CONTENT PAGE

1. COMPANY PROFILE

Ardent Computech Private Limited is an ISO 9001-2015 certified Software Development Company in India. It has been operating independently since 2003. It was recently merged with ARDENT TECHNOLOGIES.

Ardent Technologies

ARDENT TECHNOLOGIES is a Company successfully providing its services currently in the UK, USA, Canada and India. The core line of activity at ARDENT TECHNOLOGIES is to develop customized application software covering the entire responsibility of performing the initial system study, design, development, implementation and training. It also deals with consultancy services and Electronic Security systems. Its primary clientele includes educational institutes, entertainment industries, resorts, theme parks, service industry, telecommute operators, media and other business houses working in various capacities.

Ardent Collaborations

ARDENT COLLABORATIONS, the Research Training and Development Department of ARDENT COMPUTECH PVT LTD is a professional training Company offering IT enabled services & industrial training's for B-Tech, MCA, BCA, MSc and MBA freshers and experienced developers/programmers in various platforms. Summer Training / Winter Training / Industrial training will be provided for the students of B. TECH, M. TECH, MBA and MCA only. Deserving candidates may be awarded stipends, scholarships and other benefits, depending on their performance and recommendations of the mentors.

Associations

Ardent is an ISO 9001:2015 company. It is affiliated to National Council of Vocational Training (NCVT), Directorate General of Employment & Training (DGET), Ministry of Labor & Employment, and Government of India.

2. INTRODUCTION

In today's digital era, education has evolved beyond the boundaries of traditional classrooms. With the rapid advancement of technology, e-learning has emerged as a powerful tool to make education more accessible, flexible, and efficient. This project, the E-Learning App, aims to create a comprehensive and interactive platform that caters to learners of all ages and backgrounds.

Powered by MERN stack, Our project E-Learning App is designed to provide a user-friendly interface for students to access a wide range of educational content, including video lectures, quizzes, assignments, and interactive modules. The platform also enables instructors to manage courses, track student progress, and provide timely feedback. By leveraging multimedia content and smart learning features, the app ensures an engaging and effective learning experience.

This project not only bridges the gap between educators and learners but also promotes self-paced learning, making it ideal for individuals looking to acquire new skills or enhance existing knowledge at their convenience

2A.OBJECTIVE

The primary objective of the E-Learning App is to transform the traditional education system by providing a flexible, accessible, and engaging digital learning platform that serves the diverse needs of learners across various age groups and educational backgrounds. The app aims to bridge the gap between quality education and accessibility by offering a wide range of courses, interactive content, and real-time assessment tools that enhance understanding and holding of knowledge.

Ultimately, the E-Learning App seeks to empower individuals by supplying them with the skills and knowledge they need to succeed in academic, professional, and personal attempts— anytime, anywhere.

2B.SCOPE

Our project focuses on creating a mobile and/or web-based application that supports a wide range of learning materials for students of various age groups and educational levels.

1. **User Registration & Login :**
   Secure sign-up/login for students, instructors, and admins.

2. **Interactive Content:**
   Videos for engaging learning.

3. **Instructor Dashboard :**
   Tools for instructors to create, upload, and manage course content.

4. **Admin Panel :**
   Controls for user management, course approvals, reports, and analytics.

5. **Multiplatform Access :**
   Available on Android, iOS, and web browsers.

6. **Payment Integration :**
   Support for subscriptions, one-time payments, or course purchases for users.

SYSTEM ANALYSIS

3A.IDENTIFICATION OF NEED

System analysis is a crucial phase in the development of our project E-Learning App, involving creating an efficient and user-friendly platform.

With the rapid digital transformation and the increased demand for accessible education, there is a growing need for a flexible, efficient, and interactive platform that supports remote learning. Traditional classroom-based education has limitations in terms of location, time, and resources. The E- Learning App addresses these challenges by providing a scalable digital solution for students and educators.

Key Needs Identified:

1. **Accessibility to Education:**
   - Students in remote or underserved areas need access to quality educational resources.
   - Learners require flexibility to study at their own pace and schedule.
2. **Interactive and Engaging Learning:**
   - Traditional education often lacks multimedia tools that enhance understanding.
   - There is a need for videos, quizzes, assignments, and gamified learning to keep students engaged.
3. **Efficient Course Delivery:**
   - Instructors need an efficient way to upload, manage, and deliver course materials.
   - Automation in grading and assessment saves time and increases accuracy

3B.FEASIBILITY STUDY

The feasibility study of our E-Learning App project indicates that the concept is viable and promising across several key areas.

The E-Learning App project is feasible and practical across all key areas. Technically, it can be developed using widely available tools like Flutter or React Native, with scalable backend solutions like Node.js. Economically, it has strong revenue potential through subscriptions, and premium content. Operationally, it is easy to use, scalable, and meets user needs. Legally, it is viable with compliance to data protection and content licensing laws. The estimated development time is 6 to 8 months, making it achievable within a reasonable timeframe and budget.

Overall, the E-Learning App project is both realistic and promising, offering a sustainable and impactful solution in the rapidly growing EdTech industry.

3C.WORKFLOW

This Document plays a vital role in the development life cycle (SDLC) as it describes the complete requirements of the system. It is meant for use by the developers and will be the basic during the testing phase. Any changes made to the requirements in the future will have to go through a formal change approval process.

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases.

The waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow; hence it is also referred to as a linear-sequential life cycle model. This means that any phase in the development process begins only if the previous phase is complete. In the waterfall model phases do not overlap.

- **Waterfall Model Design:**

The waterfall approach was the first SDLC Model to be used widely in Software Engineering to ensure the success of the project. In "The Waterfall" approach, the whole process of software development is divided into separate phases. In the Waterfall model, typically, the Outcome of one phase acts as the input for the next phase sequentially.

- **Iterative Waterfall Design:**

Definition: The Iterative Waterfall Model is a variation of the traditional Waterfall model, which is a linear and sequential software development methodology. In the Iterative Waterfall Model, the development process is divided into small, manageable cycles, allowing for the revisiting and refinement of phases before progressing to the next stage. It combines the systematic structure of the Waterfall model with the flexibility of iterative development.

The sequential phases in Iterative Waterfall model are:

➢ **Requirement Gathering and Analysis:** All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification doc.

➢ **System Design:** The requirement specifications from the first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

➢ **Implementation:** With inputs from system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality which is referred to as Unit Testing.

➢ **Integration and Testing:** All the units developed in the implementation phase are integrated into a system after testing each unit. Post integration the entire system is tested for any faults and failures.

➢ **Deployment of the system:** Once the functional and non-functional testing is done, the product is deployed in the customer environment or released into the market.

➢ **Maintenance:** Some issues come up in the client environment. To fix those issues patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

All these phases are cascaded to each other in progress and are seen as flowing steadily downwards (like a waterfall) through the phases. The next phase is started only after the defined set of goals are achieved for the previous phase and it is signed off, so the name "Iterative Waterfall Model". In this model, phases do not overlap.

- **<u>Advantages:</u>**

1 . Flexibility: Iterations permit adjustments based on feedback.

2 . Early Delivery: Partial systems can be delivered incrementally.

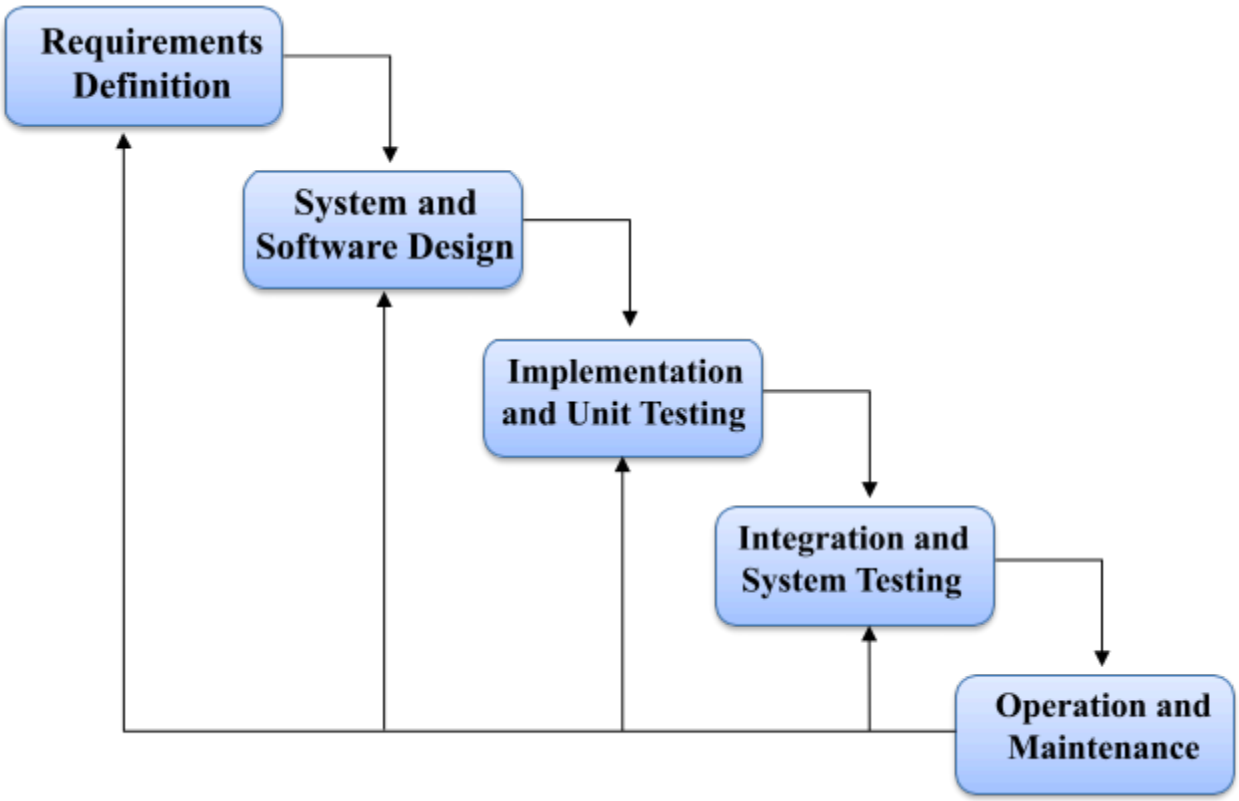
3 . Risk Management: Identifying and addressing issues early in the process.

- **<u>Disadvantages</u>:**

1. Increased Complexity: The iterative nature can make the process more complex.

2. Potential for Scope Creep: Frequent iterations may lead to scope changes.

3. Resource Intensive: Continuous revisiting of phases may demand more resources.



- **<u>Applications</u>:**

The Iterative Waterfall Model is suitable for projects with evolving or unclear requirements. It is commonly used in software development projects where regular feedback and refinement are essential.

Additionally, it is applicable in scenarios where partial system delivery is beneficial, allowing stakeholders to assess progress and make adjustments.

3D .STUDY OF THE SYSTEM

Modules: The modules used in this software are as follows:

- **Register:**

    1 . User Register: Here, the user will register to buy any course.
    2 . Admin Register: Here, the admin will register to handle all the databases.

- **Verify Account:** When a user login an OTP sends to the user Email to check the validity of the data.

- **Login:**

    1 . User Login: Here users will login to see available courses and buy any course.
    2 . Admin Login: Here admin will log in to handle all the data.

- **Home:** This page is to see feedback received from learners.

- **Courses:**
    1. **User interface:** This page shows the available courses that the users can purchase. They can also search for courses if available.
    2. **Admin interface:** In this page, the Admin can view or delete courses.

- **Lectures:**
    1. **User interface**: If the users purchase any course then they can access the lectures inside the course.
    2. **Admin interface**: Admin can add lectures of any specific course.

- **About:** This page will show the details about the website developers.

- **Account:**
    1. **User Interface:**
        a) **My Profile:** Here, the user can see the details after login.
        b) **Dashboard:** Here, the user can see the courses he/she has purchased.
    2. **Admin Interface:**
        a) **Admin Profile:** Here, the Admin can see the details after login.
        b) **Admin Dashboard:** Here the Admin can manage courses and user roles.

3E.INPUT AND OUTPUT

The main inputs, outputs and the major function the details are:

- **INPUT:**

1. Users can log in by entering their **credentials** on the login page.

- **OUTPUT:**

1. Users can view the **available courses ,buy any course and access lectures.**
2. The **admin** can access a **centralized database** that includes details of **users, add any course and add any Lecture** ensuring efficient system management.

3F.SOFTWARE REQUIREMENT SPECIFICATIONS

Software Requirements Specification provides an overview of the entire project. It is a description of a software system to be developed, laying out functional and non-functional requirements. The software requirements specification document enlists enough necessary requirements that are required for the project development. To derive the requirements we need to have a clear and thorough understanding of the project to be developed. This is prepared after detailed communication with the project team and the customer.

The developer is responsible for:

- Developing the system, which meets the SRS and solves all the requirements of the system.
- Demonstrating the system and installing the system at the client's location after acceptance testing is successful.
- Submitting the required user manual describing the system interfaces to work on it and also the documents of the system.

Functional Requirements:

A. **User Registration and Authentication:**
   1. Users should be able to create accounts securely.
   2. The system should authenticate users and manage login sessions.

B. **Browse and Search:**
   1. Users should be able to browse and search for courses.

.

C. **Courses Display:**
   1. Each Student should have detailed and up-to-date items with prices.
   2. Users should be able to view courses, helpful educational videos.

Hardware Requirements:

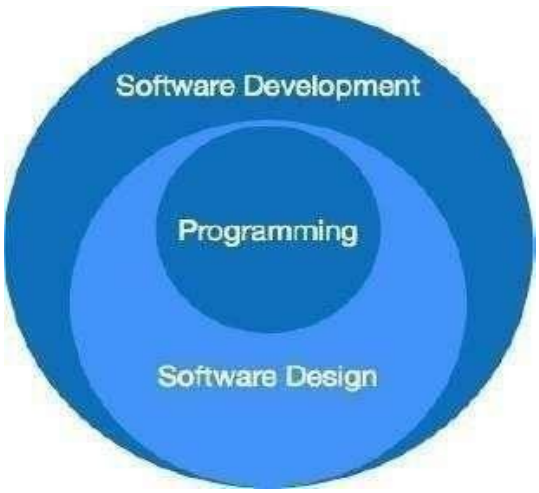
1 . Computer has Intel I3 Processor 2 .
8 GB RAM
3. SSD-ROM Drive

Software Requirements:

1. Windows 11 OS
2. Visual Studio Code
3. Mongo DB Atlas

3G.SOFTWARE ENGINEERING PARADIGM APPLIED

Software paradigms refer to the methods and steps, which are taken while designing the software. There are many methods proposed and are in work today, but we need to see where in software engineering these paradigms stand. These can be combined into various categories, though each of them is containedin one another.



The programming paradigm is a subset of Software design paradigm which is further a subset of the Software development paradigm.

There are two levels of reliability. The first is meeting the right requirements. A careful and thorough systems study is needed to satisfy this aspect of reliability. The second level of systems reliability involves the actual work delivered to the user. At this level, the system's reliability is interwoven with software engineering and development.

There are three approaches to reliability.

1. Error avoidance: Prevents errors from occurring in software.

2. Error detection and correction: In this approach, errors are recognized whenever they

are encountered, and correcting the error by the effect of the error of the system does not fail.

3. Error to lerance: In this approach, errors are recognized whenever they occur, but enables the system to keep running through degraded performance or Applying values that instruct the system to continue process.

SYSTEM DESIGN

4A. DATA FLOW DIAGRAM

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated.

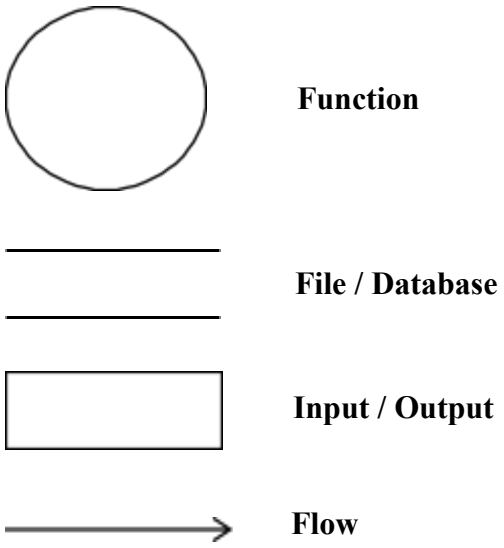
DFD can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of the process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

This context-level DFD is next "exploded", to produce a Level 1 DFD that shows some of the detail of the system being modeled. The Level 1 DFD shows how the system is divided into sub-systems (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present for the system to do its job and shows the flow of data between the various parts of the system.

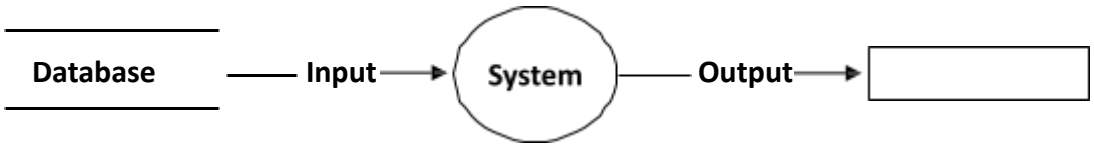
Data flow diagrams are one of the three essential perspectives of the structured-systems analysis and design method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a data flow diagram, users can visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's data-flow diagrams can be drawn up and compared with.

How any system is developed can be determined through a data flow diagram model. In the course of developing a set of leveled data flow diagrams, the analyst/designer is forced to address how the system may be decomposed into component sub-systems and to identify the transaction data in the data model. Data flow diagrams can be used in both the Analysis and Design phase of the SDLC. There are different notations to draw data flow diagrams. Defining different visual representations for processes, data stores, data flow, and external entities.

DFD Notation:



DFD Example:



Steps to Construct Data Flow Diagram:

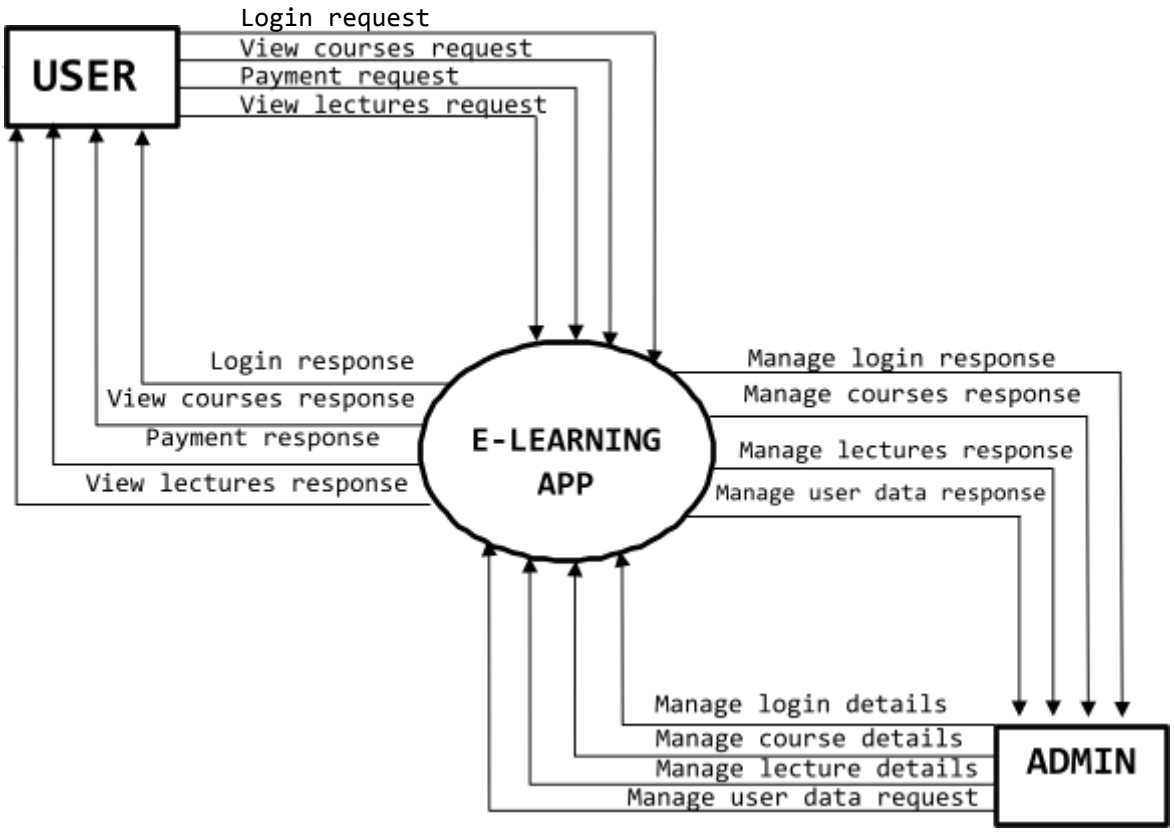
Four Steps are generally used to construct a DFD.

- Process should be named and referred for easy reference. Each name should be representative of the reference.
- The destination of flow is from top to bottom and from left to right.
- When a process is distributed into lower-level details they are numbered.
- The names of data stores, sources, and destinations are written in capital letters.

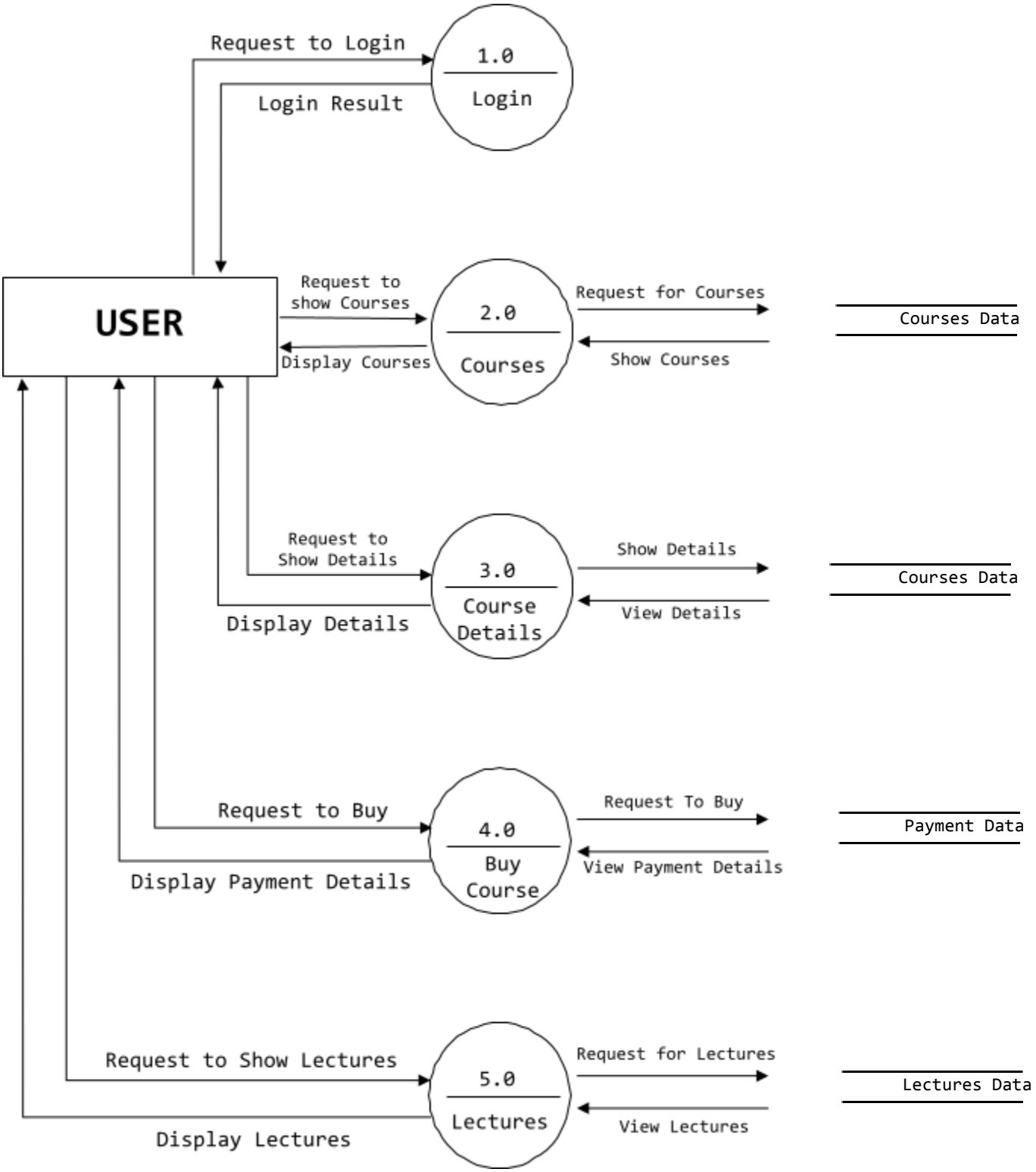
Rules for constructing a Data Flow Diagram:

- Arrows should not cross each other.
- Squares, Circles, and Files must bear a name.
- Decomposed data flow squares and circles can have the same names.
- Draw all data flow around the outside of the diagram.

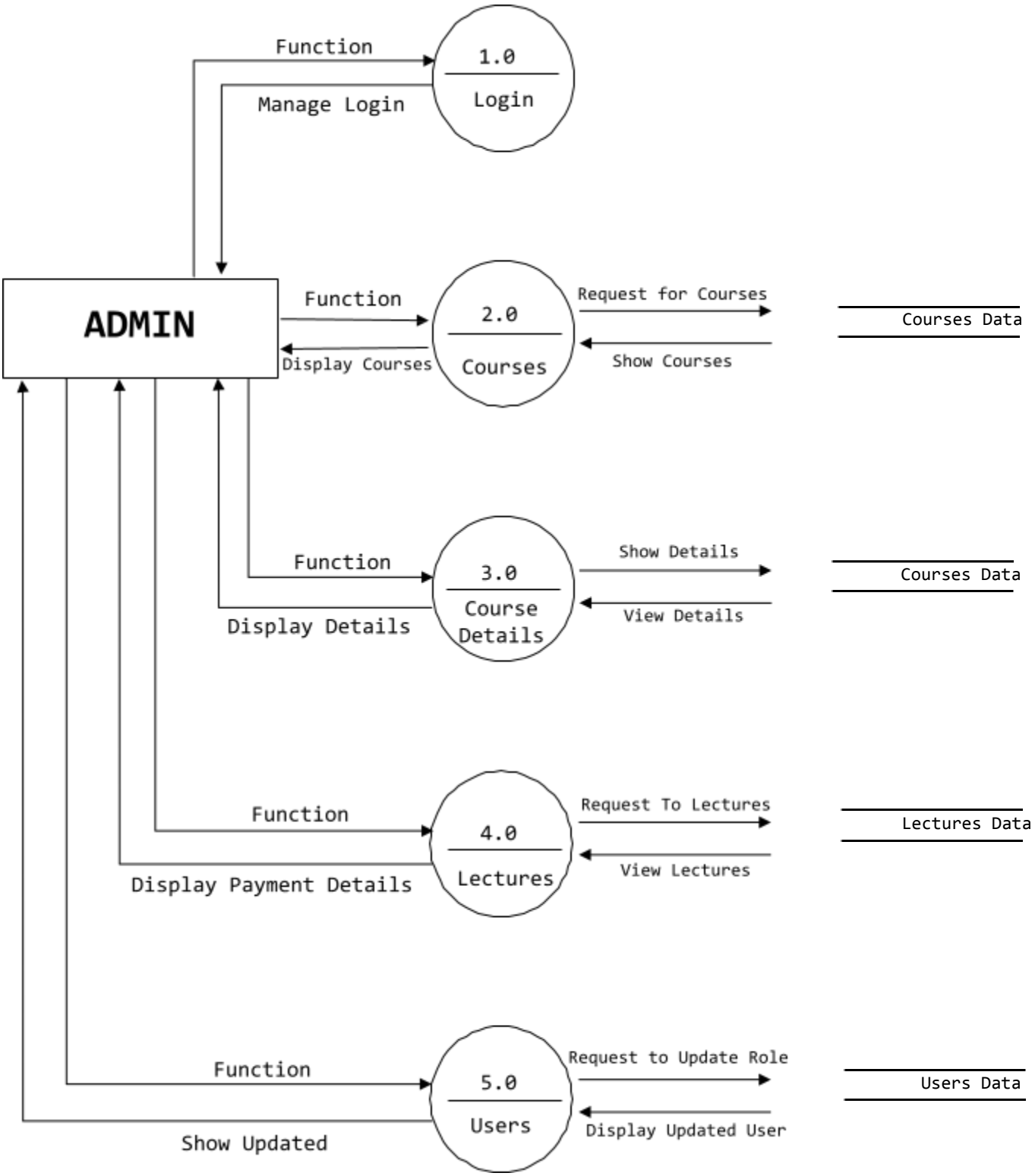
- **LEVEL 0 DFD OR CONTEXT DIAGRAM:**



- **LEVEL 1 DFD:**


Request to Login
1.0
Login
Login Result
USER
Request to
show Courses
Display Courses
2.0
Courses
Request for Courses
Show Courses
Courses Data
Request to
Show Details
Display Details
3.0
Course
Details
Show Details
View Details
Courses Data
Request to Buy
Display Payment Details
4.0
Buy
Course
Request To Buy
View Payment Details
Payment Data
Request to Show Lectures
Display Lectures
5.0
Lectures
Request for Lectures
View Lectures
Lectures Data

- **LEVEL 1 DFD:**


Function
1.0
Login
Manage Login
ADMIN
Function
2.0
Courses
Request for Courses
Show Courses
Display Courses
Courses Data
Function
3.0
Course Details
Show Details
View Details
Display Details
Courses Data
Function
4.0
Lectures
Request To Lectures
View Lectures
Display Payment Details
Lectures Data
Function
5.0
Users
Request to Update Role
Display Updated User
Show Updated
Users Data

4B.SEQUENCE DIAGRAM

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in a time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development.

Sequence diagrams are sometimes called event diagrams or event scenarios.

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

A sequence diagram is the most common kind of interaction diagram, which focuses on the message interchange between several life lines .A sequence diagram describes an interaction by focusing on the sequence of messages that are exchanged, along with their corresponding occurrence specifications on the lifelines.

The following nodes and edges are typically drawn in a UML sequence diagram: lifeline, execution specification, message, fragment, interaction, state invariant, continuation, and destruction occurrence.

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

So only static behavior is not sufficient to model a system rather dynamic behavior is more important than static behavior. In UML there are five diagrams available to model dynamic nature and a use case diagram is one of them. Now as we have to discuss that the use case diagram is dynamic in nature there should be some internal or external factors for making the interaction.

These internal and external agents are known as actors. So, use case diagrams consist of actors, use cases, and their relationships. The diagram is used to model the system/subsystem of an application. A single-use case diagram captures a particular functionality of a system.

So, to model the entire system numbers of use case diagrams are used. The purpose of a use case diagram is to capture the dynamic aspect of a system. But this definition is too generic to describe the purpose.

Because the other four diagrams (activity, sequence, collaboration, and State chart) are also having the same purpose. So, we will look into some specific purpose that will distinguish it from the other four diagrams.

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. So, when a system is analyzed to gather its functionalities use cases are prepared and actors are identified.

Now when the initial task is complete use case diagrams are modeled to present the outside view. So, in brief, the purposes of use case diagrams can be as follows:

- Used to gather requirements of a system.
- Used to get an outside view of a system.
- Identify external and internal factors influencing the system.

How to draw Use Case Diagram?

Use case diagrams are considered for high level requirement analysis of a system. So, when the requirements of a system are analyzed, the functionalities are captured in use cases.

So, we can say that uses cases are nothing but the system functionalities written in an organized manner. Now the second things which are relevant to the use cases are the actors. Actors can be defined as something that interacts with the system.

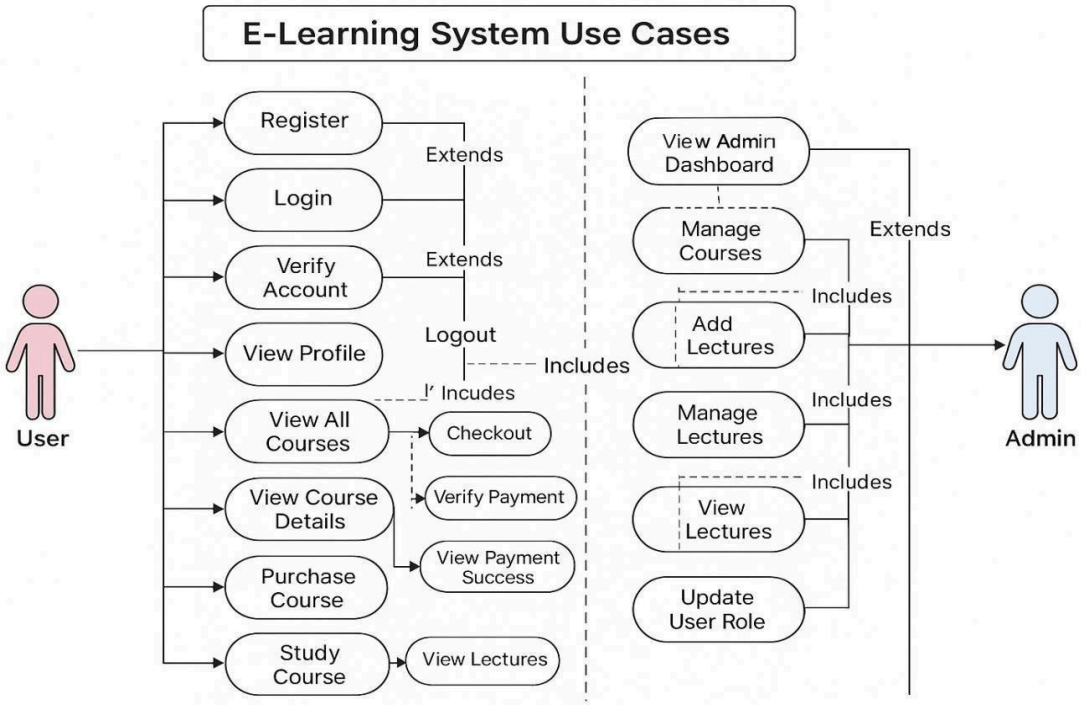
The actors can be human user, some internal applications or may be some external applications. So, in a brief when we are planning to draw use case diagram, we should have the following items identified.

- Functionalities to be represented as a use case
- Actors
- Relationships among the use cases and actors.

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

- The name of a use case is very important. So, 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. Because the main purpose of the diagram is to identify requirements.
- Use note whenever required to clarify some important point

- **USE CASE DIAGRAM:**

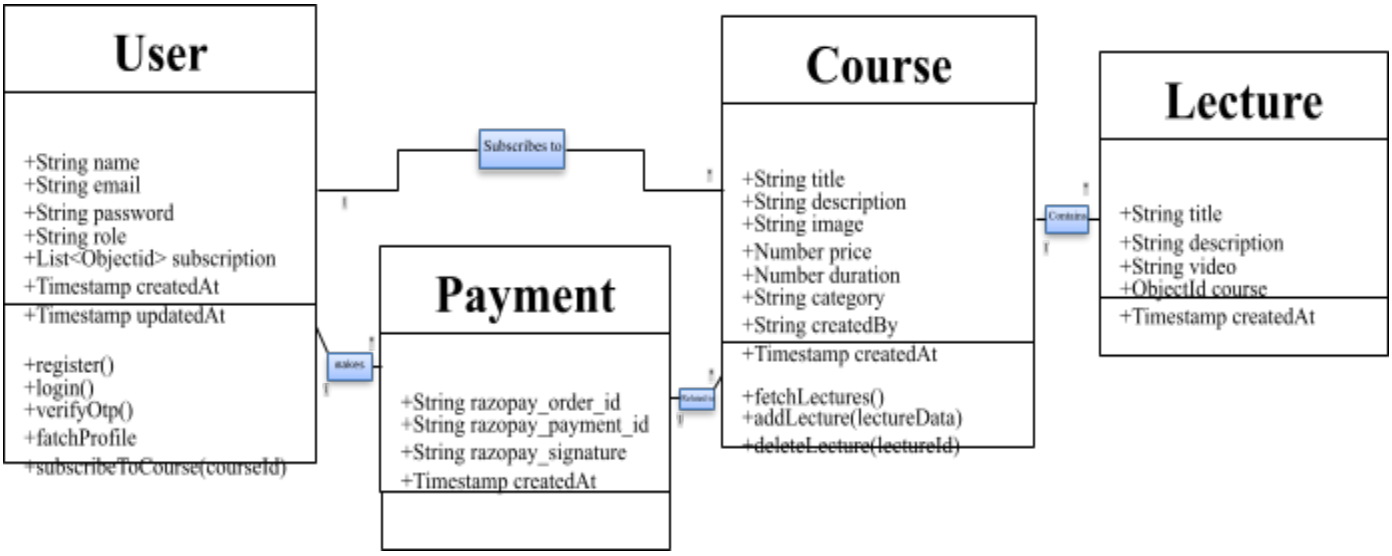


4D.SCHEMA DIAGRAM

The schema is an abstract structure or outline representing the logical view of the database as a whole. Defining categories of data and relationships between those categories, database schema design makes data much easier to retrieve, consume, manipulate, and interpret.

DB schema design organizes data into separate entities, determines how to create relationships between organized entities, and influences the applications of constraints on data. Designers create database schema to give other database users, such as programmers and analysts, a logical understanding of data.

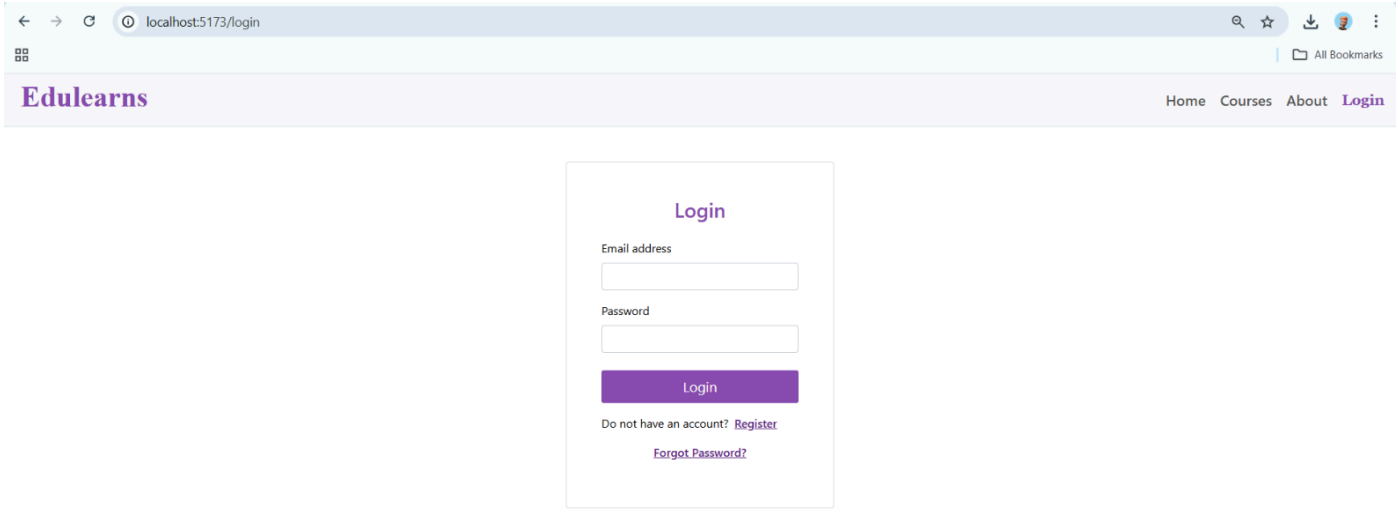
- **SCHEMA DESIGN:**



UI SNAPSHOT

❖ **FRONTEND : -**

1) **Login Page:**

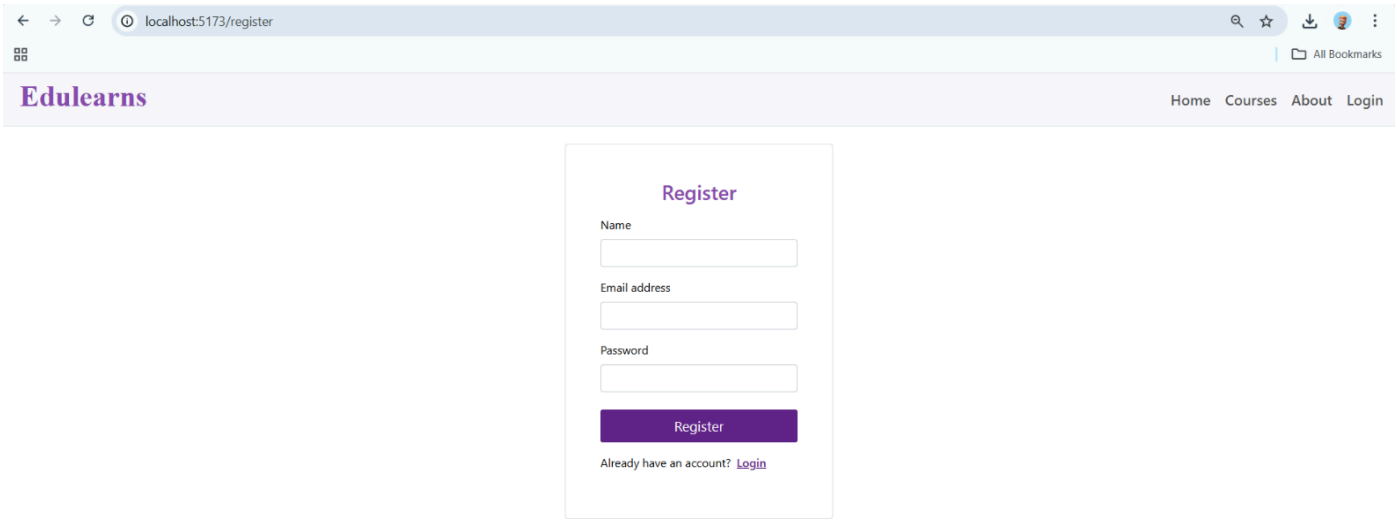


✔ **CODE**

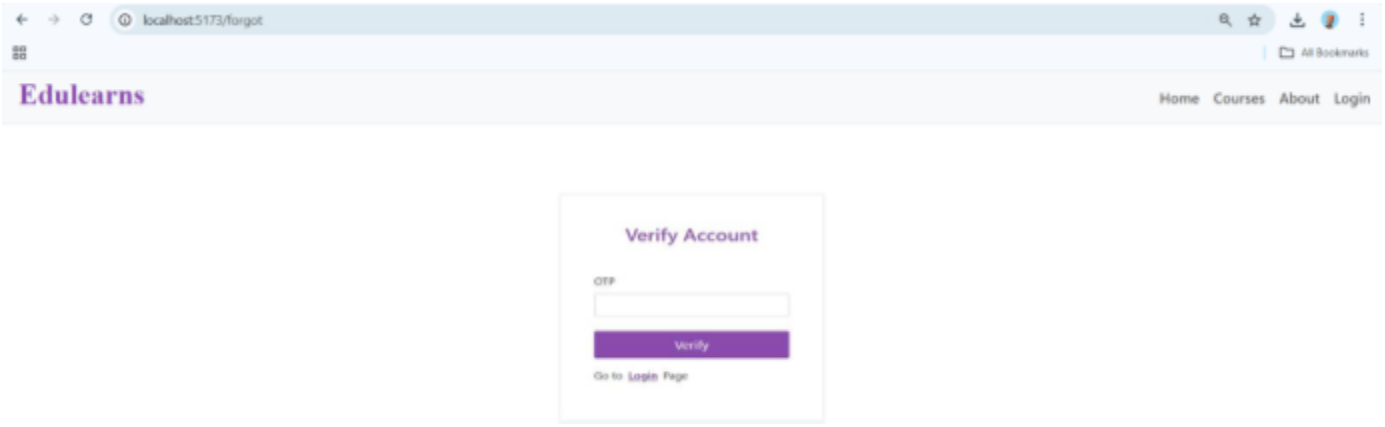
```
import React, { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import "./auth.css";
import { UserData } from "../../context/UserContext"; import
{ CourseData } from "../../context/CourseContext";

const Login = () => {
  const navigate = useNavigate();
  const { btnLoading, loginUser } = UserData();
  const [email, setEmail] = useState("");
  const [password, setPassword] = useState("");
  const { fetchMyCourse } = CourseData(); const
  submitHandler = async (e) => {
    e.preventDefault();
    await loginUser(email, password, navigate, fetchMyCourse)
  };

return (
    <>
```

```
            </h1>
            <div className="mb-3 mt-4">
              <label htmlFor="exampleInputEmail1" className="form-label"> Email
                address
              </label>
              <input
                type="email"
                value={email}
                onChange={(e) => setEmail(e.target.value)}
                className="form-control"
                aria-describedby="emailHelp"
                required
               />
             </div>
             <div className="mb-4 mt-3">
             <label htmlFor="exampleInputPassword1" className="form-label">
                Password
             </label>
             <input
                type="password"
                value={password}
                onChange={(e) => setPassword(e.target.value)}
                className="form-control"
                required
                                        /
                                        >
                                        <
}}>                                     /
                                        d
                                        i
                                        v
                                        >
                                         <button disabled={btnLoading} type="submit"
                                                   style={{ width: "100%"

                                                   {btnLoading ? "Please Wait..." :
                                                   "Login"}
                                                 </button>
                                                 <p className="mt-3 d-flex gap-2">
                                                   Do not have an account? <Link
                                                   to="/register">Register</Link>
                                                 </p>
                                                 <p className="text-center">
                                                   <Link to="/forgot">Forgot
                                                   Password?</Link>
                                                 </p>
                                               </form>
                                             </div>
                                           </div>
```

Register Page:



Verify Account Page:



✔ **CODE**

```
import React, { useState } from 'react'
import { NavLink, useNavigate } from 'react-router-dom' import
{ UserData } from '../../context/UserContext';

const Verify = () => {
  const [otp, setOtp] = useState("");
  const { btnLoading, verifyOtp } = UserData();
  const navigate = useNavigate();
  const submitHandler = async (e) => {
    e.preventDefault();
    await verifyOtp(Number(otp), navigate);
  }
  return (
    <>
        <div className="container d-flex justify-content-center mt-5">
          <div className="col-lg-4 mt-5 border p-5 rounded">
            <form onSubmit={submitHandler}>
              <h1 className='fs-3 text-center' style={{ color: "#8a4baf" }}>Verify
Account</h1>                                    <div className="mb-4 mt-5">
                                                  <label htmlFor="exampleInputEmail1"
                                                  className="form-label">OTP</label>
                                                  <input type="number" value={otp} onChange={(e) =>
                                                  setOtp(e.target.value)}
className="form-control" required />
              </div>
              <button disabled={btnLoading} type="submit" style={{ width: "100%" }}>
                {btnLoading ? "Please Wait..." : "Verify"}
              </button>
              <p className='mt-3 d-flex gap-2'>Go to <NavLink to="/login">Login</NavLink>
Page</p>
            </form
```

2) **Reset Password Page:**

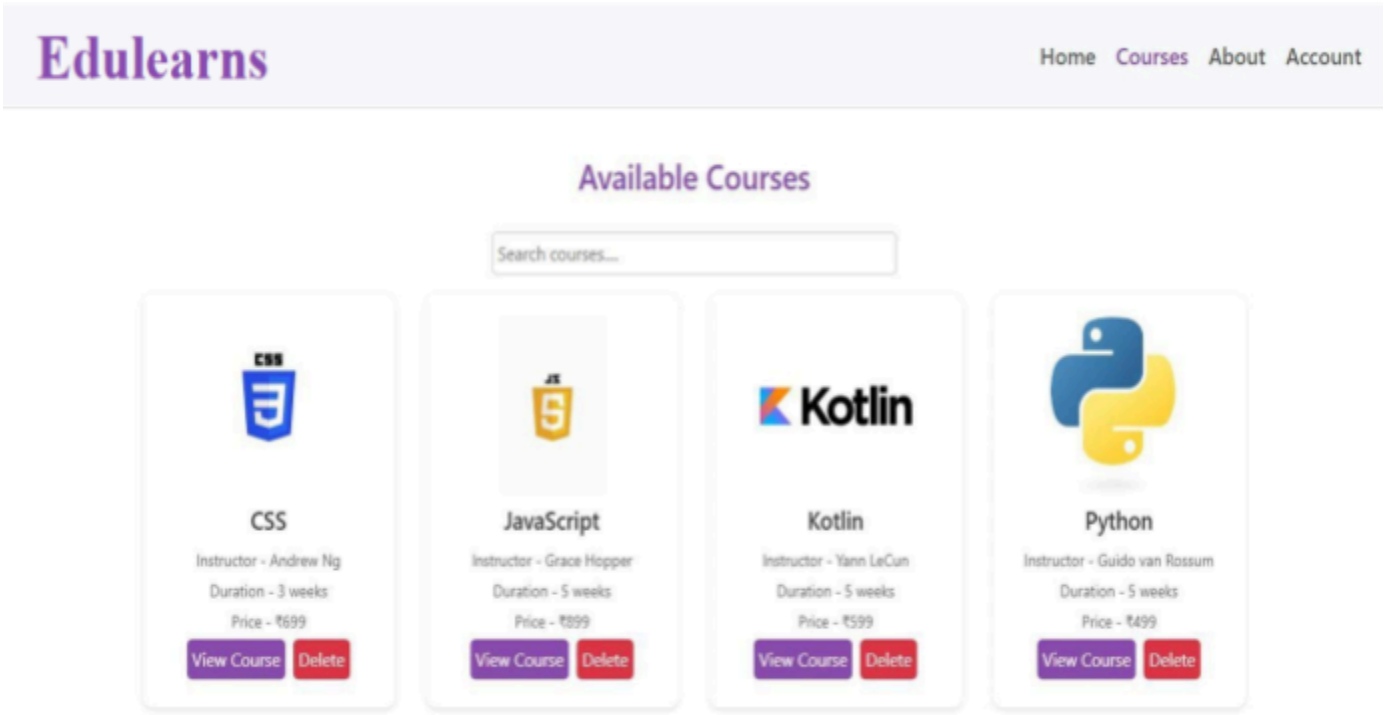
CODE :

```
import React, { useState } from 'react'
import "./auth.css"
import { useNavigate, useParams } from 'react-router-dom';
import toast from 'react-hot-toast';
import axios from 'axios';
import { server } from '../../main';

const ResetPassword = () => {
  const [password, setPassword] = useState("");
  const [btnLoading, setBtnLoading] = useState(false);
  const navigate = useNavigate();
  const params = useParams()

  const handleSubmit = async (e) => {
    e.preventDefault();
    setBtnLoading(true);
    try {
      const { data } = await
axios.post(`${server}/api/user/reset?token=${params.token}`, { password });

      toast.success(data.message);
      navigate("/login");
      setBtnLoading(false);
    } catch (error) { toast.error(error.response.data)
```

36

```
  return (
     <>
      <div className="container d-flex justify-content-center mt-5">
        <div className="col-lg-4 mt-5 border p-5 rounded">
          <form onSubmit={handleSubmit}>
            <h2 className='fs-3 text-center' style={{ color: "#8a4baf" }}>Reset
Password</h2>
            <div className="mb-4 mt-5">
              <label htmlFor="password" className="form-label">Enter
Password</label>
              <input type="password" value={password} onChange={(e) =>
setPassword(e.target.value)} className="form-control" required />
            </div>
            <button disabled={btnLoading}>{btnLoading ? "Please Wait..." : "Reset
Password"}</button>
          </form>
        </div>
      </div>
    </>
  )
}

export default ResetPassword;
```

3) COURSES PAGE(for admin and user):



- **components-pages-Courses.jsx**

```
import React, { useState } from "react";
import "./courses.css";
import { CourseData } from "../../context/CourseContext"; import
CourseCard from "../../components/coursecard/CourseCard"; import
Footer from "../../components/footer/Footer";
const Courses = () => {
  const { courses } = CourseData();
  const [search, setSearch] = useState("");

  const filterCourses = courses.filter((course) =>
    course.title.toLowerCase().includes(search.toLowerCase())
  );

  return (
    <>
      <div className="courses">
        <h2>Available Courses</h2>
        <div className="container-fluid mt-3 mb-3">
          <input type="text" placeholder="Search courses...." value={search}
            onChange={(e) => setSearch(e.target.value)}
           />
         </div>

        <div className="course-container">
          {
```

```
              <p>No Courses yet!</p>
            )
          }
        </div>
      </div>

    <Footer/>
    </>
  );
};

export default Courses;
```

- **components – CourseCard.jsx :-**

```
import React, { useEffect } from "react";
import "./courseCard.css";
import Aos from 'aos'; import
"aos/dist/aos.css";
import { server } from "../../main";
import { UserData } from "../../context/UserContext";
import { useNavigate } from "react-router-dom"; import
toast from "react-hot-toast";
import axios from "axios";
import { CourseData } from "../../context/CourseContext";

const CourseCard = ({ course }) => {
  const navigate = useNavigate(); const
  { user, isAuth } = UserData();
  useEffect(() => {
    Aos.init({ duration: 400 })
  }, [])
  const { fetchCourses } = CourseData();
  const deleteHandler = async (id) => {
    if (confirm("Are you sure want to delete this course")) {
      try {
        const { data } = await axios.delete(`${server}/api/course/${id}`, {
          headers: {
            token: localStorage.getItem("token"),
          },
        });

        toast.success(data.message); fetchCourses();
      } catch (error) {
        toast.error(error.response.data.message);
      }
    }
  };
  return (
    <>
      <div className="course-card" data-aos="fade-up">
```

```
          alt=""
          className="course-image"
        />
        <h3>{course.title}</h3>
        <p>Instructor - {course.createdBy} </p>
        <p>Duration - {course.duration} weeks</p>
        <p>Price - ₹{course.price} </p>
        {isAuth ? (
          <>
            {user && user.role !== "admin" ? (
              <>
                  {user.subscription.includes(course._id) ? (
                    <button
                      onClick={() => navigate(`/course/study/${course._id}`)}
                    >
                      Study
                    </button>
                  ) : (
                    <button onClick={() => navigate(`/course/${course._id}`)}> Get
                      Started
                    </button>
                  )}
                  </>
                ) : (
                <button onClick={() =>
navigate(`/course/study/${course._id}`)}>
                  View Course
                </button>
              )}
              </>
           ) : (
          <button onClick={() => navigate("/login")}>Get Started</button>
        )}

        {user && user.role === "admin" && (
          <button
            onClick={() => deleteHandler(course._id)}
            className="btn bg-danger ms-2 mb-1"
          >
            Delete
           </button>
         )}
       </div>
     </>
   );
 };


export default CourseCard;
```

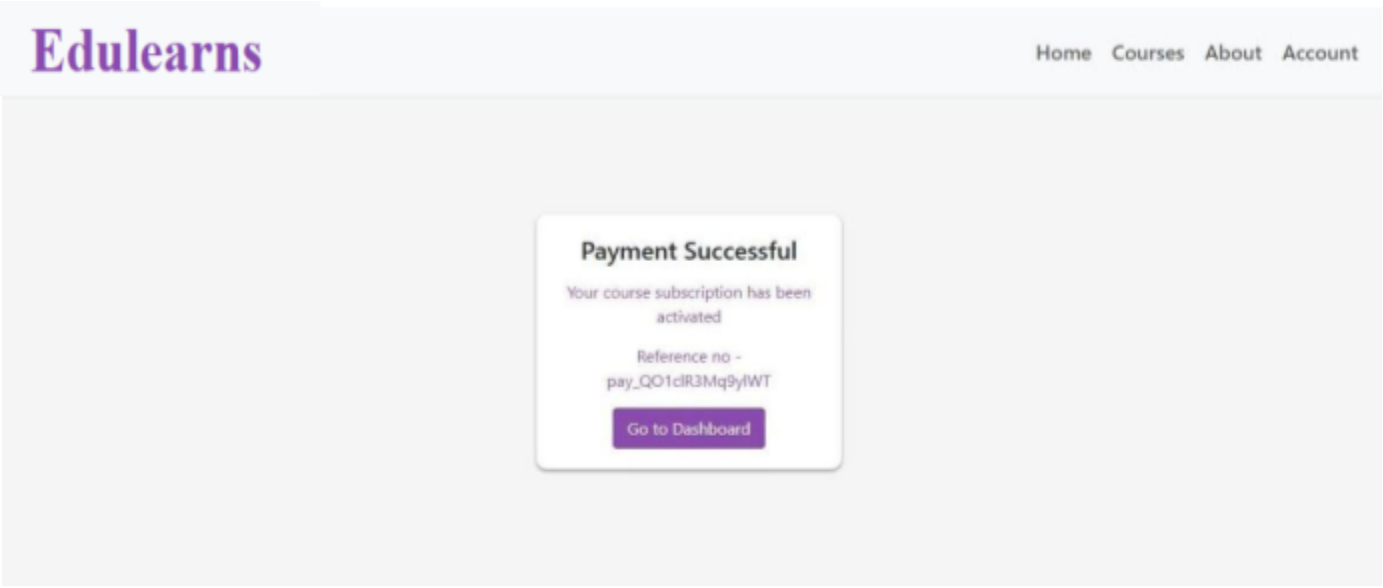
4) COURSE DESCRIPTION PAGE AND SYSTEM(for user);

✔ CODE

```
import React, { useEffect, useState } from "react";
import "./coursedescription.css";
import { useNavigate, useParams } from "react-router-dom";
import { CourseData } from "../../context/CourseContext";
import { server } from "../../main";
import axios from "axios";
import toast from "react-hot-toast";
import { UserData } from "../../context/UserContext"; import
Loading from "../../components/loading/Loading";

const CourseDescription = ({ user }) => { const
  params = useParams();
  const navigate = useNavigate();
  const { fetchCourse, course, fetchCourses, fetchMyCourse } = CourseData();
  const { fetchUser } = UserData();
  const [loading, setLoading] = useState(false);
  useEffect(() => {
    fetchCourse(params.id);
  }, []);

// Add Payment System
```

```
    const options = {
      key: "rzp_test_SIRZUl8zxB8qmW", // Enter the Key ID generated from the
Dashboard
      amount: order.id, // Amount is in currency subunits. Default currency is
INR. Hence, 50000 refers to 50000 paise
      currency: "INR",
      name: "E Learning", //your business name
      description: "Learn with us",
      order_id: order.id, //This is a sample Order ID. Pass the `id` obtained in
the response of Step 1

      handler: async function (response) {
        const { razorpay_order_id, razorpay_payment_id, razorpay_signature }
= response;

        try {
          const data = await
axios.post(`${server}/api/varification/${params.id}`, {
            razorpay_order_id, razorpay_payment_id, razorpay_signature
          },
            {
              headers: {
                token,
              },
            },
          );

          await fetchUser(); await
          fetchCourses(); await
          fetchMyCourse();
          toast.success(data.message);
          setLoading(false);
          navigate(`/payment-success/${razorpay_payment_id}`)
        } catch (error) {
          toast.error(error.response.data.message);
          setLoading(false);
        }
      },

      theme: {
        color: "#8a4baf",
      },
    };
    const razorpay = new window.Razorpay(options);
    razorpay.open();
    setLoading(false);
  };

  // End Payment
```

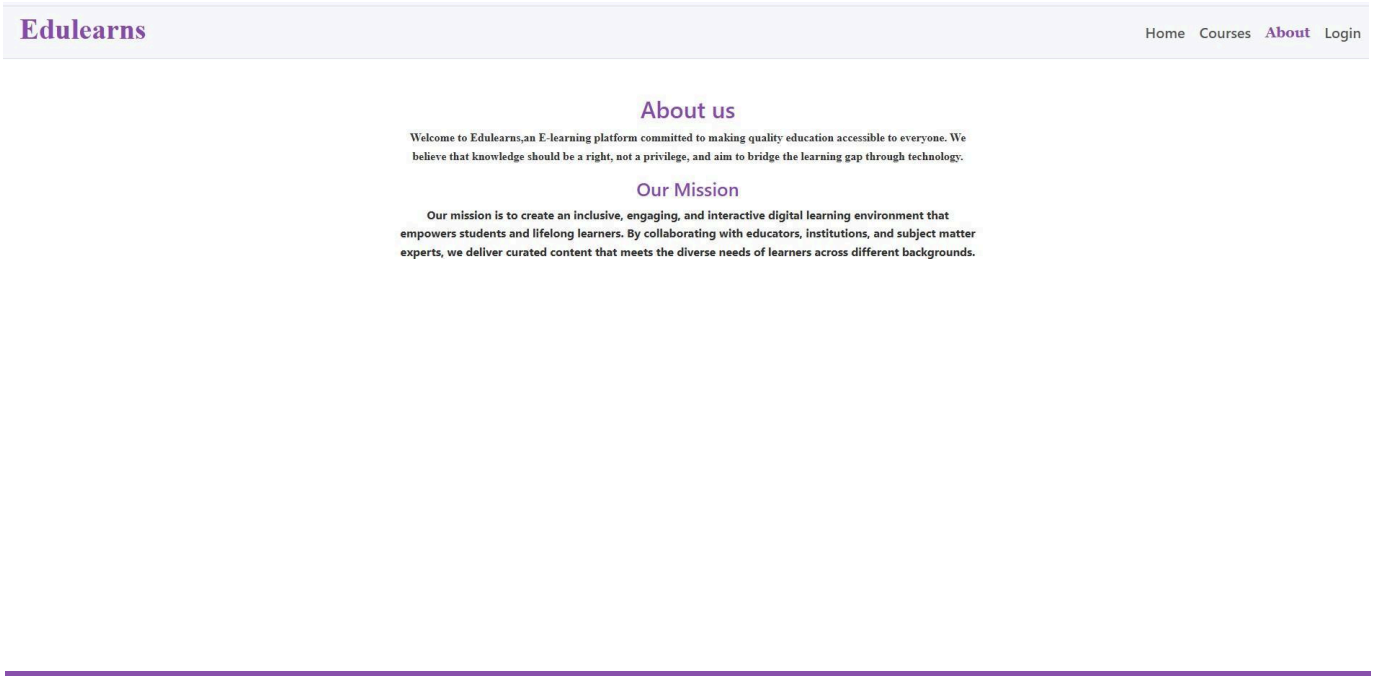
5) PAYMENT SUCCESSFULLY PAGE:



✔ CODE

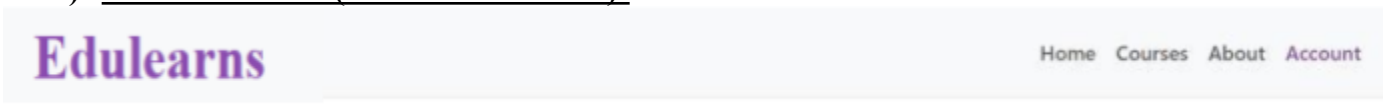
```
import React from 'react' import
"./paymentSuccess.css"
import { Link, useParams } from 'react-router-dom'

const PaymentSuccess = ({ user }) => {
  const params = useParams();
  return (
    <>
      <div className="payment-success-page">
        {user && <div className='success-message'>
          <h2>Payment Successful</h2>
          <p>Your course subscription has been activated </p>
          <p>Reference no - {params.id} </p>
          <Link to={`/${user._id}/dashboard`} className='btn btn-success'> Go to
Dashboard </Link>
        </div>}
      </div>
    </>
  )
}

export default PaymentSuccess
```

6) About Page:




```
import React from 'react'
import "./about.css"
import Footer from '../../components/footer/Footer'
const About = () => {
  return (
    <>
      <div className='about'>
        <div className='about-content'>
          <h2>About us</h2>
          <p className='p1'><b>
            Welcome to Edulearns, an E-learning platform committed to making
quality education accessible to everyone. We believe that knowledge should be a
right, not a privilege, and aim to bridge the learning gap through technology.
             </b>
          </p>
          <h3>Our Mission</h3>
          <p> <b>
            Our mission is to create an inclusive, engaging, and interactive
digital learning environment that empowers students and lifelong learners. By
collaborating with educators, institutions, and subject matter experts, we deliver
curated content that meets the diverse needs of learners across different
backgrounds.
              </b>
            </p>
          </div>
      </div>
    <Footer/>
```

7) CCOUNT PAGE(for admin and user):

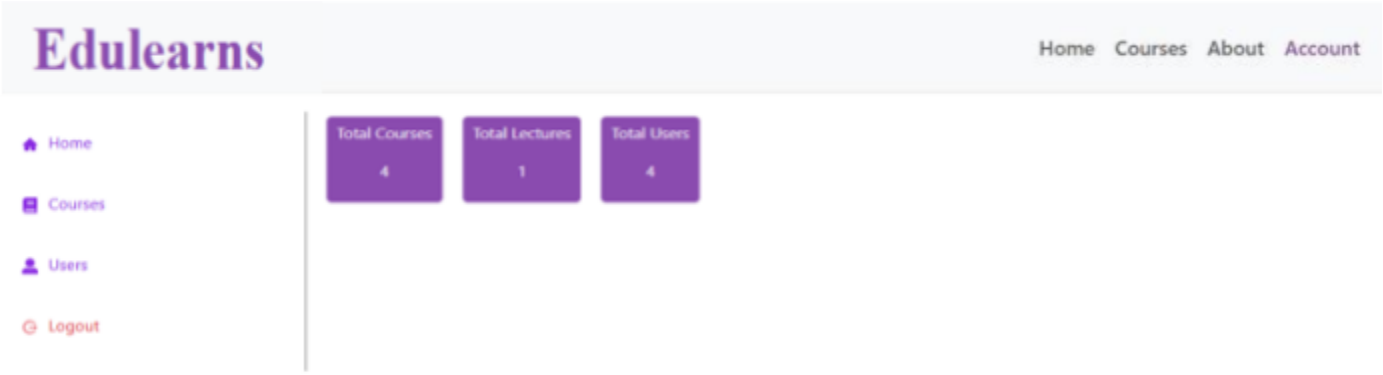


✔ **CODE:**

```
import React from 'react';
import { MdDashboard } from "react-icons/md";
import { IoMdLogOut } from "react-icons/io";
import "./account.css";
import { UserData } from '../../context/UserContext';
import toast from 'react-hot-toast';
import { useNavigate } from 'react-router-dom'; const
Account = ({ user }) => {
  const { setIsAuth, setUser } = UserData(); const
  navigate = useNavigate();

  const logoutHandler = () => {
    localStorage.clear();
    setUser([]); setIsAuth(false);
    toast.success("Logged Out");
    navigate("/login");
  }

  return (
    <>
      <div className='my-container'>
        {
          user && (
            <div className="profile">
              <h2>My Profile</h2>
              <div className="profile-info">
                <p>
                  <strong>Name - {user.name}</strong>
```

```
                {
                  user.role === "user" && (
                    <button onClick={() => navigate(`/${user._id}/dashboard`)}
className="common-btn d-flex gap-1 p-2">
                      <div>
                        <MdDashboard />
                      </div>
                      Dashboard
                    </button>
                  )
                }

                {
                  user.role === "admin" && (
                    <button onClick={() => navigate(`/admin/dashboard`)}
className="common-btn d-flex gap-1 p-2 mt-2">
                      <div>
                        <MdDashboard />
                      </div>
                      Admin Dashboard
                    </button>
                  )
                }

                <button onClick={logoutHandler} className="common-btn d-flex
gap-1 p-2 bg-danger mt-2">
                  <div>
                    <IoMdLogOut />
                  </div>
                  Logout
                </button>
              </div>
            </div>
          )
        }
      </div>
    </>
  );
};

export default Account;
```

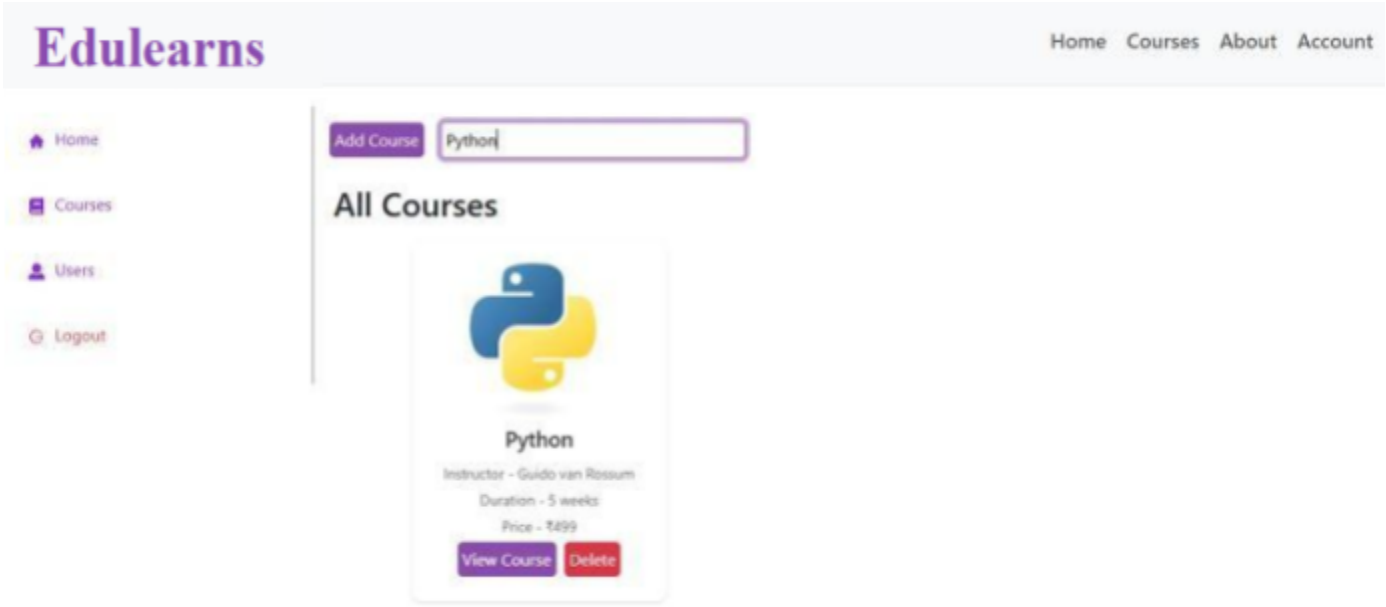
8) ADMIN DASHBOARD PAGE:



✔ **CODE:**

```
import React, { useEffect, useState } from 'react'
import "./admindashboard.css";
import { useNavigate } from 'react-router-dom' import
Layout from '../Utils/Layout';
import axios from 'axios';
import { server } from './../../main';

const AdminDashboard = ({ user }) => { const
  navigate = useNavigate();
  if (user && user.role !== "admin") return navigate("/"); const

  [stats, setStats] = useState([]);

  async function fetchStats() {
    try {
      const { data } = await axios.get(`${server}/api/stats`, {
        headers: {
          token: localStorage.getItem("token"),
        },
      });

      setStats(data.stats);
    } catch (error) {
      console.log(error);

    }
  }

  useEffect(() => { fetchStats();
  }, [])
```

9) ADMIN COURES PAGE:



✔ CODE:

```
import React, { useState } from 'react'
import "./admincourses.css"
import Layout from '../Utils/Layout'
import { useNavigate } from 'react-router-dom';
import { CourseData } from '../../context/CourseContext'; import
CourseCard from '../../components/coursecard/CourseCard'; import
toast from 'react-hot-toast';
import axios from 'axios';
import { server } from '../../main';

const categories = ["Web Development", "App Development", "Game Development",
"Data Science", "Artificial Intelligence"]

const AdminCourses = ({ user }) => {
  const navigate = useNavigate();
  if (user && user.role !== "admin") return navigate("/");

  const { courses, fetchCourses } = CourseData();

  const [title, setTitle] = useState("");
  const [description, setDescription] = useState(""); const
  [category, setCategory] = useState("")
  const [price, setPrice] = useState("")
  const [createdBy, setCreatedBy] = useState("")
  const [duration, setDuration] = useState("") const
  [image, setImage] = useState("")
  const [imagePrev, setImagePrev] = useState("") const
  [btnLoading, setBtnLoading] = useState(false) const
  [show, setShow] = useState(false);
  const [showSearch, setShowSearch] = useState(true) const
```

```
const filterCourses = courses.filter((course) =>
  course.title.toLowerCase().includes(search.toLowerCase())
);

const changeImageHandler = (e) => {
  const file = e.target.files[0];
  const reader = new FileReader();
  reader.readAsDataURL(file);
  reader.onloadend = () => {
    setImagePrev(reader.result);
    setImage(file);
  };
};

const submitHandler = async (e) => {
  e.preventDefault();
  setBtnLoading(true);
  const myForm = new FormData();

  myForm.append("title", title);
  myForm.append("description", description);
  myForm.append("category", category);
  myForm.append("price", price);
  myForm.append("createdBy", createdBy);
  myForm.append("duration", duration);
  myForm.append("file", image);

  try {
    const { data } = await axios.post(
      `${server}/api/course/new`,
      myForm,
      {
        headers: {
          token: localStorage.getItem("token"),
        },
      }
    );

    toast.success(data.message);
    setBtnLoading(false);
    await fetchCourses();
    setTitle("");
    setDescription("");
    setCategory("");
    setPrice("");
    setCreatedBy("");
    setDuration("");
    setImage("");
    setImagePrev("");
    setShow(false);
    setShowSearch(true);
  } catch (error) {
    toast.error(error.response.data.message);
    setBtnLoading(false);
```

```
    }
  };

  return (
    <>
      <Layout>
        <div className="admin-courses">
             <div className="left">
               <h2>All Courses</h2>
            <div className="dashboard-content">
              {
                filterCourses && filterCourses.length > 0 ?
                  filterCourses.map((e) => (
                    <CourseCard key={e._id} course={e} />
                  ))
                  : <p>No Courses Yet</p>
              }
            </div>
          </div>

          <div className="right">
            <div className="add-course">
              <button className="add-btn m-3" onClick={() => { setShow(!show);
setShowSearch(!showSearch) }}>
                {show ? "Close" : "Add Course"}
              </button>

                                    { showSearch && (
                                        <input type="text" placeholder="Search
                                        courses
value={search}
                                        "

                                          onChange={(e) =>
                                    }     setSearch(e.target.value)}
                                        />
                                    { )



                                      show && (
                                        <div className="course-form">
                                          <h3>Add Course</h3>
                                          <form onSubmit={submitHandler}>
                                            <label htmlFor="title">Title</label>
                                            <input type="text" value={title}
                                            onChange={e =>
setTitle(e.target.value)} required />

                        <label htmlFor="description">Description</label>
                        <input type="text" value={description} onChange={e =>
setDescription(e.target.value)} required />

                        <label htmlFor="price">Price</label>
```

```
<select value={category} onChange={e => setCategory(e.target.value)}>
                              <option value={""}>Select Categories</option>
                              {
                                  categories.map((e) => (
                                    <option value={e} key={e}>{e}</option>
                                  ))
                                }
                            </select>

                            <label htmlFor="duration">Duration</label>
                            <input type="number" value={duration} onChange={e =>
      setDuration(e.target.value)} required />

                            <input type="file" onChange={changeImageHandler}
      required />

                            {
                              imagePrev && <img src={imagePrev} alt='' width={300}
      />
                            }

                            <button type="submit" disabled={btnLoading}
      className='common-btn'>{btnLoading ? "Please Wait..." : "Add"}</button>
                          </form>
                        </div>
                      )
                    }
                  </div>
                </div>
              </div>
            </Layout>
          </>
        )
      }

      export default AdminCourses;
```

10) ADMIN USERS PAGE:

11)

✔ **CODE**

```
import React, { useEffect, useState } from 'react'
import './adminusers.css'
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import { server } from './../../main'; import
Layout from './../Utils/Layout'; import toast
from 'react-hot-toast';

const AdminUsers = ({ user }) => { const
  navigate = useNavigate();

  if (user && user.role !== "admin") return navigate("/"); const

  [users, setUsers] = useState([]);

  async function fetchUsers() { try
    {
      const { data } = await axios.get(`${server}/api/users`, {
        headers: {
          token: localStorage.getItem("token"),
           }
       });

      setUsers(data.users);
    } catch (error) {
      console.log(error);
    }
```

```
                                        }, []);
                                        const updateRole = async (id) => {
                                          if (confirm("Are you sure
                                            you want to update this
                                            user role")) { try {
                                              const { data } = await
                                                axios.put(`${server}/api/user/${i
                                                d}`, {}, { headers: {
                                                  token:
                                                  localStorage.getItem("token"),
            }
          });
        toast.success(data.message); fetchUsers();
      } catch (error) {
        toast.error(error.response.data.message);

      }
    }
  }
  return (
    <>
      <Layout>
        <div className="users">
          <h2>All Users</h2>
          <table border={"black"}>
            <thead>
              <tr>
                <th>No</th>
                <th>Name</th>
                <th>E-Mail</th>
                <th>Role</th>
                <th>Action</th>
              </tr>
            </thead>

            {
              users && users.map((e, i) => (
                <tbody>
                  <tr>
                    <td>{i + 1}</td>
                    <td>{e.name}</td>
                    <td>{e.email}</td>
                    <td>{e.role}</td>
                    <td><button onClick={() => updateRole(e._id)}
className='update-btn'>Update Role</button></td>
                  </tr>
                </tbody>
              ))
            }
```

USER DASHBOARD PAGE:

✔ **CODE:**

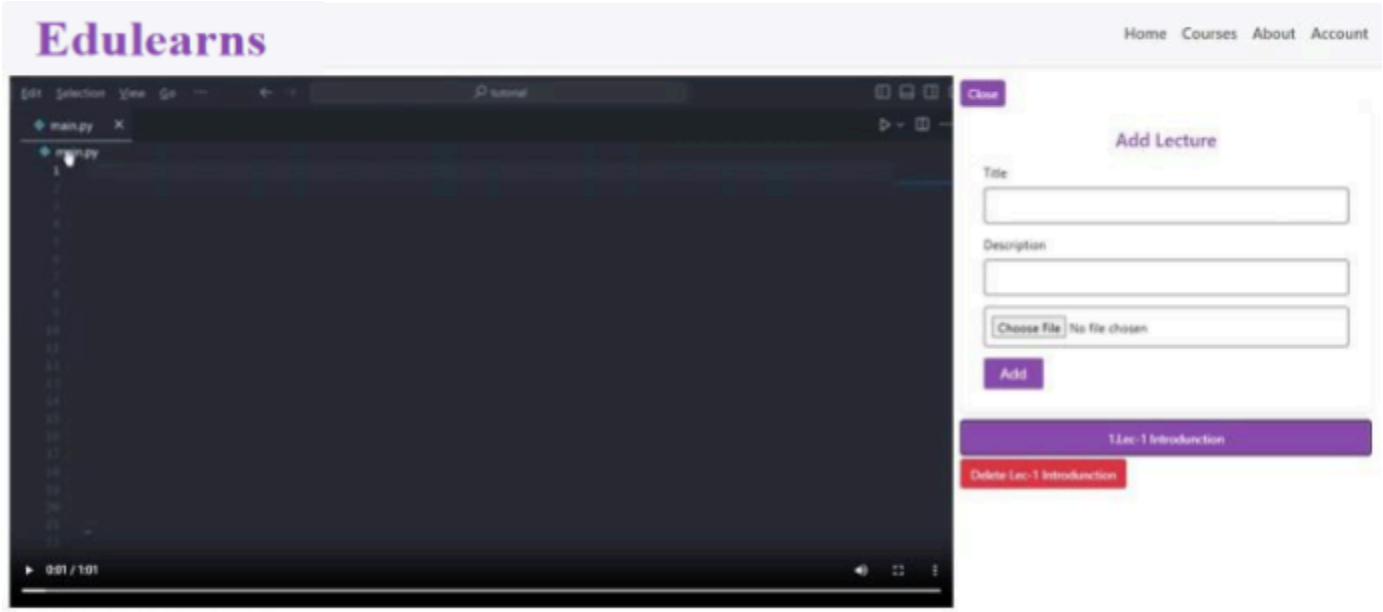
```
import React from 'react'
import "./dashboard.css"
import { CourseData } from '../../context/CourseContext'
import CourseCard from '../../components/coursecard/CourseCard';

const Dashboard = () => {
  const { mycourse } = CourseData();

  return (
    <>
      <div className="student-dashboard">
        <h2>All Enrolled Courses</h2>
        <div className="dashboard-content">
          {
              mycourse && mycourse.length > 0 ? mycourse.map((e) => (
                <CourseCard key={e._id} course={e} />
              )) : <p>No Course Enrolled Yet</p>
           }
        </div>
      </div>
    </>
  )
}

export default Dashboard;
```

12) COURSE STUDY PAGE:

✔ **CODE:**

```
import React, { useEffect } from "react";
import "./coursestudy.css";
import { Link, useNavigate, useParams } from "react-router-dom"; import
{ CourseData } from "../../context/CourseContext";
import { server } from "../../main";

const CourseStudy = ({ user }) => {
  const params = useParams();

  const { fetchCourse, course } = CourseData(); const
  navigate = useNavigate();

  if (user && user.role !== "admin" && !user.subscription.includes(params.id))
    return navigate("/");

  useEffect(() => {
    fetchCourse(params.id);
  }, []);
  return <>

    {course && <div className="course-study-page">
      <img src={`${server}/${course.image}`} />
      <h2>{course.title}</h2>
      <h4>{course.description}</h4>
      <h5>By - {course.createdBy}</h5>
      <h5>Duration - {course.duration} Weeks</h5>
      <Link to={`/lectures/${course._id}`}>Lectures</Link>
    </div>}
  </>;
```

13) LECTURE PAGE(for admin and user):



Lec-1 Introduction

✔ **CODE:**

```
import React, { useEffect, useState } from "react";
import "./lecture.css";
import { useNavigate, useParams } from "react-router-dom";
import axios from "axios";
import { server } from "./../../main";
import Loading from "../../components/loading/Loading";
import toast from "react-hot-toast";

const Lecture = ({ user }) => {
  const params = useParams(); const
  navigate = useNavigate();

  if (user && user.role !== "admin" && !user.subscription.includes(params.id))
    return navigate("/");

  const [lectures, setLectures] = useState([]);
  const [lecture, setLecture] = useState([]);
  const [loading, setLoading] = useState(true);
  const [lecLoading, setLecLoading] = useState(false);
  const [show, setShow] = useState(false);
  const [title, setTitle] = useState("");
  const [description, setDescription] = useState(""); const
  [video, setVideo] = useState("");
  const [videoPrev, setVideoPrev] = useState(""); const
  [btnLoading, setBtnLoading] = useState(false);

  async function fetchLectures() { try
    {
      const { data } = await axios.get(`${server}/api/lectures/${params.id}`, {
        headers: {
```

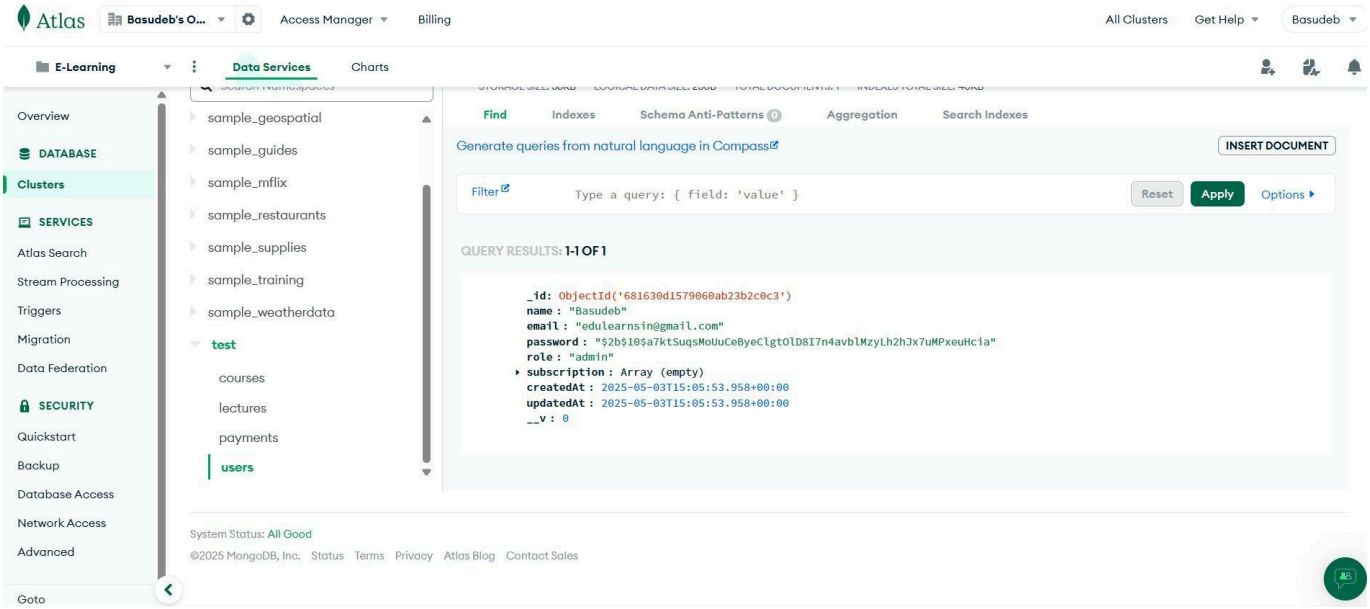
```
    });

    setLoading(false);
    setLectures(data.lectures);
  } catch (error) {
    console.log(error);
    setLoading(false);
  }
}

async function fetchLecture(id) {
  setLecLoading(true);
  try {
    const { data } = await axios.get(`${server}/api/lecture/${id}`, {
      headers: {
        token: localStorage.getItem("token"),
      },
    });

    setLecture(data.lecture);
    setLecLoading(false);
  } catch (error) {
    console.log(error);
    setLecLoading(false);
  }
}

const changeVideoHandler = (e) => {
  const file = e.target.files[0]; const
  reader = new FileReader();
  reader.readAsDataURL(file);
  reader.onloadend = () => {
    setVideoPrev(reader.result);
    setVideo(file);
  };
};

const submitHandler = async (e) => {
  e.preventDefault(); setBtnLoading(true);
  const myForm = new FormData();

  myForm.append("title", title);
  myForm.append("description", description);
  myForm.append("file", video);

  try {
    const { data } = await axios.post(
      `${server}/api/course/${params.id}`,
      myForm,
      {
        headers: {
          token: localStorage.getItem("token"),
        },
```

```
      }
    );

    toast.success(data.message);
    setBtnLoading(false);
    setShow(false);
    fetchLectures(); setTitle("");
    setDescription("");
    setVideo("");
    setVideoPrev("");
  } catch (error) {
    toast.error(error.response.data.message);
    setBtnLoading(false);
  }
};

const deleteHandler = async (id) => {
  if (confirm("Are you sure you want to delete this lecture")) { try
    {
      const { data } = await axios.delete(`${server}/api/lecture/${id}`, {
        headers: {
          token: localStorage.getItem("token"),
        },
      });

      toast.success(data.message);
      fetchLectures();
    } catch (error) {
      toast.error(error.response.data.message);
    }
  }
};

useEffect(() => {
  fetchLectures();
}, []);
return (
  <>
     {loading ? (
       <Loading />
     ) : (
       <>
         <div className="lecture-page">
           <div className="left">
             {lecLoading ? (
               <Loading />
             ) : (
               <>
                {lecture.video ? (
                  <>
                    <video
```

```
                        controlsList="nodownload noremoteplayback"
                        disablePictureInPicture disableRemotePlayback
                        autoPlay
                      ></video>
                      <h1>{lecture.title}</h1>
                      <h3>{lecture.description}</h3>
                       </>
                     ) : (
                      <h1>Please Select Your Lecture</h1>
                    )}
                  </>
                )}
              </div>
              <div className="right">
                {user && user.role === "admin" && (
                  <button className="add-btn mt-3" onClick={() =>
setShow(!show)}>                          {show ? "Close" : "Add Lecture"}
                                        </button>

                                      )}



                {show && (
                  <div className="lecture-form">
                    <h2>Add Lecture</h2>
                    <form onSubmit={submitHandler}>
                      <label htmlFor="text">Title</label>
                      <input
                        type="text"
                        value={title}
                        onChange={(e) => setTitle(e.target.value)}
                        required
                      />
                      <label htmlFor="text">Description</label>
                      <input
                        type="text" value={description}
                        onChange={(e) => setDescription(e.target.value)}
                        required
                      />
                      <input
                        type="file" placeholder="choose
                        video"
                        onChange={changeVideoHandler}
                        required
                      />

                      {videoPrev && (
                        <video src={videoPrev} width={"300"} controls></video>
                      )}

                      <button disabled={btnLoading} type="submit">
```

❖ BACKEND:-

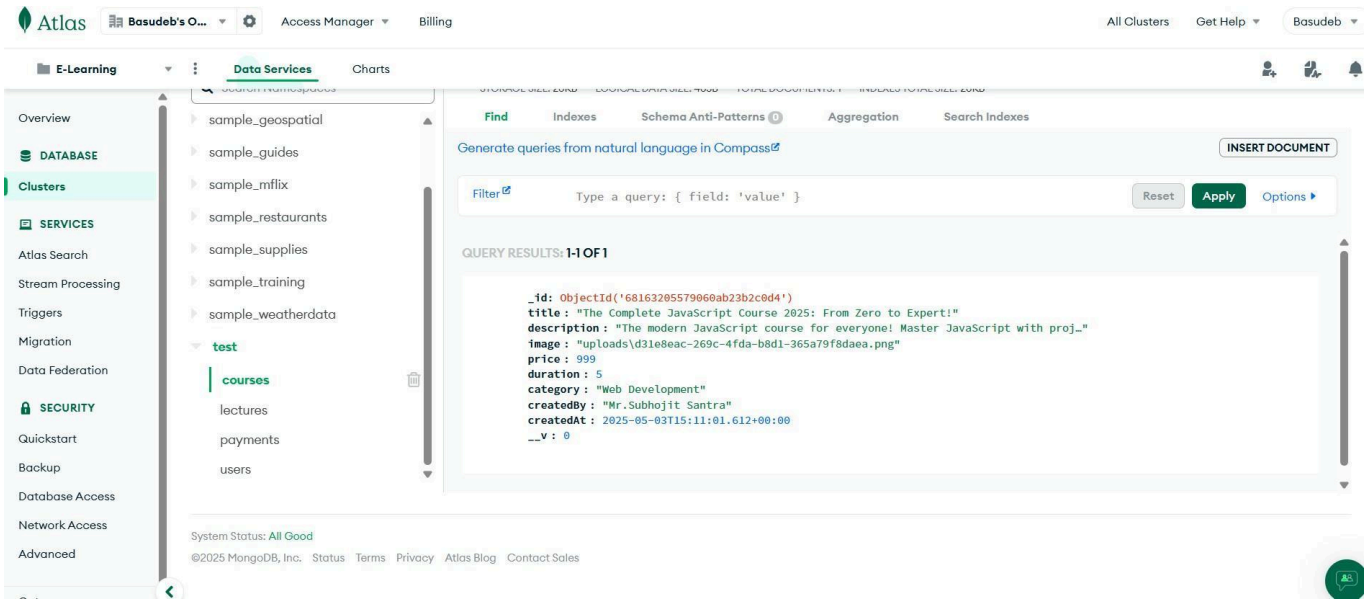
1) USER AND ADMIN DATA:



- **Database - db.js:**

```
import mongoose from "mongoose";

export const connectDB = async () => { try
  {
    await mongoose.connect(process.env.DB);
    console.log("Database Connected");
  } catch (error) {
    console.log(error);
  }
};
```

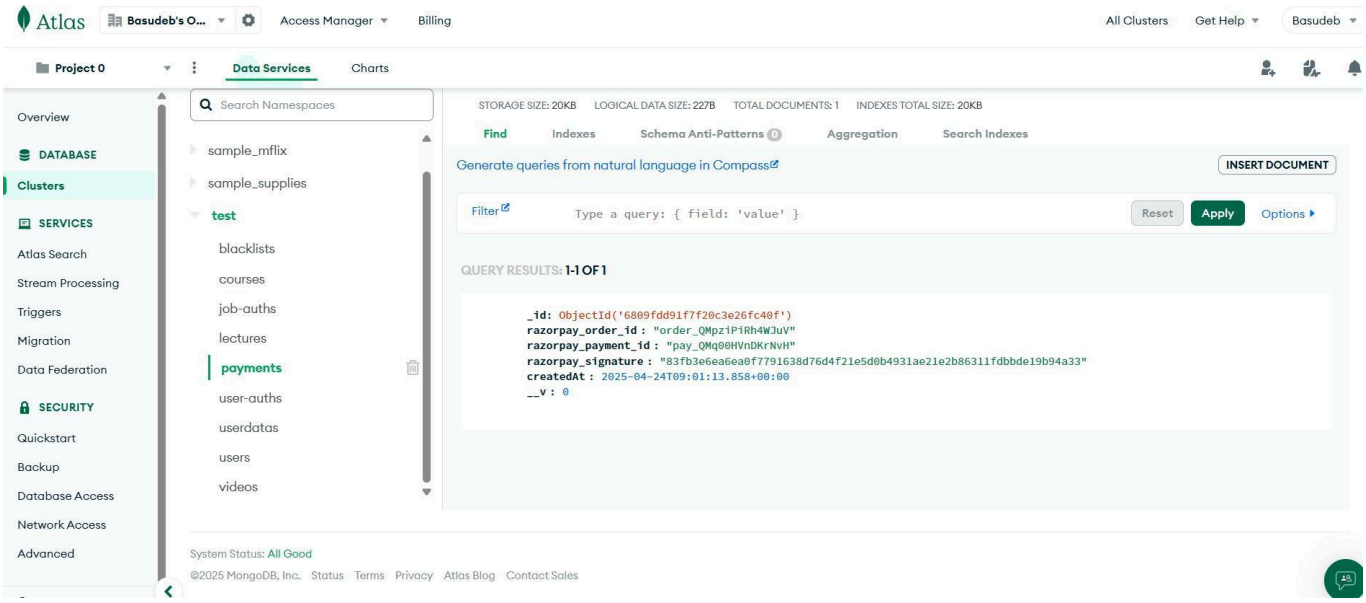
2) COURSE DATA:



- **model – Course.js :-**

```
import mongoose from "mongoose";

const schema = new mongoose.Schema({
  title: {
    type: String, required:
    true,
  },
  description: { type:
    String, required:
    true,
  },
  image: {
    type: String, required:
    true,
  },
  price: {
    type: Number, required:
    true,
  },
  duration: { type:
    Number, required:
    true,
  },
  category: { type:
    String, required:
    true,
  },
  createdBy: { type:
    String, required:
```

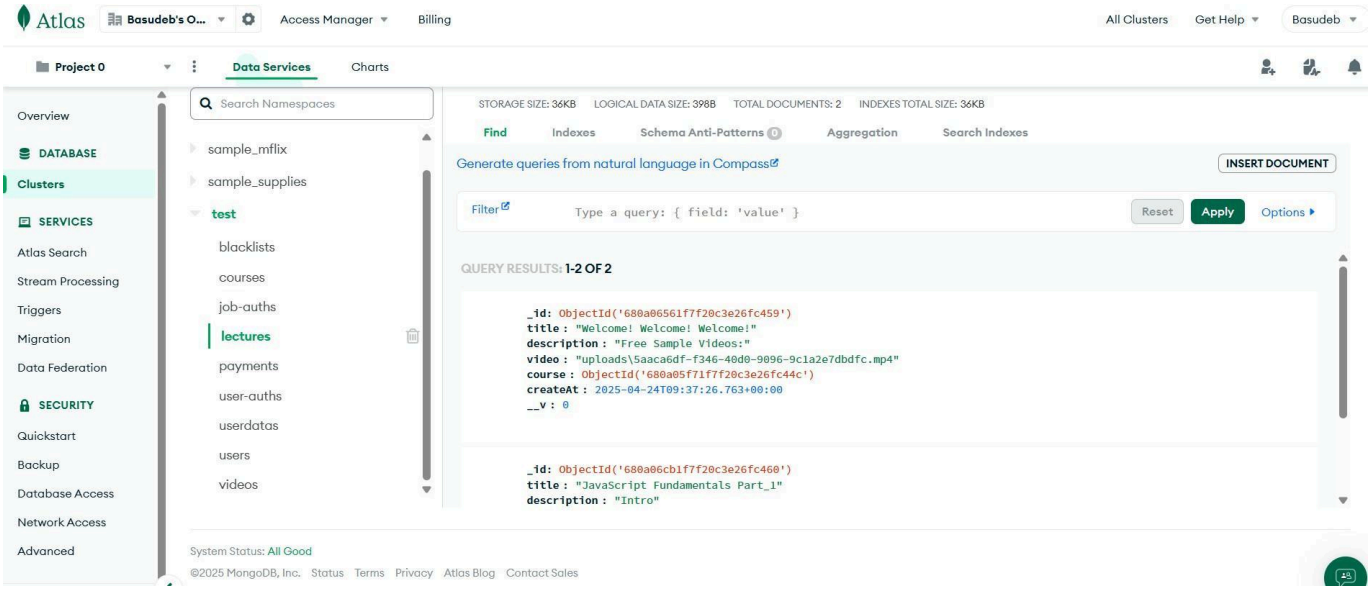
3) PAYMENT DATA



- **model – Payment.js :-**

```
import mongoose from "mongoose";

const schema = new mongoose.Schema({
  razorpay_order_id: {
    type: String, required:
    true,
  },
  razorpay_payment_id: {
    type: String, required:
    true,
  },
  razorpay_signature: {
    type: String, required:
    true,
  },

  createdAt: { type:
    Date,
    default: Date.now,
  },
});

export const Payment = mongoose.model("Payment", schema);
```

LECTURE DATA:



- **model – Lecture.js :-**

```
import mongoose from "mongoose";

const schema = new mongoose.Schema({
  title: {
    type: String,
    required: true,
  },
  description: { type:
    String, required:
    true,
  },
  video: {
    type: String,
    required: true,
  },
  course: {
    type: mongoose.Schema.Types.ObjectId,
    ref: "Courses",
    required: true,
  },
  createAt: { type:
    Date,
    default: Date.now,
  },});
export const Lecture = mongoose.model("Lecture", schema);
```

CONCLUSION

The E-Learning App serves as a powerful tool for enhancing the educational experience by making learning more accessible, interactive, and personalized. By leveraging technology, it bridges the gap between educators and learners, enabling flexible and self-paced learning anytime, anywhere. The app not only supports a wide range of subjects and courses but also incorporates features like progress tracking, quizzes, and multimedia content to enrich user engagement. As digital learning continues to grow, this app stands as a promising solution to meet the evolving educational needs of students and educators alike.

FUTURE SCOPE & FURTHER ENHANCEMENTS

❖ Future scope:-

The future of the E-Learning App is promising with numerous opportunities for growth and innovation. Key areas for development include:

- **AI-Powered Personalization:** Integrating artificial intelligence to offer personalized learning paths, smart recommendations, and adaptive assessments based on individual learning styles and progress.

- **Gamification:** Adding more interactive and game-like features to boost user engagement and motivation.

- **AR/VR Integration:** Enhancing the learning experience with augmented and virtual reality for immersive, hands-on simulations, especially in fields like medicine, engineering, and science.

- **Offline Accessibility:** Expanding features to allow content access without an internet connection, ensuring learning is uninterrupted in remote or low-connectivity areas.

- **Multilingual Support:** Supporting multiple languages to cater to a diverse, global user base.

- **Collaborative Learning Tools:** Incorporating real-time collaboration features like discussion forums, group projects, and peer assessments

❖ Further enhancement:-

1. User Experience Enhancements:

- **Personalized Dashboards**: Show progress, upcoming lessons, and tailored course suggestions.
- **Gamification:** Add badges, leaderboards, or achievement systems to increase engagement.
- **Dark Mode & Accessibility Features:** Improve usability for all users.

2. Content & Learning Tools:

- **Interactive Quizzes:** Timed tests, drag-and-drop activities, and immediate feedback.
- **Video Notes & Transcripts:** Let users take notes alongside videos; include auto-generated transcripts.
- **AI Tutors:** Use AI chatbots to answer questions or explain concepts on demand.

3. Community Features:

- **Live Classes & Webinars:** Integrate with video conferencing tools for real-time interaction.
- **Mentorship/Study Groups:** Match users with mentors or form study groups based on interest.

4. Analytics & Progress Tracking:

- Detailed Progress Reports: For students, parents, or teachers.
- Adaptive Learning Paths: Adjust difficulty and content based on user performance.

5. Tech Integrations:

- **Cloud Sync & Offline Mode**: Allow learning anytime, even without an internet connection.
- **Multilingual Support:** Expand your reach globally.
- **Integration with LMS or School Portals:** Sync with Google Classroom, Moodle, etc.

BIBLIOGRAPHY

1) www.w3schools.com

2)www.youtube.com

3)www.pexels.com

4)www.codepen.io

5)www.google.com

6)www.googlefont.com

7)www.react.our

8)www.codingworld.com