
      button="Enroll";
    }else{
      button="Pay & Enroll"
    }
    useEffect(() => {
      const isToken = localStorage.getItem("id");
      if (!isToken) {
        history.push("/login");
      } else {
        getCourses();
        getPaidCourses();
      }
    }, []);
    const afterpayment =()=>{
      alert("payment successfull");
    }
  }
  const data = {};
  return (
    <div>
      <div id="mySidenav" className="sidenav">
        <Link to="/userhome">
          <a>Dashboard</a>
        </Link>
        <Link to="/mycourse">
          <a>My Courses</a>
        </Link>

        <Link to="">
          <a>Contact</a>
        </Link>
      </div>
    </div>
  );

```

---

---

```

    <Link>
      <a onClick={logoutHandler}>Log Out</a>
    </Link>
  </div>
</Navbar />
<Modal show={showmodal} onHide={handleClose}>
  <Modal.Header closeButton>
    <Modal.Title>Course Details</Modal.Title>
  </Modal.Header>
  <Modal.Body>
    <h3>{courseName}</h3>
    <p>{courseCategory}</p>
    <h5>{courseDesc}</h5>

  </Modal.Body>
  <Modal.Footer>
    <Button
      variant="primary"
      onClick={() => {
        handleEnrol(unid,courseId, courseName);
      }}
    >

      Enroll Now
    </Button>
  </Modal.Footer>
</Modal>
<h2 style={{ color: "red", marginLeft: "20%" }}>Free Courses</h2>
<div className="cardbody">
  {courses &&
    courses.length > 0 &&
    courses.map((p) => {
      return (
        <div className="row1">
          <div className="column1">
            <div className="sub-card" style={{ marginTop: "-40%" }}>
              <h3>{p.courseName}</h3>
              <p>{p.courseCategory}</p>
              <Button
                variant="info"
                onClick={() =>
                  handleShow(
                    p._id,
                    p.courseId,
                    p.courseName,
                    p.courseCategory,
                    p.courseDesc,
                    p.courseAmount
                  )
                }
              >
                View Details
              </Button>{" "}
            </div>
          </div>
        </div>
      )
    })
  }
</div>
</div>
);
}}

```

---



---

```

        console.log("payment sucess");
        console.log('Success', paymentRequest);
    }}
    onPaymentAuthorized={ (paymentData) => {
        console.log(
            'Payment Authorised Success',
            paymentData
        );
        return { transactionState: 'SUCCESS' };
    }}

    existingPaymentMethodRequired="false"
    buttonColor="light"
    buttonType="pay"
/>
</div>
</div>
</div>
    );
    })}
</div>
</div>
);
};

export default UserHome;

```

### App.js

```

import Register from './Components/Authentication/register';
import Login from './Components/Authentication/login';
import Landing from './Components/Landing/landing';
import Dashboard from './Components/Admin/Dashboard/Dashboard'
//import Admin from './Components/Admin/adminip';
import { BrowserRouter as Router, Switch, Route } from "react-router-dom";
import Useradmin from './Components/Admin/UserAdmin/useradmin';
import Teacheradmin from
    './Components/Admin/TeacherAdmin/teacheradmin';
import Userdashboard from './Components/dashboard/dashboard';
import UserHome from './Components/UserDashboard/userhome';
import Main from './Components/Admin/Main/Main'
import Mycourse from './Components/UserDashboard/mycourse';
import Videodashhtml from
    './Components/UserDashboard/videodashhtml';
import Videodashcss from
    './Components/UserDashboard/videodashcss';

```

---

---

```
import Videodashjs from './Components/UserDashboard/videodashjs';
import Html1 from
    './Components/UserDashboard/videosCollection/html1';
import Html2 from
    './Components/UserDashboard/videosCollection/html2';
import Css1 from
    './Components/UserDashboard/videosCollection/css1';
import Css2 from
    './Components/UserDashboard/videosCollection/css2';
import Js1 from './Components/UserDashboard/videosCollection/js1';
import Js2 from './Components/UserDashboard/videosCollection/js2';
import React1 from
    './Components/UserDashboard/videosCollection/react1';
import Videodashreact from
    './Components/UserDashboard/videodashreact';
```



```

function App()
{
  return (
    <div>
      <Router>
        <Switch>
          <Route exact path = "/" component={Landing}/>
          <Route path = "/Admindashboard" component={Main}/>
          <Route path = "/login" component={Login}/>
          <Route path = "/register" component={Register}/>
          <Route path = "/useradmin" component={Useradmin}/>
          <Route path = "/userhome" component={UserHome}/>
          <Route path = "/mycourse" component={Mycourse}/>
          <Route path = "/videodashhtml" component={Videodashhtml}/>
          <Route path = "/videodashcss" component={Videodashcss}/>
          <Route path = "/videodashjs" component={Videodashjs}/>
          <Route path = "/videodashreact" component={Videodashreact}/>
          <Route path = "/html1" component={Html1}/>
          <Route path = "/html2" component={Html2}/>
          <Route path = "/css1" component={Css1}/>
          <Route path = "/css2" component={Css2}/>
          <Route path = "/js1" component={Js1}/>
          <Route path = "/js2" component={Js2}/>
          <Route path = "/react1" component={React1}/>
        </Switch>
      </Router>
    </div>
  );
}

```

```
export default App;
```

### Backend Index.js

```

var express = require("express");
const nodemailer = require('nodemailer');
const cors = require('cors');
var router = express.Router();
var courses = require("../model/addcourse");
var users = require("../model/userModel");

```

**E-Learning Platform**

```

var Enrolled = require("./model/enrolledcourses");
var tcourses=require("./model/teacherCourse");
var bcrypt = require("bcrypt");
const jwt = require("jsonwebtoken");
var Otp = require("./model/otp");
const fileupload = require("express-fileupload");
const bodyParser = require('body-parser');
const multer = require('multer');

const app = express();

app.use(cors());
app.use(fileupload());
app.use(express.static("files"));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

require("dotenv").config();

/* GET home page. */
router.get("/", function (req, res, next) {
  res.render("index", { title: "Sanju" });
});

router.post("/register", async (req, res, next) => {

  var user = new users({
    fullname: req.body.fullname,
    email: req.body.email,
    password: req.body.password,
    role:req.body.role,
  });
  var response = await user.save();

  if (response) {
    res.status(200).json({ message: "success" });
  } else {
    res.status(401).json({ status: "failed" });
  }
});
router.post("/login", async (req, res, next) => {
  try {
    //validation
    if (!req.body.email || !req.body.password)
      return res.status(400).json({
        status: false,
        message: "Validation Failed",
      });

    const user = await users.findOne({
      email: req.body.email,
    });

    if (user.status === "active") {

```

```

    console.log(user);
    if (!user)
        return res.status(404).json({
            status: false,
            message: "User does not exist",
        });
    const pwdMatch = await bcrypt.compare(req.body.password, user.password);

    if (!pwdMatch)
        return res.status(401).json({
            status: false,
            message: "Password Incorrect",
        });

    const token = jwt.sign(
        { userid: user._id, email: user.email },
        process.env.SECRET_CODE,
        { expiresIn: "1d" }
    );
    res.status(200).send({
        token: token,
        username: user.fullname,
        email: user.email,
        role: user.role,
        id: user._id,
    });
} else {
    return res.status(500).send({
        message: "User Denied",
    });
}
} catch (err) {
    return res.status(400).json({
        status: false,
        message: "Something went wrong",
        data: err,
    });
}
});
router.get("/getcourses", async (req, res) => {
    await courses
        .find()
        .exec()
        .then((response) => {
            console.log(response);
            if (response) {
                res.status(200).json({ courses: response });
            } else {
                res.status(401).json({ status: "failed" });
            }
        });
});
router.post("/getcoursebyId", async (req, res) => {
    await courses
        .findById(req.body.id)
        .exec()
        .then((response) => {
            console.log(response);
            if (response) {

```

**E-Learning Platform**

```

        res.status(200).json({ courses: response });
      } else {
        res.status(401).json({ status: "failed" });
      }
    });
  });
  router.post("/myCourses", async (req, res) => {
    let id = req.body.id;
    await Enrolled.find({ userId: id })
      .exec()
      .then((response) => {
        console.log(response);
        if (response) {
          res.status(200).json({ courses: response });
        } else {
          res.status(401).json({ status: "failed" });
        }
      });
  });
  router.get("/getStudents", async (req, res) => {
    await users
      .find({ role: "student" })
      .exec()
      .then((response) => {
        console.log(response);
        if (response) {
          res.status(200).json({ users: response });
        } else {
          res.status(401).json({ status: "failed" });
        }
      });
  });
});

// router.post("/addcourse", async (req, res, next) => {
//   console.log("dawdasd", req.body.cenddate,);
//   var course = new courses({
//     courseId: req.body.cid,
//     courseName: req.body.cname,
//     courseType: req.body.ctype,
//     courseCategory: req.body.ccategory,
//     courseStartDate: req.body.cstartdate,
//     courseEndDate: req.body.cenddate,
//     courseDesc: req.body.cdsc,
//     courseLinkVideo : req.body.clinkvideo,
//   });
//   var response = await course.save();
//   console.log(response);
//   if (response) {
//     res.status(200).json({ response });
//   } else {
//     res.status(401).json({ status: "failed" });
//   }
// });

router.post("/addcourse", async (req, res, next) => {
  let imgArr = [];

  if (req.files) {
    let coverImage = req.files.VideoLink;

```

**E-Learning Platform**

```

    coverImage.forEach((cover) => {
      let coverName = Date.now();
      cover.mv(
        "C:/Users/Futurense/Desktop/MINI_NEW/my-server/routes/videos/" +
        coverName +
        ".mp4"
      );
      imgArr.push(coverName + ".mp4");
    });
  }

  let Courses = new courses({
    courseId: req.body.cid,
    courseName: req.body.cname,
    courseType: req.body.ctype,
    courseCategory: req.body.ccategory,
    courseStartDate: req.body.cstartdate,
    courseEndDate: req.body.cenddate,
    courseDesc: req.body.cdsc,
    courseVideos: imgArr,
  });

  console.log("My" + Courses);

  Courses.save()
    .then((courses) => {
      return res.status(200).send({
        message: courses,
      });
    })
    .catch((error) => {
      return res.status(500).send({
        message: error.message,
      });
    });
});

//Deactivate course
router.post("/deleteCourse", async (req, res, next) => {
  let courseId = req.body.courseID;
  let Status = {
    courseStatus: "blocked",
  };
  courses
    .findByIdAndUpdate(courseID, { $set: Status })
    .then((user) => {
      return res.status(200).send({
        message: "Course Removed",
      });
    })
    .catch((error) => {
      return res.status(500).send({
        message: error.message,
      });
    });
});

//delete user
router.post("/deleteUser", async (req, res, next) => {
  let userID = req.body.userID;

```

**E-Learning Platform**

```

    users
      .findByIdAndDelete(userID)
      .then((user) => {
        return res.status(200).send({
          message: "User Deleted Successfully",
        });
      })
      .catch((error) => {
        return res.status(500).send({
          message: error.message,
        });
      });
  });
//block user
router.post("/blockUser", async (req, res, next) => {
  let userID = req.body.userID;
  let userStatus = {
    status: "inactive",
  };
  users
    .findByIdAndUpdate(userID, { $set: userStatus })
    .then((user) => {
      return res.status(200).send({
        message: "User Blocked",
      });
    })
    .catch((error) => {
      return res.status(500).send({
        message: error.message,
      });
    });
});
//Reactivate a User
router.post("/activateUser", async (req, res, next) => {
  let userID = req.body.userID;
  let userStatus = {
    status: "active",
  };
  users
    .findByIdAndUpdate(userID, { $set: userStatus })
    .then((user) => {
      return res.status(200).send({
        message: "User ReActivated",
      });
    })
    .catch((error) => {
      return res.status(500).send({
        message: error.message,
      });
    });
});

router.post("/updateCourse", async (req, res, next) => {
  let CourseID = req.body.courseID;

  let CourseUpdated = {
    courseName: req.body.cname,
    courseType: req.body.ctype,
    courseCategory: req.body.ccategory,
  };

```

**E-Learning Platform**

```

    courseStartDate: req.body.cstartdate,
    courseEndDate: req.body.cenddate,
    courseDesc: req.body.cdsc,
    courseLinkVideo: req.body.clinkvideo,
  };
  await courses
    .findByIdAndUpdate(CourseID, CourseUpdated, {
      new: true,
      runValidators: true,
      useFindAndModify: false,
    })
    .then((response) => {
      console.log("res", response);
      return res.status(200).send({
        message: "Course Updated",
      });
    })
    .catch((error) => {
      console.log(error.message);
      return res.status(500).send({
        message: error.message,
      });
    });
});
router.post("/addenrolledcourses", async (req, res, next) => {
  var enrolled = new Enrolled({
    userId: req.body.uid,
    courseUniqueId: req.body.unid,
    userName: req.body.uname,
    courseId: req.body.cid,
    courseName: req.body.cname,
  });
  var response = await enrolled.save();
  //console.log(response);
  if (response) {
    res.status(200).json({ response });
  } else {
    res.status(401).json({ status: "failed" });
  }
});

router.post("/sendEmail", async (req, res, next) => {
  let data = await users.findOne({
    email: req.body.email,
  });
  const responseType = {};
  if (data) {
    let otpcode = Math.floor(Math.random() * 10000 + 1);
    let otpData = new Otp({
      email: req.body.email,
      code: otpcode,
      expireIn: new Date().getTime() + 300 * 1000,
    });
    let otpResponse = await otpData.save();
    responseType.statusText = "success";
    responseType.message = "Please check your email id";
  } else {
    responseType.statusText = "error";
    responseType.message = "Email id not exist";
  }
});

```

```

    }
    res.status(200).json(responseType);
  });

  router.post("/changepass", async (req, res, next) => {
    let data = await Otp.find({ email: req.body.email, code: req.body.otpcode });
    const response = {};
    let myotp = req.body.otpcode;
    let usern = await Otp.findOne({ email: req.body.email });
    console.log(myotp)
    console.log(usern.code)
    if (myotp == usern.code) {
      if (data) {
        let currentTime = new Date().getTime();
        let diff = data.expireIn - currentTime;
        if (diff < 0) {
          response.message = "Timed out";
          response.statusText = "error";
        } else {
          let user = await users.findOne({ email: req.body.email });
          user.password = req.body.password;
          user.save();
          response.message = "Password Changed Successfully";
          response.statusText = "success";
          console.log(response.data);
        }
      } else {
        response.message = "Invalid Otp";
        response.statusText = "error";
      }
    }
    else{
      response.message = "INVALID OTP"
      response.statusText = "error";
    }
    console.log(response.data)
    res.status(200).json(response);
  });

```

```

// const mailer = (email, otp) => {
//   var nodemailer = require("nodemailer");
//   var transporter = nodemailer.createTransport({
//     service: "gmail",
//     port: 587,
//     secure: false,
//     auth: {
//       user: "sanjuabraham321@gmail.com",
//       pass: "Sanjusanju@143",
//     },
//   });
// };
// var mailOptions = {
//   from: "sanjuabraham321@gmail.com",
//   to: "sanju@gmail.com",
//   subject: "Password Reset",
//   text: "Thank you",
// };
// transporter.sendEmail(mailOptions, function (error, info) {

```



**E-Learning Platform**

```

//      if (error) {
//          console.log(error);
//      } else {
//          console.log("Email Sent: " + info.response);
//      }
//  });
// };

// const storage = multer.diskStorage({
//   destination: function (req, file, cb) {
//     cb(null, 'images') ;
//   },
//   filename: function(req, file, cb) {
//     cb(null, Date.now() + '-' + '-' + file.originalname ) ;
//   }
// });

// const upload = multer({storage}).array('file');

//add paid course
router.post("/addpaidcourse", async (req, res, next) => {

  // const newpath = __dirname + "/files/";
  // const file = req.files.file;
  // const filename = file.name;

  // file.mv(`${newpath}${filename}`, (err) => {
  //   if (err) {
  //     res.status(500).send({ message: "File upload failed", code: 200 });
  //   }
  //   res.status(200).send({ message: "File Uploaded", code: 200 });
  // });

  let imgArr = [];

  if (req.files) {
    let coverImage = req.files.VideoLink;
    coverImage.forEach((cover) => {
      let coverName = Date.now();
      cover.mv(
        "C:/Users/Futurense/Desktop/MINI_NEW/my-server/routes/videos/" +
        coverName +
        ".mp4"
      );
      imgArr.push(coverName + ".mp4");
    });
  }

  let Tcourses = new tcourses({
    courseId: req.body.cid,
    courseName: req.body.cname,
    courseCategory: req.body.cccategory,

```

**E-Learning Platform**

```

    courseStartDate: req.body.cstartdate,
    courseEndDate: req.body.cenddate,
    courseAmount: req.body.camt,
    courseDesc: req.body.cdasc,
    courseVideos: imgArr,
  });
  console.log(req.body)

  Tcourses.save()
    .then((tcourses) => {
      return res.status(200).send({
        message: tcourses,
      });
    })
    .catch((error) => {
      return res.status(500).send({
        message: error.message,
      });
    });
  // upload(req, res, (err) =>{
  //   if(err){
  //     return res.status(500).json(err)
  //   }
  //   else{
  //     return res.status(200).send(req.file)
  //   }
  // })
});

//get paid courses
router.get("/getpaidcourses", async (req, res) => {
  await tcourses
    .find()
    .exec()
    .then((response) => {
      console.log(response);
      if (response) {
        res.status(200).json({ tcourses: response });
      } else {
        res.status(401).json({ status: "failed" });
      }
    })
  });

//update paid course
router.post("/updatePaidCourse", async (req, res, next) => {
  let CourseID = req.body.courseID;

  let CourseUpdated = {
    courseName: req.body.cname,
    courseAmount: req.body.camt,
    courseCategory: req.body.ccategory,
    courseStartDate: req.body.cstartdate,
    courseEndDate: req.body.cenddate,
    courseDesc: req.body.cdasc,
    courseLinkVideo: req.body.clinkvideo,
  };
  await tcourses
    .findByIdAndUpdate(CourseID, CourseUpdated, {

```

**E-Learning Platform**

```

    new: true,
    runValidators: true,
    useFindAndModify: false,
  })
  .then((response) => {
    console.log("res", response);
    return res.status(200).send({
      message: "Course Updated",
    });
  })
  .catch((error) => {
    console.log(error.message);
    return res.status(500).send({
      message: error.message,
    });
  });
});

//get paid courses
router.get("/getpaidcourses", async (req, res) => {
  await tcourses
    .find()
    .exec()
    .then((response) => {
      console.log(response);
      if (response) {
        res.status(200).json({ tcourses: response });
      } else {
        res.status(401).json({ status: "failed" });
      }
    });
});

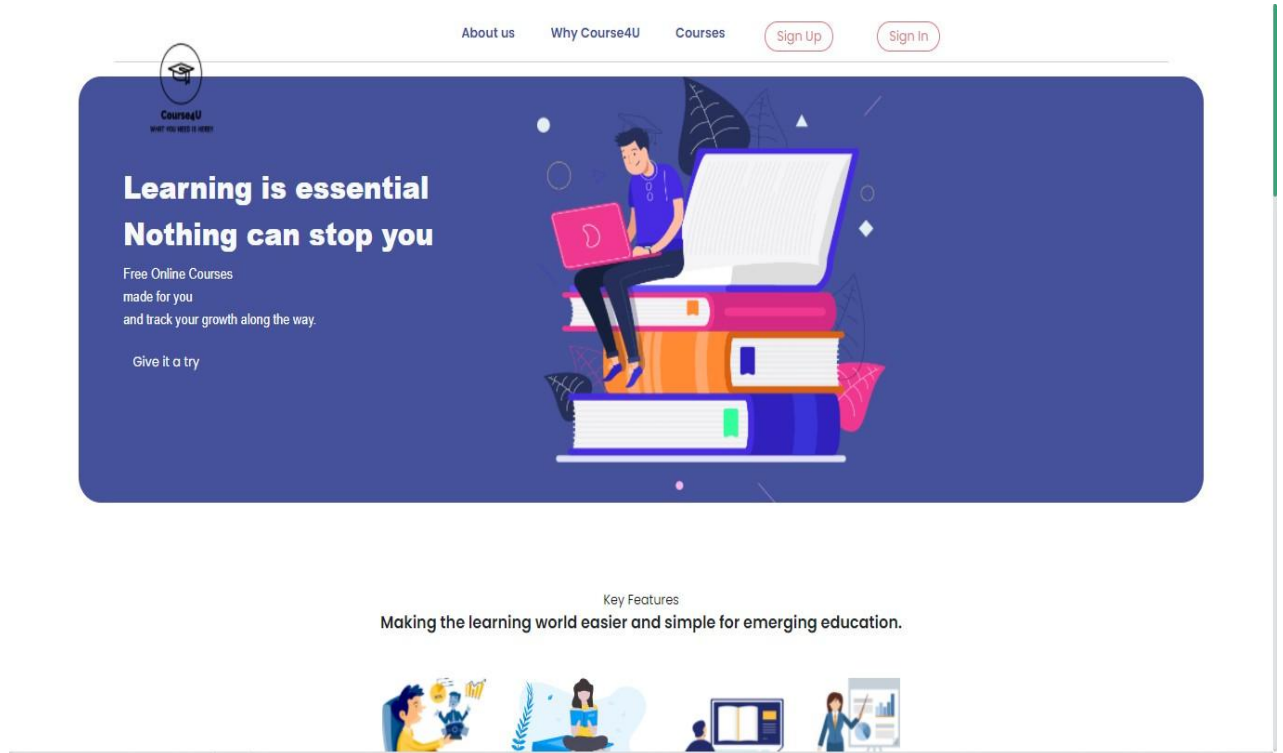
//get course video
router.post("/getcoursevideo", async (req, res) => {
  let id = req.body.cid;
  await courses.find({ _id: id })
    .exec()
    .then((response) => {
      console.log(response);
      if (response) {
        res.status(200).json({ courses: response });
      } else {
        res.status(401).json({ status: "failed" });
      }
    });
});
module.exports = router;

```



## 9.2 Screen Shots

### Index Page



### User Dashboard



## Enrolled Courses

[Dashboard](#)  
[My Courses](#)  
[Profile](#)  
[Contact](#)  
[Log Out](#)


HTML  
Computer Science  
[View Details](#)

CSS  
Computer Science  
[View Details](#)

JavaScript  
Computer Science  
[View Details](#)

ReactJS  
Computer Science  
[View Details](#)

## Admin Dashboard

  
[Manage Course](#)  
[Manage User](#)  
[LOG OUT](#)

[Back](#)

Full Name	Email	Status	Action
Jose K	jose@gmail.com	active	<a href="#">Unblock</a> <a href="#">Block</a> <a href="#">Delete</a>
Rubin Siby	rubin@gmail.com	active	<a href="#">Unblock</a> <a href="#">Block</a> <a href="#">Delete</a>
Sanju Abraham	sanju@gmail.com	active	<a href="#">Unblock</a> <a href="#">Block</a> <a href="#">Delete</a>

Dashboard
My Courses
Profile
Contact
Log Out

Video No.	Video Name
F1122	dert

Why React?

Course4U
Manage Course
Log out

Add Course

Id	Course Name
M111	Maths
M112	adsd
m111	ewqfwfw
m00111	acssc
M001	Asgsgs
V001	assacsacascascasc
F001	Search Module

Add Course

Course ID
Course Name
Select Category
dd-mm-yyyy
dd-mm-yyyy
Description
Amount
Select Video Link
Choose Files No file chosen

Start Date	End Date	Video Link	Status	Action
22-05-31	2022-05-31	1653205762740.mp4	active	Edit Deactivate
22-05-24	2022-05-24	1653217267765.mp4	active	Edit Deactivate
22-05-05	2022-06-05		active	Edit Deactivate
22-05-28	2022-05-28		active	Edit Deactivate
22-05-29	2022-05-29		active	Edit Deactivate
22-05-31	2022-05-31	1653228091367.mp4	active	Edit Deactivate
22-05-05	2022-06-05	1653278804764.mp4	active	Edit Deactivate