INDIRA GANDHI NATIONAL OPEN UNIVERSITY

MCSP - 232

ONE SPOT EDUCATION

By

Anjali Pandey

Enrolment No: 2200811729

Under Guidance

of

Satyam Srivastava

Submitted to the School of Computer and Information Sciences
in partial fulfilment of the requirements
for the degree of

Masters of Computer Applications 2024

Indira Gandhi National Open University

Maidan Garhi

New Delhi-110068.

Table of Contents

[i] Proforma for Approval of Project Proposal

SCHOOL OF COMPUTER AND INFORMATION SCIENCES
IGNOU, MAIDAN GARHI, NEW DELHI – 110 068

II. PROFORMA FOR THE APPROVAL OF MCA PROJECT PROPOSAL (MCSP-232)

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

Project Proposal No :.....

(for office use only)

Study Centre: COE (CENTER FOR ONLINE EDUCATION) **Enrolment No.:** 2200811729
Regional Centre:..... **RC Code:**..... 29 DELHI 2
E-mail: ..coe@ignou.ac.in.... **Mobile/Tel No.:** 9953885656

1. Name and Address of the Student: ANJALI PANDEY, B-788 PANDEY NIWAS NEW ASHOK NAGAR, NEW DELHI - 110096
2. Title of the Project***: ONE SPOT EDUCATION
3. Name and Address of the Guide: SATYAM SRIVASTAVA, Banjariya Purvi Khalilbad, Sant Kabir Nagar, U.P 272175
4. Educational Qualification of the Guide: (Attach bio-data also)
   (*in Computer Science / IT only)

Ph.D*	M.Tech.*	B.E*/B.Tech.*	MCA	M.Sc.*
☐	☐	☐	☑	☐

5. Working / Teaching experience of the Guide**: Good 7 years of Working Experience as Software Engineer [C.V Attached]
   (**Note: At any given point of time, a guide should not provide guidance for more than 5 MCA students of IGNOU)
6. Software used in the Project***: PHP, Laravel Framework, JQuery, Js, SASS, CSS, MySQL etc
   (*** Please refer to section VIII of these guidelines)
7. Is this your first submission? ☑ Yes ☐ No

Signature of the Student Anjali
Date: 20/07/2023

Satyam Srivastava
Signature of the Guide
Date: 25/07/2023

For Office Use Only

Name: .....

☑ Approved ☐ Not Approved

3.1

Signature, Designation, Stamp of the Project Proposal Evaluator

Date: .....

Suggestions for reformulating the Project:

[ii] Bio-data of the project guide

Curriculum Vitae

SATYAM SRIVASTAVA
Contact No. : 8948044456, 9456972323
E-mail : developer.satyamji@gmail.com
Address : B Block, New Ashok Nagar, New Delhi- 110096

Career Objective

I want to utilize my skills and potential to achieve a challenging position in the Software Industry and fulfill organizational goals.

Professional Qualification

- M.C.A. from PT University in 2015.
- B.Ed from Chaudhary Charan Singh University Meerut in 2012.
- B.C.A. from MD University, ROHTAK in 2010.

Education Qualification

- High School from Tilak Inter College Bansi (Siddharth Nagar) UP Board in 2004.
- Intermediate from Ratansen Inter College Bansi (Siddharth Nagar) UP Board in 2006.

Additional Qualification

- PHP from CETPA, Sector-2 Noida, UP in 2016.
- .NET from Ducat, Sector-16 Noida, UP in 2010.

Employers

- Worked as a Software Engineer in Cybercodex Software Solutions (P) Ltd from 12-October-2016 to 03-July-2018.
- Worked as a Software Engineer in Softlogique IT Solutions Pvt Ltd from 03-July-2018 to 28-Jan-2022.
- Currently working as a Software Engineer in OnSumaye Web Solutions Pvt. Ltd. from 28-Jan-2022 till present.

Technical Skills

- Skills: Core PHP, MySQL, HTML5, CSS, Bootstrap, Ajax, jQuery, JavaScript
- PHP Framework: CodeIgniter
- PHP CMS: WordPress, Joomla
- API: REST, SOAP
- Payment Gateways: PayPal Checkout
- WEB Server: XAMPP, WAMP, MAMP, LAMP

Satyam Srivastava

- AWS Server
- Operating Systems: Windows Enterprise, Ubuntu, Mac
- GitHub
- Tools / Software: Notepad++, Visual Studio, Dreamweaver

Key Skills

- Hands-on experience in Coding, Testing, Implementation/Maintenance Support in PHP and MySQL.
- Extensive development of jQuery, JavaScript.
- Hands-on experience in using CodeIgniter Framework to develop a web application.
- Quick learner and eager to learn new technologies.
- OOPS PHP development.
- Integration Ability: Strong in understanding and integrating third party web services.
- Platforms: Windows Server
- Client-side programming: JavaScript, AJAX, DOM, jQuery.
- Web Server installation & configuration: Apache, Microsoft IIS.

Academic Achievements & Extra Curricular Activities

- Participation in various software competitions.
- Successfully Manage Events in College.

Strengths

- Adaptability.
- Innovative.
- Quick Learner.
- Smart working.
- Positive attitude about learning new things.

Hobbies

- Reading the Daily News Paper.
- Surfing the internet for new ideas.
- Watching Hollywood Movies
- Reading books.

Projects Details

Satyam Srivastava

1. **CommLoan**

Duration: Currently Working (Tech Support and New Features)
Team Size: 10
Role played: Developer
Skills used: PHP MVC Joomla, MySQL, HTML5, CSS, Bootstrap, jQuery, Ajax
URL: https://www.commloan.com
Description:
CommLoan is a bridge between Borrowers and Lenders. CommLoan has streamlined the process of obtaining commercial real estate financing. With the touch of a button, we can sort through hundreds of lenders and thousands of loan programs to find the right loan that fits the borrower's specific needs.

Features -

Admin Management
Lender Management
Referral Source Management
Marketing Materials
Loan Pipeline
Reports
Borrower Login Portal
Lender Login Portal
Referral Login Portal

2. **Hospital Management System**

Duration: 6 Months
Team Size: 3
Role played: Developer and tester
Skills used: PHP MVC CodeIgniter, MySQL, HTML5, CSS, Bootstrap, jQuery, Ajax, REST API and SOAP API
URL: https://adminpanel.hanhphuchospital.com
Description:
Hospital Management System software offers a wide range of flexibility to manage the Products, Employees, Doctors, Patients, Patient's appointments, Patient's feedback, mobile app banner, data, etc.

Features -

Easy Mobile App Management
Flexible Appointment Management
Product Management
Patient Management
Role Management
Employee Management

Satyam Srivastava

Notifications
Dynamic Reports

3. **Online Quiz on Mobile APP**

Duration: 3 months
Team Size: 2
Role played: Developer and tester
Skills used: PHP MVC CodeIgniter, MySQL, HTML5, CSS, Bootstrap, jQuery, Ajax, REST API
URL: https://fidesrexpax.com/quizAdmin/
Description: Client can do web sitting and mobile App basic setting through Admin Panel. They can create, update, delete and set access permission of Employees. Clients make Categories, Sub Categories, Quiz sets for Users. They make Questions under the Quiz group. Then they published a quiz on mobile APP. They can see Users' results and send Notifications to Users. Users can register through mobile apps and login. Users' login with Mobile APP and start Quiz and get his quiz result.
Features -

Users Management
Roles Management
Categories Management
Sub Categories Management
Quiz Management
Questions Management
Notifications
Dynamic Reports

4. **Check Clients**

Duration: 2 months
Team Size: 2
Role played: Developer and tester
Skills used: PHP MVC Codeigniter, MySQL, HTML5, CSS, Bootstrap, JQuery, Ajax, REST API
URL: https://admin.scoro-comments.businesscontinuumsolutions.com
Description:
Check client Software is export data from workflowmax web application to excel sheet or Dropbox through API. Here we configure workflowmax API Key, workflowmax account Key and Dropbox access token. After this, we export data on Click or we schedule automatically from this web application. WorkflowMax is project management software. This software offers a wide

Satyam Srivastava

range of quoting, scheduling, time tracking, invoicing and reporting – and gets a much better picture of your people and your profit.

Features –

Users Management
Roles Management
Export Management
Dynamic Reports

5. **Neon Education**

Duration: 6 Months
Team Size: 2
Role played: Designer, Developer and tester
Skills used: PHP WordPress CMS, BuddyBoss theme, LearnDash LMS plugin, MySQL, HTML5, CSS3, Bootstrap, jQuery, Ajax, WooCommerce, custom plugins and many other Plugins. I hosted this Project on AWS server.
URL: https://neon-edu.com
Description: Neon Education is a LMS project. It has Student, parent and Teacher login Panel. Student and teacher can purchase course and study online Zoom meet live class. He has Library, Lesion Plan, and many things related to study materials.

Features -

Student Management
Parent Management
Teacher Management
Notifications
Dynamic Reports

6. **Gantt chart Plugin**

Duration: 1 month
Team Size: 1
Role played: Developer and tester
Skills used: PHP CMS WordPress, MySQL, HTML5, CSS, Bootstrap, JQuery, Ajax
URL: http://nitdemo.in/ganttchart
Description:
This plugin provides you with a Gantt chart. You can easily manage and add sectors, goals and subgoals. Assign the task to team members. Automatically send reminder mail basis of weekly and daily to goals, sub-goals creator and assign task team members.

Features -

Satyam Srivastava

Manage your sector
Manage your goals
Manage your sub-goals
Send reminder mail on a weekly and daily basis.
You can create an add goals form page; add a team member form page and a year planner page just using a shortcode.

Personal Details

Name	:	Satyam Srivastava
Father Name	:	Mr. Sunil Kumar Srivastava
Date of Birth	:	24th-Oct-1989
Sex	:	Male
Nationality	:	Indian
Marital Status	:	Married
Language Known	:	Hindi, English
Permanent Address	:	Mohalla- Banjariya (Purbi), Khalilabad Sant Kabir Nagar Utter Pradesh- 272175
Contact No.	:	+91-8948044456, +91-9456972323

Declaration

I hereby declare that all the information furnished above is true to the best of my knowledge.

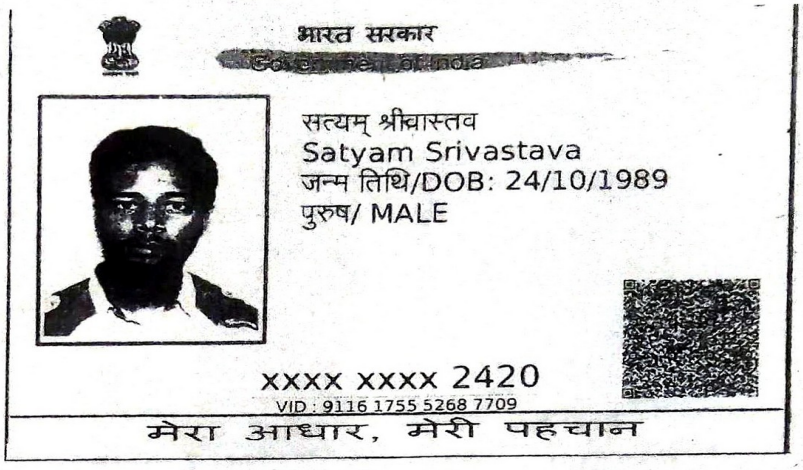
Date: 25/07/2023

Place: Delhi

SATYAM SRIVASTAVA

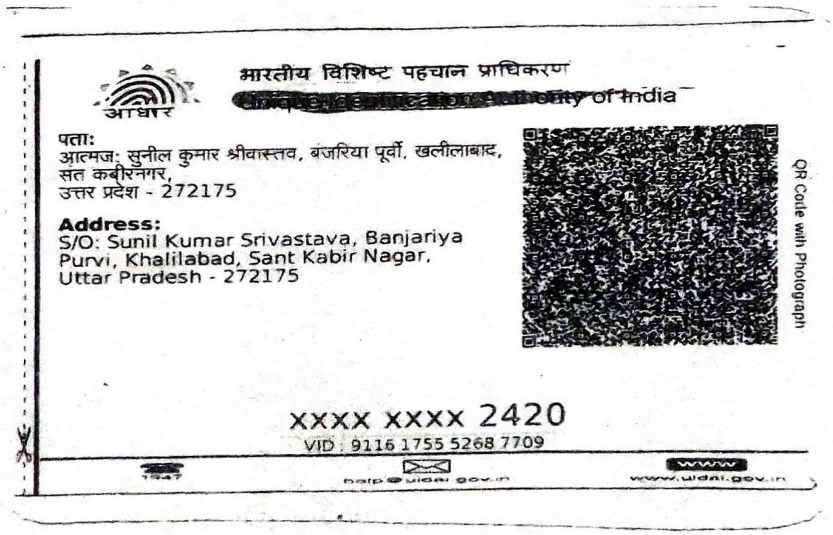
Satyam Srivastava

[iii] Identity proof of guide


भारत सरकार

सत्यम् श्रीवास्तव
Satyam Srivastava
जन्म तिथि/DOB: 24/10/1989
पुरुष/ MALE

xxxx xxxx 2420
VID : 9116 1755 5268 7709
मेरा आधार, मेरी पहचान


भारतीय विशिष्ट पहचान प्राधिकरण

पता:
आत्मज: सुनील कुमार श्रीवास्तव, बंजरिया पूर्वी, खलीलाबाद,
सत कबीरनगर,
उत्तर प्रदेश - 272175

Address:
S/O: Sunil Kumar Srivastava, Banjariya
Purvi, Khalilabad, Sant Kabir Nagar,
Uttar Pradesh - 272175

QR Code with Photograph

xxxx xxxx 2420
VID : 9116 1755 5268 7709

help@uidai.gov.in
www.uidai.gov.in

Satyam Srivastava

APPROVED

SYNOPSIS / PROJECT PROPOSAL

Title of Project

ONE SPOT EDUCATION

(A one stop revolutionary digital platform for educational institutions to transform traditional education with feature-rich innovative technologies **)**

[2] Introduction &Objectives

Introduction

In today's rapidly evolving world, the traditional education system is facing numerous challenges that hinder its ability to provide students with a dynamic and engaging learning experience. The need of the hour is a transformation that seamlessly integrates education with the power of technology. This project proposal introduces a comprehensive web portal aimed at revolutionizing education, empowering educational agencies, schools, universities, and other similar institutions to embark on a digital transformation journey. Our web portal leverages the latest advancements in technology to provide an all-in-one platform that caters to the diverse needs of students, educators, and administrators. By combining live and recorded classes, interactive features, class chat functionality, and user-friendly interfaces, our web portal aims to create an immersive and interactive learning environment that enhances student engagement, collaboration, and overall learning outcomes. The key features of our web portal include live and recorded classes, enabling students to attend real-time sessions or access course materials at their convenience. This flexibility allows learners to access high-quality education irrespective of their physical location or time constraints. Additionally, the class chat functionality, inspired by popular messaging applications, fosters real-time interaction between students and educators, promoting engagement, peer collaboration, and immediate feedback.

Automation plays a crucial role in our web portal, simplifying administrative tasks and enhancing efficiency. The platform offers an auto-attendance feature that eliminates manual tracking, reducing paperwork and streamlining the process. Furthermore, our comprehensive syllabus, assignment, and practicals management system allows educators to organize, distribute, and assess coursework effortlessly. This centralization of educational resources promotes transparency, making it easier for students and parents to stay updated on academic progress. We understand that the visual appeal and user experience significantly impact learner engagement. Therefore, the user interface of our web portal is carefully crafted to be intuitive, attractive, and easy to navigate. The impact of our web portal extends beyond the virtual classroom. With features like homework submission and personalized assessments, students are empowered to take ownership of their learning. Educators can provide tailored feedback and track individual progress, ensuring targeted support for each student's academic growth. Additionally, our platform offers a comprehensive suite of tools for collaborative projects, fostering critical thinking, creativity, and effective communication skills.

By adopting our web portal, educational agencies, schools, universities, and other institutions can revolutionize their education systems, embracing the digital age without compromising on quality or integrity. Our platform serves as a catalyst for change, empowering educators to unleash their full potential and enabling students to thrive in a technology-driven world. In conclusion, our comprehensive web portal aims to transform the traditional education system into a dynamic, interactive, and learner-centric experience. By seamlessly integrating technology into classrooms, we aim to enhance engagement, collaboration, and educational outcomes.

Objectives

- **Digitize Education:** Transform the traditional education system into a digital environment, enabling educational agencies, schools, universities, and other institutions to embrace technology and enhance the learning experience.
- **Provide Live and Recorded Classes:** Offer both live and recorded classes to cater to different learning preferences and accommodate various schedules, ensuring flexibility and accessibility for students.
- **Foster Collaboration and Engagement:** Facilitate collaboration and interaction between students and educators through class chat functionality, encouraging active participation, peer-to-peer learning, and immediate feedback.
- **Streamline Administrative Tasks:** Automate administrative tasks such as attendance tracking, homework submission, and assessment management to reduce paperwork, optimize efficiency, and free up time for educators and administrators.
- **Centralize Syllabus and Course Management:** Provide a centralized platform for syllabus organization, assignment distribution, and practical management, simplifying educational resource management and promoting transparency.
- **Enhance User Experience:** Design an intuitive, attractive, and user-friendly interface to create an engaging and enjoyable learning environment for students, educators, and administrators.
- **Personalized Learning Experience:** Offer features that allow educators to provide personalized assessments, tailored feedback, and individual progress tracking, empowering students to take ownership of their learning journey.
- **Promote Effective Communication:** Facilitate seamless communication between students, educators, and parents through integrated messaging and notification systems, fostering strong relationships and effective collaboration.
- **Optimize Learning Outcomes:** Enable educators to design interactive and immersive learning experiences that promote critical thinking, creativity, and problem-solving skills, ultimately improving overall learning outcomes.
- **Data-Driven Insights:** Generate comprehensive analytics and reports on student performance, engagement levels, and learning outcomes, empowering educators and administrators to make data-driven decisions for continuous improvement.
- **Parental Involvement:** Facilitate parental involvement and engagement in their child's education through parent portals, progress tracking, and communication channels, fostering a collaborative learning ecosystem.
- **Security and Privacy:** Implement robust security measures to protect user data, ensure privacy compliance, and maintain a secure learning environment for all stakeholders.

[3] Project Category

Our project falls under the category of an "Education Technology" or "EdTech" application. It aims to digitalize the traditional education system and provide advanced features through a web portal. The application incorporates elements of RDBMS (**Relational Database Management System**) and OOPs (**Object-Oriented Programming**) to effectively manage data and create a scalable, interactive, and user-friendly platform.

In terms of RDBMS (Relational Database Management System) and OOPs (Object-Oriented Programming), the project can be categorized as follows:

1. RDBMS : The project involves the use of a relational database management system to store, organize, and manage data related to various aspects of the education system. This includes storing information about students, teachers, courses, assignments, attendance, and other relevant data in structured tables and establishing relationships between them.The RDBMS component of the project would handle tasks such as designing an appropriate database schema, creating tables, defining relationships and constraints, implementing efficient queries for data retrieval, and ensuring data integrity through proper transaction management and normalization techniques.

2. OOPs : The project utilizes the principles of Object-Oriented Programming to design and develop the web application. The various entities and functionalities within the system, such as students, teachers, classes, assignments, and interactions, can be represented as objects with their own properties (attributes) and behaviors (methods). The OOPs aspect of the project would involve designing and implementing classes, encapsulating data and functionality within them, defining inheritance and polymorphism relationships, and applying other OOPs concepts to create modular, reusable, and maintainable code. This approach allows for the development of a flexible and extensible system that can easily accommodate future enhancements and changes.

[4] Tools/Platforms ,Hardware and Software Requirement Specification

A. *Tools/Platforms Software Requirements:-*

- **Operating System:**
  - Server: Linux (e.g., Ubuntu Server, CentOS)
  - Clients: Windows, macOS, Linux
- **Web Server:** apache (with necessary modules and configurations)
- **Programming Language:** PHP 7.x (with necessary extensions)
- **Framework:** Laravel Framework version 10
- **Database Management System:** MySQL 7 or above
- **Version Control:** Git (for source code management and collaboration)
- **Text Editor or Integrated Development Environment (IDE):** Visual Studio Code, Sublime Text, PhpStorm, or any preferred IDE for coding and development
- **Package Manager:** Composer (for managing PHP dependencies)
- **Web Browser :** Latest versions of Chrome, Firefox, Safari, or Edge (for testing and accessing the web application)
- **Deployment and Hosting:**
  - Server Management Software (e.g., SSH, SCP) for deploying the application to the server.
  - Hosting Service or Cloud Platform (e.g., AWS, DigitalOcean, Google Cloud) to host the web application.
- **Testing Framework (optional but recommended):** PHPUnit (for unit testing)

B. ***Hardware Requirements:-***

Server Configuration: -

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB.
- Storage: 256 GB SSD.
- Network Interface: Gigabit Ethernet.
- Operating System: Linux (e.g., Ubuntu Server).
- Web Server: Nginx or Apache.
- Database: MySQL.
- Backup System: Regular data backups with off-site storage.

Client Configuration : -

- IProcessor: Intel Core i3 or equivalent.
- RAM: 4 GB.
- Network Interface: Ethernet or Wi-Fi.
- Operating System: Windows, macOS, or Linux.

The clients should have a good rate of data transfer with the server for quick performance.

[5] Problem Definition, Requirements Specifications, Project Planning & Scheduling

Problem Definition: -

The Problem Definition section for the OneSpotEducation project outlines the specific challenges and shortcomings in the traditional education system that the web portal aims to address. It includes a clear description of the problems faced by educational institutions, educators, students, and parents, such as:

- **Limited access to quality education:** Many students face barriers in accessing quality education due to factors like geographical location, lack of resources, or inadequate infrastructure.
- **Ineffective communication and collaboration:** Traditional methods of communication and collaboration among students, educators, and parents can be inefficient and hinder effective knowledge sharing and engagement.
- **Inefficient administrative processes:** Manual administrative tasks such as attendance tracking, course management, and assignment grading can be time-consuming, error-prone, and inefficient.
- **Lack of interactivity and engagement:** Traditional education often lacks interactive and engaging learning experiences, leading to reduced student motivation and limited learning outcomes.
- **Limited Personalized Learning:** The one-size-fits-all approach in traditional classrooms may not cater to the unique learning styles, paces, and interests of individual students, limiting their potential for academic growth.
- **Lack of Flexibility in Learning:** Traditional education often follows rigid schedules and fixed locations, making it challenging for students to balance their academic commitments with other personal and extracurricular activities.
- **Outdated Learning Materials:** Traditional textbooks and resources may become outdated quickly, resulting in a lag in providing up-to-date and relevant learning materials to students.
- **Insufficient Assessment and Feedback:** Traditional assessment methods may not provide timely and comprehensive feedback to students, hindering their understanding of strengths and areas for improvement.
- **Difficulty in Monitoring Student Progress:** Tracking individual student progress and identifying areas where additional support may be required can be challenging in a traditional education system.
- **Lack of Data-Driven Insights:** Traditional education systems often lack robust data collection and analysis mechanisms, limiting the availability of actionable insights to inform educational decisions and improvements.

Requirements Specifications:-

A requirement specification outlines the detailed functional and non-functional requirements of a system, defining what the system should do and how it should perform. For the OneSpotEducation system, the requirement specification may include:

1. User Registration and Authentication:

- Users should be able to register an account with the system and provide necessary information.
- The system should authenticate users securely to ensure authorized access to the platform.

2. Dashboard:

- The system should provide a personalized dashboard for each user, displaying relevant information, notifications, and quick access to essential features.

3. Course Management:

- Educators should be able to create and manage courses, including adding course details, syllabus, and resources.
- Students should be able to enroll in courses, view course materials, and access resources.

4. Live and Recorded Classes:

- The system should support live online classes with video streaming and interactive features, allowing real-time communication between educators and students.
- Recorded classes should be available for on-demand viewing to accommodate flexible learning.

5. Assignment and Homework:

- Educators should be able to create assignments, set deadlines, and provide instructions.
- Students should be able to submit their assignments digitally, and educators should have a mechanism to review and grade submissions.

6. Attendance Management:

- The system should provide automated attendance tracking, allowing educators to monitor student attendance and generate reports.

7. Communication and Collaboration:

- The system should facilitate communication and collaboration between users through features such as chat functionality, discussion forums, and messaging tools.

8. Progress Tracking and Analytics:

- The system should track and analyze student progress, providing insights into performance, engagement levels, and learning outcomes.
- Educators and administrators should have access to comprehensive analytics and reports.

9. Resource Management:

- The system should provide a centralized repository for educational resources, including textbooks, multimedia materials, and supplementary resources.

10. User Notifications:

- The system should send notifications to users regarding important updates, upcoming events, assignment deadlines, and other relevant information.

11. Security and Privacy:

- The system should ensure secure data storage, transmission, and user authentication.
- It should comply with privacy regulations and protect user data.

12. Scalability and Performance:

- The system should be designed to handle increasing user load and ensure optimal performance even during peak usage.

13. Mobile Accessibility:

- The system should be accessible on mobile devices, allowing users to access course materials, participate in classes, and complete assignments on their smartphones or tablets.

14. Integration:

- The system should support integration with external tools or systems, such as learning management systems, video conferencing platforms, and online payment gateways.

15. User Interface and User Experience:

- The system should have an intuitive, user-friendly interface that is visually appealing and easy to navigate.
- It should provide a seamless and engaging user experience across different devices and screen sizes.

This requirement specification provides an overview of the key features and functionalities that the OneSpotEducation system should possess. It serves as a foundation for the system's design, development, and implementation, ensuring that the system meets the needs and expectations of its users, whether they are educators, students, or administrators.

Functional Requirements:-

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that show how a use case is to be fulfilled. They are supported by non-functional requirements, which impose constraints on the design or implementation (such as performance requirements, security, or reliability).As defined in requirements engineering, functional requirements specify particular behaviors of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability.

Functional requirements define the specific features, functionalities, and behaviors that the system must exhibit to fulfill its intended purpose. They describe what the system should do and how it should behave in response to user interactions. Here are some functional requirements for the OneSpotEducation system:

- **User Registration and Authentication:** Users should be able to create accounts, provide necessary information, and authenticate securely.
- **Dashboard:** Users should have access to a personalized dashboard displaying enrolled courses, assignments, announcements, and progress tracking.
- **Course Management:** Educators should be able to create and manage courses, add course materials, resources, and syllabic.
- **Live and Recorded Classes:** The system should support scheduling and delivery of live online classes, with interactive features like chat and Q&A. Recorded classes should be accessible for on-demand viewing.
- **Assignment and Homework Management:** Educators should be able to create and assign homework and assignments, students should be able to submit their work electronically, and educators should be able to provide feedback and grades.
- **Attendance Management:** The system should automate attendance tracking, allowing educators to record and monitor student attendance. Reporting and analytics on attendance should be available.
- **Communication and Collaboration:** The system should provide communication channels like discussion forums and private messaging for students, educators, and parents to interact and collaborate.
- **Reporting and Analytics:** The system should generate comprehensive reports and analytics on student performance, engagement levels, and course progress. Administrators and educators should have access to these reports.

Technical Requirements:-

Technical requirements define the underlying technology, infrastructure, and software components needed to implement and operate the system effectively. Here are some technical requirements for the OneSpotEducation system:

- **Web Application Framework:** The system should be developed using Laravel, a PHP web application framework.
- **Database Management System:** The system should utilize MySQL or PostgreSQL for data storage and retrieval.
- **Server Environment:** The system should be deployed on a server with sufficient processing power, memory, and storage to handle user traffic and data storage requirements.
- **Web Server:** The system should be compatible with popular web servers like Apache or Nginx.
- **Security Measures:** The system should implement secure user authentication, data encryption, and access control measures to ensure data security and user privacy.
- **Scalability:** The system should be designed to scale horizontally or vertically to handle increased user traffic and growing data storage needs.
- **Compatibility:** The system should be compatible with modern web browsers (e.g., Chrome, Firefox, Safari, Edge) and operating systems (e.g., Windows, macOS, Linux).

Nonfunctional Requirements:-

In systems engineering and requirements engineering, non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements". Qualities, of Non-functional requirements can be divided into two main categories. Execution qualities, such as security and usability, are observable at run time. Evolution qualities, such as extensibility and scalability, embody in the static structure of the software system. Non-functional requirements define the attributes or qualities that the system should possess, rather than specific functionalities. They describe how the system should perform and its characteristics. Here are some non-functional requirements for the OneSpotEducation system:

Project Planning & Scheduling:-

PERT Chart/ Task Network Chart:-

The Program Evaluation Review Technique (PERT) is the cost and time management system. PERT organizes that project is complex that some task must be completed before other can be stated and that the appropriate way to manage a project is to define and control each task. Because projects often fall behind schedules, PERT is designed to facilitate getting a project back on schedule. The PERT chart gives a graphical representation of this information.

Depending on the working priorities, the entire project can be subdivided into the following main modules, those are:-

- Users Module
- Course Module
- Class Module
- Assignment Assessment Module
- Activity Calendar Module
- System Admin and Config Module.

We can construct our activities plain as follows:-

Activity	Activity Name
A	User Management
B	Course Management
C	Live Classes
D	Assignment Assessment
E	Activity Calendar
F	System Admin and Config

Chart:-

Activity	Predecessor Activity	Time Estimated Weeks (Individual)
A	……	5
B	A	6
C	B	3
D	C	7
E	C	7
F	D,E	4

Critical Path Method (CPM): -

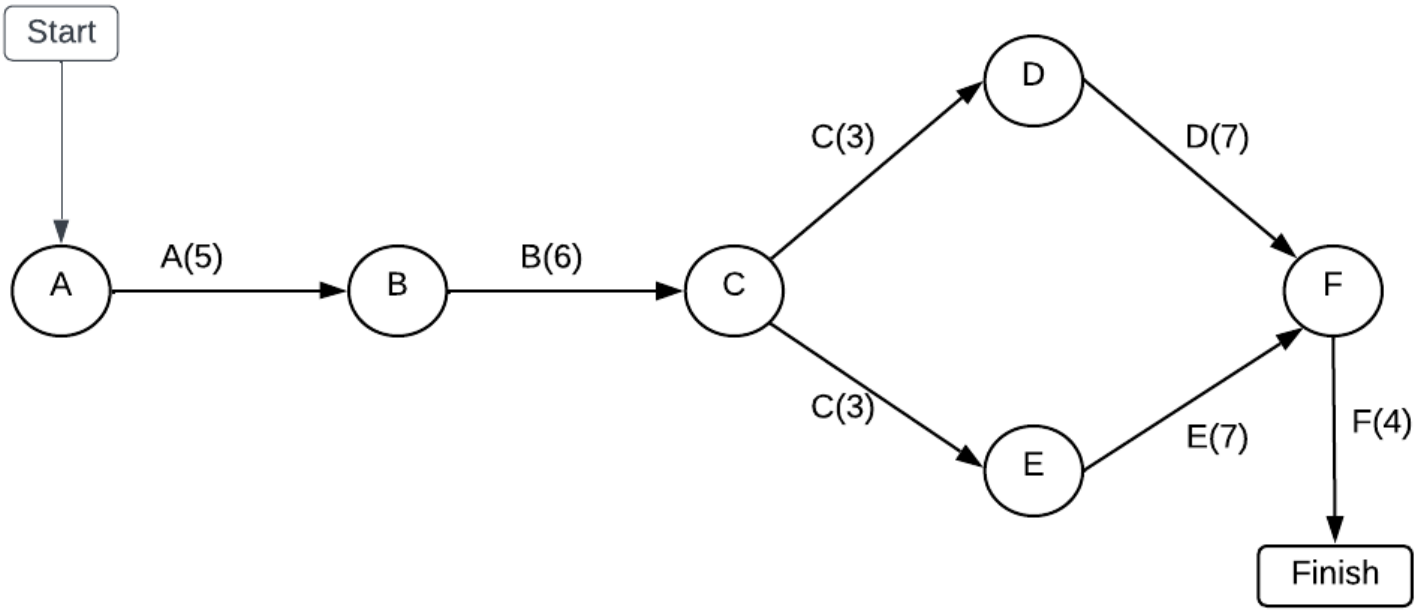


Fig: PERT Chart

Time Line Path: -

Path	Length Of Time
Start-A-B-C-D-F-Finish	5+6+3+7+4 = 25 WEEKS
Start-A-B-C-E-F-Finish	5+6+3+7+4 = 25 WEEKS

GANTT Chart:-

A Gantt chart is a popular type of bar chart that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project.

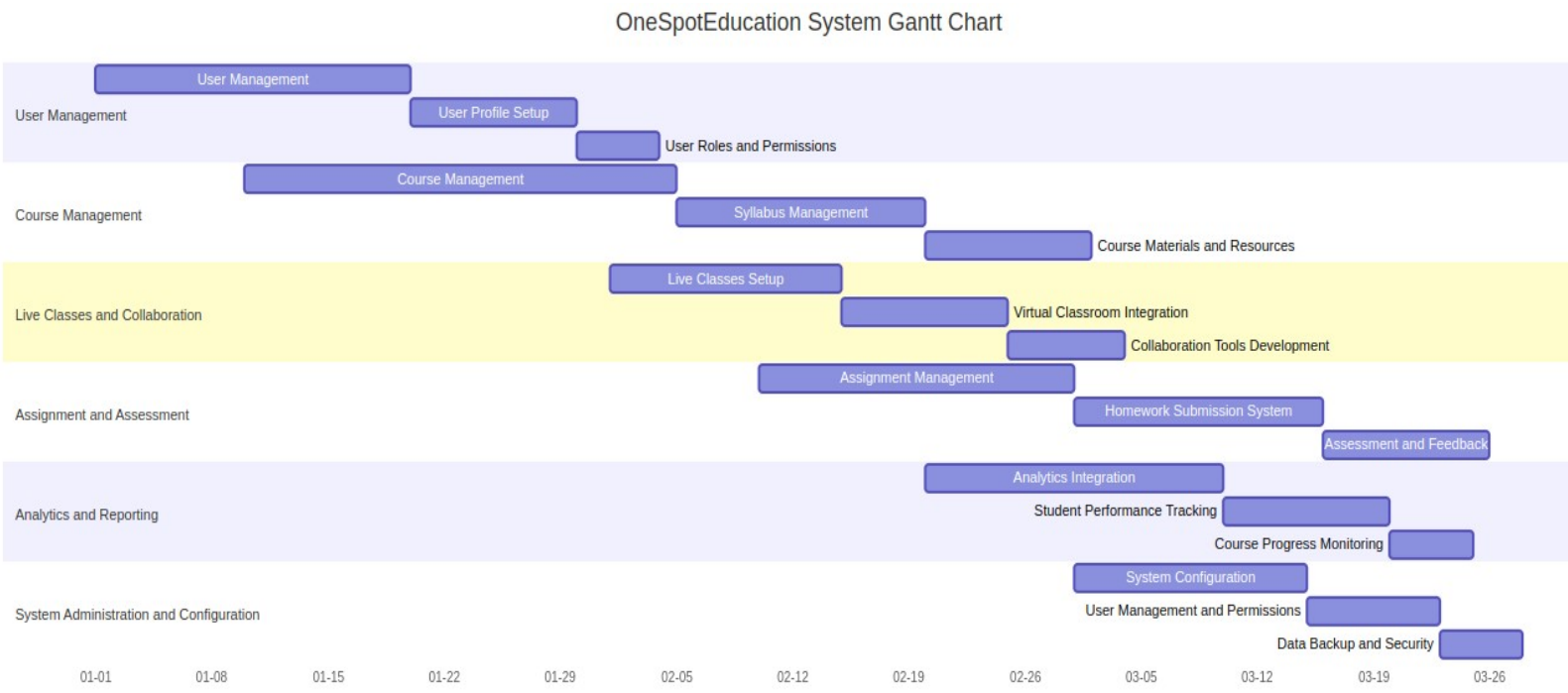


Fig: GANTT Chart

[6] Scope of the Solution

The scope of the OneSpotEducation system is to provide a comprehensive and integrated platform for digitizing and enhancing the traditional education system. It aims to revolutionize the way educational institutions, such as schools, universities, and other similar agencies, manage their teaching and learning processes. The solution encompasses a wide range of features and functionalities designed to streamline various aspects of education delivery and administration. The system's scope includes user management, allowing seamless registration, authentication, and profile management for students, teachers, administrators, and other stakeholders. It offers a centralized hub for course management, enabling efficient creation, organization, and distribution of course materials, syllabi, and resources. The platform supports both live and recorded classes, providing a virtual classroom environment with interactive tools for engaging and effective teaching.

Another key aspect of the solution is assignment and homework management, enabling instructors to create and distribute assignments while facilitating student submissions and grading. The system also encompasses attendance management, simplifying the process of recording and tracking student attendance for accurate reporting and analysis. Communication and collaboration features are integral to the solution, enabling seamless interaction between teachers, students, and administrators. These include chat functionality, discussion forums, messaging systems, and announcement tools, fostering effective communication and knowledge sharing within the educational community. Reporting and analytics capabilities form a crucial part of the system, providing insights into student performance, course progress, and attendance patterns. This data-driven approach empowers educators and administrators to make informed decisions, identify areas for improvement, and drive student success. System administration and configuration features ensure the smooth operation of the platform. They encompass settings management, user access control, and data security measures to safeguard sensitive information.

The scope of the OneSpotEducation system extends to integration with external learning resources, such as learning management systems (LMS) and educational content providers. This allows for a seamless and enriched learning experience, leveraging existing educational tools and content. Overall, the scope of the OneSpotEducation system is to offer a comprehensive, user-friendly, and scalable solution that digitalizes the education ecosystem, enhances collaboration, improves administrative efficiency, and promotes effective teaching and learning outcomes.

[7] Analysis (DFDs, ER Diagrams/ Class Diagrams etc.)

The whole approach of analysis of problem should however be based around critical factors like the availability of information for making the decision, the time available for processing the data i.e. the realism. System Requirement Specification or SRS had been prepared after proper discussion with the persons attached with the mentioned "**OSCM**". Software project management begins with a set of activities collectively called PROJECT PLANNING. Software project planning actually encompasses all of the activities. Planning involves estimation- to determine how much money, how much effort, how many resources, and how much time it will take to build a specific software-based system or product.

Phases Cover:-

- Pre–Analysis Studies
- System Analysis
- System Design
- Project Coding
- Project Testing
- Implementation & Documentation

- **Pre–Analysis Phase:** In this phase problems with existing system are to be determined and do the investigation to make the solutions.
- **System Analysis Phase:** In this phase system analysis is done by preparation of Software Requirement Specification.
- **System Design Phase:** The purpose of the design phase is to plan a solution for the problem specified in the requirements documents.
- **Project Coding Phase:** The goal of the coding phase is to translate the design of the system into a program code by help of a programming language like Visual Studio, Java, etc.
- **Project Testing Phase:** Testing concerned with the elimination of errors introduced during coding phase.
- **Implementation & Documentation Phase: -** This phase includes all the activities performed to keep the system operational after the installation of the software.

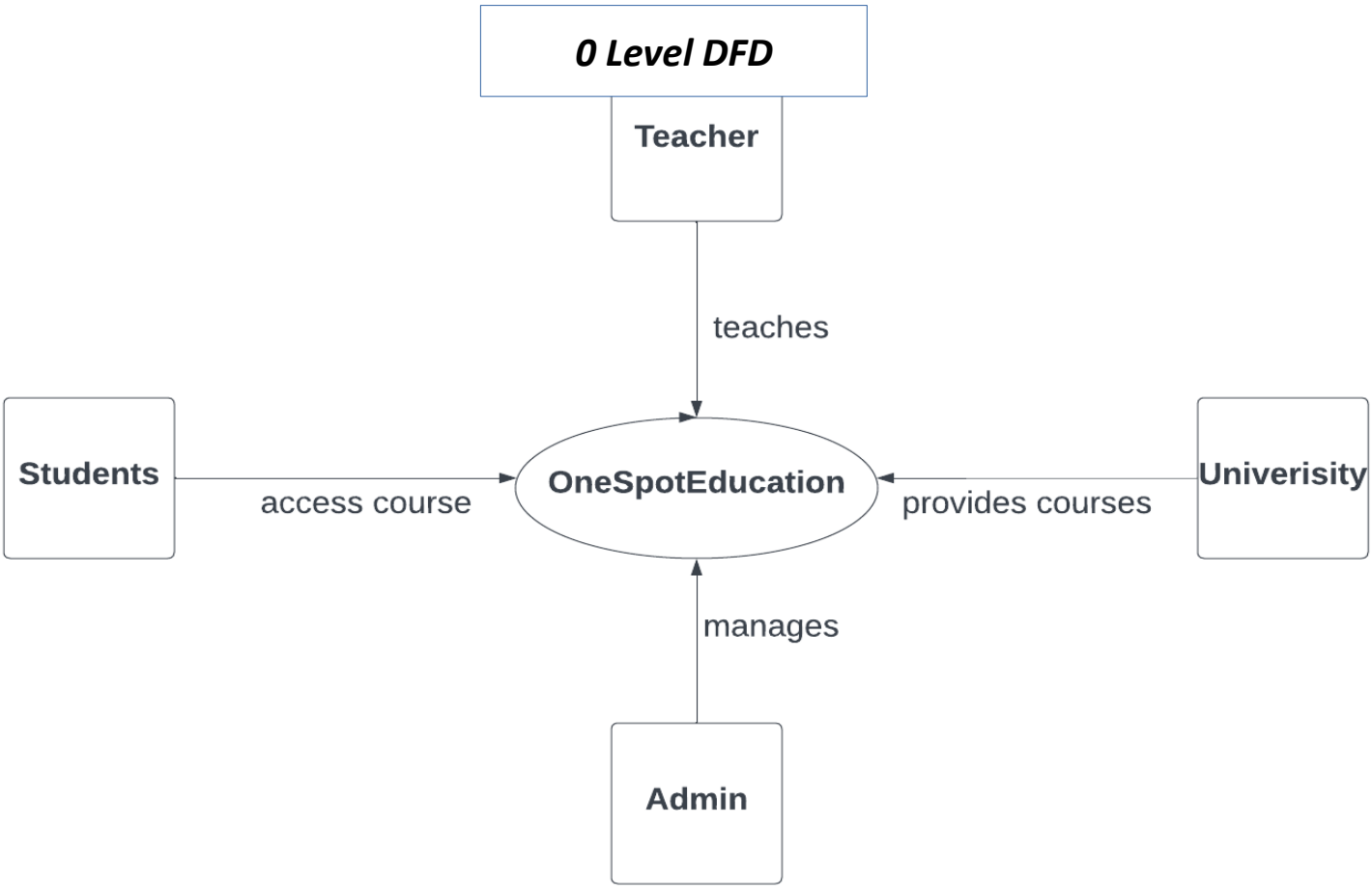
Data Flow Diagram (DFDs):-

Data flow diagrams models the passage of data in the system and are represented by lines joining system components. Flows of data in the system can take place between:-

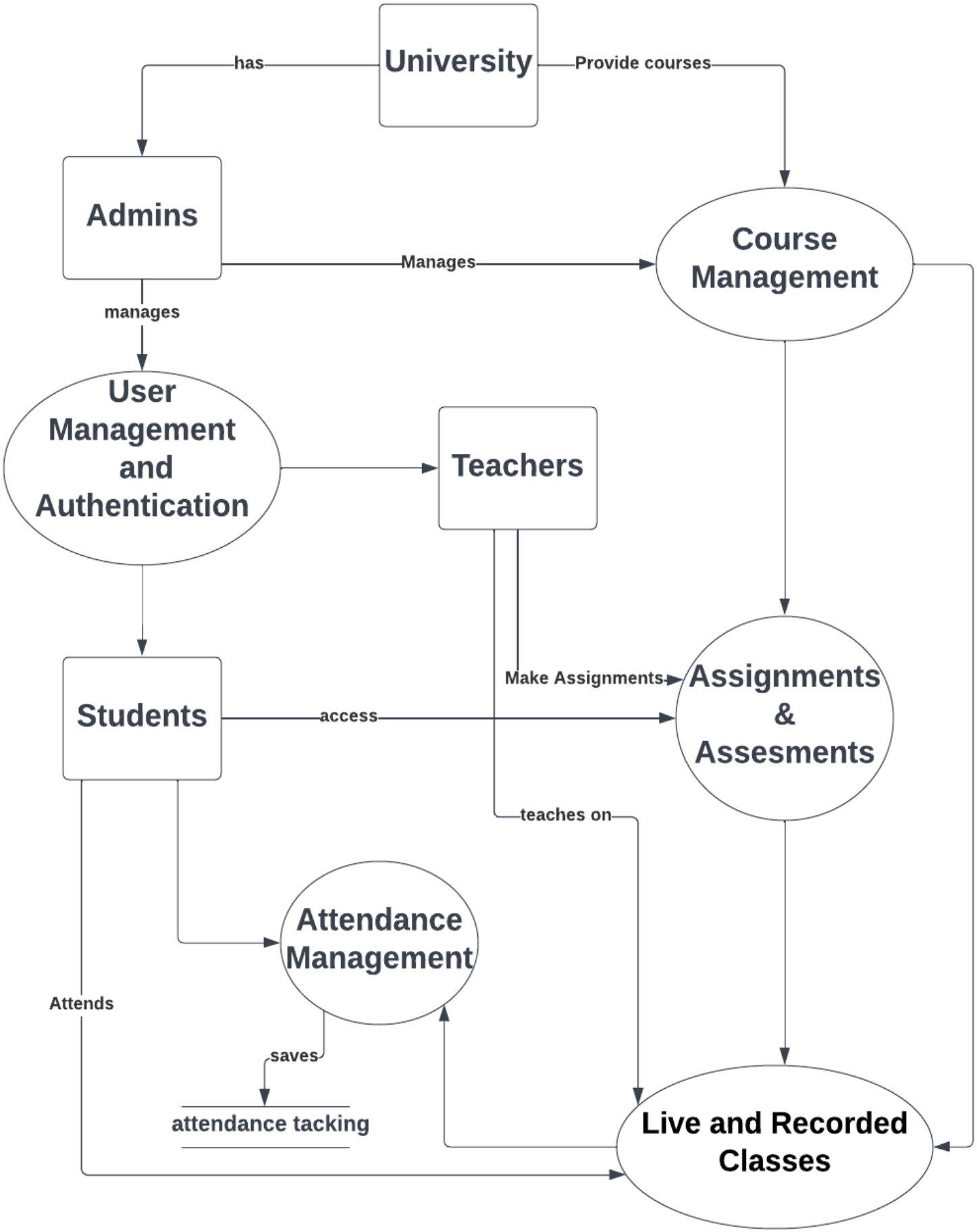
- Between two processes,
- From a data store to a process,
- From a process to data store,
- From a source to process and
- From a process to a sink.

Though the system mainly consists of two parts viz. online admission and online examination and other parts are going to be automated gradually…, so DFD is also, illustrated in two parts, respectively…

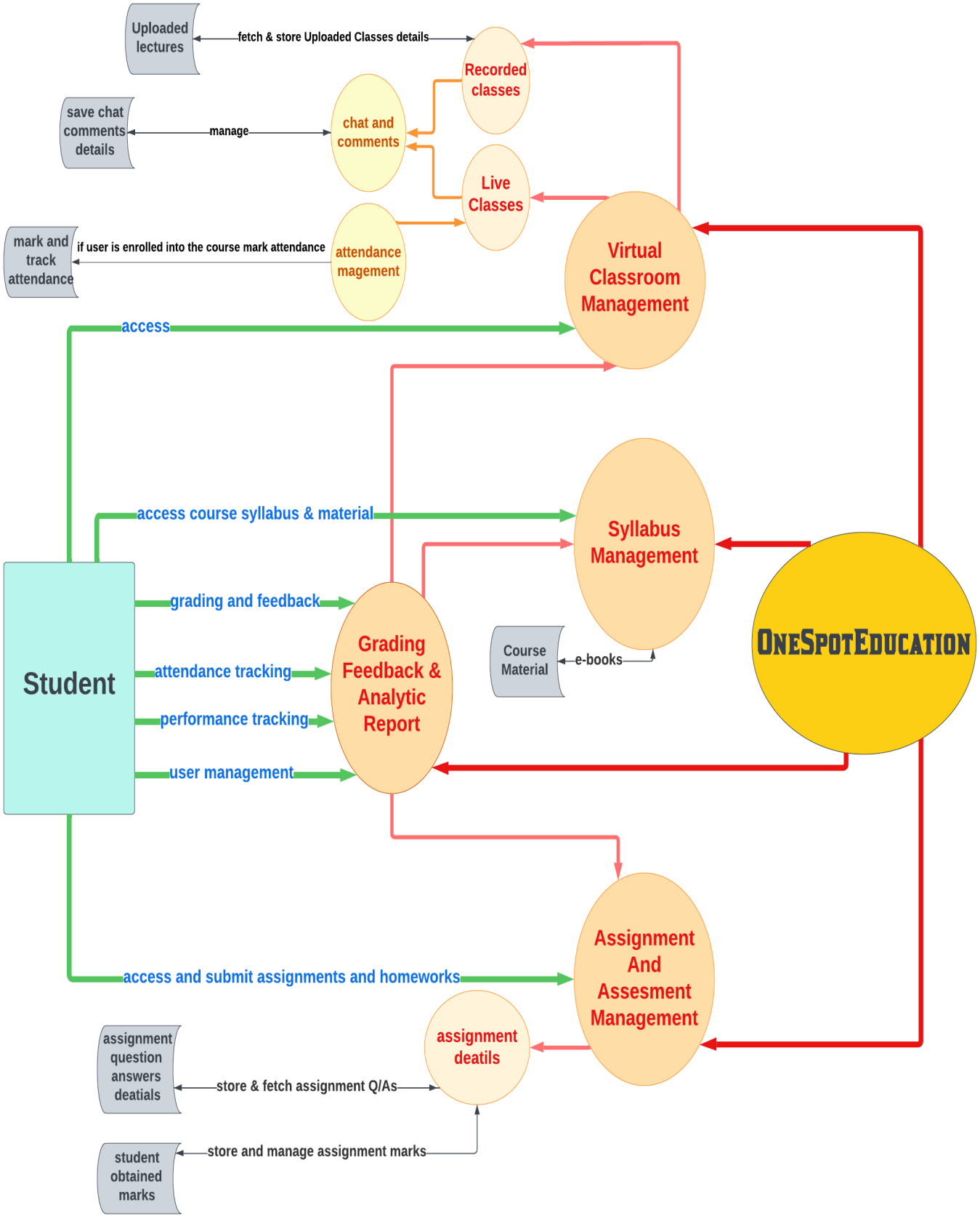
Context Level Diagram:-



1st Level DFD



2nd Level DFD



Class Diagram:-

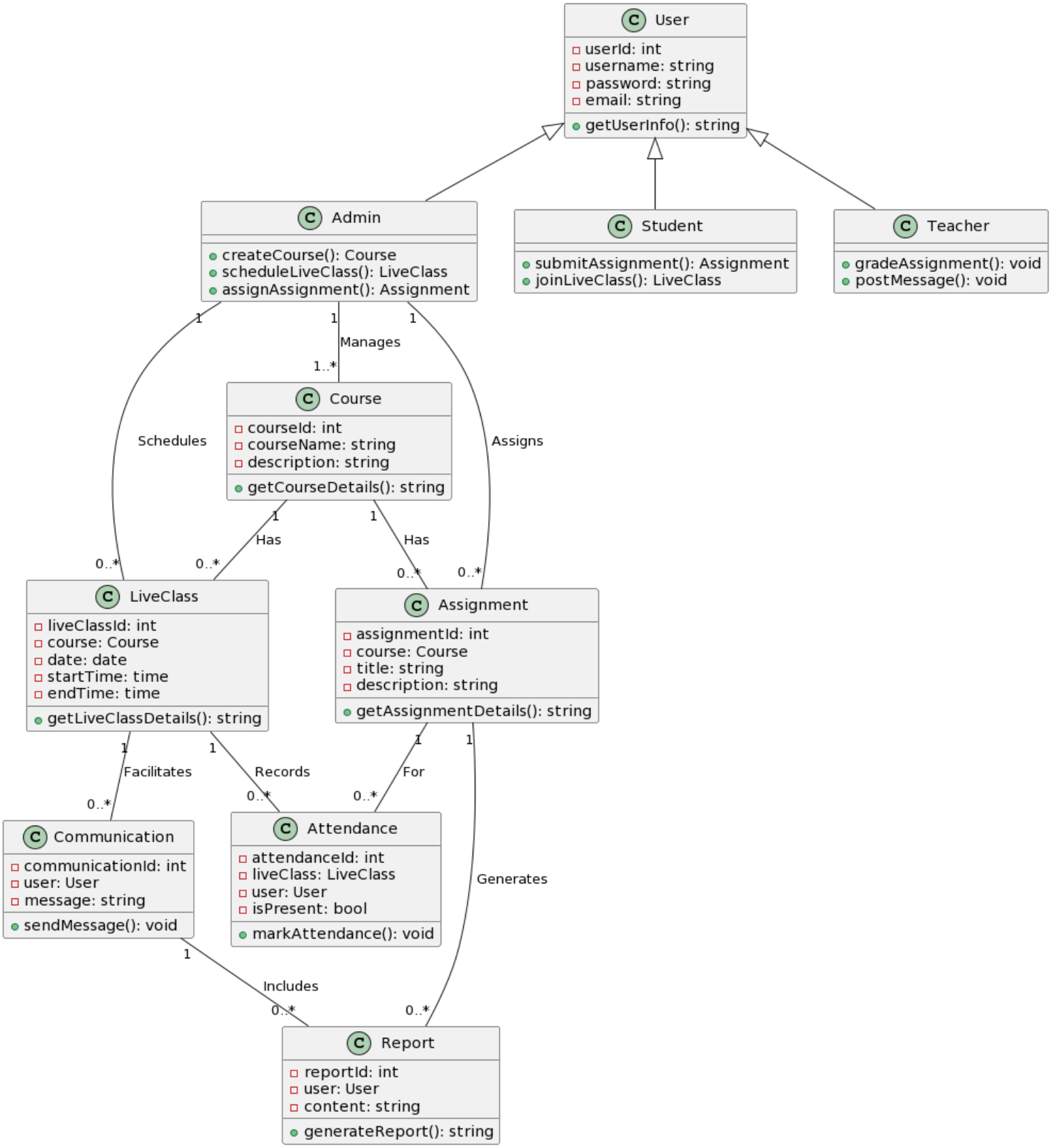
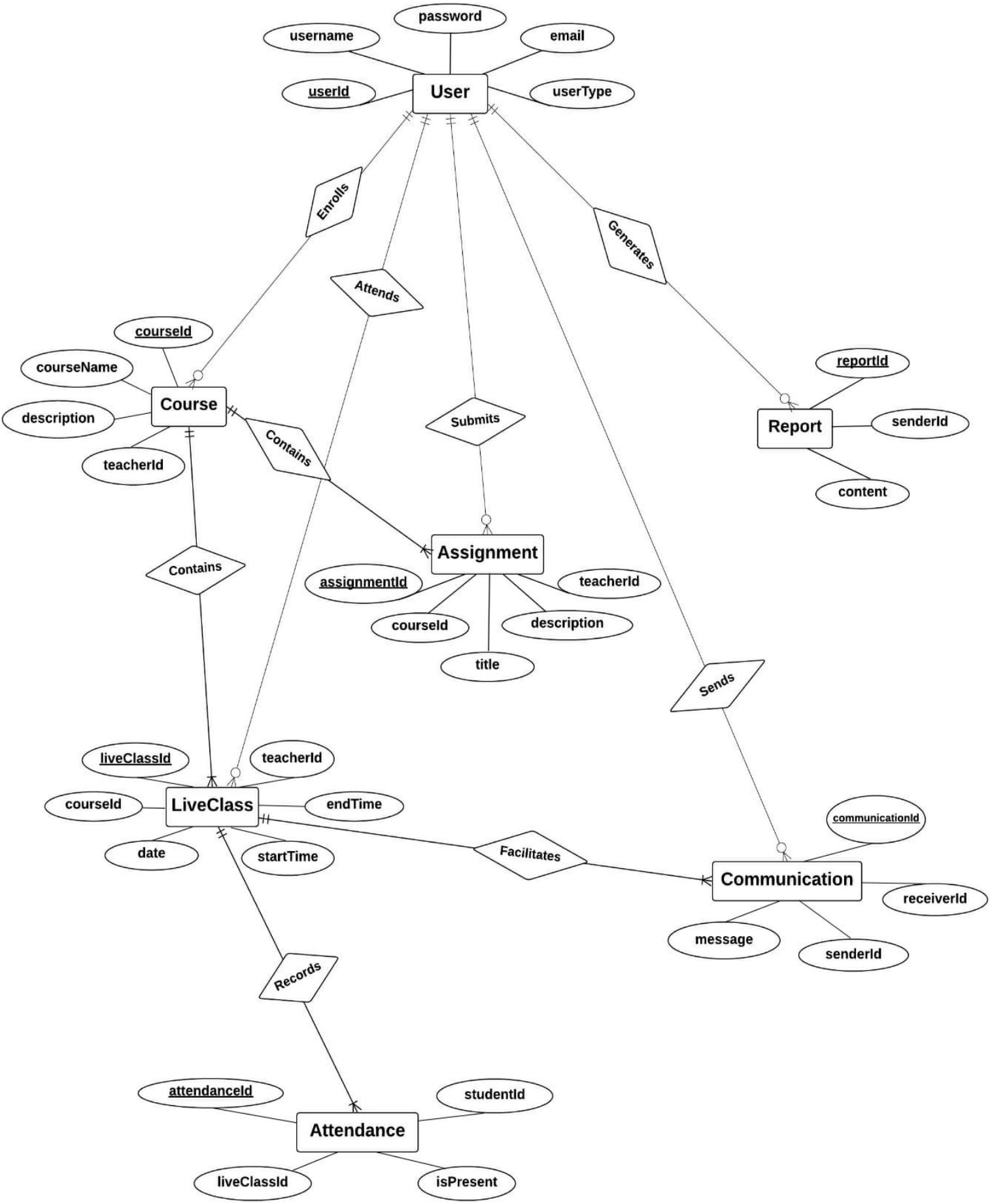


Fig: Class Diagram of the entire system

Entity Relationship Diagram (ERD):-



[8] A Complete Database & Tables

This is a large enough application, and for building such a type numbers of tables are required. The number of tables is gradually increasing in course of development, as well as with the application of Normalization. Here only a few of those are mentioned, mainly the masters one, just to clear an overview of the application.

Table Name: *Users* - Stores information about all users in the system.

Column Name	Data Type (Size)	Constraint
userId	int	Primary Key, Auto Increment
username	varchar(50)	Unique, Not Null
password	varchar(100)	Not Null
email	varchar(100)	Unique, Not Null
userType	varchar(20)	Not Null
firstName	varchar(50)	
lastName	varchar(50)	
dateOfBirth	date	
phoneNumber	varchar(15)	
address	varchar(200)	

Table Name: *Courses* - Stores information about courses offered.

Column Name	Data Type (Size)	Constraint
courseId	int	Primary Key, Auto Increment
courseName	varchar(100)	Unique, Not Null
description	text	
teacherId	int	Foreign Key (Users.userId)
startDate	date	
startDate	date	

Table Name: *LiveClasses* - Stores details of scheduled live classes.

Column Name	Data Type (Size)	Constraint
liveClassId	int	Primary Key, Auto Increment
courseId	int	Foreign Key (Courses.courseId)
date	date	Not Null

Column Name	Data Type (Size)	Constraint
startTime	time	Not Null
endTime	time	Not Null
teacherId	int	Foreign Key (Users.userId)

Table Name: *Assignments* - Stores details of assignments for courses.

Column Name	Data Type (Size)	Constraint
assignmentId	int	Primary Key, Auto Increment
courseId	int	Foreign Key (Courses.courseId)
title	varchar(200)	Not Null
description	text	
dueDate	date	
maxPoints	int	
teacherId	int	Foreign Key (Users.userId)

Table Name: *Attendance* - Records attendance for live classes.

Column Name	Data Type (Size)	Constraint
attendanceId	int	Primary Key, Auto Increment
liveClassId	int	Foreign Key (LiveClasses.liveClassId)
studentId	int	Foreign Key (Users.userId)
isPresent	boolean	Not Null

Table Name: *Communication* - Stores user communication messages.

Column Name	Data Type (Size)	Constraint
communicationId	int	Primary Key, Auto Increment
senderId	int	Foreign Key (Users.userId)
receiverId	int	Foreign Key (Users.userId)
message	text	

Table Name: *Reports* - Stores various system-generated reports.

Column Name	Data Type (Size)	Constraint
reportId	int	Primary Key, Auto Increment
senderId	int	Foreign Key (Users.userId)
content	text	

UserRole **- Stores the relationship between users and roles, mapping users to their associated roles in the system.**

Column Name	Data Type	Constraints
userRoleId	int	Primary Key
roleId	int	Foreign Key to Role.roleId
userId	int	Foreign Key to User.userId

CourseEnrollment **– Stores details about the course student is enrolled into.**

Column Name	Data Type	Constraints
enrollmentId	int	Primary Key
courseId	int	Foreign Key to Course,courseId
studentId	int	Foreign Key to User.userId
enrollmentDate	date	

Role**: Contains various roles available in the system, defining different access privileges and permissions for users.**

Column Name	Data Type	Constraints
roleId	int	Primary Key
roleName	varchar(50)	
roleDescription	varchar(250)	

Grade**: Records the grades or scores achieved by students for specific assignments in courses.**

Column Name	Data Type(size)	Constraint
gradeId	int	Primary Key, Auto Increment
assignmentId	int	Foreign Key(Assignment.assignmentId), Not Null
studentId	int	Foreign Key(User.userId), Not Null
score	decimal(5, 2)	Not Null

CourseMaterial: Stores the educational materials (e.g., documents, presentations) associated with each course.

Column Name	Data Type(size)	Constraint
materialId	int	Primary Key, Auto Increment
courseId	int	Foreign Key(Course.courseId), Not Null
materialTitle	varchar(200)	Not Null
materialContent	text	Not Null

CourseSchedule: Manages the schedule and location information for different classes or sessions within courses.

Column Name	Data Type(size)	Constraint
scheduleId	int	Primary Key, Auto Increment
courseId	int	Foreign Key(Course.courseId), Not Null
scheduleDate	date	Not Null
startTime	time	Not Null
endTime	time	Not Null
location	varchar(100)	Not Null

Homework: Keeps track of homework assignments associated with courses, including titles, descriptions, and deadlines.

Column Name	Data Type(size)	Constraint
homeworkId	int	Primary Key, Auto Increment
courseId	int	Foreign Key (Course, courseId), Not Null
title	varchar(200)	Not Null
description	text	Not Null
deadline	date	Not Null

[9] A Complete Structure

Number of Modules & Their Description:-

The application consists of number of modules and sub modules, of which, the most important are discussed briefly…

Here's a list of major to minor modules of the OneSpotEducation system along with their sub modules and descriptions. This list covers various modules and submodules that collectively form the OneSpotEducation system. Each module plays a crucial role in delivering an efficient and comprehensive online education platform.

- **User Management**
  - **User Registration:** Allows users to register for the platform by providing essential details.
  - **User Login:** Enables users to log in to their accounts using credentials.
  - **User Profile:** Provides users with the option to manage their profile information.
  - **Edit Profile** : Allows users to edit their user information like mobile number, password etc.

- **Course Management**
  - **Course Creation:** Allows administrators to create new courses in the system.
  - **Course Details:** Displays information about each course, such as the syllabus and instructor.

- **Live Classes Collaboration**
  - **Live Class Scheduling:** Allows teachers to schedule live classes with date, time, and topic.
  - **Live Class Attendance:** Tracks student attendance during live classes.
  - **Live Class Interaction:** Facilitates real-time communication between teachers and students.

- **Assignment Assessment**
  - **Assignment Creation:** Enables teachers to create assignments with titles and descriptions.
  - **Assignment Submission:** Allows students to submit their assignments for evaluation.
  - **Assignment Grading:** Enables teachers to grade and provide feedback on submitted assignments.

- **Analytics and Reporting**
  - **Student Progress Reports:** Generates reports on individual student progress.
  - **Course Analytics:** Provides insights into course performance and engagement.
  - **Attendance Reports:** Generates reports on attendance and participation.

- **System Administration and Configuration**
  - **User Roles and Permissions:** Allows administrators to manage user roles and access rights.
  - **Database Management:** Manages the system's database, backups, and maintenance.
  - **System Settings:** Enables administrators to configure system-wide settings.

- **Communication and Collaboration**
  - **Messaging System:** Facilitates private messaging between users.
  - **Discussion Forums:** Provides platforms for public discussions and Q&A sessions.
  - **Announcement Center:** Allows teachers and administrators to make announcements to students.

- **Mobile Application Support**
  - **Mobile App Development:** Develops and maintains mobile applications for the system.
  - **Mobile User Experience:** Ensures a user-friendly and responsive interface for mobile users.

- **User Support and Helpdesk**
  - **Helpdesk Management:** Handles user queries and support tickets.
  - **Knowledge Base:** Provides a repository of FAQs and self-help resources for users.
  - **Technical Support:** Offers technical assistance to users facing system-related issues.

- **Security and Access Control**
  - **User Authentication:** Ensures secure login mechanisms for user authentication.
  - **Data Encryption:** Implements encryption techniques to protect sensitive data.
  - **Access Restrictions:** Enforces access control to prevent unauthorized data access.

- **Integration with Learning Resources**
  - **External Content Integration:** Allows integration with external educational resources.
  - **Third-Party Tools Integration:** Integrates external tools for enhanced functionality.

- **Course Material Management**
  - **Course Materials Repository:** Stores and manages course-related resources.
  - **File Upload and Sharing:** Enables teachers to upload and share study materials.

Data Structures as per the project requirements:-

Based on the project requirements of the OneSpotEducation system, here are some of the essential data structures that may be used:

1. **User Data Structure:**
   - userId: int
   - username: string
   - password: string
   - email: string
   - userType: string (enum: Student, Teacher, Administrator)
   - additional attributes for user profile management
2. **Course Data Structure:**
   - courseId: int
   - courseName: string
   - description: string
   - teacherId: int (reference to the User who is the teacher of the course)
   - additional attributes for course details
3. **LiveClass Data Structure:**
   - liveClassId: int
   - courseId: int (reference to the Course to which the live class belongs)
   - date: date
   - startTime: time
   - endTime: time
   - teacherId: int (reference to the User who is conducting the live class)
   - additional attributes for live class details
4. **Assignment Data Structure:**
   - assignmentId: int
   - courseId: int (reference to the Course to which the assignment belongs)
   - title: string
   - description: string
   - teacherId: int (reference to the User who assigned the assignment)
   - additional attributes for assignment details
5. **Attendance Data Structure:**
   - attendanceId: int
   - liveClassId: int (reference to the LiveClass for which attendance is marked)
   - studentId: int (reference to the User who is marked present/absent)
   - isPresent: bool (true for present, false for absent)

6. Communication Data Structure:

- communicationId: int
- senderId: int (reference to the User who sent the message)
- receiverId: int (reference to the User who received the message)
- message: string
- additional attributes for message details

7. Report Data Structure:

- reportId: int
- senderId: int (reference to the User who generated the report)
- content: string
- additional attributes for report details

8. Enrollment Data Structure:

- enrollmentId: int
- courseId: int (reference to the Course in which the student enrolled)
- studentId: int (reference to the User who enrolled in the course)
- enrollmentDate: date
- additional attributes for enrollment details

9. Grade Data Structure:

- gradeId: int
- assignmentId: int (reference to the Assignment for which the grade is given)
- studentId: int (reference to the User for whom the grade is given)
- score: float
- additional attributes for grade details

10. CourseMaterial Data Structure:

- materialId: int
- courseId: int (reference to the Course to which the material belongs)
- materialTitle: string
- materialContent: string (text of the material)

11. CourseSchedule Data Structure:

- scheduleId: int
- courseId: int (reference to the Course for which the schedule is created)
- scheduleDate: date
- startTime: time
- endTime: time
- location: string
- additional attributes for schedule details

12. Homework Data Structure:

- homeworkId: int
- courseId: int (reference to the Course for which the homework is assigned)
- title: string
- description: string
- deadline: date
- additional attributes for homework details

These data structures represent the main entities and relationships required for the OneSpotEducation system. Depending on the system's complexity and specific requirements, additional data structures and attributes may be incorporated to support all functionalities and features effectively.

Implementation Methodology **:-**

Agile Development is an iterative and incremental approach to software development that emphasizes flexibility and collaboration. It breaks the development process into small, manageable tasks that are continuously tested and delivered in short cycles called iterations or sprints. Each iteration adds new features or improvements based on feedback from users and stakeholders.

Implementation Methodology (MVC – Model-View-Controller):

MVC is suitable for OneSpotEducation as it allows clear separation of user interface, business logic, and data management. This promotes code organization, scalability, and maintainability, making it easier to develop and enhance the application over time.

- **Model:** Represents the data and business logic of the application. It manages the data and provides methods to access and modify it.
- **View:** Deals with the user interface and how data is presented to users. It displays information from the Model and receives user inputs.
- **Controller:** Acts as an intermediary between the Model and View. It processes user inputs, updates the Model accordingly, and controls the flow of data to the View.

Advantages of MVC:

- **Separation of Concerns:** It divides the application into distinct components, making it easier to manage and maintain.
- **Code Reusability:** Each component can be reused across different parts of the application.
- **Collaboration:** Multiple developers can work on different components simultaneously, promoting teamwork.

***List of Reports that are Likely to be Generated*:-**

- **Student Progress Report:** Provides an overview of individual student's academic performance, including grades, attendance, and participation.
- **Attendance Report:** Summarizes attendance data for live classes and recorded sessions, highlighting students' attendance patterns.
- **Assignment Submission Report:** Lists the status of assignments, indicating which students have submitted and which are pending.
- **Teacher Performance Report:** Evaluates the performance of teachers based on student feedback, assignment grading, and live class ratings.
- **Course Material Usage Report:** Tracks the usage of course materials, including downloads and views by students.
- **Communication Activity Report:** Provides statistics on communication activities, such as the number of messages sent and received by users.
- **System Health Report:** Monitors the system's performance, identifying potential bottlenecks or technical issues.
- **Helpdesk Support Report:** Tracks user queries, response times, and resolution rates of the helpdesk support team.
- **Financial Report:** Summarizes revenue generated from course enrollments, subscriptions, and other financial transactions.

These reports offer valuable insights to administrators, teachers, and students, allowing them to make data-driven decisions, evaluate performance, and enhance the overall educational experience within the OneSpotEducation system.

[10] Overall Network Architecture

Technical Architecture:-

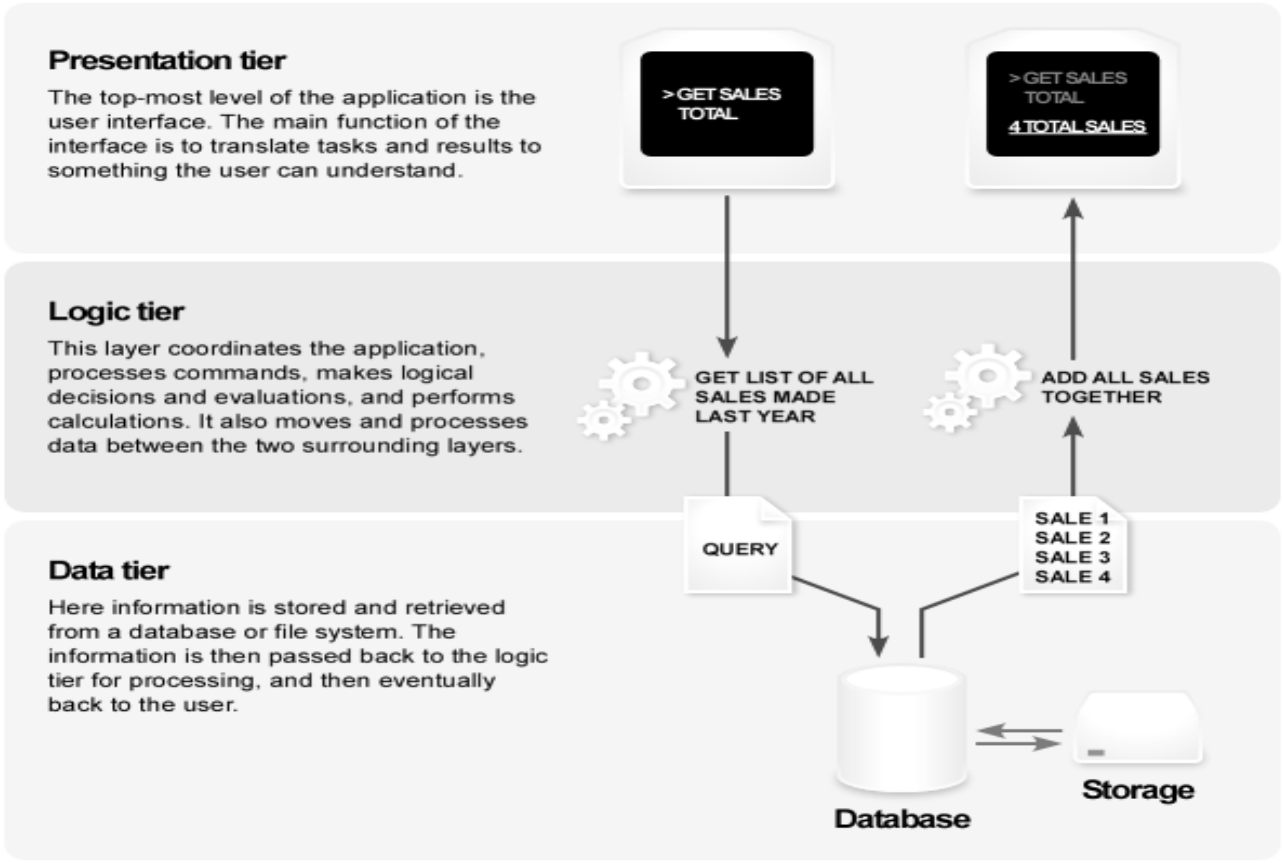


Fig: Three-Tier Architecture

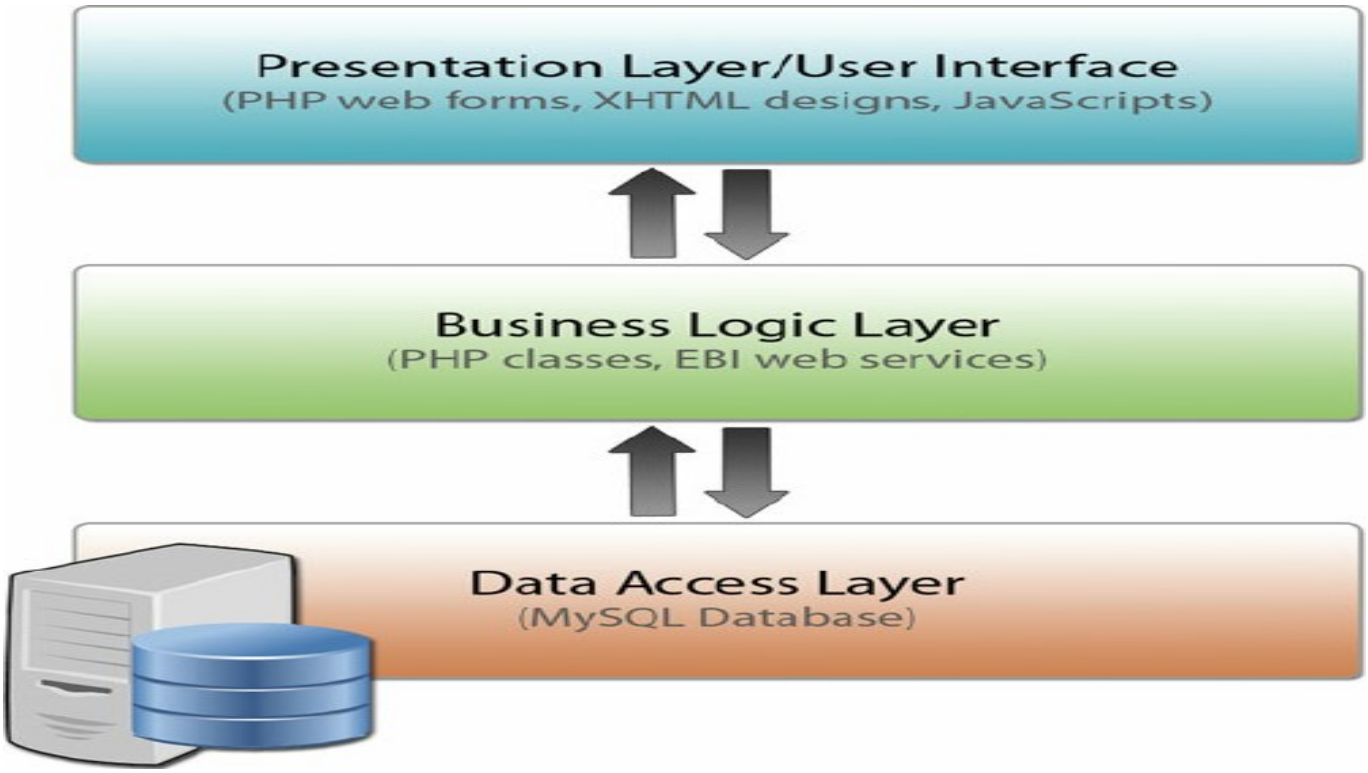


Figure: 3-tier Architecture (Illustrative View)

Three tier:-

Three-Tier Architecture is a popular architectural pattern that separates the application into three logical layers: presentation, business logic, and data storage. Each layer is independent and communicates with the other layers through well-defined interfaces. This separation of concerns makes the system more manageable and allows for easier maintenance and scalability.

Three-tier is a client-server architecture in which the user interface, functional process logic, computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. The 3-Tier architecture has the following three tiers:

- **Presentation Tier-**

  This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network.

- **Application Tier (Business Logic/Logic Tier)-**

  The logic tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.

- **Data Tier-**

  This tier consists of Database Servers. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

Advantages of Three-Tier Architecture:

1. **Modularity:** Allows independent development and maintenance of each layer.
2. **Scalability:** Facilitates scaling individual layers as needed.
3. **Security:** Helps enforce access control and protect sensitive data.
4. **Reusability**: Components in each layer can be reused in different parts of the application.

Using Three-Tier Architecture can be beneficial for complex systems like OneSpotEducation as it provides a structured approach to development. However, the specific choice of architecture depends on the project's requirements, the team's expertise, and other considerations like time and budget constraints.

[11] Implementation of Security Mechanisms at Various Levels

Implementation of security Machanisms

- **User Authentication and Authorization:**
  - User Authentication: Implement secure login mechanisms (e.g., username/password, multi-factor authentication) to ensure that only authorized users can access the system.
  - User Authorization: Enforce role-based access control (RBAC) to grant appropriate privileges to users based on their roles (e.g., student, teacher, administrator).
- **Data Encryption:** Use encryption algorithms (e.g., AES, RSA) to protect sensitive data stored in databases, ensuring data confidentiality.
- **Secure Communication:** Implement secure communication protocols (e.g., HTTPS) to encrypt data transmitted between the client and server, preventing eavesdropping and man-in-the-middle attacks.
- **Input Validation and Sanitization:** Validate and sanitize all user inputs to prevent common vulnerabilities like SQL injection and cross-site scripting (XSS) attacks.
- **Password Security:** Enforce strong password policies (e.g., minimum length, complexity) and store passwords securely using hashing algorithms (e.g., bcrypt).
- **Session Management:** Implement secure session management to handle user sessions, including session timeouts and session token management.
- **Secure File Uploads:** Validate and restrict file uploads to prevent uploading of malicious files or unauthorized access.
- **Preventing Brute Force Attacks:** Implement mechanisms to detect and prevent brute force attacks on login pages or other vulnerable endpoints.
- **Cross-Site Request Forgery (CSRF) Protection:** Implement CSRF tokens to protect against CSRF attacks that exploit authenticated users to perform unintended actions.
- **Regular Security Assessments:** Conduct regular security assessments and penetration testing to identify vulnerabilities and address potential security risks.

By implementing security mechanisms at various levels, the OneSpotEducation system can safeguard sensitive data, protect against potential threats, and ensure a secure and reliable environment for users and administrators. It's essential to continuously monitor and update the security measures to stay vigilant against emerging threats and ensure a robust security posture.

[12] Future Scope & Further Enhancement

The future scope and further enhancements of the OneSpotEducation system are vast, providing opportunities for continuous growth and improvement. Some potential future scope and enhancements include:

1. **Artificial Intelligence Integration:**
   - Implement AI-based adaptive learning algorithms to personalize courses for each student based on their learning preferences and progress.
   - Utilize AI-powered chatbots for instant user support and answering common queries.
2. **Virtual Reality and Augmented Reality (VR/AR) Expansion:**
   - Expand VR/AR learning experiences across various subjects to provide immersive and interactive education.
   - Develop virtual laboratories for practical subjects like science, engineering, and medicine.
3. **Blockchain-Based Credentialing:**
   - Integrate blockchain technology to issue secure and verifiable certificates, diplomas, and badges upon course completion.
   - Establish a decentralized credentialing system to enhance the credibility of achievements.
4. **Enhanced Data Analytics and Reporting:**
   - Develop advanced analytics dashboards with real-time insights into user engagement, course performance, and learning trends.
   - Introduce predictive analytics to identify at-risk students and provide early interventions.
5. **Mobile App Offline Mode and Mobile-First Approach:**
   - Improve the mobile application to support offline access to course materials, allowing students to study without an internet connection.
   - Adopt a mobile-first approach to optimize the user experience on smartphones and tablets.
6. **Continuous Learning and Professional Development:**
   - Offer continuous learning resources and courses for professionals to upgrade their skills and stay competitive in their industries.
   - Partner with industry experts and organizations to provide certifications and skill development programs.
7. **Multilingual Support and Internationalization:**
   - Translate course content and the user interface into multiple languages to cater to a global audience.
   - Customize the platform to comply with the educational standards and regulations of different countries.
8. **Social Learning Enhancements:**
   - Foster more extensive social learning communities with discussion forums, study groups, and collaborative projects.
   - Encourage peer-to-peer learning and knowledge sharing among students and educators.
9. **Enhanced Security and Privacy Measures:**
   - Implement advanced security protocols to protect user data, prevent cyber threats, and ensure compliance with data privacy regulations.

- Conduct regular security audits and penetration testing to identify and address potential vulnerabilities.

10. Integration with External Learning Platforms and Resources:

- Collaborate with reputable educational content providers, open educational resources (OER), and Massive Open Online Course (MOOC) platforms to expand course offerings.
- Facilitate seamless transfer of credits and certifications from other recognized educational institutions.

11. Continuous User Feedback and Improvement:

- Conduct user surveys and feedback loops to gather insights for continuous improvement of the platform's features and usability.
- Regularly update and optimize the system based on user preferences and needs.

Conclusion:-

In conclusion, the OneSpotEducation system presents a comprehensive and cutting-edge solution to revolutionize the traditional education landscape. By leveraging the power of technology and advanced features, the platform aims to digitize and modernize the learning experience for students, educators, and institutions alike. Through its user-friendly interface, interactive live classes, and personalized learning paths, OneSpotEducation endeavors to foster a dynamic and engaging educational environment. The emphasis on security mechanisms ensures the safety and privacy of user data, building trust and confidence in the platform. As we continually strive for improvement, the future scope of the system includes integrating emerging technologies like AI, VR/AR, and blockchain, further enhancing the learning process and making education accessible to a global audience. With a commitment to excellence and a vision for continuous development, OneSpotEducation aspires to shape the future of education, empowering learners to explore their full potential and embrace a lifetime of knowledge acquisition and growth. As the educational landscape continues to evolve, OneSpotEducation remains dedicated to pushing the boundaries of online education, creating a collaborative and innovative environment that inspires lifelong learning and empowers individuals to thrive in an ever-changing world.

[13] Bibliography

BOOKS

- **Data Flow Diagram –** Ref - Object Oriented Analysis and Design (MCS-219),Block 2:Modeling,UNIT 8: Functional Modeling
- **ER Diagram –** Ref - **Modeling the System Architecture** Unit 2, Block 1 of MCS 213 (Software Engineering), Study Material by IGNOU
- Ref - **Software Engineering** MCS 213 , Study Material by IGNOU
- Ref - **Class Diagram-** Unit 2: Structural Modeling using UML, Block 1 of MCS 219 (Object Oriented Analysis and Design), Study Material by IGNOU

Articles and Documents

- Ref Cloudways– Tips to Write Web Development Proposal Template in 2022by Najam Ahmed
- Ref - MCSP – 232 project guidelines Programme Guide For Master Of Computer Applications (Online) (Programme Code: Mcaol)
- Ref – Laravel Documentation : https://laravel.com/docs/10.x

Websites

- Ref – W3school - https://www.w3schools.com/
- Ref – Tutorial Point- https://www.tutorialspoint.com/
- Ref - Real Time Chat With Laravel Broadcast, Pusher and Vuejs : https://www.udemy.com/course/laravel-real-time-chat-with/
- Ref: FreeCodeCamp - https://www.freecodecamp.org/learn/
- Ref- Teach Yourself in SQL Server Database Development-Techmedia.Ref - RDBMS -Bipin C. Desai.

The End

Index of comments

3.1 1) All the figures, tables, should be numbered properly and it must be referred in the text.
2) ERD, DFDs (All levels) and Tables must be explained w.r.t your project and synchronize entities and Tables. Also draw and explained related UML diagrams.
3) Follow the SDLC model.
4) Must follow the all the guidelines of MCSP-232.

Prof Sandeep Singh Rawat
SOCIS, INGNOU, New Delhi.
Date: 26/09/2023

X. CERTIFICATE OF ORIGINALITY

This is to certify that the project report entitled ***One Spot Education*** submitted to **Indira Gandhi National Open University** in partial fulfilment of the requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS,** is an authentic and original work carried out by Mr. / Ms. ***Anjali Pandey*** with enrolment no. ***2200811729*** under my guidance.

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this University or to any other University / Institute for the fulfilment of the requirements of any course of study.

Anjali

Signature of the Student:

Date: 06/06/2024

Name and Addressthe student: Anjali Pandey

B-788,
New Ashok Nagar
Delhi-96

Enrolment No...***2200811729***.....

Satyam Srivastava

Signature of the Guide

Date: .....

Name, Designation of and Address of the

Guide:

Project Report

Title of the project

ONE SPOT EDUCATION

(A one stop revolutionary digital platform for educational institutions to transform traditionaleducation with feature-rich innovative technologies **)** 4

Table of Contents

[1] Introduction & Objective(s)

Introduction

In today's rapidly evolving world, the traditional education system is facing numerous challenges that hinder its ability to provide students with a dynamic and engaging learning experience. The need of the hour is a transformation that seamlessly integrates education with the power of technology. This project proposal introduces a comprehensive web portal aimed at revolutionizing education, empowering educational agencies, schools, universities, and other similar institutions to embark on a digital transformation journey. Our web portal leverages the latest advancements in technology to provide an all-in-one platform that caters to the diverse needs of students, educators, and administrators. By combining live and recorded classes, interactive features, class chat functionality, and user-friendly interfaces, our web portal aims to create an immersive and interactive learning environment that enhances student engagement, collaboration, and overall learning outcomes. The key features of our web portal include live and recorded classes, enabling students to attend real-time sessions or access course materials at their convenience. This flexibility allows learners to access high-quality education irrespective of their physical location or time constraints. Additionally, the class chat functionality, inspired by popular messaging applications, fosters real-time interaction between students and educators, promoting engagement, peer collaboration, and immediate feedback.

Automation plays a crucial role in our web portal, simplifying administrative tasks and enhancing efficiency. The platform offers an auto-attendance feature that eliminates manual tracking, reducing paperwork and streamlining the process. Furthermore, our comprehensive syllabus, assignment, and practicals management system allows educators to organize, distribute, and assess coursework effortlessly. This centralization of educational resources promotes transparency, making it easier for students and parents to stay updated on academic progress. We understand that the visual appeal and user experience significantly impact learner engagement. Therefore, the user interface of our web portal is carefully crafted to be intuitive, attractive, and easy to navigate. The impact of our web portal extends beyond the virtual classroom. With features like homework submission and personalized assessments, students are empowered to take ownership of their learning. Educators can provide tailored feedback and track individual progress, ensuring targeted support for each student's academic growth. Additionally, our platform offers a comprehensive suite of tools for collaborative projects, fostering critical thinking, creativity, and effective communication skills.

By adopting our web portal, educational agencies, schools, universities, and other institutions can revolutionize their education systems, embracing the digital age without compromising on quality or integrity. Our platform serves as a catalyst for change, empowering educators to unleash their full potential and enabling students to thrive in a technology-driven world. In conclusion, our comprehensive web portal aims to transform the traditional education system into a dynamic, interactive, and learner-centric experience. By seamlessly integrating technology into classrooms, we aim to enhance engagement, collaboration, and educational outcomes.

Objectives

- **Digitize Education:** Transform the traditional education system into a digital environment, enabling educational agencies, schools, universities, and other institutions to embrace technology and enhance the learning experience.
- **Provide Live and Recorded Classes:** Offer both live and re System Analysis scored classes to cater to different learning preferences and accommodate various schedules, ensuring flexibility and accessibility for students.
- **Foster Collaboration and Engagement:** Facilitate collaboration and interaction between students and educators through class chat functionality, encouraging active participation, peer-to-peer learning, and immediate feedback.
- **Streamline Administrative Tasks:** Automate administrative tasks such as attendance tracking, homework submission, and assessment management to reduce paperwork, optimize efficiency, and free up time for educators and administrators.
- **Centralize Syllabus and Course Management:** Provide a centralized platform for syllabus organization, assignment distribution, and practical management, simplifying educational resource management and promoting transparency.
- **Enhance User Experience:** Design an intuitive, attractive, and user-friendly interface to create an engaging and enjoyable learning environment for students, educators, and administrators.
- **Personalized Learning Experience:** Offer features that allow educators to provide personalized assessments, tailored feedback, and individual progress tracking, empowering students to take ownership of their learning journey.
- **Promote Effective Communication:** Facilitate seamless communication between students, educators, and parents through integrated messaging and notification systems, fostering strong relationships and effective collaboration.
- **Optimize Learning Outcomes:** Enable educators to design interactive and immersive learning experiences that promote critical thinking, creativity, and problem-solving skills, ultimately improving overall learning outcomes.
- **Data-Driven Insights:** Generate comprehensive analytics and reports on student performance, engagement levels, and learning outcomes, empowering educators and administrators to make data-driven decisions for continuous improvement.
- **Parental Involvement:** Facilitate parental involvement and engagement in their child's education through parent portals, progress tracking, and communication channels, fostering a collaborative learning ecosystem.
- **Security and Privacy:** Implement robust security measures to protect user data, ensure privacy compliance, and maintain a secure learning environment for all stakeholders.

[2] System Analysis

[A] Identification of Needs:-

The One Spot Education project is designed to address several critical needs within the educational landscape. In an era where technology is rapidly transforming every aspect of our lives, traditional educational systems often struggle to keep pace with the demands for flexibility, accessibility, and enhanced communication. The need for a comprehensive digital platform like One Spot Education arises from these evolving requirements. Modern students and educators require a platform that supports both live and recorded classes, allowing for flexible learning schedules that accommodate diverse lifestyles and commitments. Traditional classroom settings do not always offer this level of flexibility, which can hinder the learning process for many students, especially those who balance their studies with work or other responsibilities. One Spot Education provides a solution by offering a centralized hub where students can access educational content at any time, ensuring that learning can continue beyond the physical classroom.

Furthermore, effective communication and collaboration are vital components of successful education. Traditional methods such as emails and in-person meetings often lack the immediacy and interactivity required for effective learning. One Spot Education addresses this by incorporating robust communication tools, including class chats and real-time collaboration features. These tools facilitate better interaction between students and teachers, promoting a more engaging and interactive learning environment. Efficient management of educational resources and activities is another critical need. Tasks such as managing syllabus, assignments, practicals, and attendance can be cumbersome and error-prone when handled manually. One Spot Education automates these processes, ensuring accuracy and saving valuable time for both students and educators. Features like auto-attendance tracking and streamlined assignment management reduce administrative burdens and enhance the overall efficiency of educational institutions.

Additionally, in today's data-driven world, educational institutions need sophisticated analytics and reporting tools to monitor and evaluate student performance and engagement. One Spot Education provides detailed insights through its reporting and analytics module, enabling educators to make informed decisions that enhance teaching and learning outcomes.

Finally, with the increasing concerns around data security and privacy, One Spot Education incorporates advanced security measures to protect sensitive information. By implementing robust security protocols, the platform ensures a safe and secure learning environment, addressing the growing need for data protection in digital education. In essence, the One Spot Education project is designed to meet the modern demands of flexibility, interactivity, efficiency, data-driven insights, and security in education, thereby revolutionizing the learning experience for both students and educators.

[2] Project Planning & Scheduling:-

[A] PERT Chart/ Task Network Chart:-

The Program Evaluation Review Technique (PERT) is the cost and time management system. PERT organizes that project is complex that some task must be completed before other can be stated and that the appropriate way to manage a project is to define and control each task. Because projects often fall behind schedules, PERT is designed to facilitate getting a project back on schedule. The PERT chart gives a graphical representation of this information.

Depending on the working priorities, the entire project can be subdivided into the following main modules, those are:-

- Users Module
- Course Module
- Class Module
- Assignment Assessment Module
- Activity Calendar Module
- System Admin and Config Module.

We can construct our activities plain as follows:-

Activity	Activity Name
A	User Management
B	Course Management
C	Live Classes
D	Assignment Assessment
E	Activity Calendar
F	System Admin and Config

The Program Evaluation Review Technique (PERT) is a project management tool used to plan, schedule, and control tasks within a project. It is particularly useful for projects that are complex and involve multiple interdependent tasks. PERT helps in organizing the project by identifying the sequence of tasks that must be completed before others can begin, thus ensuring a systematic approach to project execution.

One of the key features of PERT is its focus on cost and time management. By breaking down the project into individual tasks and estimating the time and resources required for each task, PERT helps project managers allocate resources efficiently and track progress effectively. This approach is crucial in managing complex projects where tasks may have dependencies and delays can impact the overall project timeline.

The PERT chart, also known as the Task Network Chart, provides a visual representation of the project's tasks, their dependencies, and the estimated time required for each task. It helps project teams understand the critical path, which is the sequence of tasks that determines the shortest possible duration for completing the project.

Now, let's delve into how the One Spot Education project can be subdivided into main modules using the PERT approach:

[A] Users Module: This module focuses on user management, including registration, authentication, and profile management functionalities. It ensures that users can access the platform securely and efficiently.

[B] Course Module: The course module deals with managing courses offered on the platform. It includes functionalities such as course creation, editing, enrollment management, and course-related notifications.

[C] Class Module: This module handles live and recorded classes. It includes functionalities for scheduling classes, conducting live sessions, recording sessions for later access, and managing class materials.

[D] Assignment Assessment Module: The assignment assessment module facilitates the creation, submission, grading, and feedback process for assignments. It ensures that students and teachers can manage assignments effectively.

[E] Activity Calendar Module: The activity calendar module provides a calendar view of all upcoming events, classes, assignments, and deadlines. It helps users stay organized and plan their activities efficiently.

[F] System Admin and Config Module: This module is responsible for system administration, configuration settings, and user roles management. It ensures that the platform operates smoothly, and administrators have the necessary tools to manage the system effectively.

Chart:-

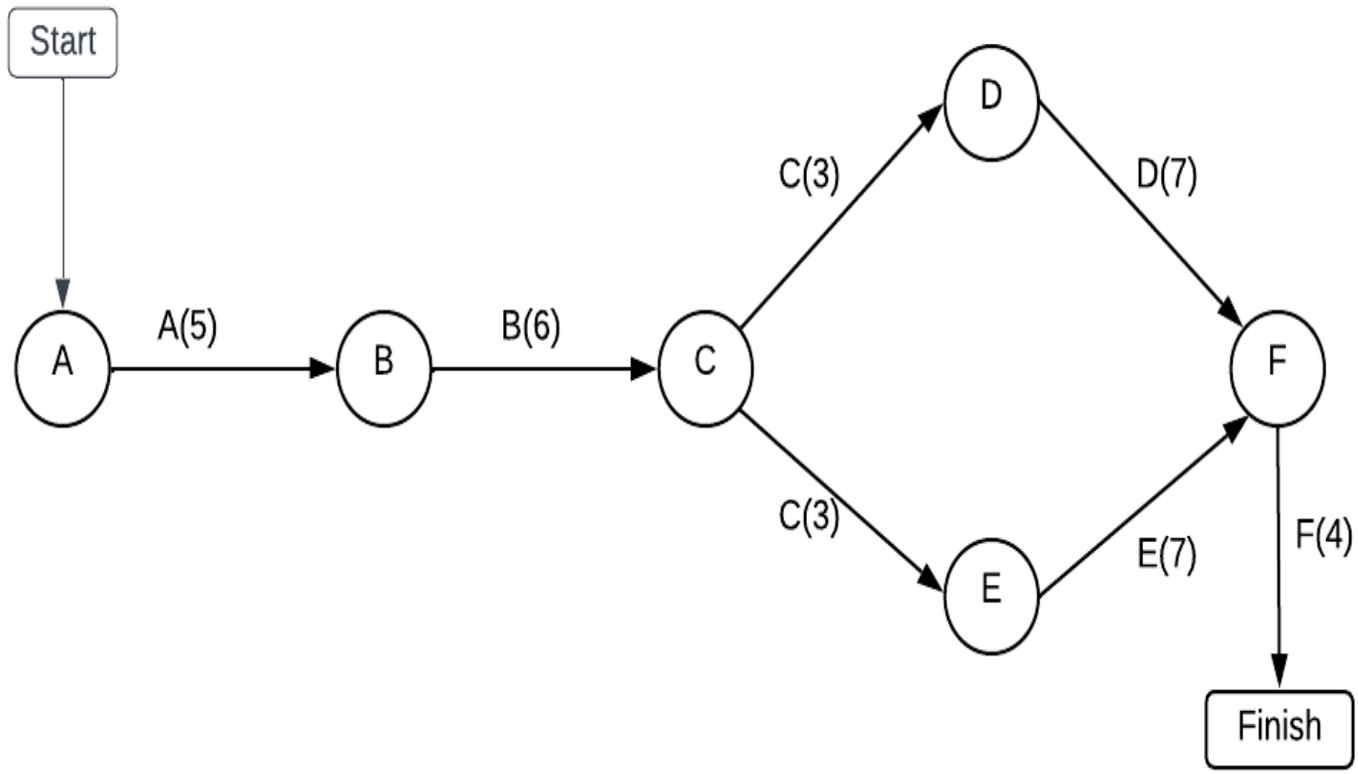
Activity	Predecessor Activity	Time Estimated Weeks (Individual)
A	……	5
B	A	6
C	B	3
D	C	7
E	C	7
F	D,E	4

Critical Path Method (CPM) diagram illustrates the flow of activities for the One Spot Education system. Each activity represents a key module of the project. Here's a brief explanation of the given CPM diagram tailored to our project:

- **Start Node:** The project initiates at the "Start" node and progresses to the first activity.
- **Activity A:** User Management (Duration: 5 weeks)
  - This initial task involves setting up and managing user accounts, authentication, and user profiles. Once completed, the project moves to the next activity.
- **Activity B:** Course Management (Duration: 6 weeks)
  - This task involves creating, editing, and managing courses offered on the platform. Completion of this activity allows the project to progress to activity C.
- **Activity C:** Live Classes (Duration: 3 weeks)
  - This module focuses on scheduling and managing live and recorded classes. After this, the project can move forward into two parallel paths:
    - Path 1: Assignment Assessment (Activity D)
    - Path 2: Activity Calendar (Activity E)
- **Activity D**: Assignment Assessment (Duration: 7 weeks)
  - This task involves managing assignments, including creation, submission, grading, and feedback. It is critical for assessing student performance. This leads to the next activity, System Admin and Config.
- **Activity E:** Activity Calendar (Duration: 7 weeks)
  - This module involves managing the calendar view of all events, classes, assignments, and deadlines to help users plan their activities. This also leads to the final activity.

- **Activity F:** System Admin and Config (Duration: 4 weeks)
  - The final task involves setting up and managing system configurations and administrative settings to ensure smooth operation of the platform. Completion of this activity marks the end of the project.

PERT Chart



Critical Path:

- The critical path is the sequence of tasks that determines the shortest time to complete the project. For this project, both identified paths are critical due to their equal durations.

Path Durations:

- Path 1: Start → A (User Management) → B (Course Management) → C (Live Classes) → D (Assignment Assessment) → F (System Admin and Config) → Finish
  - Duration: 5 (A) + 6 (B) + 3 (C) + 7 (D) + 4 (F) = 25 weeks
- Path 2: Start → A (User Management) → B (Course Management) → C (Live Classes) → E (Activity Calendar) → F (System Admin and Config) → Finish
  - Duration: 5 (A) + 6 (B) + 3 (C) + 7 (E) + 4 (F) = 25 weeks

Time Line Path: -

Path	Length Of Time
Start-A-B-C-D-F-Finish	5+6+3+7+4 = 25 WEEKS
Start-A-B-C-E-F-Finish	5+6+3+7+4 = 25 WEEKS

Table 2.2.2: Time Line Path of One Spot Education system

Since both paths have the same duration, both are critical paths, meaning any delay in any activity along these paths will delay the project.

Key Points:

- Critical Path: Both paths outlined above are critical.
- Dependencies: Each module depends on the completion of the preceding modules.
- Parallel Activities: Activities D (Assignment Assessment) and E (Activity Calendar) can be worked on concurrently after the completion of activity C (Live Classes).

The CPM helps ensure that the project is managed effectively, with close monitoring of critical tasks to keep the project on schedule.

[B] GANTT Chart:-

A Gantt chart is a popular type of bar chart that illustrates a project schedule. Gantt charts illustrate the start and finish dates of the terminal elements and summary elements of a project. Terminal elements and summary elements comprise the work breakdown structure of the project. The Gantt chart provides a comprehensive timeline for the One Spot Education project, clearly outlining the sequence and duration of each task. It highlights the dependencies between tasks and ensures that critical functionalities are developed in a logical order. This structured approach helps in managing the project efficiently and ensures timely completion of each module.

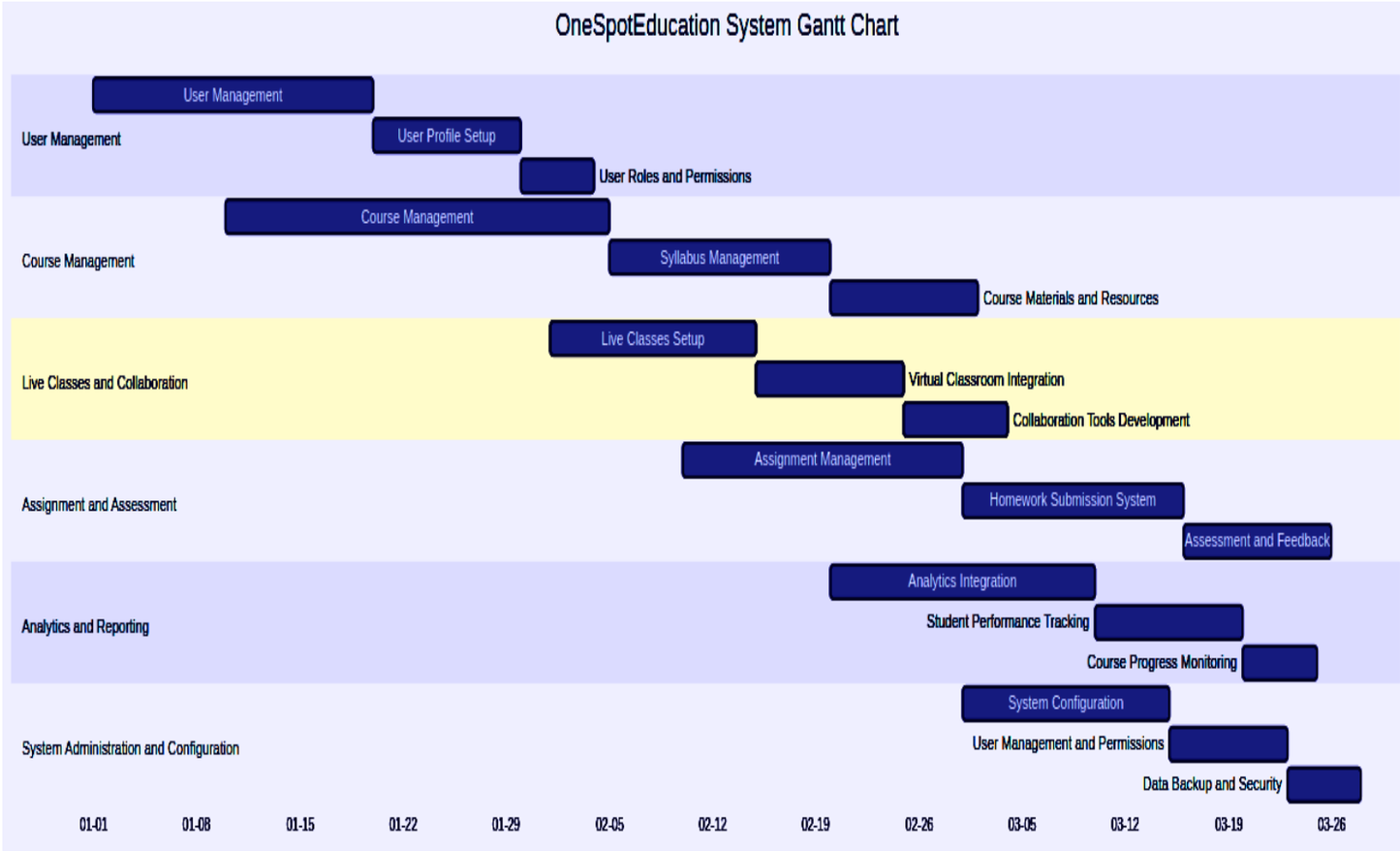


Fig 2.2.3: Gaint Chart

The provided Gantt chart visually represents the project timeline for the development of the One Spot Education system. It delineates various project phases and tasks, highlighting their start and end dates, durations, and dependencies.

Here’s a brief description of each section and its corresponding tasks:

1. User Management

- **User Management** (01-01 to 01-14): Initial setup and configuration of user management functionalities.
- **User Profile Setup** (01-05 to 01-19): Configuration of user profile settings, including data fields and customization options.
- **User Roles and Permissions** (01-12 to 01-26): Setting up different user roles and defining permissions for each role.

2. Course Management

- **Course Management** (01-15 to 01-28): Establishing the course management module, including creation, editing, and deletion of courses.
- **Syllabus Management** (01-22 to 02-05): Developing tools for managing course syllabi, including uploading and organizing course content.
- **Course Materials and Resources** (01-28 to 02-11): Integrating course materials and resources for easy access by users.

3. Live Classes and Collaboration

- **Live Classes Setup** (02-01 to 02-10): Setting up infrastructure and functionalities for live classes.
- **Virtual Classroom Integration** (02-05 to 02-19): Integrating virtual classroom tools like Zoom or Google Meet.
- **Collaboration Tools Development** (02-10 to 02-24): Developing tools to facilitate collaboration among students and teachers, such as discussion forums and chat.

4. Assignment and Assessment

- **Assignment Management** (02-12 to 02-25): Establishing a system for creating, assigning, and managing assignments.
- **Homework Submission System** (02-20 to 03-05): Developing a system for students to submit homework online.
- **Assessment and Feedback** (02-25 to 03-12): Implementing assessment tools and feedback mechanisms for assignments and tests.

5. Analytics and Reporting

- **Analytics Integration** (02-26 to 03-12): Integrating analytics tools to monitor user activities and system performance.
- **Student Performance Tracking** (03-01 to 03-19): Developing tools to track and analyze student performance.
- **Course Progress Monitoring** (03-05 to 03-20): Implementing tools to monitor the progress of courses and student engagement.

6. System Administration and Configuration

- **System Configuration** (03-10 to 03-20): Configuring system settings and parameters for optimal performance.
- **User Management and Permissions** (03-12 to 03-25): Fine-tuning user management functionalities and permissions.
- **Data Backup and Security** (03-15 to 03-30): Implementing data backup procedures and security measures to protect user data and ensure system integrity.

[3] Software requirement specifications (SRS)

A requirement specification outlines the detailed functional and non-functional requirements of a system, defining what the system should do and how it should perform. For the One Spot Education system, the requirement specification may include:

1. User Registration and Authentication:

- Users should be able to register an account with the system and provide necessary information.
- The system should authenticate users securely to ensure authorized access to the platform.

2. Dashboard:

- The system should provide a personalized dashboard for each user, displaying relevant information, notifications, and quick access to essential features.

3. Course Management:

- Educators should be able to create and manage courses, including adding course details, syllabus, and resources.
- Students should be able to enroll in courses, view course materials, and access resources.

4. Live and Recorded Classes:

- The system should support live online classes with video streaming and interactive features, allowing real-time communication between educators and students.
- Recorded classes should be available for on-demand viewing to accommodate flexible learning.

5. Assignment and Homework:

- Educators should be able to create assignments, set deadlines, and provide instructions.
- Students should be able to submit their assignments digitally, and educators should have a mechanism to review and grade submissions.

6. Attendance Management:

- The system should provide automated attendance tracking, allowing educators to monitor student attendance and generate reports.

7. Communication and Collaboration:

- The system should facilitate communication and collaboration between users through features such as chat functionality, discussion forums, and messaging tools.

8. Progress Tracking and Analytics:

- The system should track and analyze student progress, providing insights into performance, engagement levels, and learning outcomes.
- Educators and administrators should have access to comprehensive analytics and reports.

9. Resource Management:

- The system should provide a centralized repository for educational resources, including textbooks, multimedia materials, and supplementary resources.

10. User Notifications:

- The system should send notifications to users regarding important updates, upcoming events, assignment deadlines, and other relevant information.

11. Security and Privacy:

- The system should ensure secure data storage, transmission, and user authentication.
- It should comply with privacy regulations and protect user data.

12. Scalability and Performance:

- The system should be designed to handle increasing user load and ensure optimal performance even during peak usage.

13. Mobile Accessibility:

- The system should be accessible on mobile devices, allowing users to access course materials, participate in classes, and complete assignments on their smartphones or tablets.

14. Integration:

- The system should support integration with external tools or systems, such as learning management systems, video conferencing platforms, and online payment gateways.

15. User Interface and User Experience:

- The system should have an intuitive, user-friendly interface that is visually appealing and easy to navigate.
- It should provide a seamless and engaging user experience across different devices and screen sizes.

This requirement specification provides an overview of the key features and functionalities that the One Spot Education system should possess. It serves as a foundation for the system's design, development, and implementation, ensuring that the system meets the needs and expectations of its users, whether they are educators, students, or administrators.

Functional Requirements:-

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that show how a use case is to be fulfilled. They are supported by non-functional requirements, which impose constraints on the design or implementation (such as performance requirements, security, or reliability).As defined in requirements engineering, functional requirements specify particular behaviors of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability.

Functional requirements define the specific features, functionalities, and behaviors that the system must exhibit to fulfill its intended purpose. They describe what the system should do and how it should behave in response to user interactions. Here are some functional requirements for the One Spot Education system:

- **User Registration and Authentication:** Users should be able to create accounts, provide necessary information, and authenticate securely.
- **Dashboard:** Users should have access to a personalized dashboard displaying enrolled courses, assignments, announcements, and progress tracking.
- **Course Management:** Educators should be able to create and manage courses, add course materials, resources, and syllabic.
- **Live and Recorded Classes:** The system should support scheduling and delivery of live online classes, with interactive features like chat and Q&A. Recorded classes should be accessible for on-demand viewing.
- **Assignment and Homework Management:** Educators should be able to create and assign homework and assignments, students should be able to submit their work electronically, and educators should be able to provide feedback and grades.
- **Attendance Management:** The system should automate attendance tracking, allowing educators to record and monitor student attendance. Reporting and analytics on attendance should be available.
- **Communication and Collaboration:** The system should provide communication channels like discussion forums and private messaging for students, educators, and parents to interact and collaborate.
- **Reporting and Analytics:** The system should generate comprehensive reports and analytics on student performance, engagement levels, and course progress. Administrators and educators should have access to these reports.

Technical Requirements:-

Technical requirements define the underlying technology, infrastructure, and software components needed to implement and operate the system effectively. Here are some technical requirements for the One Spot Education system:

- **Web Application Framework:** The system should be developed using Laravel, a PHP web application framework.
- **Database Management System:** The system should utilize MySQL or PostgreSQL for data storage and retrieval.
- **Server Environment:** The system should be deployed on a server with sufficient processing power, memory, and storage to handle user traffic and data storage requirements.
- **Web Server:** The system should be compatible with popular web servers like Apache or Nginx.
- **Security Measures:** The system should implement secure user authentication, data encryption, and access control measures to ensure data security and user privacy.
- **Scalability:** The system should be designed to scale horizontally or vertically to handle increased user traffic and growing data storage needs.
- **Compatibility:** The system should be compatible with modern web browsers (e.g., Chrome, Firefox, Safari, Edge) and operating systems (e.g., Windows, macOS, Linux).

Nonfunctional Requirements:-

In systems engineering and requirements engineering, non-functional requirements are requirements which specify criteria that can be used to judge the operation of a system, rather than specific behaviors. This should be contrasted with functional requirements that specify specific behavior or functions. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements". Qualities, of Non-functional requirements can be divided into two main categories. Execution qualities, such as security and usability, are observable at run time. Evolution qualities, such as extensibility and scalability, embody in the static structure of the software system. Non-functional requirements define the attributes or qualities that the system should possess, rather than specific functionalities. They describe how the system should perform and its characteristics. Here are some non-functional requirements for the One Spot Education system:

[4] Software Engineering Paradigm applied

For the One Spot Education system, the chosen software engineering paradigm is the **Waterfall Model**. This model is particularly suitable for a student final year project due to its straightforward, sequential approach, which simplifies project planning and implementation. Given that there are no real clients or organizational constraints, the Waterfall Model allows for a clear, phase-by-phase development process, ensuring that each stage is thoroughly completed before moving on to the next.

Phases of the Waterfall Model Applied:

1. **Requirements Analysis:**
    - This initial phase involves gathering and documenting all the requirements for the One Spot Education system. Detailed Software Requirement Specifications (SRS) are created to ensure a clear understanding of what the system needs to achieve. Since this is a self-project, requirements are based on the envisioned functionality and features.
2. **System Design:**
    - Based on the requirements, a system architecture is designed. This includes defining the overall system structure, creating data models, and designing the system's user interface. For the One Spot Education system, this phase includes designing modules like User Management, Course Management, Live Classes, and others.
3. **Implementation:**
    - The system design is translated into code in this phase. Each module is developed according to the design specifications. The development is done sequentially, ensuring that each part of the system is functional before moving on to the next.
4. **Integration and Testing:**
    - Once individual modules are developed, they are integrated and tested as a complete system. Testing ensures that all parts of the system work together seamlessly and that any bugs or issues are identified and resolved.
5. **Deployment:**
    - After successful testing, the One Spot Education system is deployed in a controlled environment where it can be used and accessed. This phase ensures that the system is fully functional and ready for use by students and educators.
6. **Maintenance:**
    - Although maintenance might be minimal for a student project, this phase involves making any necessary updates, fixing bugs that may arise, and possibly adding new features based on feedback.

Suitability of the Waterfall Model:

- **Clear Structure:** The Waterfall Model's clear, linear progression makes it easier for students to follow and manage their project timeline.
- **Documentation:** Extensive documentation in each phase ensures that the project goals are well-defined and can be easily reviewed.
- **Simplified Management:** The sequential nature of the Waterfall Model allows for straightforward project management, which is ideal for a self-directed student project

.

Every new software engineering paradigm that gets invented seems to be greeted with the battle cry "Revolution!" as if everywhere software engineers and programmers were freed suddenly of all the evils that the previous paradigm had foisted upon them. Each new generation of unspoiled programming acolytes is admonished to go out into the world, to spread the word, and to suffer not the sins of the previous generation. The truth is that each and every one of these so-called revolutions was built unequivocally on the features of the paradigm that was its predecessor. Software engineering paradigms did not revolt; they evolved! None of the lessons learned was ever forgotten. In many cases, the only things that were changed were those features that had been left completely neglected by an earlier paradigm.

In some respects, that explains why we have progressive shifts anyhow. At the end of any one paradigm phase, it may seem that everyone knows everything needed to write complete, consistent, and clear code. So, we are completely at the mercy of …. *researchers*. Those are the people who can watch what it is that master programmers do differently from other programmers and put a catchy name onto it. Of course, there are many researchers (more than there are master programmers), and some of the researchers are better than others. Their keen observation and insightful analysis is vital in order to tell us how to write the software in the future. It didn't happen overnight. It took many lines of code, many master programmers, and scores of researchers for us to be able to look back at what we can see as the mainstay of computer programming paradigms.

Benefits of the Paradigm:-

Re-usability: anyone that needs a particular functionality can use an appropriate module, without having to code the algorithm from scratch.

Specialization : one person can concentrate on writing a best possible module (function) for a particular task while others look after other areas.

Upgradability : if a programmer comes up with a better way to implement a module then he/she simply replace the code within the function. Provided the interface remains the same - in other words the module name and the order and type of each parameter are unchanged - then no changes should be necessary in the rest of the application.

In summary, the Waterfall Model provides a structured and disciplined approach to developing the One Spot Education system, ensuring that each phase is completed thoroughly before moving on to the next. This model's simplicity and clarity make it an excellent choice for a student final year project without real client or organizational dependencies.

[5] Data models (like DFD), Control Flow diagrams, State Diagrams/Sequence diagrams, Entity Relationship Model, Class Diagrams/CRC Models/Collaboration Diagrams/Use- case Diagrams/Activity Diagrams and other models depending upon your project requirements

The whole approach of analysis of problem should however be based around critical factors like the availability of information for making the decision, the time available for processing the data i.e. the realism. Software project management begins with a set of activities collectively called PROJECT PLANNING. Software project planning actually encompasses all of the activities. Planning involves estimation- to determine how much money, how much effort, how many resources, and how much time it will take to build a specific software-based system or product.

Phases Cover:-

- Pre–Analysis Studies
- System Analysis
- System Design
- Project Coding
- Project Testing
- Implementation & Documentation

- **Pre–Analysis Phase:** In this phase problems with existing system are to be determined and do the investigation to make the solutions.
- **System Analysis Phase:** In this phase system analysis is done by preparation of Software Requirement Specification.
- **System Design Phase:** The purpose of the design phase is to plan a solution for the problem specified in the requirements documents.
- **Project Coding Phase:** The goal of the coding phase is to translate the design of the system into a program code by help of a programming language like Visual Studio, Java, etc.
- **Project Testing Phase:** Testing concerned with the elimination of errors introduced during coding phase.
- **Implementation & Documentation Phase: -** This phase includes all the activities performed to keep the system operational after the installation of the software.

Data Model: -

Data modeling is a method used to define and analyze data requirements needed to support the business processes of an organization. The data requirements are recorded as a conceptual data model with associated data definitions. Actual implementation of the conceptual model is called a logical data model. To implement one conceptual data model may require multiple logical data models. Data modeling defines the relationships between data elements and structures, Data modeling techniques are used to model data in a standard, consistent, predictable manner in order to manage it as a resource. The use of this standard is strongly recommended for all projects requiring a standard means of defining and analyzing the data resources within an organization.

Data Design: -

Data Flow Diagram (DFD):

Data flow diagrams can be used to provide a clear representation of any software function and the technique starts with an overall picture of the software and continues by analysing each of the functional areas of interest. This analysis can be carried out to precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

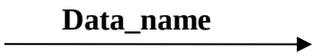
External Entity:-

An external entity is a source or destination of a data flow, which is outside the area of study. Only those entities, which originate or receive data are represented on a business process diagram. The symbol used is an oval containing a meaningful and unique identifier.

Process:-

A process shows a transformation or manipulation of data flows within the system. Firstly an identification number appears in the upper corner. This is allocated arbitrarily at the top level and serves as a unique reference. Finally, a descriptive title is placed in the center of the box. This should be a simple imperative sentence with a specific verb, for example 'SignIn' or 'Signup'.

Data Flow:-



A data flow shows the flow of information from its source to its destination. A data flow is represented by a line, with arrowheads showing the direction of flow. Information always flows to or from a process and may be written, verbal. The processes or data stores at its head or tail may reference each data flow.

Data Store:-

A data store is a holding place for information within the system. It is represented by an open ended narrow rectangle. Data stores may be long-term Database such as user detail. Each data store should be given a reference followed by an arbitrary number.

Context Diagram:-

- The context diagram represents the entire system under investigation. This diagram should be drawn first, and used to clarify and agree the scope of the investigation.
- The components of a context diagram are clearly shown on this screen. The system under investigation is represented as a single process, connected to external entities by data flow, resources flow.
- The context diagram clearly shows the interfaces between the system under investigation and the external entities with which it communicates. Therefore, whilst it is often conceptually trivial, a context diagram serves to focus attention on the system boundary and can help in clarifying the precise scope of the analysis.
- The context diagram shown below represents a book of chat web server. Chat web server receives details of users, and shows every detail of users who are currently on line.

Data Flow Diagram (DFDs) of One Spot Education:-

Data flow diagrams models the passage of data in the system and are represented by lines joining system components. Flows of data in the system can take place between:-

- Between two processes,
- From a data store to a process,
- From a process to data store,
- From a source to process and
- From a process to a sink.

Though the system mainly consists of two parts viz. online admission and online examination and other parts are going to be automated gradually…, so DFD is also, illustrated in two parts, respectively…

Context Level Diagram of One Spot Education:-

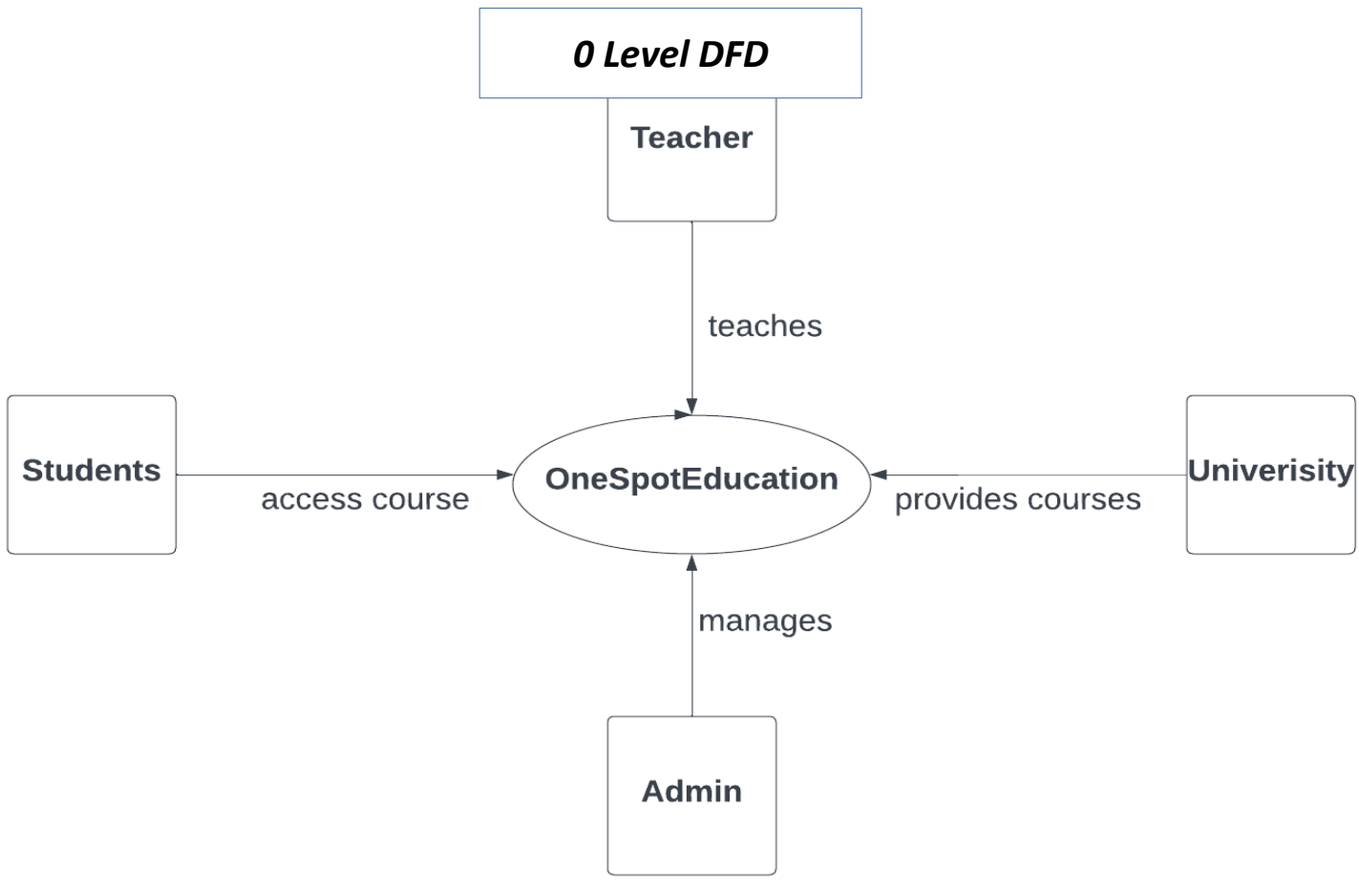


Fig 2.5.1: Context level DFD of One Spot Education system

The provided diagram is a Context Level Data Flow Diagram (DFD) for the One Spot Education system, also known as a Level 0 DFD. This diagram gives a high-level overview of the system and its interactions with external entities.

1. **External Entities**:
    - **Students**: Represents the end-users who access the One Spot Education system to enroll in and participate in courses. They interact with the system primarily to access course materials and live classes.
    - **Teacher**: Represents educators who use the system to deliver courses. They interact with the system by uploading course materials, conducting live classes, and assessing students' performance.
    - **University**: Represents the institution that provides courses to the One Spot Education system. They manage course content, upload syllabus, and possibly provide accreditation.
    - **Admin**: Represents the system administrators who manage and maintain the One Spot Education platform. They handle user management, system configuration, and overall platform maintenance.
2. **System (One Spot Education)**:
    - The central system, One Spot Education, acts as a hub where all interactions occur. It facilitates the connection and flow of data between the external entities.
3. **Interactions**:
    - **Students → One Spot Education**: Students access courses through the One Spot Education platform. This includes enrolling in courses, attending live classes, and submitting assignments.
    - **Teacher → One Spot Education**: Teachers provide educational content and conduct live sessions through the platform. They also assess and grade students' assignments.
    - **University → One Spot Education**: Universities provide the courses, including course materials and syllabus, to the One Spot Education platform.
    - **Admin → One Spot Education**: Administrators manage the platform, ensuring all users have appropriate access and maintaining the system's functionality and security.

1st Level DFD

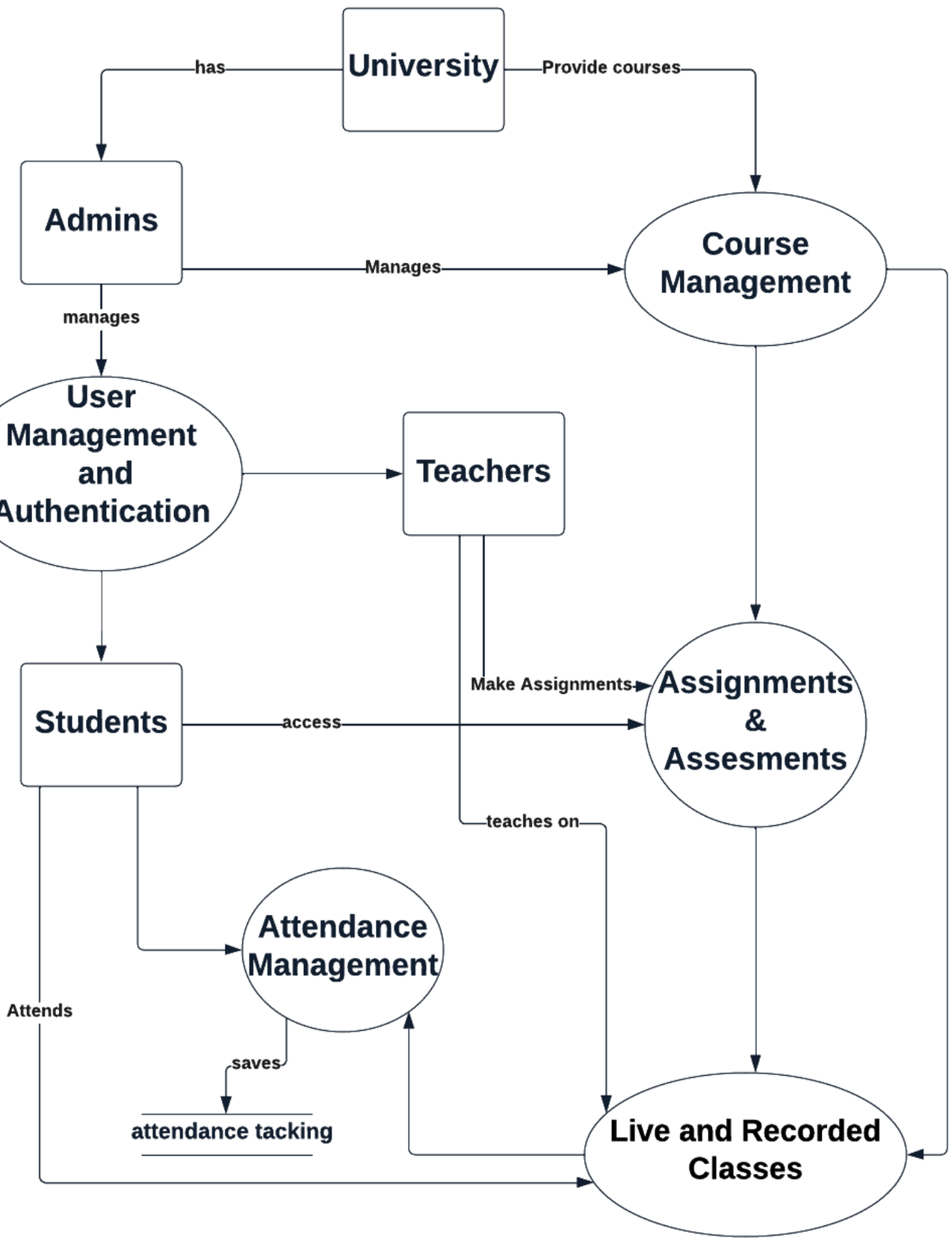


Fig 2.5.2: 1st level DFD of One Spot Education system

The Level 1 Data Flow Diagram (DFD) for the One Spot Education system expands on the Context Level DFD (Level 0 DFD) by detailing the major sub-processes within the system and showing how data flows between these sub-processes and external entities. Here's a brief explanation of the Level 1 DFD for the One Spot Education system:

1. **Sub-processes within One Spot Education:**

    1) **User Management and Authentication:** This sub-process handles all user-related activities such as user registration, authentication, profile management, and user role assignments.
    2) **Course Managemen**t: This sub-process is responsible for managing course information, including course creation, syllabus management, and course material uploads.
    3) **Live Classes and Collaboration:** This sub-process deals with scheduling and conducting live classes, as well as facilitating collaboration tools such as discussion forums and chat rooms.
    4) **Assignment and Assessment**: This sub-process covers the management of assignments, including assignment creation, submission, and grading.
    5) **Attendance management:** This sub-process generates various reports and analytics related to student performance, course progress, and attendance.

2. **Interactions and Data Flows:**

    1) **Students**:
        - Interact with the User Management sub-process to register and manage their profiles.
        - Access courses through the Course Management sub-process.
        - Attend live classes and use collaboration tools through the Live Classes and Collaboration sub-process.
        - Submit assignments and receive grades through the Assignment and Assessment sub-process.
        - View performance reports generated by the Analytics and Reporting sub-process.
    2) **Teachers**:
        - Interact with the User Management sub-process for user-related tasks.
        - Manage courses and upload materials through the Course Management sub-process.
        - Conduct live classes and use collaboration tools through the Live Classes and Collaboration sub-process.
        - Grade assignments and provide feedback through the Assignment and Assessment sub-process.
    3) **University**:
        - Provides courses and materials through the Course Management sub-process.
    4) **Admin**:
        - Manages system configurations and security through the System Administration and Configuration sub-process.
        - Ensures data integrity and performs system backups through the System Administration and Configuration sub-process.

2nd Level DFD

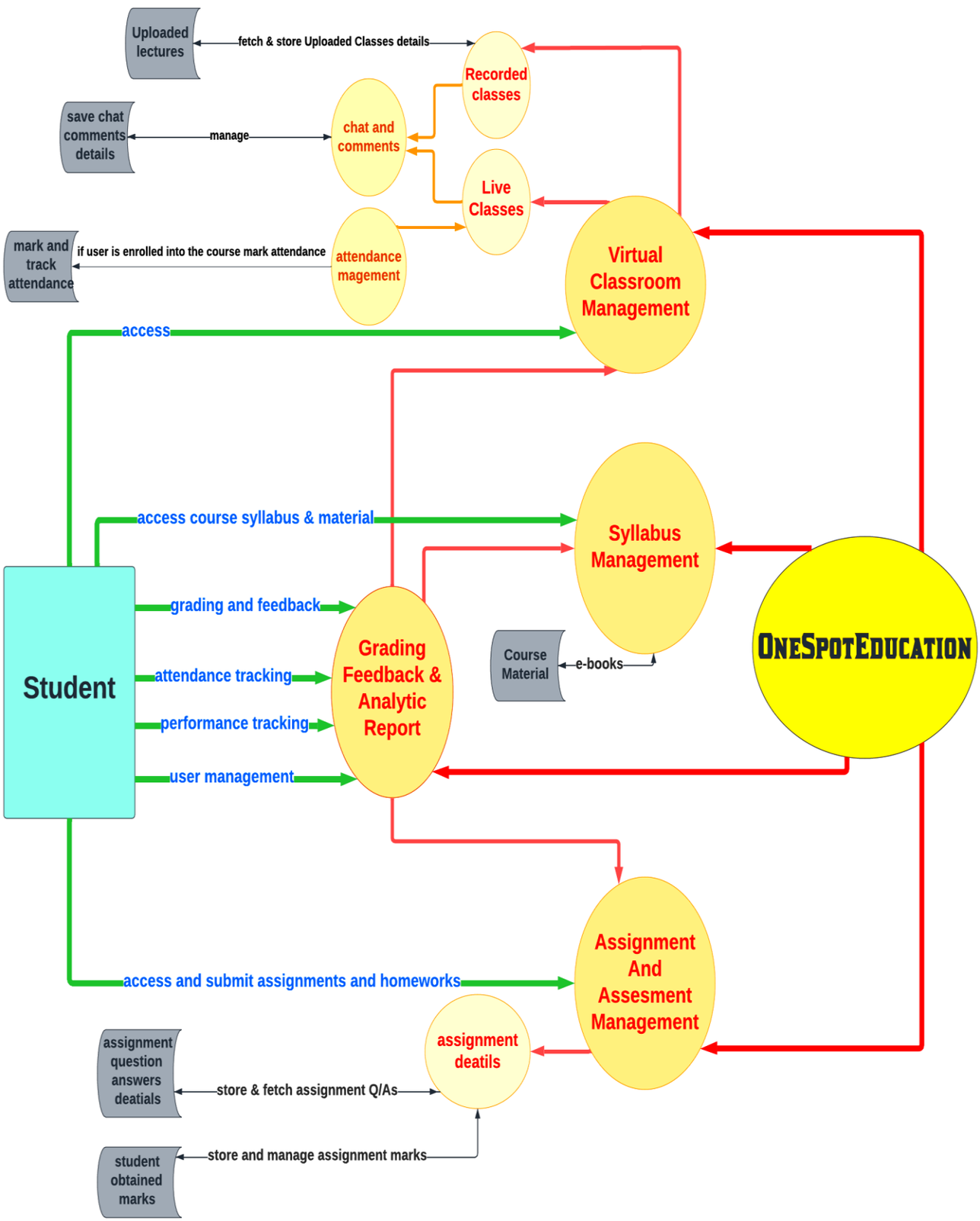


Fig 2.5.3 : 2nd level DFD of One Spot Education System

Above Level 2 Data Flow Diagram (DFD) outlines the interactions between various components of the OneSpotEducation web application, aimed at digitalizing the traditional education system with advanced features. Here's a breakdown and brief explanation of the DFD components and their interactions:

Data Flows

- **Green Arrows (Access and Fetch)**: Represent data being accessed by the student, such as course syllabus, assignments, and feedback.
- **Red Arrows (Manage and Store)**: Indicate internal management and storage processes within the system components.
- **Yellow Ovals (Processes and Data Stores)**: Represent various processes like live classes, recorded classes, and assignment management.
- **Gray Boxes (Data Stores)**: Represent data storage for uploaded lectures, assignment details, and student marks.

Components shown in level 2 diagrams

1. **Student**:
    - The central user interacting with the system.
    - Accesses and submits assignments, homeworks, and course material.
    - Engages with live and recorded classes.
    - Receives grading and feedback.
    - Tracks attendance and performance.
2. **Virtual Classroom Management**:
    - Manages live and recorded classes.
    - Handles chat and comments during live classes.
    - Manages attendance for live classes.
    - Stores uploaded lectures and recorded class details.
3. **Syllabus Management**:
    - Manages course syllabus and material.
    - Provides access to e-books and other course-related content.
4. **Assignment and Assessment Management**:
    - Manages assignment details, submissions, and assessments.
    - Stores and fetches assignment Q&As.
    - Records and manages assignment marks.
5. **Grading Feedback & Analytic Report**:
    - Provides grading and feedback to students.
    - Tracks attendance and performance.
    - Handles user management.

6. **OneSpotEducation**:

    - Acts as the overarching system that integrates all these components.
    - Ensures seamless interaction and data flow between components.

Data Flow and Interactions

- **Student Interaction**:

    - Students access the system to engage in live or recorded classes managed by the Virtual Classroom Management component.
    - Attendance is automatically managed during live classes.
    - Students can view and download the course syllabus and materials from the Syllabus Management.
    - Assignments and assessments can be accessed, submitted, and tracked through the Assignment and Assessment Management.
    - Performance tracking and feedback are managed and provided by the Grading Feedback & Analytic Report component.

- **Virtual Classroom Management**:

    - Handles both live classes and recorded classes.
    - Facilitates chat and comments during live sessions.
    - Automatically marks attendance if the student is enrolled in the course.
    - Interacts with Syllabus Management to fetch and store uploaded lectures and details.

- **Syllabus Management**:

    - Provides course materials and e-books.
    - Interacts with the Virtual Classroom Management to provide necessary course content.

- **Assignment and Assessment Management**:

    - Stores details of assignments, including questions and answers.
    - Manages student marks for assignments.
    - Provides this data to the Grading Feedback & Analytic Report for feedback and performance tracking.

- **Grading Feedback & Analytic Report**:

    - Provides grading and feedback based on assignments and assessments.
    - Tracks student performance and attendance.
    - Manages user data and integrates this information to provide comprehensive reports.

Class Diagram:-

Class diagram outlines the core structure of the OneSpotEducation system, emphasizing the relationships between users (Admins, Students, Teachers) and key entities like Courses, LiveClasses, and Assignments. It showcases how the system manages educational activities, communications, and attendance tracking, providing a comprehensive overview of the system's architecture.

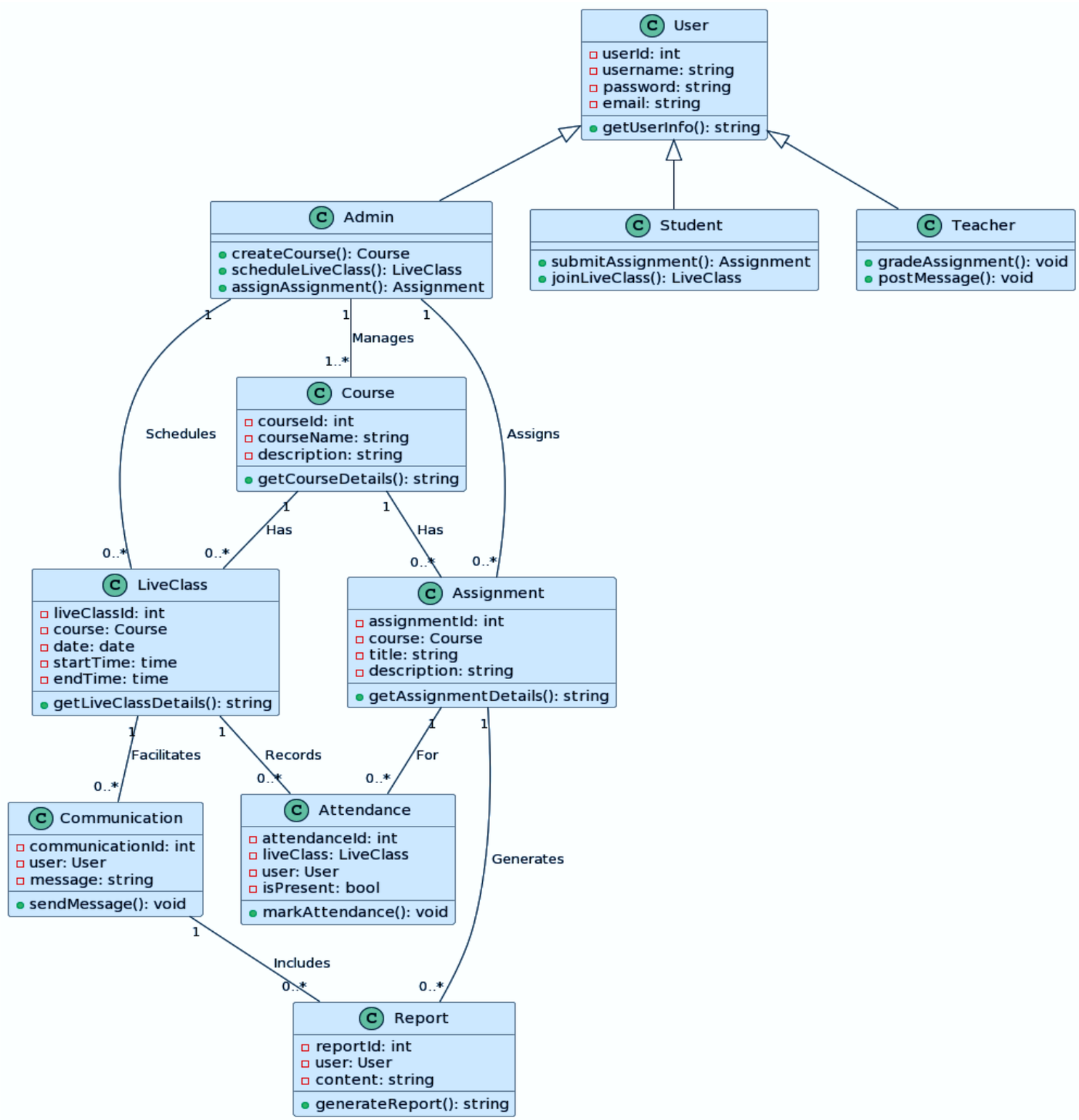


Fig: Class Diagram of the entire system

The class diagram for the OneSpotEducation project depicts the relationships and interactions between different classes within the system. Here's a brief description of each class and its role within the system:

Classes and Relationships

1. **User**:
    - Attributes: `userId`, `username`, `password`, `email`.
    - Methods: `getUserInfo()`.
    - The base class for all types of users (Admin, Student, Teacher).

2. **Admin** (inherits from User):
    - Methods: `createCourse()`, `scheduleLiveClass()`, `assignAssignment()`.
    - Manages courses, schedules live classes, and assigns assignments to courses.

3. **Student** (inherits from User):
    - Methods: `submitAssignment()`, `joinLiveClass()`.
    - Can submit assignments and join live classes.

4. **Teacher** (inherits from User):
    - Methods: `gradeAssignment()`, `postMessage()`.
    - Grades assignments and posts messages.

5. **Course**:
    - Attributes: `courseId`, `courseName`, `description`.
    - Methods: `getCourseDetails()`.
    - Represents a course managed by an admin and linked to assignments and live classes.

6. **LiveClass**:
    - Attributes: `liveClassId`, `course`, `date`, `startTime`, `endTime`.
    - Methods: `getLiveClassDetails()`.
    - Represents a live class scheduled for a course, can be joined by students and attended.

7. **Assignment**:
    - Attributes: `assignmentId`, `course`, `title`, `description`.
    - Methods: `getAssignmentDetails()`.
    - Represents an assignment linked to a course, which can be submitted by students and graded by teachers.
8. **Communication**:
    - Attributes: `communicationId`, `user`, `message`.
    - Methods: `sendMessage()`.
    - Facilitates communication (chat/messages) between users.
9. **Attendance**:
    - Attributes: `attendanceId`, `liveClass`, `user`, `isPresent`.
    - Methods: `markAttendance()`.
    - Records attendance for live classes.
10. **Report**:
    - Attributes: `reportId`, `user`, `content`.
    - Methods: `generateReport()`.
    - Generates reports based on user data and system interactions.

Relationships

- **Admin and Course**:
    - One Admin manages multiple Courses.
    - A Course is managed by one Admin.
- **Course and LiveClass**:
    - One Course can have multiple LiveClasses.
    - Each LiveClass is associated with one Course.
- **Course and Assignment**:
    - One Course can have multiple Assignments.Each Assignment is associated with one Course.
- **LiveClass and Communication**:
    - A LiveClass can facilitate multiple Communications (messages).
- **LiveClass and Attendance**:
    - One LiveClass can record multiple Attendance entries.
    - Each Attendance entry is for one LiveClass and one User.
- **Assignment and Report**:
    - An Assignment can generate multiple Reports.
    - Each Report is associated with one Assignment.

Use Case Diagram

The use case diagram for the OneSpotEducation system provides a visual representation of the various interactions between users and the system. It highlights the primary actors involved (students, teachers, admins, and university) and their respective interactions with different system functionalities. The key use cases include user management (registration, login), course management (creation, enrollment), live classes (scheduling, attending), assignment handling (creation, submission, grading), and system administration (configuration, data backup). This diagram helps in understanding the user requirements and the overall functionality of the system, ensuring all user interactions are effectively captured and addressed.

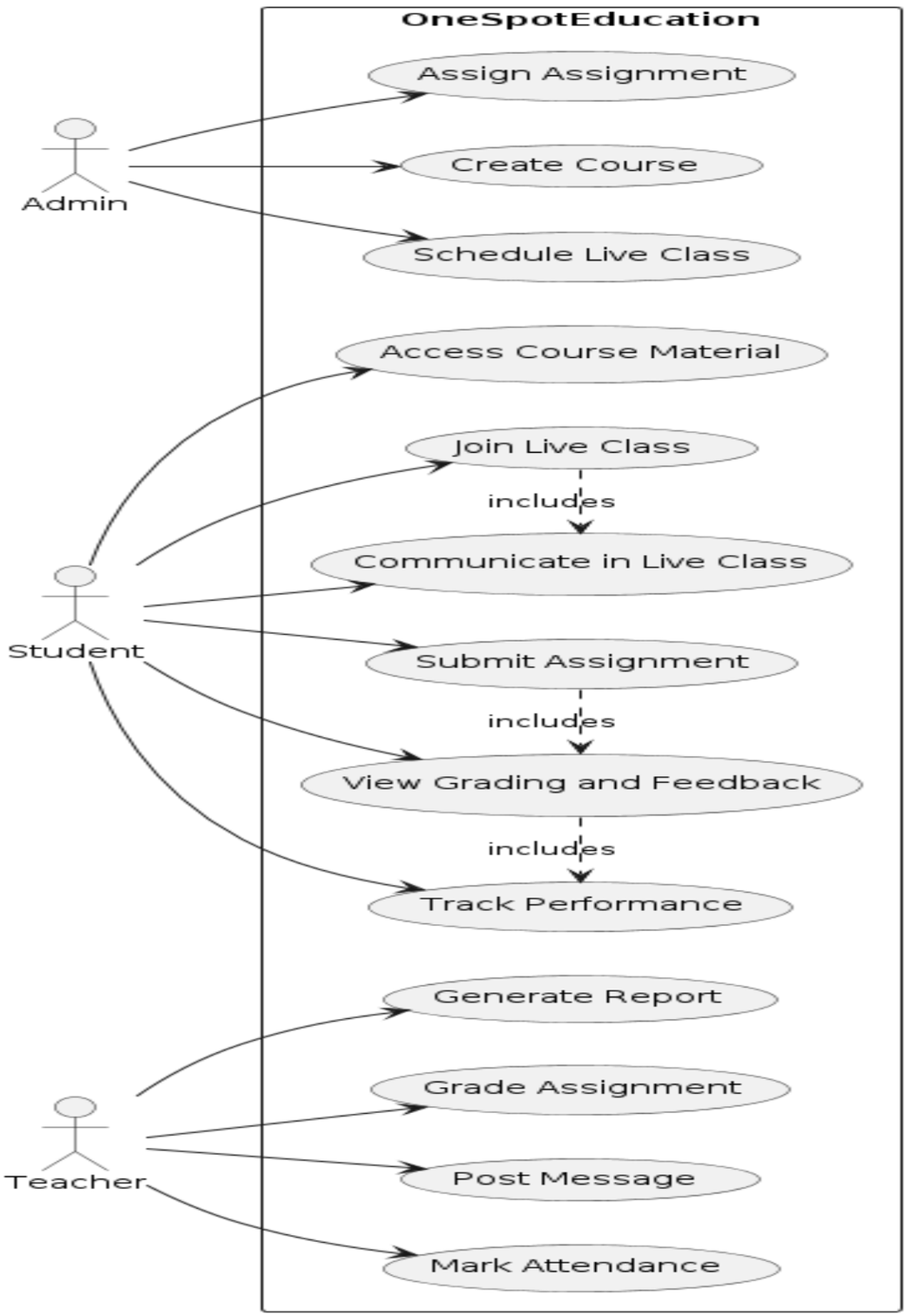


Fig 2.5.5 : Use Case Diagram

Entity Relationship Diagram (ERD):-

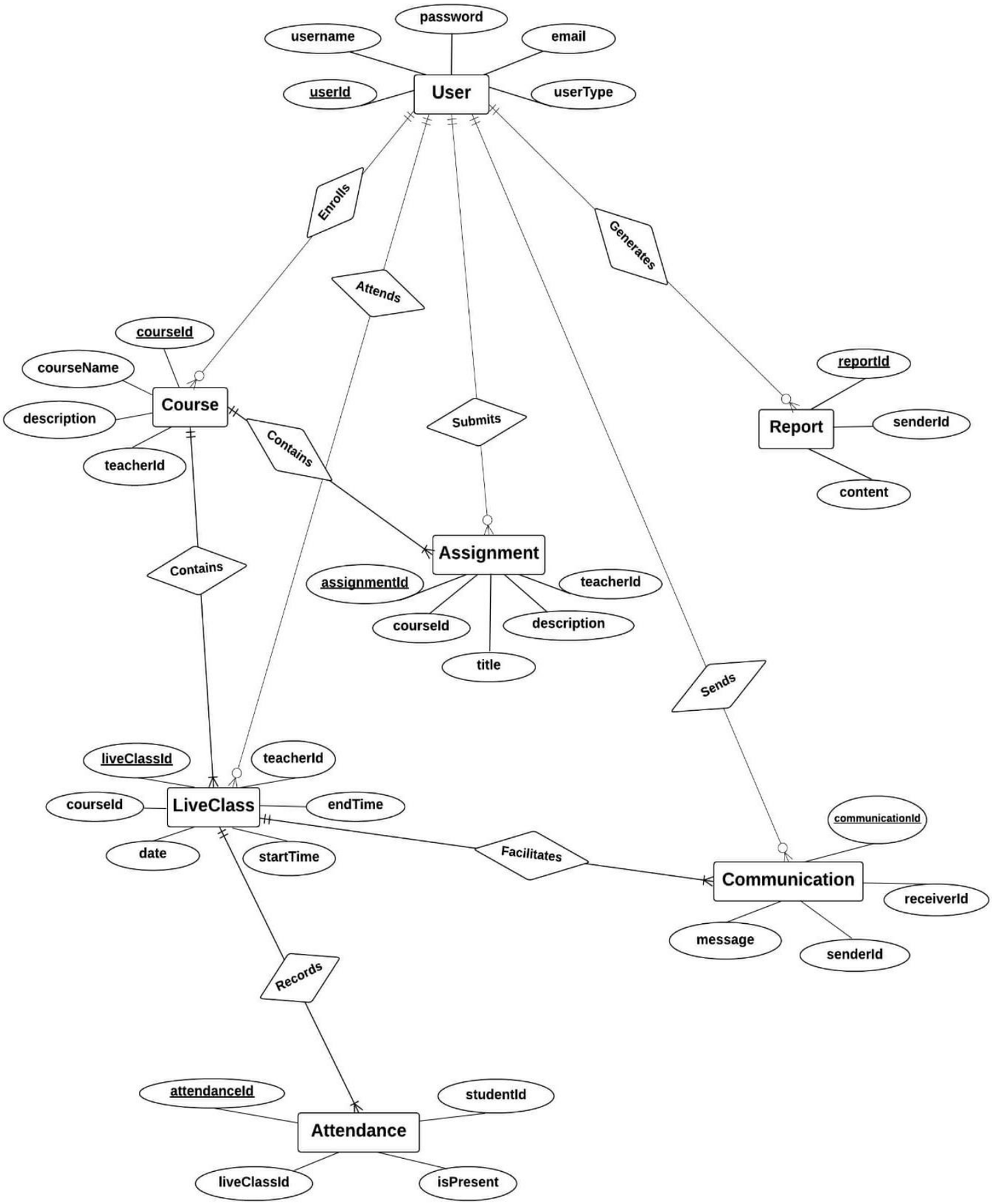


Fig: Entity Relationship Diagram (ERD) of system

The Entity-Relationship Diagram (ERD) for the OneSpotEducation system is a detailed graphical representation that illustrates the various entities within the system and the relationships between them. The major entities include Users, Courses, Live Classes, Assignments, Attendance, and Reports, each of which has specific attributes and identifiers. For instance, the Users entity captures details like user ID, username, password, and email, while the Courses entity includes course ID, course name, and description.

Relationships between these entities are also depicted, such as:

- **Users** to **Courses**: Users can enroll in multiple courses.
- **Courses** to **Live Classes**: Courses can have multiple live classes.
- **Live Classes** to **Attendance**: Each live class can have multiple attendance records.
- **Assignments** to **Courses**: Assignments are linked to specific courses.
- **Assignments** to **Reports**: Assignments generate reports based on student submissions.

The ERD helps in visualizing how data is structured and interlinked within the OneSpotEducation system, aiding in the database design process and ensuring that all necessary data relationships are adequately captured.

3. System Design

[1] Modularization Details :-

The OneSpotEducation system is a comprehensive platform designed to facilitate various aspects of online education for students, teachers, and administrators. The system is modular in nature, allowing each functional area to be developed, managed, and maintained independently. Below is a brief description of the modularization details, including the number of modules and their detailed descriptions.

Number of Modules

The OneSpotEducation system is divided into six main modules, each comprising several submodules to handle specific functionalities. The main modules are:

1. **User Management**
2. **Course Management**
3. **Live Classes and Collaboration**
4. **Assignment and Assessment**
5. **Analytics and Reporting**
6. **System Administration and Configuration**

Detailed Description of Each Module

1. **User Management**

    - **Description:** This module handles all aspects related to user accounts and roles within the system.
    - **Submodules:**
        - **User Registration and Authentication:** Manages user sign-up, login, and password management.
        - **User Profiles:** Allows users to create and update their profiles.
        - **User Roles and Permissions:** Assigns roles (student, teacher, admin) and manages access control.
2. **Course Management**

    - **Description:** This module focuses on creating, managing, and organizing courses.
    - **Submodules:**
        - **Course Creation and Management:** Allows administrators and teachers to create and manage courses.
        - **Syllabus Management:** Handles the creation and update of course syllabi.
        - **Course Materials and Resources:** Uploads and manages course-related materials and resources.
3. **Live Classes and Collaboration**

    - **Description:** Facilitates live virtual classes and collaboration tools for students and teachers.
    - **Submodules:**
        - **Live Class Scheduling:** Allows teachers to schedule live classes.
        - **Virtual Classroom Integration:** Integrates virtual classroom tools for live teaching.
        - **Collaboration Tools:** Provides tools like chat, forums, and group activities.

4. **Assignment and Assessment**

    - **Description:** Manages assignments, submissions, and assessments.
    - **Submodules:**
        - **Assignment Management:** Allows teachers to create and assign homework.
        - **Homework Submission System:** Enables students to submit their assignments.
        - **Assessment and Feedback:** Facilitates grading and providing feedback on assignments.
5. **Analytics and Reporting**

    - **Description:** Provides insights and reports based on various data points within the system.
    - **Submodules:**
        - **Analytics Integration:** Integrates analytical tools for data analysis.
        - **Student Performance Tracking:** Tracks and reports student progress and performance.
        - **Course Progress Monitoring:** Monitors and reports on course completion and engagement levels.
6. **System Administration and Configuration**

    - **Description:** Handles the overall administration and configuration of the system.
    - **Submodules:**
        - **System Configuration:** Manages system settings and configurations.
        - **User Management and Permissions:** Oversees user roles and permissions at a higher level.
        - **Data Backup and Security:** Ensures data security and regular backups for system integrity.

Each of these modules plays a crucial role in ensuring that the OneSpotEducation system operates smoothly and meets the needs of its users. The modular structure also allows for easy maintenance, updates, and scalability.

[2] ***Data Integrity & Constraints:-***

Based upon the previously drawn ERD, some of the proposed tables are given below, the alteration is desired in course of development and Based on the project requirements of the One Spot Education system, here are some of the essential data structures that may be used:

1. **User Data Structure:**
   - userId: int
   - username: string
   - password: string
   - email: string
   - userType: string (enum: Student, Teacher, Administrator)
   - additional attributes for user profile management
2. **Course Data Structure:**
   - courseId: int
   - courseName: string
   - description: string
   - teacherId: int (reference to the User who is the teacher of the course)
   - additional attributes for course details
3. **LiveClass Data Structure:**
   - liveClassId: int
   - courseId: int (reference to the Course to which the live class belongs)
   - date: date
   - startTime: time
   - endTime: time
   - teacherId: int (reference to the User who is conducting the live class)
   - additional attributes for live class details
4. **Assignment Data Structure:**
   - assignmentId: int
   - courseId: int (reference to the Course to which the assignment belongs)
   - title: string
   - description: string
   - teacherId: int (reference to the User who assigned the assignment)
   - additional attributes for assignment details

5. Attendance Data Structure:

- attendanceId: int
- liveClassId: int (reference to the LiveClass for which attendance is marked)
- studentId: int (reference to the User who is marked present/absent)
- isPresent: bool (true for present, false for absent)

6. Communication Data Structure:

- communicationId: int
- senderId: int (reference to the User who sent the message)
- receiverId: int (reference to the User who received the message)
- message: string
- additional attributes for message details

7. Report Data Structure:

- reportId: int
- senderId: int (reference to the User who generated the report)
- content: string
- additional attributes for report details

8. Enrollment Data Structure:

- enrollmentId: int
- courseId: int (reference to the Course in which the student enrolled)
- studentId: int (reference to the User who enrolled in the course)
- enrollmentDate: date
- additional attributes for enrollment details

9. Grade Data Structure:

- gradeId: int
- assignmentId: int (reference to the Assignment for which the grade is given)
- studentId: int (reference to the User for whom the grade is given)
- score: float
- additional attributes for grade details

10. CourseMaterial Data Structure:

- materialId: int
- courseId: int (reference to the Course to which the material belongs)
- materialTitle: string
- materialContent: string (text of the material)

11. CourseSchedule Data Structure:

- scheduleId: int
- courseId: int (reference to the Course for which the schedule is created)
- scheduleDate: date
- startTime: time
- endTime: time
- location: string
- additional attributes for schedule details

12. Homework Data Structure:

- homeworkId: int
- courseId: int (reference to the Course for which the homework is assigned)
- title: string
- description: string
- deadline: date
- additional attributes for homework details

These data structures represent the main entities and relationships required for the One Spot Education system. Depending on the system's complexity and specific requirements, additional data structures and attributes may be incorporated to support all functionalities and features effectively.

[3] *Database design, Procedural Design/Object Oriented Design:-*

[A] Complete Database & Table Structure:-

This is a large enough application, and for building such a type numbers of tables are required. The number of tables is gradually increasing in course of development, as well as with the application of Normalization. Here only a few of those are mentioned, mainly the masters one, just to clear an overview of the application.

1. **Table Name: *Users* - Stores information about all users in the system.**

Column Name	Data Type (Size)	Constraint
userId	int	Primary Key, Auto Increment
username	varchar(50)	Unique, Not Null
password	varchar(100)	Not Null
email	varchar(100)	Unique, Not Null
userType	varchar(20)	Not Null
firstName	varchar(50)	
lastName	varchar(50)	
dateOfBirth	date	
phoneNumber	varchar(15)	
address	varchar(200)	

2. **Table Name: *Courses* - Stores information about courses offered.**

Column Name	Data Type (Size)	Constraint
courseId	int	Primary Key, Auto Increment
courseName	varchar(100)	Unique, Not Null
description	text	
teacherId	int	Foreign Key (Users.userId)
startDate	date	
startDate	date	

3. **Table Name: *LiveClasses* - Stores details of scheduled live classes.**

Column Name	Data Type (Size)	Constraint
liveClassId	int	Primary Key, Auto Increment
courseId	int	Foreign Key (Courses.courseId)
date	date	Not Null
startTime	time	Not Null
endTime	time	Not Null
teacherId	int	Foreign Key (Users.userId)

4. **Table Name: *Assignments* - Stores details of assignments for courses.**

Column Name	Data Type (Size)	Constraint
assignmentId	int	Primary Key, Auto Increment
courseId	int	Foreign Key (Courses.courseId)
title	varchar(200)	Not Null
description	text	
dueDate	date	
maxPoints	int	
teacherId	int	Foreign Key (Users.userId)

5. **Table Name: *Attendance* - Records attendance for live classes.**

Column Name	Data Type (Size)	Constraint
attendanceId	int	Primary Key, Auto Increment
liveClassId	int	Foreign Key (LiveClasses.liveClassId)
studentId	int	Foreign Key (Users.userId)
isPresent	boolean	Not Null

6. **Table Name: *Communication* - Stores user communication messages.**

Column Name	Data Type (Size)	Constraint
communicationId	int	Primary Key, Auto Increment
senderId	int	Foreign Key (Users.userId)
receiverId	int	Foreign Key (Users.userId)
message	text	

7. **Table Name: *Reports* - Stores various system-generated reports.**

Column Name	Data Type (Size)	Constraint
reportId	int	Primary Key, Auto Increment
senderId	int	Foreign Key (Users.userId)
content	text	

8. ***UserRole*** **- Stores the relationship between users and roles, mapping users to their associated roles in the system.**

Column Name	Data Type	Constraints
userRoleId	int	Primary Key
roleId	int	Foreign Key to Role.roleId
userId	int	Foreign Key to User.userId

9. ***CourseEnrollment*** **– Stores details about the course student is enrolled into.**

Column Name	Data Type	Constraints
enrollmentId	int	Primary Key
courseId	int	Foreign Key to Course,courseId
studentId	int	Foreign Key to User.userId
enrollmentDate	date	

10. ***Role*****: Contains various roles available in the system, defining different access privileges and permissions for users.**

Column Name	Data Type	Constraints
roleId	int	Primary Key
roleName	varchar(50)	
roleDescription	varchar(250)	

11. ***Grade*: Records the grades or scores achieved by students for specific assignments in courses.**

Column Name	Data Type(size)	Constraint
gradeId	int	Primary Key, Auto Increment
assignmentId	int	Foreign Key(Assignment.assignmentId), Not Null
studentId	int	Foreign Key(User.userId), Not Null
score	decimal(5, 2)	Not Null

12. **CourseMaterial: Stores the educational materials (e.g., documents, presentations) associated with each course.**

Column Name	Data Type(size)	Constraint
materialId	int	Primary Key, Auto Increment
courseId	int	Foreign Key(Course.courseId), Not Null
materialTitle	varchar(200)	Not Null
materialContent	text	Not Null

13. **CourseSchedule: Manages the schedule and location information for different classes or sessions within courses.**

Column Name	Data Type(size)	Constraint
scheduleId	int	Primary Key, Auto Increment
courseId	int	Foreign Key(Course.courseId), Not Null
scheduleDate	date	Not Null
startTime	time	Not Null
endTime	time	Not Null
location	varchar(100)	Not Null

14. **Homework: Keeps track of homework assignments associated with courses, including titles, descriptions, and deadlines.**

Column Name	Data Type(size)	Constraint
homeworkId	int	Primary Key, Auto Increment
courseId	int	Foreign Key (Course, courseId), Not Null
title	varchar(200)	Not Null
description	text	Not Null
deadline	date	Not Null

[B] Procedural Design:-

Technical Architecture:-

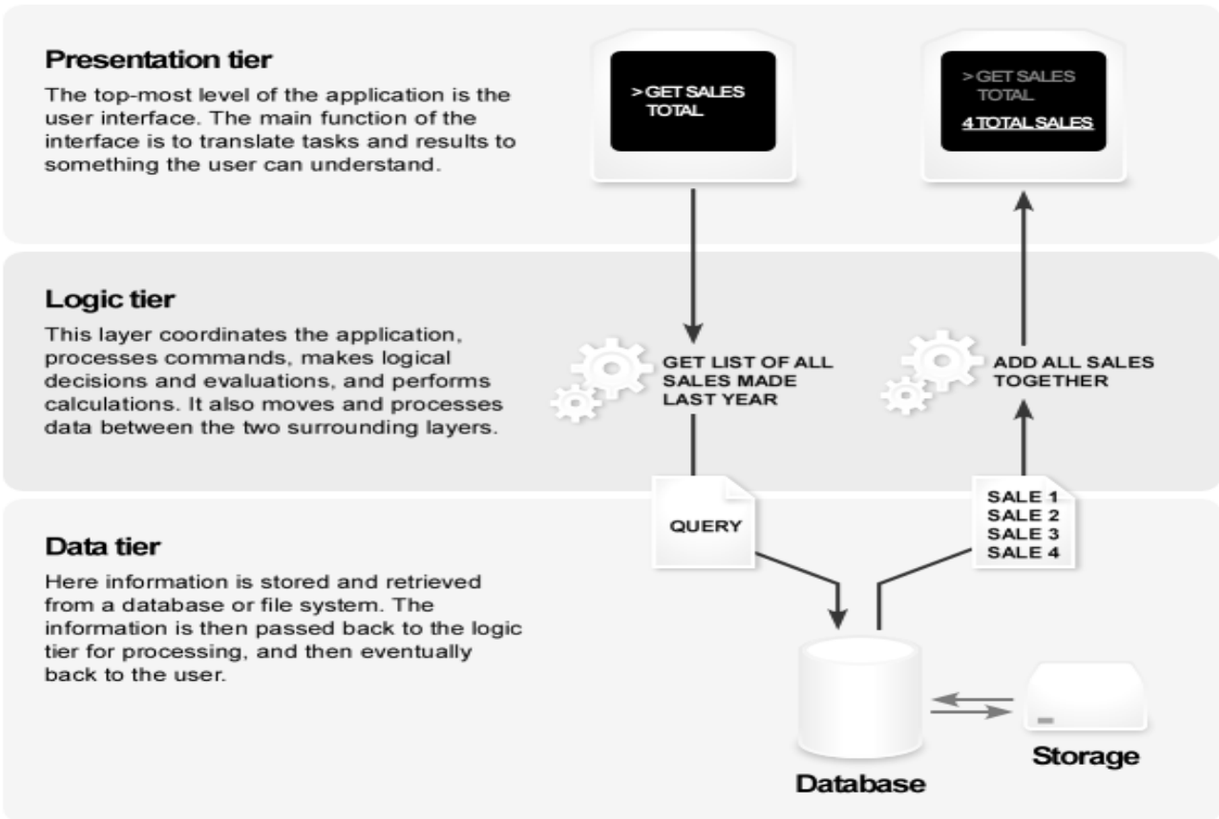


Fig: Three-Tier Architecture

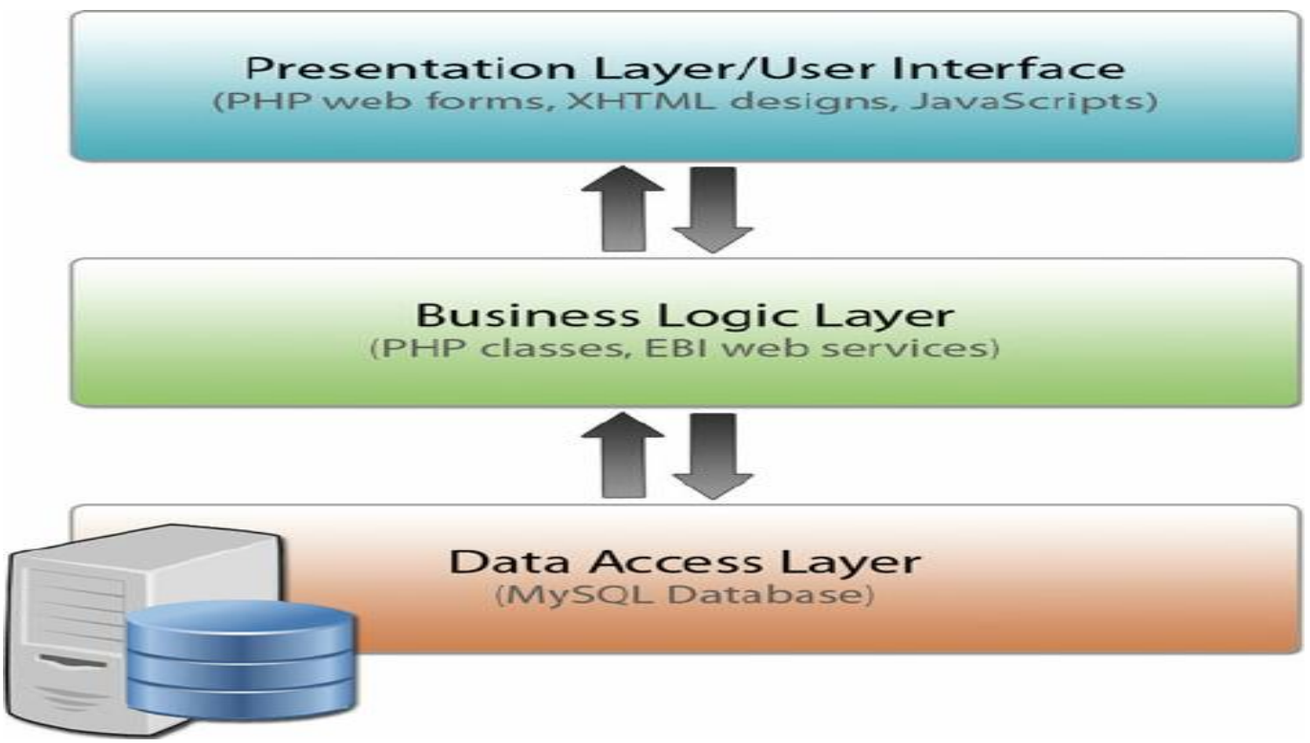


Figure: 3-tier Architecture (Illustrative View)

Three tier:-

Three-Tier Architecture is a popular architectural pattern that separates the application into three logical layers: presentation, business logic, and data storage. Each layer is independent and communicates with the other layers through well-defined interfaces. This separation of concerns makes the system more manageable and allows for easier maintenance and scalability.

Three-tier is a client-server architecture in which the user interface, functional process logic, computer data storage and data access are developed and maintained as independent modules, most often on separate platforms. The 3-Tier architecture has the following three tiers:

- **Presentation Tier-**

  This is the topmost level of the application. The presentation tier displays information related to such services as browsing merchandise, purchasing, and shopping cart contents. It communicates with other tiers by outputting results to the browser/client tier and all other tiers in the network.

- **Application Tier (Business Logic/Logic Tier)-**

  The logic tier is pulled out from the presentation tier and, as its own layer, it controls an application's functionality by performing detailed processing.

- **Data Tier-**

  This tier consists of Database Servers. Here information is stored and retrieved. This tier keeps data neutral and independent from application servers or business logic. Giving data its own tier also improves scalability and performance.

Advantages of Three-Tier Architecture:

1. **Modularity:** Allows independent development and maintenance of each layer.
2. **Scalability:** Facilitates scaling individual layers as needed.
3. **Security:** Helps enforce access control and protect sensitive data.
4. **Reusability**: Components in each layer can be reused in different parts of the application.

Using Three-Tier Architecture can be beneficial for complex systems like One Spot Education as it provides a structured approach to development. However, the specific choice of architecture depends on the project's requirements, the team's expertise, and other considerations like time and budget constraints.

Procedural design in software development refers to the detailed planning of procedures or methods to solve specific problems within a system. For the OneSpotEducation project, the procedural design will outline the steps and methods necessary to implement the various features and functionalities of the system.

Here are the procedural designs for the key features of the OneSpotEducation project:

1. Create Course

Procedure:

1. Admin logs into the system.
2. Admin navigates to the "Create Course" section.
3. Admin fills out the course details (name, description, etc.).
4. Admin submits the form.
5. The system validates the input.
6. If validation passes, the system saves the course details to the database.
7. Confirmation message is displayed to the Admin.

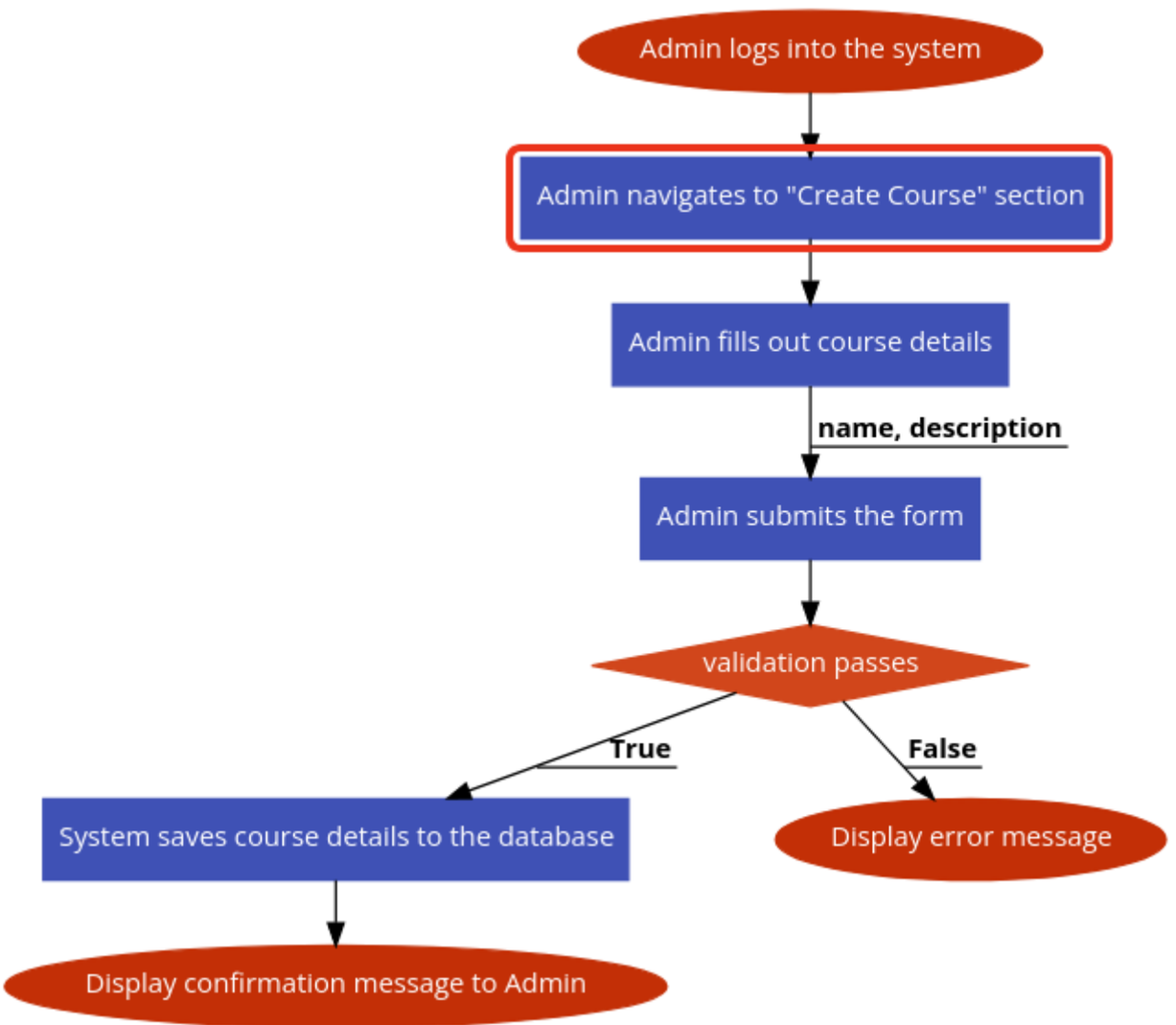


Figure: Procedural Diagram to create course

2. Schedule Live Class

Procedure:

1. Admin logs into the system.
2. Admin navigates to the "Schedule Live Class" section.
3. Admin selects the course for which the live class is to be scheduled.
4. Admin fills out the live class details (date, start time, end time).
5. Admin submits the form.
6. The system validates the input.
7. If validation passes, the system saves the live class details to the database.
8. Notification is sent to all enrolled students about the scheduled live class.
9. Confirmation message is displayed to the Admin.

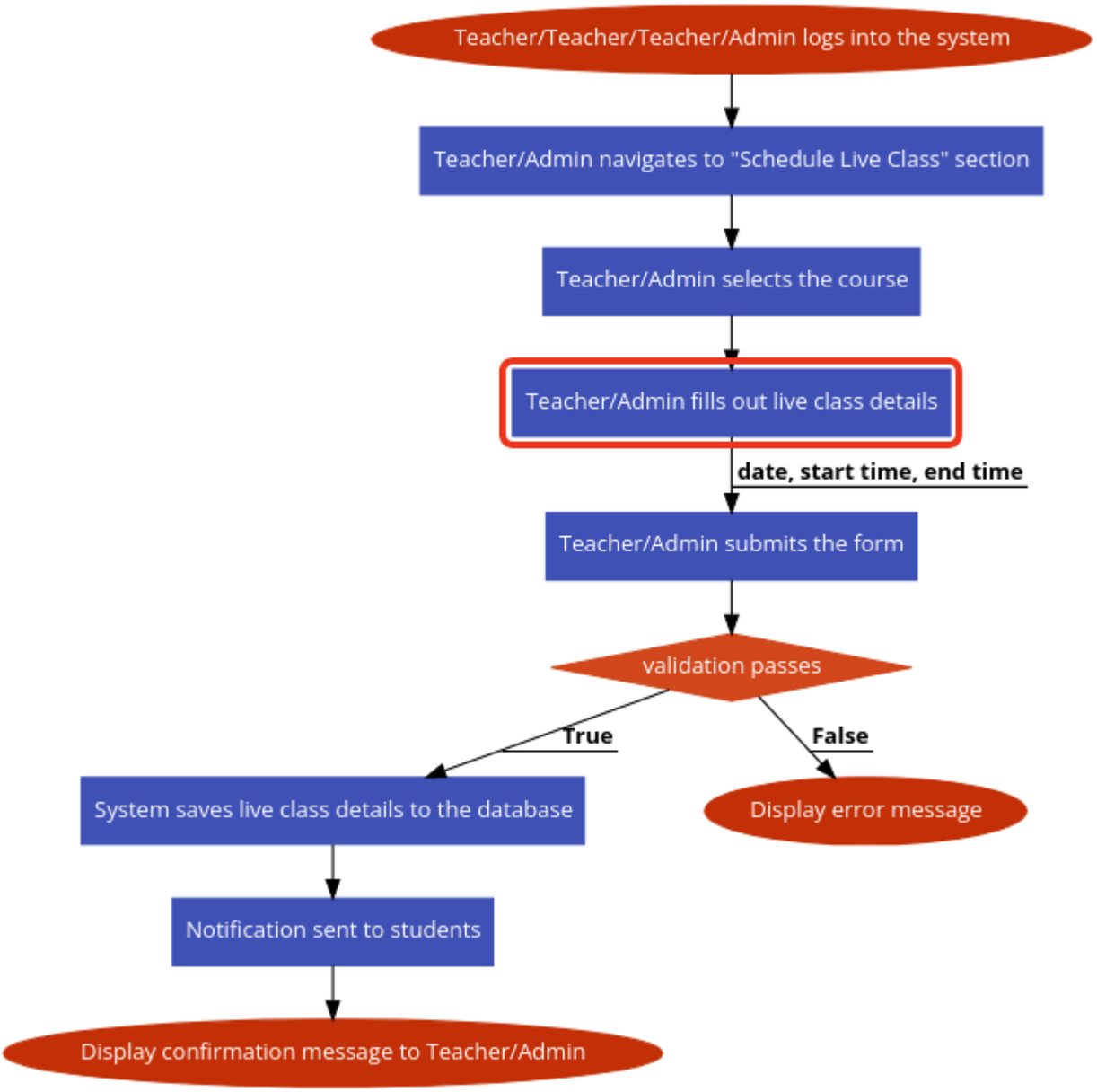


Figure: Procedural Diagram to Schedule Live Class

3. Assign Assignment

Procedure:

1. Admin logs into the system.
2. Admin navigates to the "Assign Assignment" section.
3. Admin selects the course for which the assignment is to be created.
4. Admin fills out the assignment details (title, description, due date).
5. Admin submits the form.
6. The system validates the input.
7. If validation passes, the system saves the assignment details to the database.
8. Notification is sent to all enrolled students about the new assignment.
9. Confirmation message is displayed to the Admin.

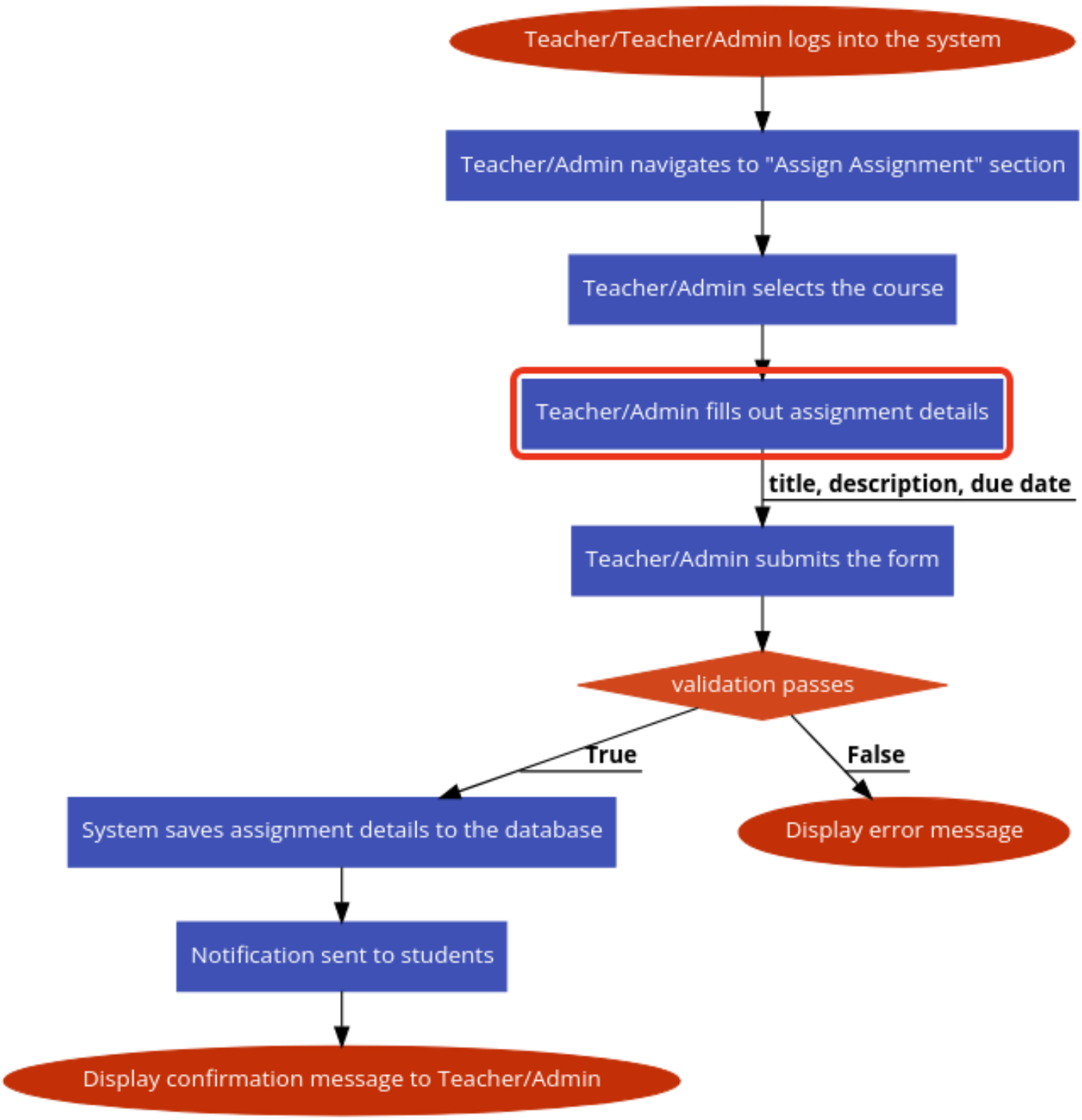


Figure: Procedural Diagram to Assign Assignments

4. Join Live Class

Procedure:

1. Student logs into the system.
2. Student navigates to the "Live Classes" section.
3. Student selects the live class to join.
4. The system checks if the class is currently live.
5. If the class is live, the student is granted access to the live class interface.
6. Student can participate in the live class and use the communication features to interact with the teacher and other students.

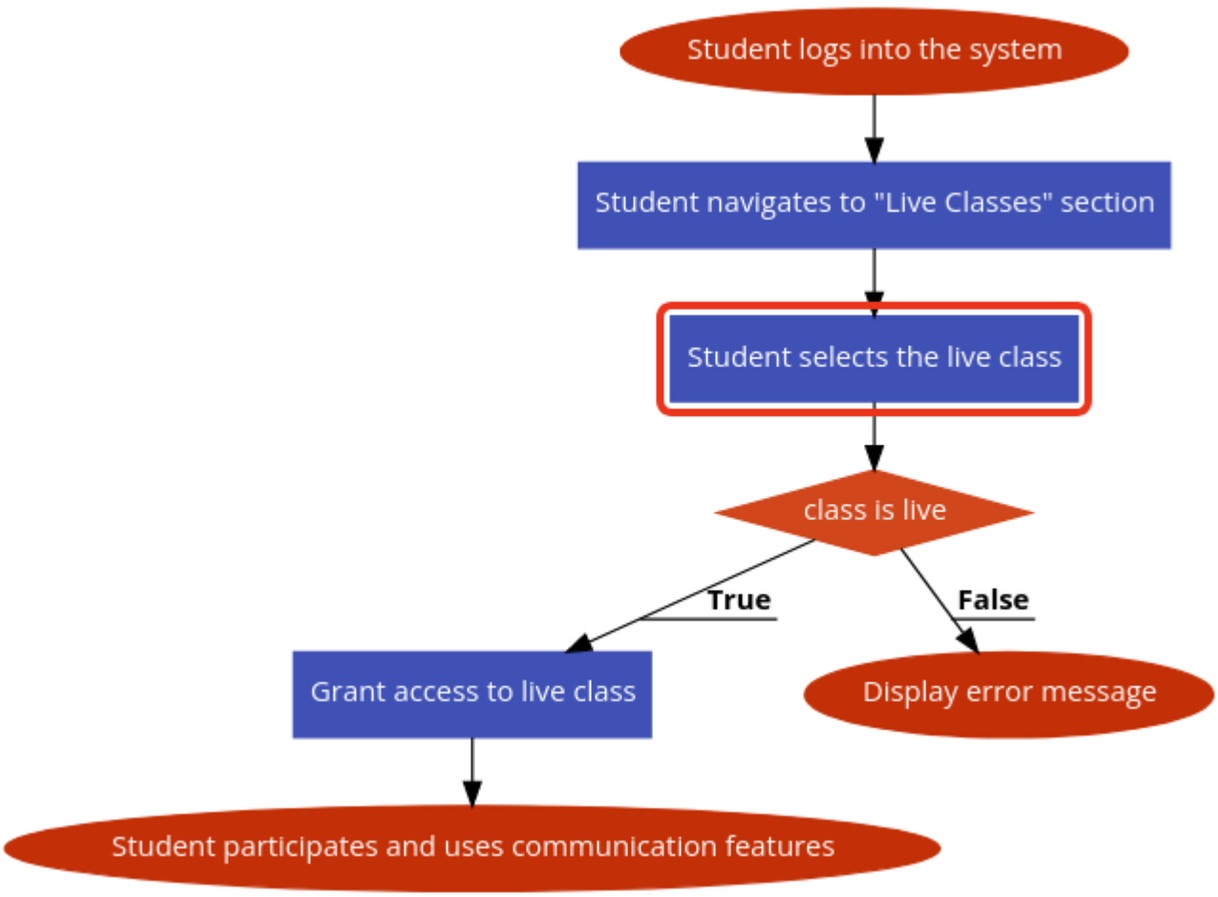


Figure: Procedural Diagram to Join live class

5. Submit Assignment

Procedure:

1. Student logs into the system.
2. Student navigates to the "Assignments" section.
3. Student selects the assignment to submit.
4. Student uploads the assignment file and adds any necessary comments.
5. Student submits the assignment.
6. The system validates the input.
7. If validation passes, the system saves the assignment submission details to the database.
8. Confirmation message is displayed to the Student.

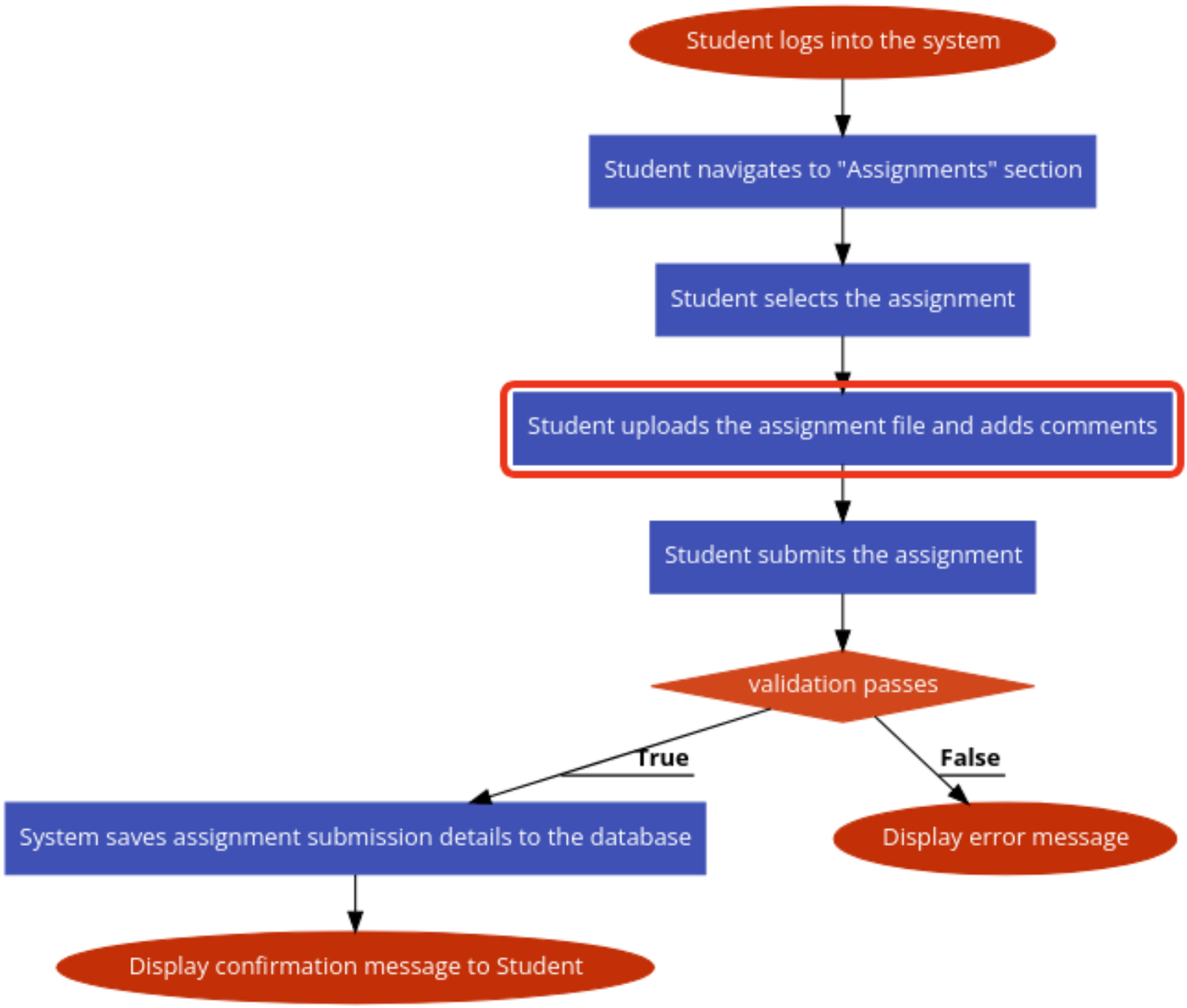


Figure: Procedural Diagram of Assignment Submission

6. Grade Assignment

Procedure:

1. Teacher logs into the system.
2. Teacher navigates to the "Assignments" section.
3. Teacher selects the assignment to grade.
4. Teacher reviews the submitted assignment.
5. Teacher enters the grade and any feedback.
6. Teacher submits the grade.
7. The system saves the grading details to the database.
8. Notification is sent to the student about the graded assignment.
9. Confirmation message is displayed to the Teacher.

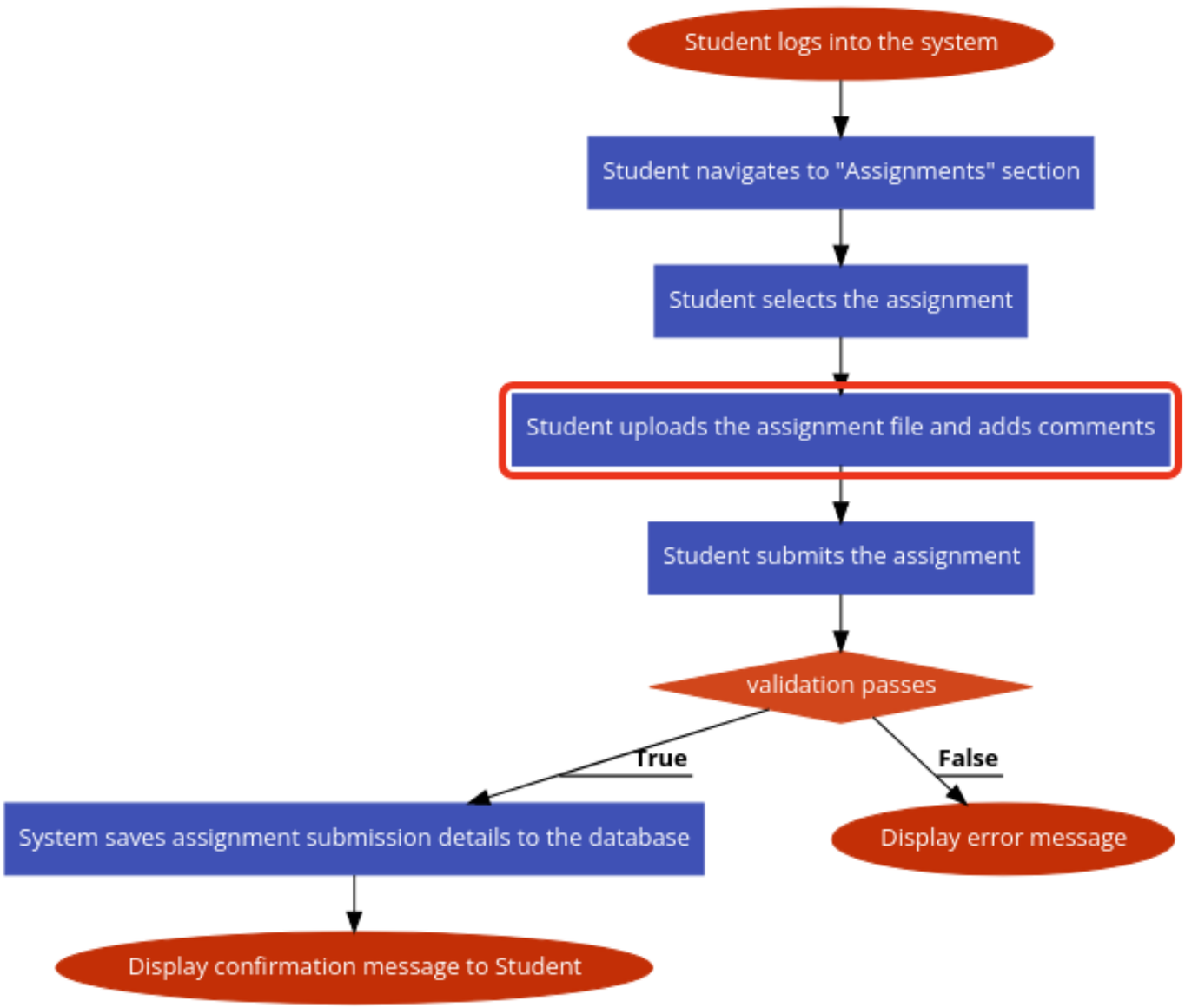


Figure: Procedural Diagram to Grade Assignment

7. Post Message

Procedure:

1. Teacher logs into the system.
2. Teacher navigates to the "Live Classes" or "Course Communication" section.
3. Teacher selects the relevant course or live class.
4. Teacher composes a message.
5. Teacher submits the message.
6. The system saves the message details to the database.
7. Notification is sent to all relevant users (students and other teachers).
8. Confirmation message is displayed to the Teacher.

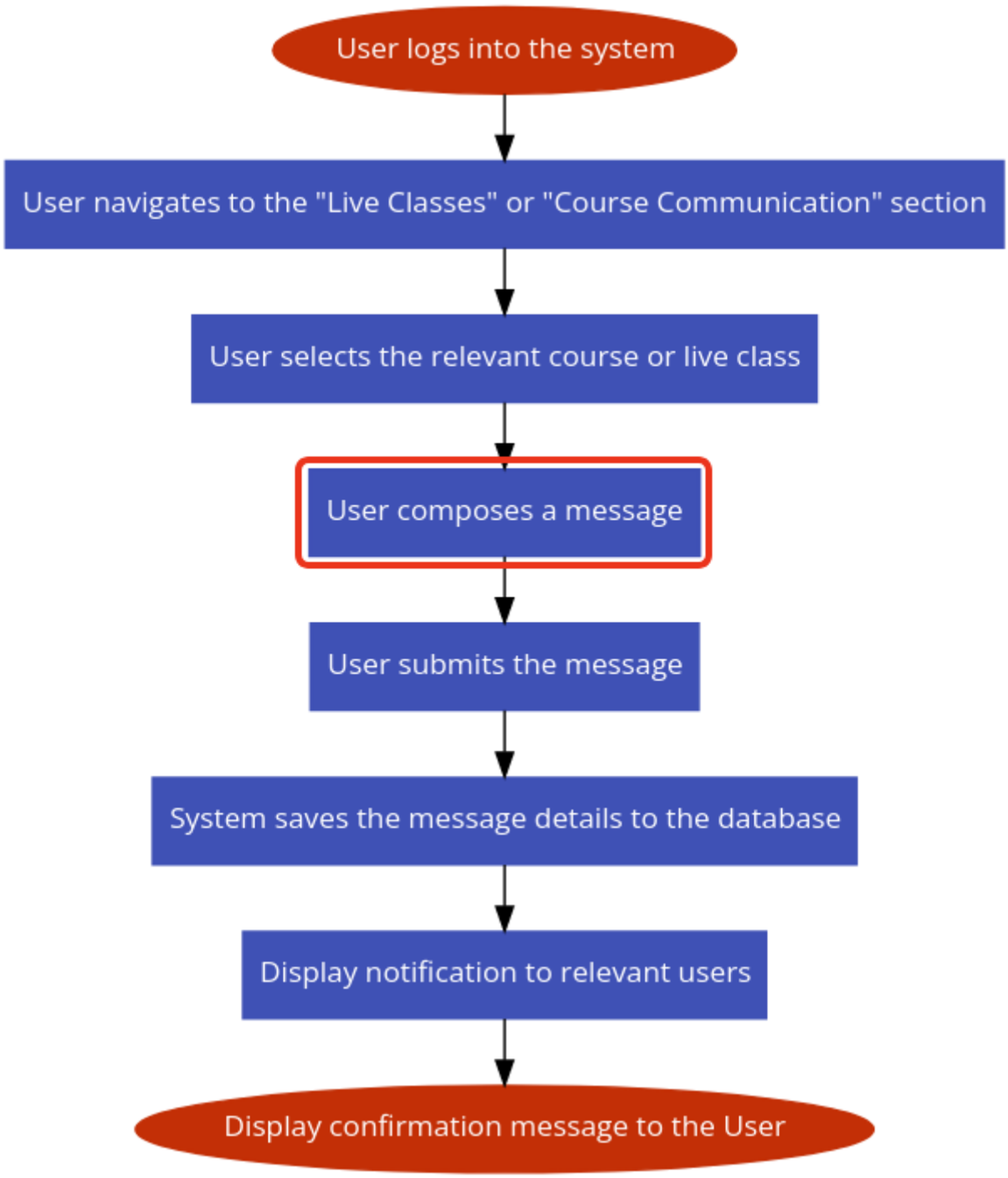


Figure: Procedural Diagram to Post Message

9. Generate Report

Procedure:

1. Teacher logs into the system.
2. Teacher navigates to the "Reports" section.
3. Teacher selects the criteria for the report (course, student, date range).
4. Teacher submits the report request.
5. The system processes the request and generates the report.
6. The report is displayed to the Teacher, and an option to download is provided.

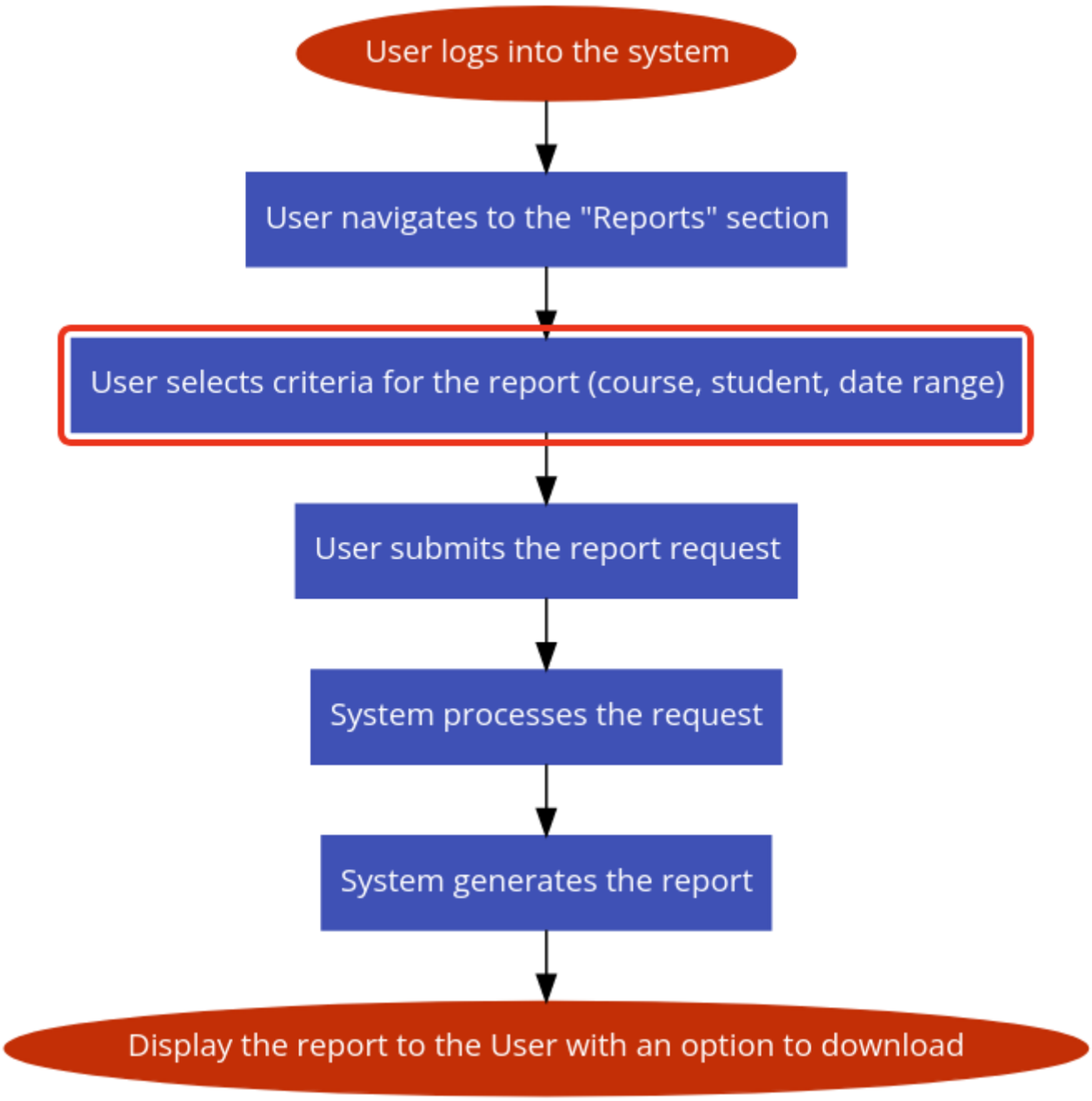


Figure: Procedural Diagram of Report Generation

These procedural designs provide a step-by-step approach to implementing the core functionalities of the OneSpotEducation project. Each procedure ensures that the interactions between the user and the system are clear and that the necessary validations and notifications are in place.

[C] Object Oriented Design

The **Model-View-Controller (MVC)** design pattern is a well-established architectural pattern used for developing scalable and maintainable software applications. It is particularly effective in the context of object-oriented design. MVC separates an application into three main interconnected components: the Model, the View, and the Controller. This separation facilitates modular development, making it easier to manage complexity and promote reusability.

Components of MVC Object Oriented Design:

1. **Model:**

   - **Role:** Represents the core functionality and data of the application. It directly manages the data, logic, and rules of the application.
   - **Details:**
     - Encapsulates the data and business logic.
     - Responsible for interacting with the database or other data sources.
     - Notifies the View of any state changes, typically using observer patterns or events.

2. **View:**

   - **Role:** Handles the presentation of the data. It is responsible for displaying the data provided by the Model in a specific format.
   - **Details:**
     - Retrieves data from the Model and renders it to the user.
     - Reflects the current state of the Model.
     - The View updates dynamically in response to changes in the Model.

3. **Controller:**

   - **Role:** Acts as an intermediary between the Model and the View. It receives user input, processes it (often updating the Model), and returns the output display to the View.
   - **Details:**
     - Handles user interactions and inputs.
     - Interprets user actions, such as clicks and keyboard inputs.
     - Updates the Model based on user input and then updates the View accordingly.

Advantages of MVC in Object-Oriented Design:

- **Separation of Concerns:** By dividing an application into the Model, View, and Controller, each component can be developed, tested, and maintained independently, reducing complexity and enhancing code manageability.

- **Reusability:** Components can be reused across different parts of the application or in different projects, promoting efficient code reuse.

- **Modularity:** MVC supports modular development, making it easier to manage large-scale applications. Changes in one component (e.g., the View) typically do not impact others (e.g., the Model).

- **Maintainability:** Clear separation of concerns and modularity make it easier to debug, update, and maintain the application. Developers can work on different components simultaneously without conflicts.

- **Scalability:** The modular structure of MVC applications supports scalability. As the application grows, new features and components can be added without significant changes to the existing codebase.

Application in OneSpotEducation System:

For the OneSpotEducation system, implementing the MVC pattern would involve:

- **Model:** Classes like `User`, `Course`, `LiveClass`, `Assignment`, etc., encapsulating the data and business logic.
- **View:** User interfaces (UIs) such as web pages and mobile screens that present data to students, teachers, and administrators.
- **Controller:** Classes that handle user inputs, such as `UserController`, `CourseController`, etc., which process requests, update the Model, and refresh the View.

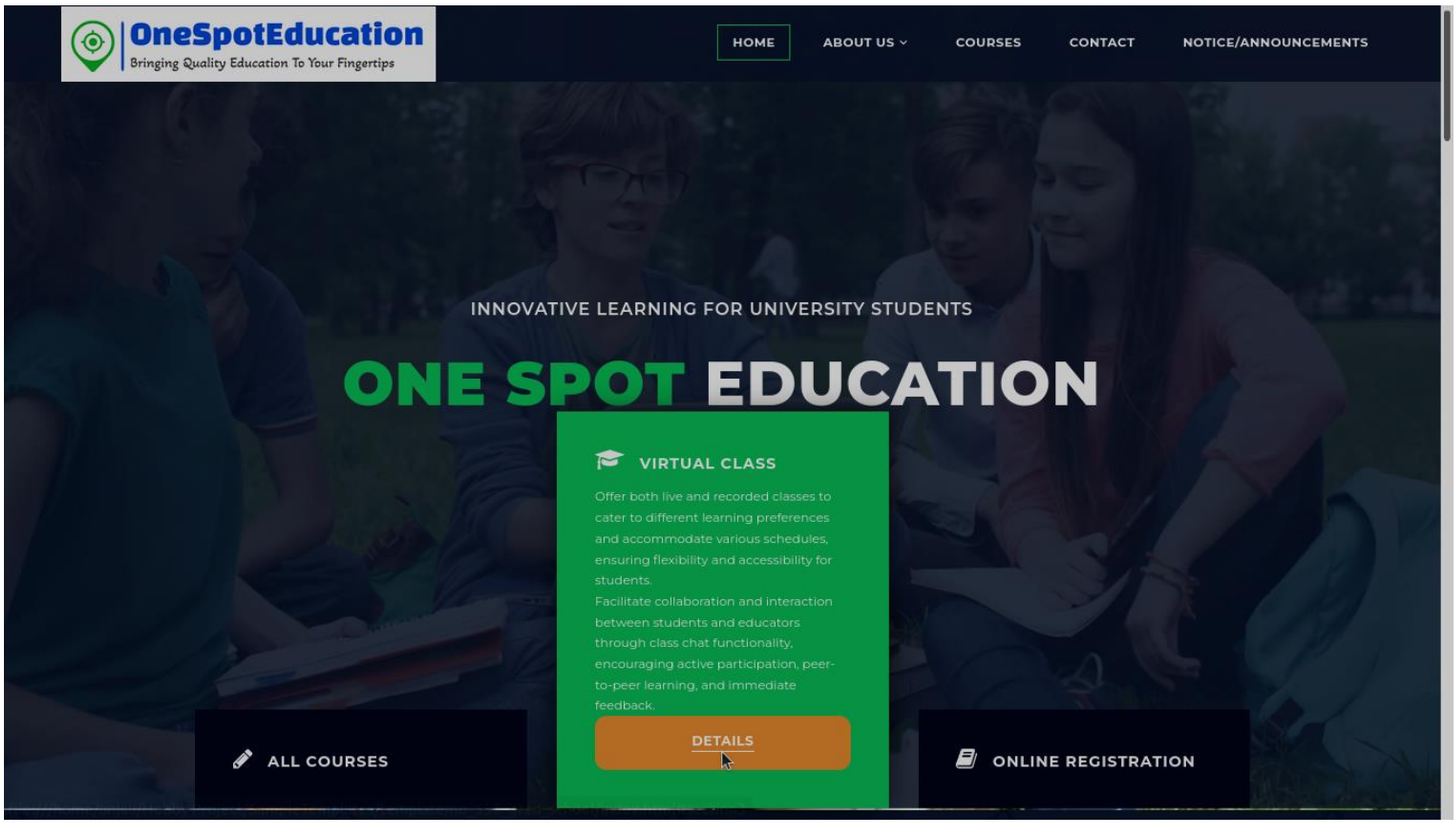
By leveraging MVC, the OneSpotEducation system ensures that the design is clean, maintainable, and scalable, ultimately providing a robust framework for managing educational content and user interactions effectively.

[4] *User Interface Design*

[1] HOME PAGE

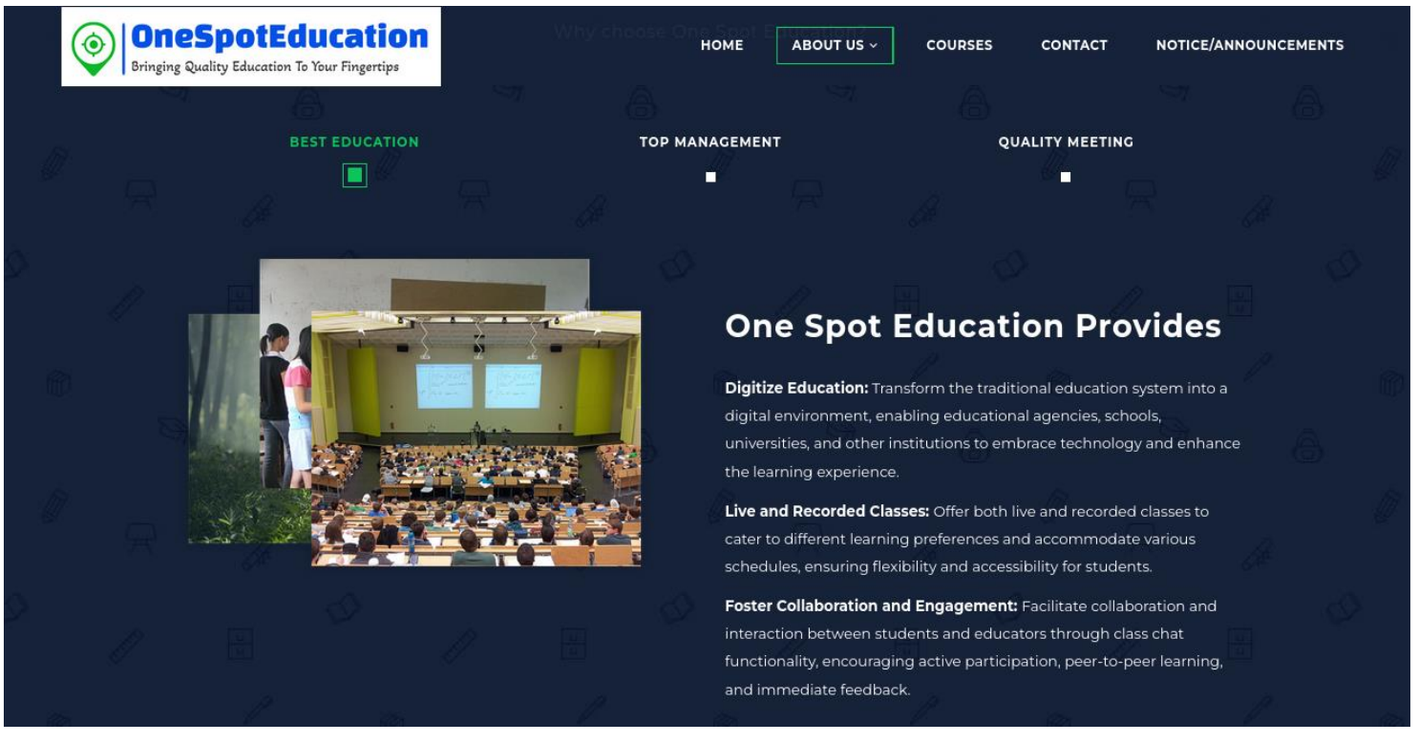
[2] HOME PAGE SECTION ONE [About all courses]

[3] HOME PAGE SECTION TWO [About virtual Classes]

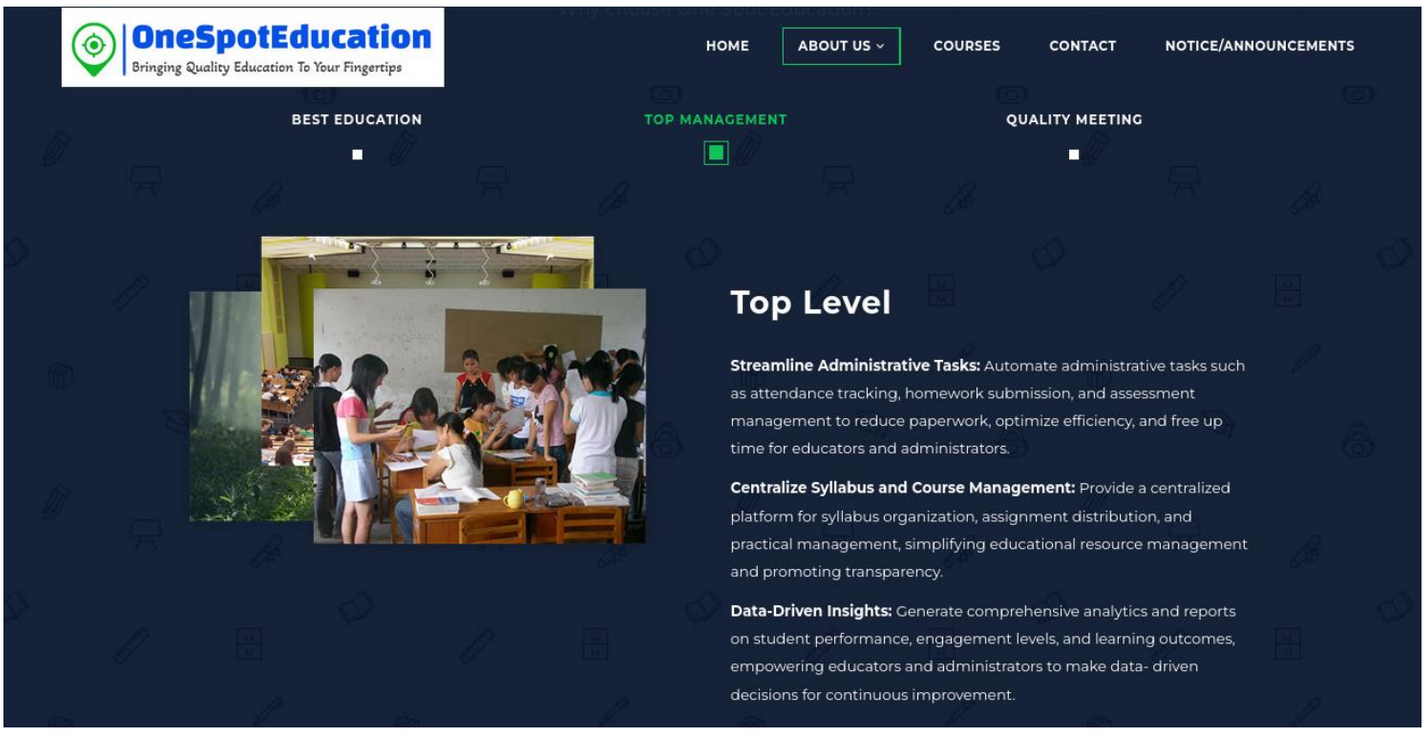


[4] HOME PAGE SECTION THREE [About Online Admission]

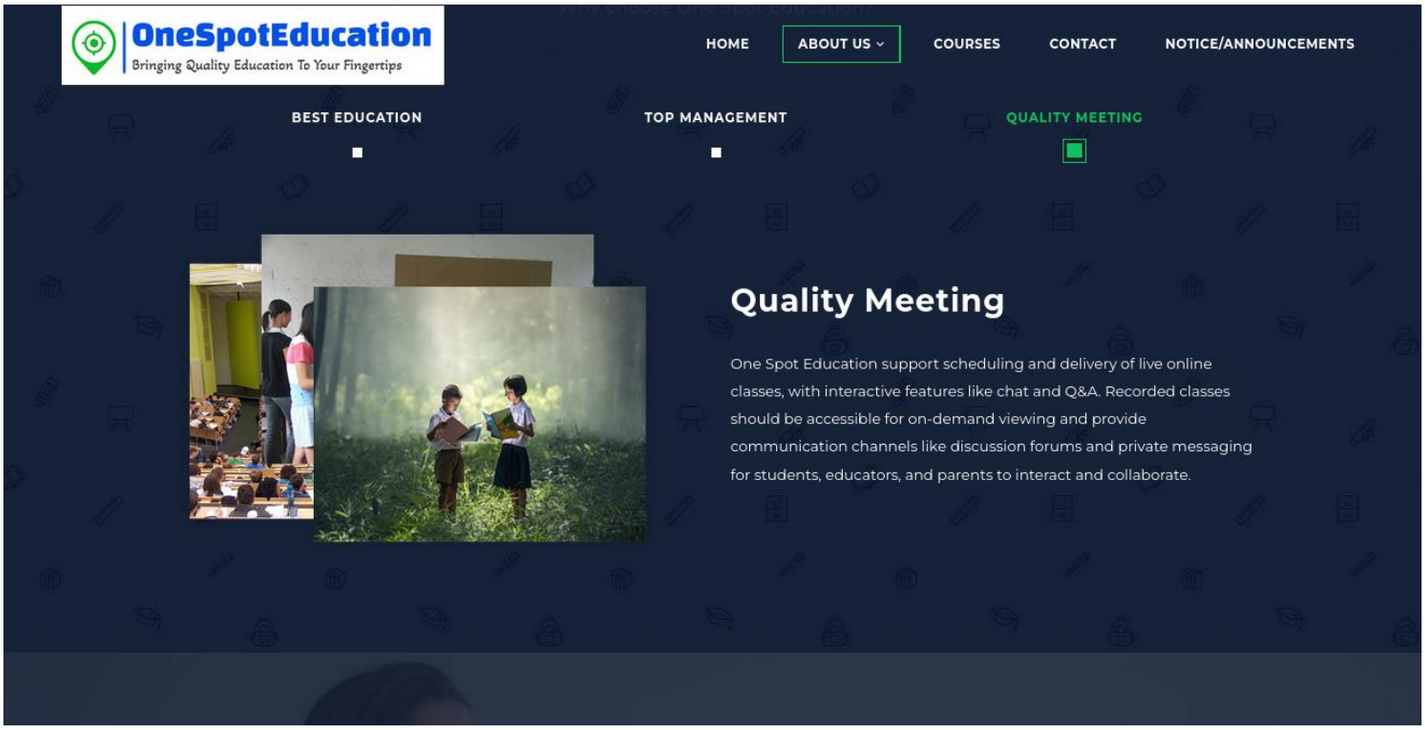
[5] OneSpotEducation Provides Best Education



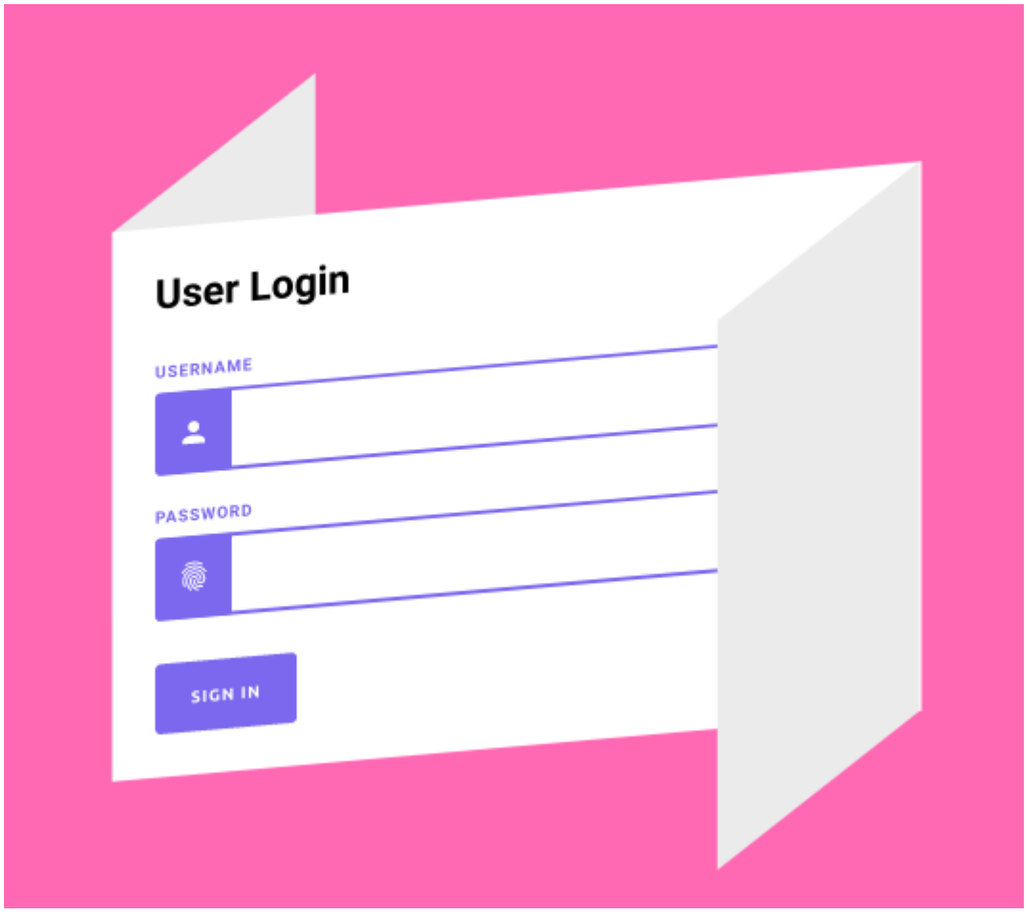
[6] Top Management



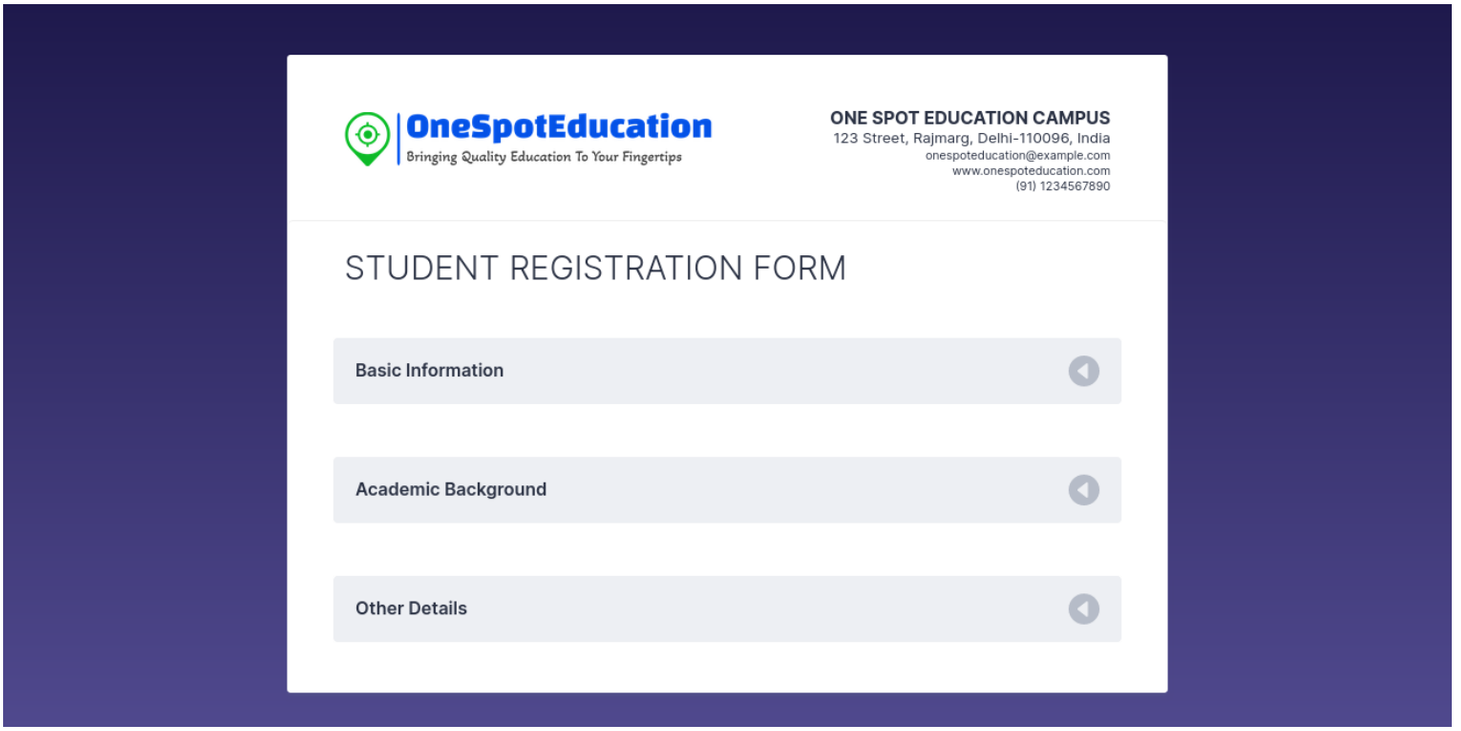
[7] Quality Meeting



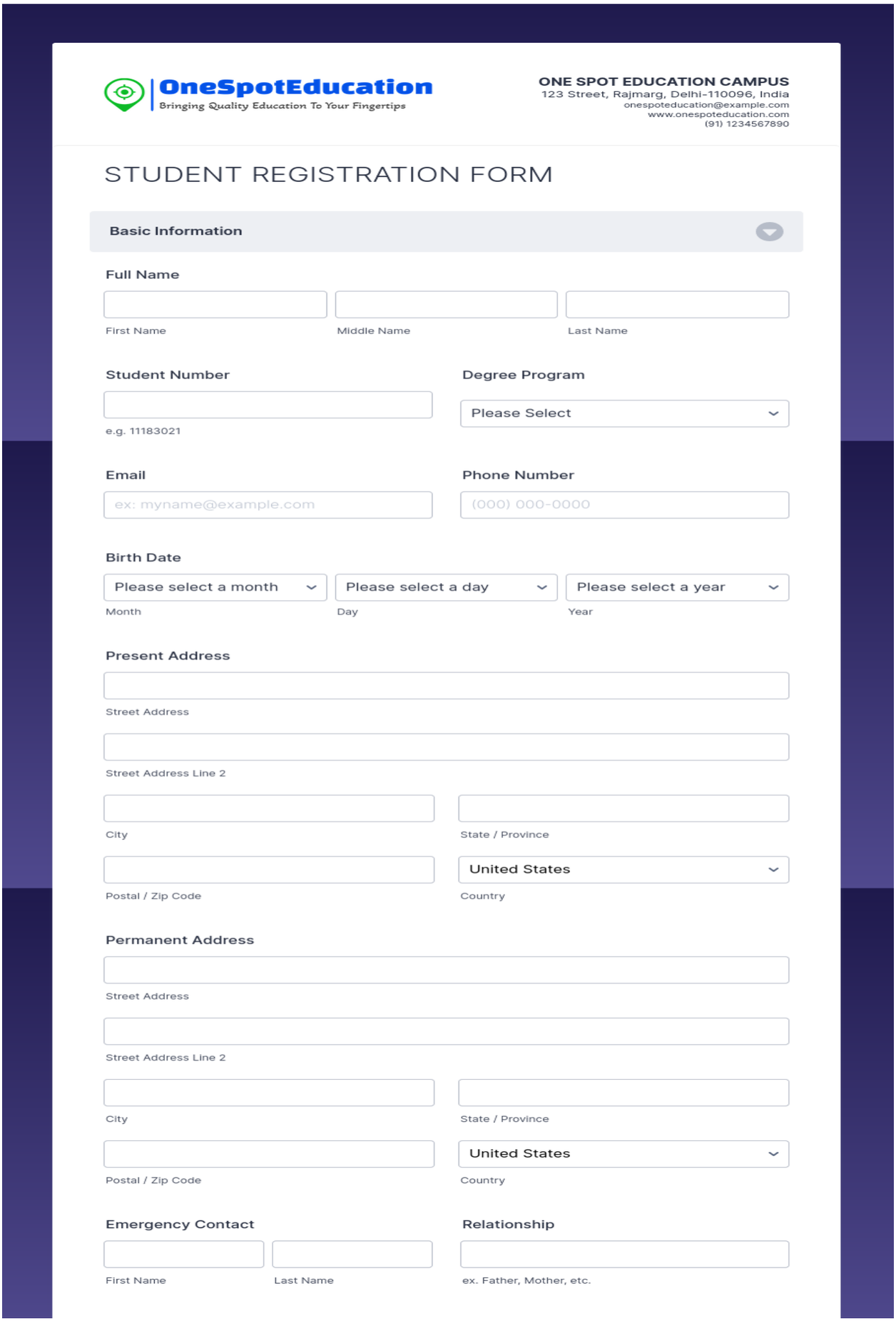
[8] User Login with unique id & password



[9] Student Online Registration Multilevel Form



[10.1] Basic Information Student Registration Form



OneSpotEducation
Bringing Quality Education To Your Fingertips

ONE SPOT EDUCATION CAMPUS
123 Street, Rajmarg, Delhi-110096, India
onespoteducation@example.com
www.onespoteducation.com
(91) 1234567890

STUDENT REGISTRATION FORM

Basic Information

Full Name
First Name | Middle Name | Last Name

Student Number
e.g. 11183021

Degree Program
Please Select

Email
ex: myname@example.com

Phone Number
(000) 000-0000

Birth Date
Please select a month (Month) | Please select a day (Day) | Please select a year (Year)

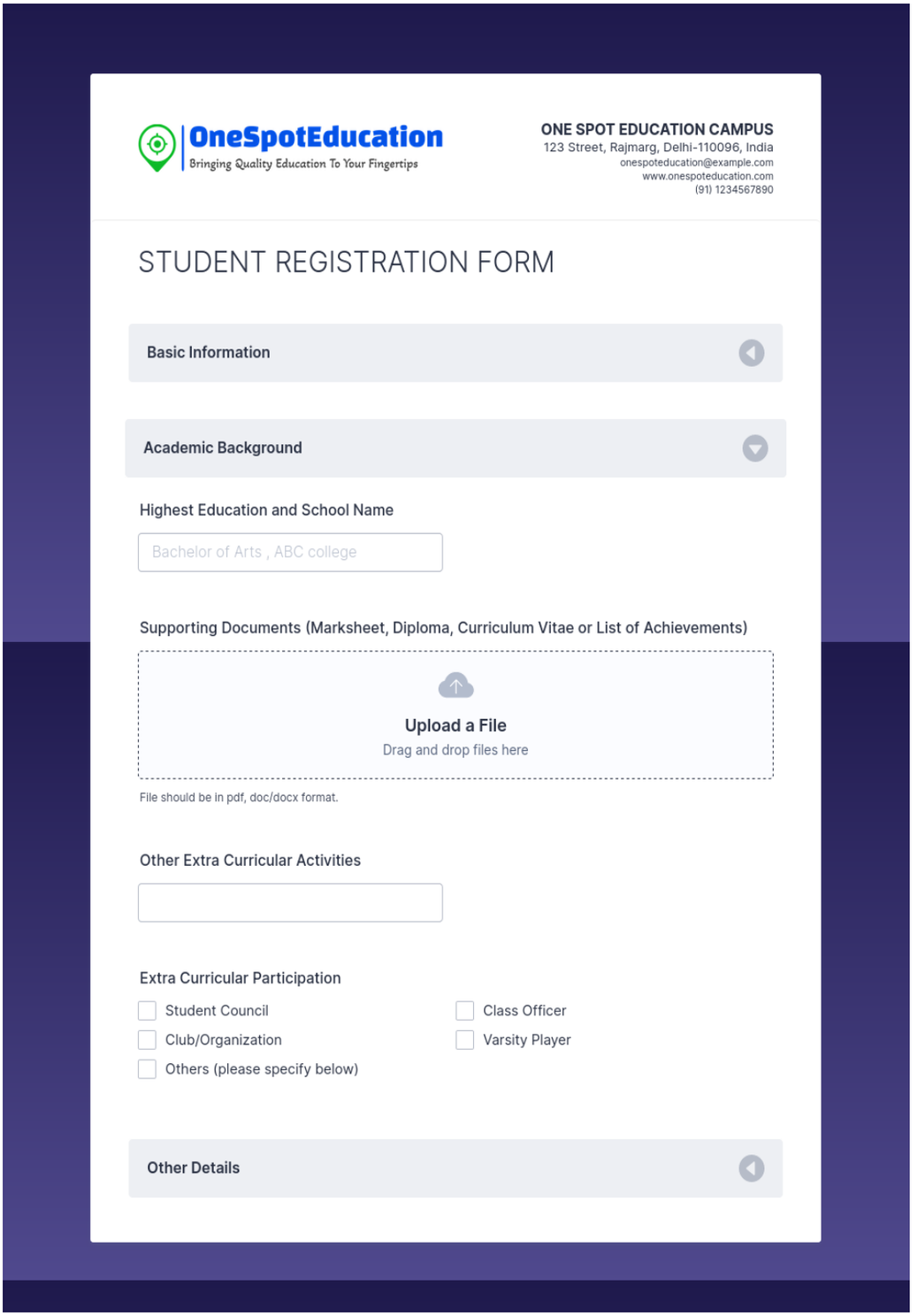
Present Address
Street Address
Street Address Line 2
City | State / Province
Postal / Zip Code | United States (Country)

Permanent Address
Street Address
Street Address Line 2
City | State / Province
Postal / Zip Code | United States (Country)

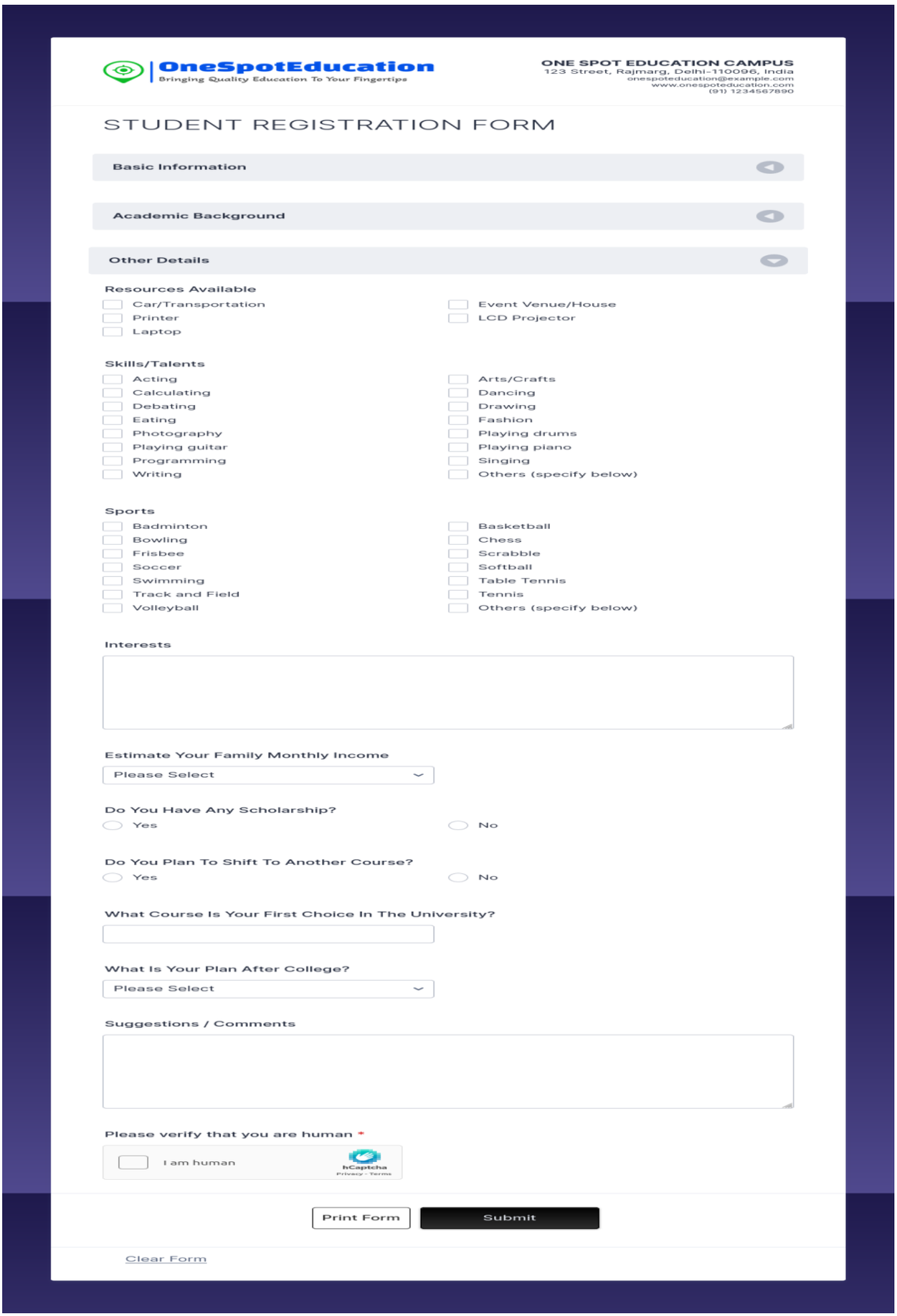
Emergency Contact
First Name | Last Name

Relationship
ex. Father, Mother, etc.

[10.2] Academic Background Student Registration Form



[10.3] Other Details in Student Registration Form



OneSpotEducation
Bringing Quality Education To Your Fingertips

ONE SPOT EDUCATION CAMPUS
123 Street, Rajmarg, Delhi-110096, India
onespoteducation@example.com
www.onespoteducation.com
(91) 1234567890

STUDENT REGISTRATION FORM

Basic Information

Academic Background

Other Details

Resources Available
- [ ] Car/Transportation
- [ ] Printer
- [ ] Laptop
- [ ] Event Venue/House
- [ ] LCD Projector

Skills/Talents
- [ ] Acting
- [ ] Calculating
- [ ] Debating
- [ ] Eating
- [ ] Photography
- [ ] Playing guitar
- [ ] Programming
- [ ] Writing
- [ ] Arts/Crafts
- [ ] Dancing
- [ ] Drawing
- [ ] Fashion
- [ ] Playing drums
- [ ] Playing piano
- [ ] Singing
- [ ] Others (specify below)

Sports
- [ ] Badminton
- [ ] Bowling
- [ ] Frisbee
- [ ] Soccer
- [ ] Swimming
- [ ] Track and Field
- [ ] Volleyball
- [ ] Basketball
- [ ] Chess
- [ ] Scrabble
- [ ] Softball
- [ ] Table Tennis
- [ ] Tennis
- [ ] Others (specify below)

Interests

Estimate Your Family Monthly Income
Please Select

Do You Have Any Scholarship?
- ( ) Yes
- ( ) No

Do You Plan To Shift To Another Course?
- ( ) Yes
- ( ) No

What Course Is Your First Choice In The University?

What Is Your Plan After College?
Please Select

Suggestions / Comments

Please verify that you are human *
- [ ] I am human (hCaptcha Privacy - Terms)

Print Form | Submit

Clear Form

[11] Daily Virtual Class Report

Daily Online Class Reporting Form

Date *

05-06-2024

Date

Faculty Name

Department

Details of Classes Engaged *

	Class	Subject	Topic	Platform	Participation	Link
Hour 1						
Hour 2						
Hour 3						
Hour 4						
Hour 5						
Tutorial						

Unexpected error occuredcontrol_textarea

Print Form Save Submit

Clear Form

[12] All Courses View

All | Master | Bachelor | Diploma | Certificate

Total 5 records are found.

Bachelor of Social Work (BSWOL)

The Bachelor Degree Programme in Social Work (BSWOL) is meant for people who are int...

Know More

Bachelor of Computer Applications (BCAOL)

The basic objective of the programme is to open a channel of admission for computing...

Know More

Bachelor of Commerce (BCOMOL)

The Bachelor of Commerce programme is a broad based programme with a mix of discipli...

Know More

Bachelor of Library and Information Sciences (BLISOL)

The programme comprises of 9 courses of which five courses (BLI 224, BLIE 225, BLIE ...

Know More

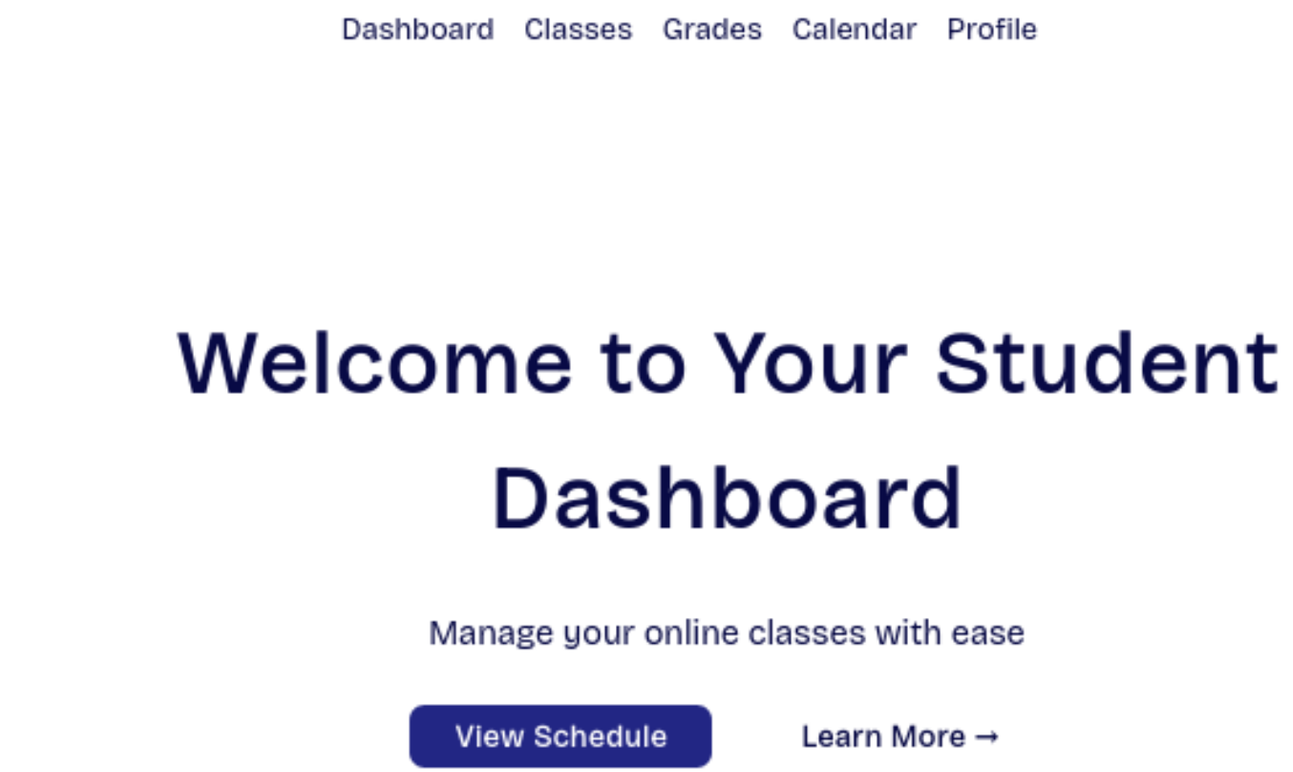
Bachelor of Tourism (BTSOL)

BTS is a 3-year Degree Programme. The programme is of 96 credits. The BTS Programme ...

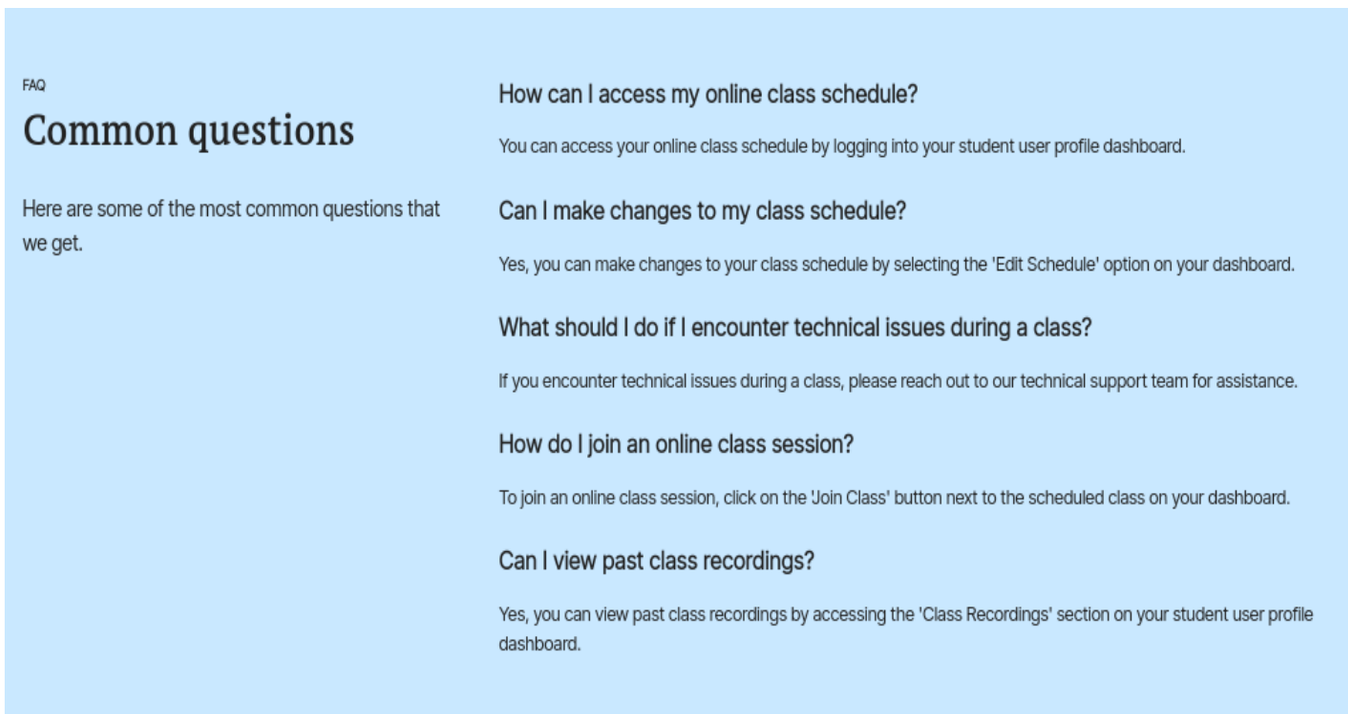
Know More

First 1

[13] Student Dashboard



[14] Student FAQs



[15] Student User Profile

[5] Test Cases
(Unit Test Cases and System Test Cases)

Unit Testing: -

Unit Testing is a software verification and validation method, where the programmer gains confidence that individual units of source codes are fit for use. A unit is the smallest testable part of an application. In procedural programming, a unit may be an individual program, function, procedure, etc., while in object-oriented programming, the smallest unit is a method, which may belong to a base/super class, abstract class or derived/child class. Unit testing can be done by something as simple as stepping through code in a debugger; modern applications include the use of a test framework such as xUnit.

Ideally, each test case is independent from the others: substitutes like method stubs, mock objects, fakes and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its requirements and behaves as intended. Its implementation can vary from being very manual (pencil and paper) to being formalized as part of build automation.

System Testing: -

System Testing (sometimes called System and Testing, abbreviated **I&T**) is the activity of software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing.

System testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an System test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

The purpose of System testing, is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using Black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test that all components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing.

4. Coding

a. Complete Project Coding, Comments & Description , Standardization Of Coding, Parameters Calling/Passing, Validation Checks:-

[1] User.php [MODEL]

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;
use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable
{
    use HasApiTokens, HasFactory, Notifiable;

    protected $fillable = [
        'username',
        'email',
        'password',
        'role_id',
    ];

    protected $hidden = [
        'password',
        'remember_token',
    ];

    public function role()
```

```
    {
        return $this->belongsTo(Role::class);
    }

    public function courses()
    {
        return $this->belongsToMany(Course::class, 'user_courses');
    }
}
```

[2] UserController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\User;
use Illuminate\Support\Facades\Hash;

class UserController extends Controller
{
    public function register(Request $request)
    {
        $validatedData = $request->validate([
            'username' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
            'role_id' => 'required|exists:roles,id',
        ]);

        $user = User::create([
            'username' => $validatedData['username'],
```

```
        'email' => $validatedData['email'],
        'password' => Hash::make($validatedData['password']),
        'role_id' => $validatedData['role_id'],
    ]);

    return response()->json(['user' => $user], 201);
}

public function login(Request $request)
{
    $credentials = $request->only('email', 'password');

    if (Auth::attempt($credentials)) {
        $user = Auth::user();
        $token = $user->createToken('auth_token')->plainTextToken;

        return response()->json(['token' => $token], 200);
    }

    return response()->json(['message' => 'Unauthorized'], 401);
}

public function getUserInfo($id)
{
    $user = User::with('role')->findOrFail($id);
    return response()->json(['user' => $user], 200);
}

public function updateUser(Request $request, $id)
{
    $user = User::findOrFail($id);
```

```
        $user->update($request->all());

        return response()->json(['user' => $user], 200);
    }

    public function deleteUser($id)
    {
        $user = User::findOrFail($id);
        $user->delete();

        return response()->json(['message' => 'User deleted'], 200);
    }
}
```

[3] Course Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Course extends Model
{
    use HasFactory;

    protected $fillable = [
        'courseName',
        'description',
    ];

    public function users()
```

```
    {
        return $this->belongsToMany(User::class, 'user_courses');
    }

    public function liveClasses()
    {
        return $this->hasMany(LiveClass::class);
    }

    public function assignments()
    {
        return $this->hasMany(Assignment::class);
    }
}
```

[4] CourseController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Course;

class CourseController extends Controller
{
    public function createCourse(Request $request)
    {
        $validatedData = $request->validate([
            'courseName' => 'required|string|max:255',
            'description' => 'required|string',
        ]);
```

```
    $course = Course::create($validatedData);

    return response()->json(['course' => $course], 201);
}

public function getCourses()
{
    $courses = Course::all();
    return response()->json(['courses' => $courses], 200);
}

public function getCourse($id)
{
    $course = Course::with(['liveClasses', 'assignments'])->findOrFail($id);
    return response()->json(['course' => $course], 200);
}

public function updateCourse(Request $request, $id)
{
    $course = Course::findOrFail($id);
    $course->update($request->all());

    return response()->json(['course' => $course], 200);
}

public function deleteCourse($id)
{
    $course = Course::findOrFail($id);
    $course->delete();

    return response()->json(['message' => 'Course deleted'], 200);
```

```
    }
}
```

[5] LiveClass Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class LiveClass extends Model
{
    use HasFactory;

    protected $fillable = [
        'course_id',
        'date',
        'startTime',
        'endTime',
    ];

    public function course()
    {
        return $this->belongsTo(Course::class);
    }
}
```

[6] LiveClassController

```
<?php

namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
use App\Models\LiveClass;

class LiveClassController extends Controller
{
    public function scheduleLiveClass(Request $request)
    {
        $validatedData = $request->validate([
            'course_id' => 'required|exists:courses,id',
            'date' => 'required|date',
            'startTime' => 'required',
            'endTime' => 'required',
        ]);

        $liveClass = LiveClass::create($validatedData);

        return response()->json(['liveClass' => $liveClass], 201);
    }

    public function getLiveClasses()
    {
        $liveClasses = LiveClass::all();
        return response()->json(['liveClasses' => $liveClasses], 200);
    }

    public function getLiveClass($id)
    {
        $liveClass = LiveClass::findOrFail($id);
        return response()->json(['liveClass' => $liveClass], 200);
    }
```

```
    public function updateLiveClass(Request $request, $id)
    {
        $liveClass = LiveClass::findOrFail($id);
        $liveClass->update($request->all());

        return response()->json(['liveClass' => $liveClass], 200);
    }

    public function deleteLiveClass($id)
    {
        $liveClass = LiveClass::findOrFail($id);
        $liveClass->delete();

        return response()->json(['message' => 'Live class deleted'], 200);
    }
}
```

[7] Assignment Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Assignment extends Model
{
    use HasFactory;

    protected $fillable = [
        'course_id',
```

```
        'title',
        'description',
        'dueDate',
    ];

    public function course()
    {
        return $this->belongsTo(Course::class);
    }

    public function users()
    {
        return $this->belongsToMany(User::class, 'user_assignments');
    }
}
```

[8] AssignmentController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Assignment;

class AssignmentController extends Controller
{
    public function createAssignment(Request $request)
    {
        $validatedData = $request->validate([
            'course_id' => 'required|exists:courses,id',
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'dueDate' => 'required|date',
        ]);

        $assignment = Assignment::create($validatedData);

        return response()->json(['assignment' => $assignment], 201);
    }

    public function getAssignments()
    {
        $assignments = Assignment::all();
        return response()->json(['assignments' => $assignments], 200);
    }
```

```
    public function getAssignment($id)
    {
        $assignment = Assignment::findOrFail($id);
        return response()->json(['assignment' => $assignment], 200);
    }

    public function updateAssignment(Request $request, $id)
    {
        $assignment = Assignment::findOrFail($id);
        $assignment->update($request->all());

        return response()->json(['assignment' => $assignment], 200);
    }

    public function deleteAssignment($id)
    {
        $assignment = Assignment::findOrFail($id);
        $assignment->delete();

        return response()->json(['message' => 'Assignment deleted'], 200);
    }
}
```

[9] Attendance Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Attendance extends Model
```

```
{
    use HasFactory;

    protected $fillable = [
        'live_class_id',
        'user_id',
        'is_present',
    ];

    public function liveClass()
    {
        return $this->belongsTo(LiveClass::class);
    }

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

[10] AttendanceController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Attendance;

class AttendanceController extends Controller
{
    public function markAttendance(Request $request)
    {
```

```
        $validatedData = $request->validate([
            'live_class_id' => 'required|exists:live_classes,id',
            'user_id' => 'required|exists:users,id',
            'is_present' => 'required|boolean',
        ]);

        $attendance = Attendance::create($validatedData);

        return response()->json(['attendance' => $attendance], 201);
    }

    public function getAttendanceByClass($live_class_id)
    {
        $attendance = Attendance::where('live_class_id', $live_class_id)->get();
        return response()->json(['attendance' => $attendance], 200);
    }

    public function getAttendanceByUser($user_id)
    {
        $attendance = Attendance::where('user_id', $user_id)->get();
        return response()->json(['attendance' => $attendance], 200);
    }
}
```

[11] Communication Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
```

```
class Communication extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'message',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

[12] CommunicationController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Communication;

class CommunicationController extends Controller
{
    public function sendMessage(Request $request)
    {
        $validatedData = $request->validate([
            'user_id' => 'required|exists:users,id',
            'message' => 'required|string',
        ]);
```

```
        $communication = Communication::create($validatedData);

        return response()->json(['communication' => $communication], 201);
    }

    public function getMessages()
    {
        $communications = Communication::all();
        return response()->json(['communications' => $communications], 200);
    }

    public function getMessage($id)
    {
        $communication = Communication::findOrFail($id);
        return response()->json(['communication' => $communication], 200);
    }
}
```

[13] Report Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Report extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
```

```
        'content',
    ];


    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

[14] ReportController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Report;

class ReportController extends Controller
{
    public function generateReport(Request $request)
    {
        $validatedData = $request->validate([
            'user_id' => 'required|exists:users,id',
            'content' => 'required|string',
        ]);

        $report = Report::create($validatedData);

        return response()->json(['report' => $report], 201);
    }

    public function getReports()
    {
        $reports = Report::all();
        return response()->json(['reports' => $reports], 200);
    }

    public function getReport($id)
    {
```

```
        $report = Report::findOrFail($id);
        return response()->json(['report' => $report], 200);
    }
}
```

[15] Role Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Role extends Model
{
    use HasFactory;

    protected $fillable = [
        'role_name',
    ];

    public function users()
    {
        return $this->hasMany(User::class);
    }
}
```

[16] RoleController

```
<?php

namespace App\Http\Controllers;
```

```
use Illuminate\Http\Request;
use App\Models\Role;

class RoleController extends Controller
{
    public function createRole(Request $request)
    {
        $validatedData = $request->validate([
            'role_name' => 'required|string|max:255|unique:roles',
        ]);

        $role = Role::create($validatedData);

        return response()->json(['role' => $role], 201);
    }

    public function getRoles()
    {
        $roles = Role::all();
        return response()->json(['roles' => $roles], 200);
    }

    public function getRole($id)
    {
        $role = Role::findOrFail($id);
        return response()->json(['role' => $role], 200);
    }

    public function updateRole(Request $request, $id)
    {
        $role = Role::findOrFail($id);
```

```
        $role->update($request->all());

        return response()->json(['role' => $role], 200);
    }

    public function deleteRole($id)
    {
        $role = Role::findOrFail($id);
        $role->delete();

        return response()->json(['message' => 'Role deleted'], 200);
    }
}
```

[17] System Configuration Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class SystemConfig extends Model
{
    use HasFactory;

    protected $fillable = [
        'config_name',
        'config_value',
    ];
}
```

[18] SystemConfigController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\SystemConfig;

class SystemConfigController extends Controller
{
    public function setConfig(Request $request)
    {
        $validatedData = $request->validate([
            'config_name' => 'required|string|max:255|unique:system_configs',
            'config_value' => 'required|string',
        ]);

        $config = SystemConfig::create($validatedData);

        return response()->json(['config' => $config], 201);
    }

    public function getConfig($config_name)
    {
        $config = SystemConfig::where('config_name', $config_name)->firstOrFail();
        return response()->json(['config' => $config], 200);
    }

    public function updateConfig(Request $request, $config_name)
    {
        $config = SystemConfig::where('config_name', $config_name)->firstOrFail();
```

```
        $config->update($request->all());

        return response()->json(['config' => $config], 200);
    }

    public function deleteConfig($config_name)
    {
        $config = SystemConfig::where('config_name', $config_name)->firstOrFail();
        $config->delete();

        return response()->json(['message' => 'Config deleted'], 200);
    }
}
```

[19] Activity Calendar Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class ActivityCalendar extends Model
{
    use HasFactory;

    protected $fillable = [
        'title',
        'description',
        'start_date',
        'end_date',
        'user_id',
```

[20] ActivityCalendarController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\ActivityCalendar;

class ActivityCalendarController extends Controller
{
    public function createActivity(Request $request)
    {
        $validatedData = $request->validate([
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'start_date' => 'required|date',
            'end_date' => 'required|date|after_or_equal:start_date',
            'user_id' => 'required|exists:users,id',
        ]);

        $activity = ActivityCalendar::create($validatedData);

        return response()->json(['activity' => $activity], 201);
    }

    public function getActivities()
    {
        $activities = ActivityCalendar::all();
        return response()->json(['activities' => $activities], 200);
    }
```

```
    public function getActivity($id)
    {
        $activity = ActivityCalendar::findOrFail($id);
        return response()->json(['activity' => $activity], 200);
    }

    public function updateActivity(Request $request, $id)
    {
        $activity = ActivityCalendar::findOrFail($id);
        $activity->update($request->all());

        return response()->json(['activity' => $activity], 200);
    }

    public function deleteActivity($id)
    {
        $activity = ActivityCalendar::findOrFail($id);
        $activity->delete();

        return response()->json(['message' => 'Activity deleted'], 200);
    }
}
```

[21] Notification Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
```

```
class Notification extends Model
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'message',
        'read_at',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

[22] NotificationController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\Notification;

class NotificationController extends Controller
{
    public function sendNotification(Request $request)
    {
        $validatedData = $request->validate([
            'user_id' => 'required|exists:users,id',
            'message' => 'required|string',
```

```
        ]);

        $notification = Notification::create($validatedData);

        return response()->json(['notification' => $notification], 201);
    }

    public function getNotifications($user_id)
    {
        $notifications = Notification::where('user_id', $user_id)->get();
        return response()->json(['notifications' => $notifications], 200);
    }

    public function markAsRead($id)
    {
        $notification = Notification::findOrFail($id);
        $notification->read_at = now();
        $notification->save();

        return response()->json(['notification' => $notification], 200);
    }
}
```

[23] Feedback Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Feedback extends Model
```

```
{
    use HasFactory;

    protected $fillable = [
        'user_id',
        'course_id',
        'rating',
        'comment',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }

    public function course()
    {
        return $this->belongsTo(Course::class);
    }
}
```

[24] UserProfile Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class UserProfile extends Model
{
```

```
    use HasFactory;

    protected $fillable = [
        'user_id',
        'first_name',
        'last_name',
        'dob',
        'address',
        'phone',
        'profile_picture',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

[25] UserProfileController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\UserProfile;

class UserProfileController extends Controller
{
    public function updateProfile(Request $request, $user_id)
    {
        $validatedData = $request->validate([
            'first_name' => 'required|string|max:255',
```

```
            'last_name' => 'required|string|max:255',
            'dob' => 'required|date',
            'address' => 'required|string',
            'phone' => 'required|string|max:15',
            'profile_picture' => 'nullable|image|max:2048',
        ]);

        $profile = UserProfile::where('user_id', $user_id)->firstOrFail();
        $profile->update($validatedData);

        return response()->json(['profile' => $profile], 200);
    }

    public function getProfile($user_id)
    {
        $profile = UserProfile::where('user_id', $user_id)->firstOrFail();
        return response()->json(['profile' => $profile], 200);
    }
}
```

[26] FileUpload Model

```
<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class FileUpload extends Model
{
    use HasFactory;
```

```
    protected $fillable = [
        'user_id',
        'filename',
        'path',
    ];

    public function user()
    {
        return $this->belongsTo(User::class);
    }
}
```

[27] FileUploadController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use App\Models\FileUpload;
use Illuminate\Support\Facades\Storage;

class FileUploadController extends Controller
{
    public function uploadFile(Request $request)
    {
        $validatedData = $request->validate([
            'user_id' => 'required|exists:users,id',
            'file' => 'required|file|max:10240',
        ]);

        $path = $request->file('file')->store('uploads');

        $fileUpload = FileUpload::create([
            'user_id' => $request->user_id,
            'filename' => $request->file('file')->getClientOriginalName(),
            'path' => $path,
        ]);

        return response()->json(['file' => $fileUpload], 201);
    }

    public function getUserFiles($user_id)
```

```
    {
        $files = FileUpload::where('user_id', $user_id)->get();
        return response()->json(['files' => $files], 200);
    }

    public function downloadFile($id)
    {
        $file = FileUpload::findOrFail($id);
        return Storage::download($file->path, $file->filename);
    }

    public function deleteFile($id)
    {
        $file = FileUpload::findOrFail($id);
        Storage::delete($file->path);
        $file->delete();

        return response()->json(['message' => 'File deleted'], 200);
    }
}
```

CLINET SITE CODING:-

[1] Master layout file

File Path: resources/views/layouts/app.blade.php)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>OneSpotEducation</title>
    <link rel="stylesheet" href="{{ asset('css/app.css') }}">
    <script src="{{ asset('js/app.js') }}" defer></script>
</head>
<body>
    <div id="app">
        @include('partials.navbar')

        <main class="py-4">
            @yield('content')
        </main>
    </div>
</body>
</html>
```

[2] Navbar Partial

File Path: resources/views/partials/navbar.blade.php

```
<nav class="navbar navbar-expand-md navbar-light bg-white shadow-sm">
  <div class="container">
    <a class="navbar-brand" href="{{ url('/') }}">
      OneSpotEducation
    </a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="{{ __('Toggle navigation') }}">
```

```
    <span class="navbar-toggler-icon"></span>
</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">
    <!-- Left Side Of Navbar -->
    <ul class="navbar-nav mr-auto">
        <li class="nav-item">
            <a class="nav-link" href="{{ route('courses.index') }}">Courses</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{ route('live-classes.index') }}">Live Classes</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="{{ route('assignments.index') }}">Assignments</a>
        </li>
    </ul>

    <!-- Right Side Of Navbar -->
    <ul class="navbar-nav ml-auto">
        <!-- Authentication Links -->
        @guest
            <li class="nav-item">
                <a class="nav-link" href="{{ route('login') }}">{{ __('Login') }}</a>
            </li>
            @if (Route::has('register'))
                <li class="nav-item">
                    <a class="nav-link" href="{{ route('register') }}">{{ __('Register') }}</a>
                </li>
            @endif
        @else
            <li class="nav-item dropdown">
```

```
                <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button" data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false" v-pre>
                    {{ Auth::user()->name }} <span class="caret"></span>
                </a>

                <div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">
                    <a class="dropdown-item" href="{{ route('logout') }}"
                       onclick="event.preventDefault();
                                     document.getElementById('logout-form').submit();">
                        {{ __('Logout') }}
                    </a>

                    <form id="logout-form" action="{{ route('logout') }}" method="POST" class="d-none">
                        @csrf
                    </form>
                </div>
            </li>
        @endguest
    </ul>
  </div>
 </div>
</nav>
```

[3] Home Page View

File Path: `resources/views/home.blade.php`

```
@extends('layouts.app')

@section('content')
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-8">
      <div class="card">
```

```
            <div class="card-header">{{ __('Dashboard') }}</div>

            <div class="card-body">
                @if (session('status'))
                    <div class="alert alert-success" role="alert">
                        {{ session('status') }}
                    </div>
                @endif

                {{ __('You are logged in!') }}
            </div>
        </div>
    </div>
  </div>
</div>
@endsection
```

[4] Courses Index View

File Path: `resources/views/courses/index.blade.php`

```
@extends('layouts.app')

@section('content')
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-8">
      <div class="card">
        <div class="card-header">{{ __('Courses') }}</div>

        <div class="card-body">
          @if (session('status'))
            <div class="alert alert-success" role="alert">
              {{ session('status') }}
```

```
    </div>
@endif

<a href="{{ route('courses.create') }}" class="btn btn-primary">Add New Course</a>
<table class="table mt-4">
    <thead>
        <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Description</th>
            <th>Actions</th>
        </tr>
    </thead>
    <tbody>
        @foreach ($courses as $course)
            <tr>
                <td>{{ $course->id }}</td>
                <td>{{ $course->name }}</td>
                <td>{{ $course->description }}</td>
                <td>
                    <a href="{{ route('courses.edit', $course->id) }}" class="btn btn-warning">Edit</a>
                    <form action="{{ route('courses.destroy', $course->id) }}" method="POST" class="d-inline">
                        @csrf
                        @method('DELETE')
                        <button type="submit" class="btn btn-danger">Delete</button>
                    </form>
                </td>
            </tr>
        @endforeach
    </tbody>
</table>
```

```
            </div>
        </div>
    </div>
</div>
</div>
@endsection
```

[5] Live Classes Index View

File Path: `resources/views/live-classes/index.blade.php`

```
@extends('layouts.app')


@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Live Classes') }}</div>

                <div class="card-body">
                    @if (session('status'))
                        <div class="alert alert-success" role="alert">
                            {{ session('status') }}
                        </div>
                    @endif

                    <a href="{{ route('live-classes.create') }}" class="btn btn-primary">Schedule New Live Class</a>
                    <table class="table mt-4">
                        <thead>
                            <tr>
                                <th>ID</th>
                                <th>Course</th>
                                <th>Date</th>
```

```
                    <th>Time</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                @foreach ($liveClasses as $liveClass)
                    <tr>
                        <td>{{ $liveClass->id }}</td>
                        <td>{{ $liveClass->course->name }}</td>
                        <td>{{ $liveClass->date }}</td>
                        <td>{{ $liveClass->start_time }} - {{ $liveClass->end_time }}</td>
                        <td>
                            <a href="{{ route('live-classes.edit', $liveClass->id) }}" class="btn btn-warning">Edit</a>
                            <form action="{{ route('live-classes.destroy', $liveClass->id) }}" method="POST" class="d-inline">
                                @csrf
                                @method('DELETE')
                                <button type="submit" class="btn btn-danger">Delete</button>
                            </form>
                        </td>
                    </tr>
                @endforeach
            </tbody>
        </table>
    </div>
</div>
</div>
</div>
@endsection
```

[6] Assignments Index View

File Path: resources/views/assignments/index.blade.php

```
@extends('layouts.app')


@section('content')
<div class="container">
   <div class="row justify-content-center">
      <div class="col-md-8">
         <div class="card">
            <div class="card-header">{{ __('Assignments') }}</div>


            <div class="card-body">
               @if (session('status'))
                  <div class="alert alert-success" role="alert">
                     {{ session('status') }}
                  </div>
               @endif


               <a href="{{ route('assignments.create') }}" class="btn btn-primary">Create New Assignment</a>
               <table class="table mt-4">
                  <thead>
                     <tr>
                        <th>ID</th>
                        <th>Course</th>
                        <th>Title</th>
                        <th>Description</th>
                        <th>Actions</th>
                     </tr>
                  </thead>
                  <tbody>
```

```
                @foreach ($assignments as $assignment)
                  <tr>
                    <td>{{ $assignment->id }}</td>
                    <td>{{ $assignment->course->name }}</td>
                    <td>{{ $assignment->title }}</td>
                    <td>{{ $assignment->description }}</td>
                    <td>
                      <a href="{{ route('assignments.edit', $assignment->id) }}" class="btn btn-warning">Edit</a>
                      <form action="{{ route('assignments.destroy', $assignment->id) }}" method="POST" class="d-inline">
                        @csrf
                        @method('DELETE')
                        <button type="submit" class="btn btn-danger">Delete</button>
                      </form>
                    </td>
                  </tr>
                @endforeach
              </tbody>
            </table>
          </div>
        </div>
      </div>
    </div>
</div>
@endsection
```

[7] User Profile View

File Path: `resources/views/profile/index.blade.php`

```
@extends('layouts.app')

@section('content')
<div class="container">
  <div class="row justify-content-center">
```

```
<div class="col-md-8">
  <div class="card">
    <div class="card-header">{{ __('User Profile') }}</div>

    <div class="card-body">
      @if (session('status'))
        <div class="alert alert-success" role="alert">
          {{ session('status') }}
        </div>
      @endif

      <form action="{{ route('profile.update', $user->id) }}" method="POST" enctype="multipart/form-data">
        @csrf
        @method('PUT')

        <div class="form-group">
          <label for="first_name">First Name</label>
          <input type="text" class="form-control" id="first_name" name="first_name" value="{{ $user->profile->first_name }}">
        </div>

        <div class="form-group">
          <label for="last_name">Last Name</label>
          <input type="text" class="form-control" id="last_name" name="last_name" value="{{ $user->profile->last_name }}">
        </div>

        <div class="form-group">
          <label for="dob">Date of Birth</label>
          <input type="date" class="form-control" id="dob" name="dob" value="{{ $user->profile->dob }}">
        </div>
```

```
                <div class="form-group">
                    <label for="address">Address</label>
                    <input type="text" class="form-control" id="address" name="address" value="{{ $user->profile->address }}">
                </div>

                <div class="form-group">
                    <label for="phone">Phone</label>
                    <input type="text" class="form-control" id="phone" name="phone" value="{{ $user->profile->phone }}">
                </div>

                <div class="form-group">
                    <label for="profile_picture">Profile Picture</label>
                    <input type="file" class="form-control-file" id="profile_picture" name="profile_picture">
                </div>

                <button type="submit" class="btn btn-primary">Update Profile</button>
            </form>
        </div>
    </div>
  </div>
</div>
@endsection
```

[8] Course Create/Edit View

File Path: `resources/views/courses/form.blade.php`

```
@extends('layouts.app')

@section('content')
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-8">
```

```
        <div class="card">
            <div class="card-header">{{ isset($course) ? 'Edit Course' : 'Create Course' }}</div>

            <div class="card-body">
                @if (isset($course))
                    <form action="{{ route('courses.update', $course->id) }}" method="POST">
                    @method('PUT')
                @else
                    <form action="{{ route('courses.store') }}" method="POST">
                @endif
                    @csrf

                    <div class="form-group">
                        <label for="name">Course Name</label>
                        <input type="text" class="form-control" id="name" name="name" value="{{ old('name', isset($course) ? $course->name : '') }}">
                    </div>

                    <div class="form-group">
                        <label for="description">Description</label>
                        <textarea class="form-control" id="description" name="description">{{ old('description', isset($course) ? $course->description : '') }}</textarea>
                    </div>

                    <button type="submit" class="btn btn-primary">{{ isset($course) ? 'Update Course' : 'Create Course' }}</button>
                </form>
            </div>
        </div>
    </div>
  </div>
</div>
```

@endsection

[9] Live Class Create/Edit View

File Path: resources/views/live-classes/form.blade.php

```
@extends('layouts.app')


@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ isset($liveClass) ? 'Edit Live Class' : 'Schedule Live Class' }}</div>

                <div class="card-body">
                    @if (isset($liveClass))
                        <form action="{{ route('live-classes.update', $liveClass->id) }}" method="POST">
                        @method('PUT')
                    @else
                        <form action="{{ route('live-classes.store') }}" method="POST">
                    @endif
                        @csrf

                        <div class="form-group">
                            <label for="course_id">Course</label>
                            <select class="form-control" id="course_id" name="course_id">
                                @foreach($courses as $course)
                                    <option value="{{ $course->id }}" {{ isset($liveClass) && $liveClass->course_id == $course->id ? 'selected'
: '' }}>{{ $course->name }}</option>
                                @endforeach
                            </select>
                        </div>
```

```
                <div class="form-group">
                    <label for="date">Date</label>
                    <input type="date" class="form-control" id="date" name="date" value="{{ old('date', isset($liveClass) ? $liveClass->date : '') }}">
                </div>

                <div class="form-group">
                    <label for="start_time">Start Time</label>
                    <input type="time" class="form-control" id="start_time" name="start_time" value="{{ old('start_time', isset($liveClass) ? $liveClass->start_time : '') }}">
                </div>

                <div class="form-group">
                    <label for="end_time">End Time</label>
                    <input type="time" class="form-control" id="end_time" name="end_time" value="{{ old('end_time', isset($liveClass) ? $liveClass->end_time : '') }}">
                </div>

                <button type="submit" class="btn btn-primary">{{ isset($liveClass) ? 'Update Live Class' : 'Schedule Live Class' }}</button>
            </form>
        </div>
    </div>
</div>
</div>
@endsection
```

[10] Assignment Create/Edit View

File Path: `resources/views/assignments/form.blade.php`

```
@extends('layouts.app')

@section('content')
```

```
<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-8">
      <div class="card">
        <div class="card-header">{{ isset($assignment) ? 'Edit Assignment' : 'Create Assignment' }}</div>

        <div class="card-body">
          @if (isset($assignment))
            <form action="{{ route('assignments.update', $assignment->id) }}" method="POST">
            @method('PUT')
          @else
            <form action="{{ route('assignments.store') }}" method="POST">
          @endif
            @csrf

            <div class="form-group">
              <label for="course_id">Course</label>
              <select class="form-control" id="course_id" name="course_id">
                @foreach($courses as $course)
                  <option value="{{ $course->id }}" {{ isset($assignment) && $assignment->course_id == $course->id ?
'selected' : '' }}>{{ $course->name }}</option>
                @endforeach
              </select>
            </div>

            <div class="form-group">
              <label for="title">Title</label>
              <input type="text" class="form-control" id="title" name="title" value="{{ old('title', isset($assignment) ?
$assignment->title : '') }}">
            </div>

            <div class="form-group">
```

```
                    <label for="description">Description</label>

                    <textarea class="form-control" id="description" name="description">{{ old('description', isset($assignment) ? $assignment->description : '') }}</textarea>

                </div>


                <button type="submit" class="btn btn-primary">{{ isset($assignment) ? 'Update Assignment' : 'Create Assignment' }}</button>

            </form>

        </div>

    </div>

</div>

</div>

@endsection
```

[11] Calendar View

File Path: resources/views/calendar/index.blade.php

```
@extends('layouts.app')


@section('content')
<div class="container">
   <div class="row justify-content-center">
      <div class="col-md-12">
         <div class="card">
            <div class="card-header">{{ __('Activity Calendar') }}</div>


            <div class="card-body">
               <div id="calendar"></div>
            </div>
         </div>
      </div>
   </div>
</div>


@section('scripts')
<script>
   document.addEventListener('DOMContentLoaded', function() {
      var calendarEl = document.getElementById('calendar');


      var calendar = new FullCalendar.Calendar(calendarEl, {
         initialView: 'dayGridMonth',
         events: @json($events)
      });


      calendar.render();
```

```
    });
</script>
@endsection
@endsection
```

[12] Admin Dashboard View

File Path: `resources/views/admin/dashboard.blade.php`

```
@extends('layouts.app')


@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Admin Dashboard') }}</div>


                <div class="card-body">
                    <a href="{{ route('users.index') }}" class="btn btn-primary">Manage Users</a>
                    <a href="{{ route('courses.index') }}" class="btn btn-primary">Manage Courses</a>
                    <a href="{{ route('live-classes.index') }}" class="btn btn-primary">Manage Live Classes</a>
                    <a href="{{ route('assignments.index') }}" class="btn btn-primary">Manage Assignments</a>
                    <a href="{{ route('calendar.index') }}" class="btn btn-primary">Manage Calendar</a>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

[13] User Management View

File Path: resources/views/admin/users/index.blade.php

```
@extends('layouts.app')


@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Manage Users') }}</div>


                <div class="card-body">
                    @if (session('status'))
                        <div class="alert alert-success" role="alert">
                            {{ session('status') }}
                        </div>
                    @endif


                    <table class="table mt-4">
                        <thead>
                            <tr>
                                <th>ID</th>
                                <th>Name</th>
                                <th>Email</th>
                                <th>Actions</th>
                            </tr>
                        </thead>
                        <tbody>
                            @foreach ($users as $user)
                                <tr>
```

```
                    <td>{{ $user->id }}</td>
                    <td>{{ $user->name }}</td>
                    <td>{{ $user->email }}</td>
                    <td>
                      <a href="{{ route('users.edit', $user->id) }}" class="btn btn-warning">Edit</a>
                      <form action="{{ route('users.destroy', $user->id) }}" method="POST" class="d-inline">
                        @csrf
                        @method('DELETE')
                        <button type="submit" class="btn btn-danger">Delete</button>
                      </form>
                    </td>
                  </tr>
                @endforeach
              </tbody>
            </table>

            <a href="{{ route('users.create') }}" class="btn btn-primary mt-3">Create User</a>
          </div>
        </div>
      </div>
    </div>
</div>
@endsection
```

5. Standardization of the coding

Standardization of the Coding for OneSpotEducation Project

In any software development project, maintaining coding standards is crucial to ensure the quality, reliability, and maintainability of the code. This is particularly important in a student project like OneSpotEducation, where learning best practices can set the foundation for future professional development. Here's a detailed note on various aspects of coding standardization relevant to our project:

Code Efficiency

Definition: Code efficiency refers to how well the code optimizes resource usage, such as memory and CPU, while achieving its intended functionality.

Application in OneSpotEducation:

- **Optimized Queries:** Ensuring database queries are efficient, avoiding unnecessary data retrieval, and utilizing indexing where appropriate. For example, using Eloquent ORM's eager loading in Laravel to minimize the number of queries.
- **Code Optimization:** Writing concise and clear code that eliminates redundancy. Using loops, conditional statements, and functions effectively to reduce the overall complexity and execution time.
- **Resource Management:** Proper management of resources like memory and processing power. For instance, ensuring that large data sets are processed in chunks rather than loading everything into memory at once.

Error Handling

Definition: Error handling is the process of responding to and managing errors that occur during the execution of a program.

Application in OneSpotEducation:

- **Try-Catch Blocks:** Utilizing try-catch blocks to gracefully handle exceptions and prevent the application from crashing. For instance, catching database connection errors and providing meaningful messages to the user.
- **Custom Error Pages:** Creating custom error pages to enhance user experience during unexpected failures. For example, showing a user-friendly error page when a 404 error occurs.
- **Logging:** Implementing logging mechanisms to record errors and track their occurrences. This helps in debugging and improving the application. Laravel's built-in logging can be used for this purpose.

Parameters Calling/Passing

Definition: Parameters calling/passing refers to how arguments are passed to functions and methods within the code.

Application in OneSpotEducation:

- **Consistent Parameter Passing:** Ensuring that parameters are passed consistently across functions and methods. This includes using named parameters for clarity and maintaining the order of parameters.

- **Default Values:** Providing default values for optional parameters to ensure functions can be called with varying numbers of arguments without causing errors.

Validation Checks

Definition: Validation checks are the processes of verifying that the data being processed meets the required criteria before performing any operations on it.

Application in OneSpotEducation:

- **Form Validation:** Implementing robust validation rules for user input forms to prevent invalid data from being submitted. Laravel's validation features can be utilized to define rules for various fields.
- **Backend Validation:** Ensuring that data is validated not only on the frontend but also on the backend to prevent malicious data from being processed. This adds an additional layer of security.
- **Sanitization:** Sanitizing input data to remove any potentially harmful content. For instance, stripping HTML tags from user input to prevent XSS (Cross-Site Scripting) attacks.

By adhering to these coding standards in the OneSpotEducation project, we can ensure that the codebase is efficient, reliable, and maintainable. These practices will not only improve the quality of the current project but also provide a solid foundation for future projects.

[6] Testing

a. Principles of Testing:-

Testing is a comprehensive organized effort to exercise all aspects of a software system for possible failure.

- Successful testing is not a proof of correctness (unless we try every input exhaustively), but it can provide useful reassurance.
- Design test cases, Testing should not be **"random."** When the number of possible inputs is small, try them Test all **"branches"** of the code. For example, be sure to include test cases that go through the if part as well as the else part of a conditional statement. Test boundary cases, like 0, 1, -1, etc.
- Look for ways to **"break"** the program, and try those to be sure they don't.
- Know what the expected results of our tests should be. Otherwise, we might not recognize an incorrect result.
- Always be aggressive about testing. If we test gingerly, we won't uncover as many errors. Remember, the objective in testing is to **find** errors.
- Save our test program (and the expected results), so we can run it again later. Whenever modifying a program to fix one problem, there is a risk that we might have caused a different problem, so it's best to repeat all the tests.

Program Errors: -

Compilation Errors:

Syntax errors -- The compiler cannot understand our program because it does not follow the **syntax** (the rules for a valid program). Common syntax errors include:

Missing or misplaced; or },
Missing return type for a procedure,
Missing or duplicate variable declaration.

Type errors -- these include:

Type mismatch on assignment,
Type mismatch between actual and formal parameters (the Java compiler might say "no such method..." instead).

Run-time Errors:

Output errors -- the program runs but produces an incorrect result. This indicates an error in the meaning of the program (logic error).

Exceptions -- the program terminates abnormally. Examples include division by zero, null pointer (we'll see what this means later), out of memory.

These also indicate an error in the semantics (logic error).

Non-termination -- the program does not terminate as expected, but continues running "forever." (We'll see how this could happen later.)

Correcting Compiler Errors:-

The compiler reports the approximate location of the error and a vague description of it. In Symantec Cafe, double click on the error message to find that location in the code. Then study that line, as well as a few previous lines, for the error. Check any procedure call on that line for proper number, order, and types of arguments.

Note: Errors may cause the compiler to go "off course." Try correcting the first few errors and running the compiler again.

b. Correcting Run-time Errors (Debugging):-

1. Avoid it. Taking the time to write a correct and well-documented program from the outset will pay back many times over. Finding and correcting bugs takes a lot of time, and it can be frustrating, especially if documentation is insufficient. However, nobody writes perfect software all the time.
2. Debug on a small scale. Thoroughly test each component of application (each method of each object) piece by piece, correcting errors as we go. This is much easier than finding errors in a large program.
3. Understand the symptom. Before changing a program, understand the error and where in the program it occurred. Look at what the program tried to do and where it went wrong.

4. Reason about each error, trying to understand how it could have arisen. This is an important problem-solving task. Form several hypotheses about how the error might have arisen and explore them.
5. Find the source of the error. Don't waste time making "random" changes to our program. After reasoning about the error, we can start searching back through the execution of our program to see where things went wrong.

A **Debugger** can help :-

(a). Examine the call stack and the values of parameters and local variables. See if it matches our expectation of what the program "should" do.

(b). Trace through an execution of our program **by hand** and with a debugger, stopping at certain points in the execution and/or printing out values along the way, to correct the errors and test throughly.

General Concept: -

A good rule of thumb is that testing should take approximately half the effort that was expended in developing and implementing the algorithms, including testing performed by the software development team. Common estimates of the cost of finding and fixing errors in programs range from 40% to 80% of the total development costs. In addition, it has estimated that programs released to testing have one to three bugs per every 100 lines of code, and this is after the programmers have fixed 99% of their mistakes.

Many likely and less-than likely input variations should be tried during testing with the emphasis on tests that are likely to discover bugs. It is not cost effective, and often not possible, to test software so completely that it is bug-free. Instead, the strategy is to find the bugs that are the most likely to cause problems and those that cause the largest problems. Generally, a bug which is the most important to catch will be one that delivers some or many incorrect numbers but does not cause the program to crash or issue warning messages. This includes output values reported in the wrong format, in an incorrect location, or outputs which are just plain wrong. This is important to fix first since the user has a reasonable expectation that any number delivered is **"correct".** The next most important type bug to catch is one that causes the program to crash, or not deliver the expected reports, or to issue false error messages. In this case, the program at least indicates to the user that there is a problem, and they will not depend on the results.

The effort of testing should not be limited by a specific plan. In many cases ,the majority of the testing that is performed and found useful (discovered bugs) is testing that is done on an ad hoc basis following the flow of the program and the intuition of the tester. Adequate time for testing should be provided, since rushing this step serves only to increase the number of likely bugs at the final release.

Glass Box Testing:-

Testing performed by a programmer with access to and understanding of the source code is called Glass Box Testing or structural Testing. Glass Box testing is usually performed during source code development and prior to Black Box testing which is also known as Functional Testing. The original programmers typically perform some Glass Box testing. Unlike Glass Box testing, Black Box testing is testing of the program from the user's perspective with no access to source code. Glass Box testing may include the following:

Code Review- A careful line-by-line review of the source code by another programmer familiar with the program's overall objectives. In the development teams, exchanging source code with another programmer best performs this.

Complete Coverage- A coverage monitor ensures that all lines of the source code are executed. Different types of coverage monitoring programs exist which look at executing all lines, all branches, or all logic conditions for each branch.

Drivers- Adding code that calls the module being developed with artificial data and getting a known response. This also allows for incremental or evolutionary programming, which allows the entire program to be run even when intermediate modules have not yet been developed.

Module deletion- To narrow down how crashes occur, modules are removed and stubs (non-performing modules with the same name) are substituted.

Assertion checks- Conditional statements of a premise at a certain point in the program. The result of an assertion that is incorrect should be a special error condition. Assertion checks may be conditionally compiled, so that they do not exist in the final program.

In addition, the programmer should identify the following during Glass Box Testing, if not already defined in the specification:

Input ranges – The maximum and minimum allowable values for every data entry field.

Internal boundaries – Threshold values of inputs that cause different program segments to be used.

Error handler triggers – Values of input that should cause error handling routines to be executed.

Black Box Testing:-

Black Box testing that is also known as Functional or Behavioral testing. This is a testing pre- release or release version of the program and trying various inputs looking for incorrect outputs or program crashes. It is one of the best ways to diagnose bugs before users discover them. Given the complexity of many software packages, much thought must be devoted to optimizing the testing process. Ideally only one test will be performed for each possible set of software conditions. Of all possible sets of software conditions there are groupings of conditions that if all tested would be testing, the same code and reveal the same bugs (or hopefully the lack of bugs). These groups of conditions are considered an equivalent class or each is called an equivalent class partition.

Software Testing Standards and Guiding Documents:

The Institute of Electrical and Electronics Engineers has a number of publications on quality assurance procedures and standards for developing software. The most applicable standard, "Standard 829-1998 IEEE Standard for Software Test Documentation,"should be reviewed, when developing a testing plan.

Acceptance Testing;-

A range of simple buildings should be simulated with the building simulation program undergoing testing automatically prior to all other tests. These should be considered an acceptance test intended to weed out unstable versions of the software that would not be fruitful to use for further debugging. Acceptance tests are often automated and may be provided to the programmers as a way to reduce the number of versions submitted for testing.

Regression Testing:-

Tests are performed multiple times, including once after each major source code change. The changes are usually due to inconsistencies found during the testing, and to implement new features and are likely to contain problems. The first tests to be run with each new version are considered Regression tests and compare the results before and after the series of tests. The results of the Regression tests are all concatenated into a text file and are compared to a text file prepared using the identical method on the previous version. The comparison will be performed using a standard text file comparison utility and will report any new differences. The inconsistencies that have been fixed by the development team should be easy to identify and confirm that the fixes have been made correctly. The series of regression tests should consist of automated quick-to-perform tests and all tests that have previously found errors. The likelihood of an old error creeping back into the code, when new ones are fixed is high enough to justify keeping tests in the regression suite that have been long since fixed.

Release Tests:-

Special consideration must be made just prior to a public release of the program to ensure that all bugs that were intended to be fixed were actually fixed. A Release test is the most comprehensive automated test that, in large part, consists of a mixture of previously failed-and-fixed tests and tests that have always passed. It is critical that prior to any form of public release that known problems are specifically identified. The public release should include a **"readme"** file that describes all known problems at the time of release, and it is usually the tester that is responsible for compiling this list of problems and any work-around that exist. Release tests should include virus checking of the final installation package. Too many cases of distribution of viruses have been reported to not take this additional precaution. One type of Release test that also needs to be performed prior to a final public release is a comparison of all features actually working reliably with prepared literature. It is crucial the literature reflect all design decision made during development and testing.

Beta Tests:-

Other than the developers and testers, most software developed is not used by anyone else until Beta tests commence. A common practice is to recruit a group of Beta testers who are knowledgeable users of similar products. The Beta test group should be sent the program in executable form for their target environment and should include documentation. They should be warned extensively that the product still has bugs and should not be used for production purposes. It is unrealistic to expect the Beta tester to contribute more than half a day of testing per week. They should be informed of a clear path to make reports on bugs, general problems, and possible enhancements. This should include both e-mail and telephone. Often the lead tester is responsible for managing the feedback from the Beta testers. The lead tester works with the Beta testers to verify any reported bugs by trying to reproduce them. It is not uncommon for the Beta test support to require full time effort, especially if the Beta tester list is above 20 people. Beta testers should get new versions of the program not more often than once every other week otherwise the effort in installing and uninstalling the program is most of the time spent. At times, a critical bug is found and Beta testing needs to be halted, so all Beta testers should be available by e-mail. Beta testers have a few different motivations for volunteering to be a Beta tester and each needs to be catered to:

- Desire to become an expert quickly,
- Anticipation of a free copy of the final program,
- Professional curiosity,
- Complementary product, or
- Competitive product.

At times, it even makes sense to pay the Beta tester to provide expert advice, if there are a limited number of experts in a field. Getting a final version of the program is very common in Beta testing and if it is not part of the plan, the Beta testers should be explicitly told.

Test Suspension and Resumption Criteria:-

Some bugs are so fundamental that the credibility of the results of other tests is affected, and tests need to be suspended. These include tests that appear to reveal general bugs in the input and output processing, tests that show non-repeatable results using identical inputs, tests that produce order of magnitude errors in fundamental algorithms, and tests that produce order of magnitude errors in basic elements. These tests should result in an urgent bug report and a follow up call to the responsible developer and the overall project manager. Both should be informed that no further testing could be fruitful, until that bug is fixed. The overall project manager is informed to let them know of the critical path that may be getting delayed due to the bug ,so that additional programmers may be utilized if necessary. Upon receipt of an updated version, the tester should perform the normal regression testing that is performed on every new version, and repeat the tests that originally revealed the critical bug. Upon passing these tests, both the programmer and the overall project manager should again be informed that tests have resumed.

Full Code Tests:-

Full code tests are designed to exercise all lines of code by exercising combinations of inputs and tracking which lines of code have been executed. This is a glass box testing technique and must be performed by the programmer using software designed to aid in the testing process. Many of the range tests may also be appropriate for performing these tests. Full code tests are no panacea, since they cannot be as comprehensive as full logic flow tests. Full logic flow tests test every possible set of preceding conditions that have been executed prior to a particular portion of the code and, by definition, require an almost infinite amount of effort.

Documentation Tests:-

Comparing how the program operates and the Documentation describing the program is often left to the tester. This is a crucial step. Even though many people don’t read the Documentation, one can expect that any inconsistencies will be costly and embarrassing to fix.

Comparative Tests:-

Comparative tests compare a program to it or to other simulation programs. This type of testing accomplishes results on two different levels, both Validation and Debugging. From a Validation perspective, Comparative tests will show, if the software is computing solutions that are reasonable compared to similar programs. This is a very powerful method of assessment, but it is no substitute for determining , if the program is absolutely correct, since it may be just as equally incorrect as the Benchmark program or programs. The biggest strength of Comparative testing is the ability to compare any cases that two or more programs can both model. This is much more flexible than analytical tests, when only specific solutions exist for simple models, and much more flexible than empirical tests, when only specific data sets have been collected for usually a very narrow band of operation. Comparative testing is also useful for field-by-field input Debugging. Complex programs have so many inputs and outputs that the results are often difficult to interpret. To ascertain , if a given test passes or fails, engineering judgment or hand calculations are often needed. Field by field Comparative testing eliminates any calculation requirements for the subset of fields that are equivalent in two or more simulation programs. The equivalent fields are exercised using equivalent inputs and relevant outputs are directly compared.

Analytical Tests:-

Analytical tests compare results to mathematical solutions for simple cases.

Empirical Tests:-

Empirical tests compare results to experimental data. In many respects, these have proven to be the most difficult type of tests to do. It is important that high-quality data be used as the basis for comparison along with complete and accurate information for developing a simulation model that represents the test building or module as closely as possible.

Range Tests:-

Range tests check the operation of the code over the complete range of valid inputs. The tests will also go beyond all valid ranges to ensure that adequate error messages are generated.

Sensitivity Tests:-

Sensitivity tests compare results to a baseline case and exhaustively test the functioning of every modeling input, including weather data for a full range of climate zones.

Executable Tests:-

Executable Tests interrupt and restart the program, including tests that remove selective binary or input files, looking for graceful program stops with appropriate error messages. Executable Tests also include:--

Load Tests– Testing the program's ability to handle large tasks.

Error Recovery– Attempting to make the program generate as many error messages as possible.

Compatibility– Checking the functioning of the program, simultaneously with other applications such as word processors, spreadsheets, day planners, etc.

Installation– The installation program must be tested, to see if it properly installs in a variety of environments.

Testing Deliverables:-

The ultimate deliverable from any testing effort is software that has fewer bugs than before the testing. In order to understand the value of specific testing activities for use in planning necessary revisions to the software, additional details should be provided in the form of :-

- Test Logs,
- Incident Reports,
- Summary Report.

According to IEEE-829, the Summary Report should include:-

- Summary,
- Variances,
- Comprehensiveness assessment,
- Summary of results,
- Evaluation,
- Summary of activities.

The Summary Report should include what software was tested including version number and a description of the software and hardware environment. It should reference the test plan, test logs and test incident reports. It should describe in the variance section how certain tests described in the test plan were not followed exactly and what

additional tests were performed beyond the test plan. The comprehensiveness assessment should include an evaluation of what features or feature combinations of the software were not sufficiently tested and are expected to contain a subjective assessment. The summary of results and evaluation should include a description of all incidents and whether they were resolved, and provide an overall evaluation of the testing and an estimate of the general reliability of the software. The summary of activities describes the effort level and calendar time needed for performing different aspects of the testing.

c. Testing Techniques & Strategies:-

Testing is a set of activities that can be planned in advance and conducted systematically.

No program or system design is perfect, communication between the user and the designer is not always complete or clear, and time is usually short. So, the result is error and more errors. The number and nature of errors in a new design depend on several factors:-

- Communication between the user and the designer.
- The programmer's ability to generate a code that reflects exactly the system specification.
- The time frame for design.

Theoretically, a newly designed system should have all the pieces in working order, but in reality, each piece works independently. Now is the time to put all the pieces into one system and test it to determine whether it meets the user's requirements. This is the last chance to detect and correct errors, before the system is installed for user acceptance testing. The purpose of system testing is to consider all the likely variations to which it will be subjected and then push the system to its limit. It is a tedious but necessary step in system development.

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. Inadequate testing or contesting leads to errors that may not appear until months later.

Another reason for system testing is its utility as a user-oriented vehicle before implementation. The best program is worthless if it does not meet user needs. Unfortunately, the user's demands are often compromised by efforts to facilitate program or design efficiency in terms of processing time or memory utilization. Often the computer technician and the user have communication barriers due to different backgrounds, interests, priorities, and perhaps languages. The system tester (designer, programmer, or user) who has developed some computer mastery can bridge this barrier.

Before going to test, it is very important to know some basic terms:-

- Unit Testing- It is a testing changes made in an existing or a new program.

- Sequential or Series Testing- It is checking the logic of one or more programs in the candidate system, where the output of one program will affect the processing done by another program.

- System Testing- It is executing a program to check logic changes made in it and with the intention of finding errors-making the program fail. Effective testing does not guarantee reliability. Reliability is a design consideration.

- Acceptance Testing- It is running the system with live data by the actual user.

The first test of a system is to see whether it produces the correct outputs. No other test can be more crucial. Practically it is to be kept in mind that Testing has two broad aspects or sites:-

Development Site:

Database Testing - Testing the database with some tests data, to check how the queries are working?.

Interface Testing - Testing to ensure that the components of the interface are performing properly.

Connection Testing - Testing to check the connection between GUI and database (JDBC-ODBC Connectivity) is working properly.

Implementation Site:

After the implementation of the demo of the package, the users are asked to use the package in the way , they want it without hesitation to test the robustness of the package. Any discrepancies identified during this time are to be fixed before the original implementation.

While testing, the more common errors in computation are….

- Misunderstood or incorrect arithmetic precedence.
- Mixed mode operations.
- Incorrect initialization.
- Precision inaccuracy.
- Incorrect symbolic representation of an expression.
- Test cases should uncover errors such as: -
    - Comparison of different data types
    - Incorrect logical operators or precedence
    - Expectation of equality when precision error makes equality unlikely.
    - Incorrect comparison of variables.
    - Improper or nonexistent loop termination.
    - Failure to exit when divergent iteration is encountered.
    - Improperly modified loop variables.
- Among the potential errors that should be tested when error handling is evaluated are-
    - Error description is unintelligible.
    - Error noted does not correspond to error encountered.
    - Error condition causes system intervention prior to error handling.
    - Exception-condition processing is incorrect.
    - Error description does not provide enough information to assist in the location of the cause of the error.

d. Test Case Design:-

Considering our project the following testing are took place:-

*** Database Testing:**

Considering the entire database i.e. all the tables created in the above project, some test or sampledatas are input and tested the entire modules of the system behaves OK.

*** User Data Input Testing:**

Some invalid datas are input and tested if the system behaves properly to check or resist against the input of invalid datas.

*** Connection Testing:**

Every connection between tables and JDBC-ODBC connections are tested here, also the system behaves properly.

*** Robustness:**

After all testing, all the units and modules and integration testing is to be considered that the system is flexible and can be recovered if crashed.

Test Reports:-

here's a test case table for the OneSpotEducation project based on the provided functionalities:

Test Case Description	Input Data	Expected Output	Error Notification	Measures Taken
User Registration	User details (name, email, password, etc.)	Successful registration	Nil	Nil
User Login	User credentials (username, password)	Successful login as a registered user	Nil	Nil
User Profile	Edit profile information	Profile information updated successfully	Nil	Nil
Course Creation	Course details (title, description, etc.)	New course created	Nil	Nil
Live Class Scheduling	Class details (date, time, topic, etc.)	Class scheduled successfully	Nil	Nil
Live Class Attendance	Student attendance during live class	Attendance recorded	Nil	Nil
Assignment Creation	Assignment details (title, description)	New assignment created	Nil	Nil
Assignment Submission	Submitted assignment	Assignment submitted successfully	Nil	Nil
Assignment Grading	Graded assignment	Assignment graded and feedback provided	Nil	Nil
Student Progress Reports	Student ID or name	Progress report generated	Nil	Nil
Course Analytics	Course ID or name	Analytics and insights provided	Nil	Nil
Attendance Reports	Class ID or date	Attendance report generated	Nil	Nil

User Roles and Permissions	Role assignment and access control	Roles and permissions updated	Nil	Nil
Database Management	Backup and maintenance	Database operations completed	Nil	Nil
System Settings	Configuration changes	Settings updated	Nil	Nil
Messaging System	Message content and recipient	Message sent successfully	Nil	Nil
Discussion Forums	Topic, content, participants	Forum post created	Nil	Nil
Announcement Center	Announcement content	Announcement published	Nil	Nil
Mobile App Development	App features, compatibility	App developed and launched	Nil	Nil
Mobile User Experience	App usability, responsiveness	Smooth user experience	Nil	Nil
Helpdesk Management	User query or ticket	Query resolved or ticket closed	Nil	Nil
Knowledge Base	FAQ search	Relevant FAQ displayed	Nil	Nil
Technical Support	User-reported issue	Issue resolved or assistance provided	Nil	Nil
User Authentication	Login credentials	Secure login achieved	Nil	Nil
Data Encryption	Sensitive data handling	Data encrypted securely	Nil	Nil
Access Restrictions	User access control	Unauthorized access prevented	Nil	Nil
External Content Integration	Integration with external resources	Seamless integration	Nil	Nil
Third-Party Tools Integration	Integration with external tools	Enhanced functionality achieved	Nil	Nil
Course Materials Repository	Storage and management of course materials	Materials accessible and organized	Nil	Nil
File Upload and Sharing	**File upload and sharing**	**Files uploaded and shared successfully**	**Nil**	**Nil**

Each test case should have specific inputs, expected outputs, and measures to be taken in case of errors or issues. Adjustments can be made based on the actual functionalities and requirements of the OneSpotEducation project.

Test Case Report:-

Here we specify all the test cases that are used for system testing. The different conditions that need to be tested along with the test case used for testing those conditions and the expected outputs are given. The goal is to test the different functional requirement document. Test cases have been selected for both valid and invalid inputs.

S.No	Test case	Condition	Expected Output
1	Get Systems	Input Domain name	**Print list of all system in current domains & response time**
2	Get User	Input Domain name	**System id, user id, port no, domain name**
3	Get Processes details	Select process	**Output the details of processes**
4	Get modules details	Select process & select thread opt	**Details of modules**
5	Get thread details	Select process & select thread opt	**Details of threads**
6	Stop the processes	System id, user id, password	**Process close**
7	**Stop the system**	**System id**	**System close**

Table: Test reports with varied output

Testing Analysis:-

S.No	Testing object	Expected value	Simulated value	Explanation	Remarks
1	User name & Password	AEIND GUEST	AEIND GUEST	Equal of expected and simulated values	**Pass**
2	User name & Password	AEIND GUEST	AEIND GUEST	Unequal of expected and simulated value	**Fail**
3	Change password	GUESS	GUEST (Old password)	Equal of these two passwords	**Pass**
4	Start time and end time	0.46	0.46	Equal of these times	**Pass**
5	**Start date and end date**	**43151**	**43151**	**Equal of these dates**	**Pass**

Table: Test Analysis Reports with varied data

7. System Security Measures

Implementation of security Machanisms

- **User Authentication and Authorization:**
  - User Authentication: Implement secure login mechanisms (e.g., username/password, multi-factor authentication) to ensure that only authorized users can access the system.
  - User Authorization: Enforce role-based access control (RBAC) to grant appropriate privileges to users based on their roles (e.g., student, teacher, administrator).
- **Data Encryption:**Use encryption algorithms (e.g., AES, RSA) to protect sensitive data stored in databases, ensuring data confidentiality.
- **Secure Communication:**Implement secure communication protocols (e.g., HTTPS) to encrypt data transmitted between the client and server, preventing eavesdropping and man-in-the-middle attacks.
- **Input Validation and Sanitization:**Validate and sanitize all user inputs to prevent common vulnerabilities like SQL injection and cross-site scripting (XSS) attacks.
- **Password Security:**Enforce strong password policies (e.g., minimum length, complexity) and store passwords securely using hashing algorithms (e.g., bcrypt).
- **Session Management:**Implement secure session management to handle user sessions, including session timeouts and session token management.
- **Secure File Uploads:**Validate and restrict file uploads to prevent uploading of malicious files or unauthorized access.
- **Preventing Brute Force Attacks:**Implement mechanisms to detect and prevent brute force attacks on login pages or other vulnerable endpoints.
- **Cross-Site Request Forgery (CSRF) Protection:**Implement CSRF tokens to protect against CSRF attacks that exploit authenticated users to perform unintended actions.
- **Regular Security Assessments:**Conduct regular security assessments and penetration testing to identify vulnerabilities and address potential security risks.

By implementing security mechanisms at various levels, the One Spot Education system can safeguard sensitive data, protect against potential threats, and ensure a secure and reliable environment for users and administrators. It's essential to continuously monitor and update the security measures to stay vigilant against emerging threats and ensure a robust security posture.

Managing Security:-

Our project falls under the category of an "Education Technology" or "EdTech" application. It aims to digitalize the traditional education system and provide advanced features through a web portal. The application incorporates elements of RDBMS (**Relational Database Management System**) and OOPs (**Object-Oriented Programming**) to effectively manage data and create a scalable, interactive, and user-friendly platform.

In terms of RDBMS (Relational Database Management System) and OOPs (Object-Oriented Programming), the project can be categorized as follows:

1. RDBMS : The project involves the use of a relational database management system to store, organize, and manage data related to various aspects of the education system. This includes storing information about students, teachers, courses, assignments, attendance, and other relevant data in structured tables and establishing relationships between them.The RDBMS component of the project would handle tasks such as designing an appropriate database schema, creating tables, defining relationships and constraints, implementing efficient queries for data retrieval, and ensuring data integrity through proper transaction management and normalization techniques.

2. OOPs : The project utilizes the principles of Object-Oriented Programming to design and develop the web application. The various entities and functionalities within the system, such as students, teachers, classes, assignments, and interactions, can be represented as objects with their own properties (attributes) and behaviors (methods). The OOPs aspect of the project would involve designing and implementing classes, encapsulating data and functionality within them, defining inheritance and polymorphism relationships, and applying other OOPs concepts to create modular, reusable, and maintainable code. This approach allows for the development of a flexible and extensible system that can easily accommodate future enhancements and changes.

A database must have a solid security system to control which activities can be performed and which information can be viewed and modified. A solid security system ensures the protection of data, regardless of how users gain access to the database.

This section describes the security tools built into Microsoft® SQL Server™ 2000 and includes information about:-

- Security Architecture.
- Planning Security.
- Creating Security Accounts.
- Managing Security Accounts.
- Managing Permissions.
- Advanced Security Topics.
- Auditing SQL Server Activity.

Security Architecture:

Tools/Platforms Software Encorporated:-

- **Operating System:**
  - ➔ Server: Linux (e.g., Ubuntu Server, CentOS)
  - ➔ Clients: Windows, macOS, Linux
- **Web Server:** apache (with necessary modules and configurations)
- **Programming Language:** PHP 7.x (with necessary extensions)
- **Framework:** Laravel Framework version 10
- **Database Management System:** MySQL 7 or above
- **Version Control:** Git (for source code management and collaboration)
- **Text Editor or Integrated Development Environment (IDE):** Visual Studio Code, Sublime Text, PhpStorm, or any preferred IDE for coding and development
- **Package Manager:** Composer (for managing PHP dependencies)
- **Web Browser :** Latest versions of Chrome, Firefox, Safari, or Edge (for testing and accessing the web application)

- **Deployment and Hosting:**
  - Server Management Software (e.g., SSH, SCP) for deploying the application to the server.
  - Hosting Service or Cloud Platform (e.g., AWS, DigitalOcean, Google Cloud) to host the web application.
- **Testing Framework (optional but recommended):** PHPUnit (for unit testing)

Hardware Used :-

Server Configuration: -

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB.
- Storage: 256 GB SSD.
- Network Interface: Gigabit Ethernet.
- Operating System: Linux (e.g., Ubuntu Server).
- Web Server: Nginx or Apache.
- Database: MySQL.
- Backup System: Regular data backups with off-site storage.

Client Configuration : -

- IProcessor: Intel Core i3 or equivalent.
- RAM: 4 GB.
- Network Interface: Ethernet or Wi-Fi.
- Operating System: Windows, macOS, or Linux.

The clients should have a good rate of data transfer with the server for quick performance.

The architecture of a security system is based on users and groups of users. The following illustration shows how users and local and global groups in Microsoft® Windows NT® 4.0 and Windows® 2000 can map to security accounts in Microsoft SQL Server™, and how SQL Server can handle security accounts independently of the accounts in Windows NT 4.0 and Windows 2000.

SQL Server also provides security at the application level through the use of individual database application roles.

Creating Security Accounts: -

Each user must gain access to an instance of Microsoft® SQL Server™ through a login account that establishes the user's ability to connect (authentication). This login then has to be mapped to a SQL Server user account, which is used to control activities performed in the database (permissions validation). Therefore, a single login is mapped to one user account created in each database the login is accessing. If no user account exists in a database, the user cannot access the database even though the user may be able to connect to an instance of SQL Server.

Managing Permissions:-

When users connect to an instance of Microsoft® SQL Server™, the activities, they can perform, are determined by the permissions granted to the security account
The user must have the appropriate permissions to perform any activity that involves changing the database definition or accessing data.

Managing Permissions includes granting or revoking user rights to:-

- Work with data and execute procedures (object permissions).
- Create a database or an item in the database (statement permissions).
- Utilize permissions granted to predefined roles (implied permissions).

Object Permissions:-

Working with data or executing a procedure requires a class of permissions known as Object Permissions….

1. SELECT, INSERT, UPDATE, and DELETE statement permissions, which can be applied to the entire table and view.
2. SELECT and UPDATE statement permissions, which can be selectively applied to individual columns of a table or view.
3. SELECT permissions, which may be applied to user-defined functions.
4. INSERT and DELETE statement permissions, which affect the entire row, and therefore can be applied only to the table and view and not to individual columns.
5. EXECUTE statement permissions, which affect stored procedures and functions.

Statement Permissions:-

Activities involved in creating a database or an item in a database, such as a table or stored procedure, require a different class of permissions called Statement Permissions. For example, if a user must be able to create a table within a database, then grant the CREATE TABLE statement permission to the user. Statement permissions, such as CREATE DATABASE, are applied to the statement itself, rather than to a specific object defined in the database.

Statement Permissions are:-

1. BACKUP DATABASE.
2. BACKUP LOG.
3. CREATE DATABASE.
4. CREATE DEFAULT.
5. CREATE FUNCTION.
6. CREATE PROCEDURE.
7. CREATE RULE.
8. CREATE TABLE.
9. CREATE VIEW.

Implied Permissions:-

Implied Permissions control those activities that can be performed only by members of predefined system roles or owners of database objects. For example, a member of the **sysadmin** fixed server role inherits automatically full permission to do or see anything in a SQL Server installation.

Database object owners also have Implied permissions that allow them to perform all activities with the object they own. For example, a user who owns a table can view, add, or delete data, alter the table definition, or control permissions that allow other users to work with the table.

BACKUP:

Backs Up an entire database, transaction log, or one or more files or filegroups. For more information about database backup and restore operations, see Backing Up and Restoring Databases.

b. Creation of User Profiles and Access Rights: -

Userprofile, or simply **profile** when used in-context is a collection of personal data associated to a specific user. A profile refers therefore to the explicit digital representation of a person's identity. A user profile can also be considered as the computer representation of a user model.

A profile can be used to store the description of the characteristics of person. This information can be exploited by systems taking into account the persons' characteristics and preferences.

Profiling is the process that refers to construction of a profile via the extraction from a set of data.

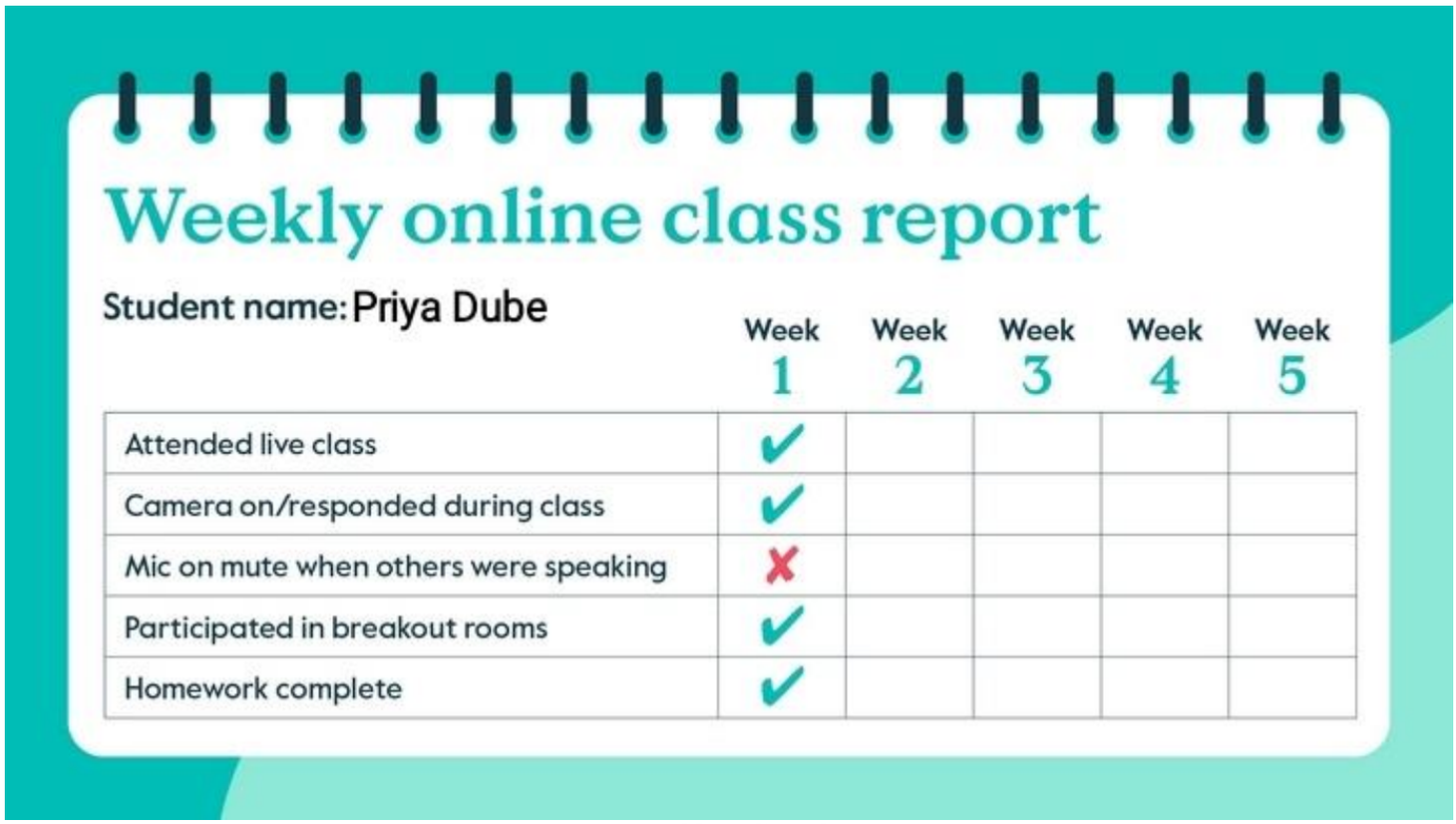
[8] Reports

Some of the reports that are going to be produced automatically are listed below…

Report is well thought-out here as a split module to be developed. This module allows printing different reports as the name specifies. The reports generated are listed below; the functionalities of these are carried accordingly:-

- Attendance Recorded
- Homework /Assignment completion recorded
- Academic Performance
- Marks Scored in Course
- Grade record in Report Card

Sample Layout of Reports: -



Layout: Weekly Online Class Report

Term End Report

12-5-2024

Student Name : Pooja Pandey **Roll No :** 2200811729

Course: Bachelor of Science in Liberal Arts

Scholastic Areas	Term 1					Term 2					Total
Subject	FA1	FA2	FA3	SA1	Total	FA1	FA2	FA3	SA1	Total	Overall %
History	60.0	77.69	58.0	70.0	67.91	60.0	77.69	58.0	70.0	67.91	67.91
English	79.0	44.5	90.0	66.0	66.95	79.0	44.5	90.0	66.0	66.95	66.95
Mathematics	86.0	36.5	96.0	44.0	60.55	86.0	36.5	96.0	44.0	60.55	60.55
Social Studies	85.71	65.0	60.0	86.0	74.44	85.71	65.0	60.0	86.0	74.44	74.44
Physics	78.57	93.08	82.0	90.0	87.03	78.57	93.08	82.0	90.0	87.03	87.03
Economics	82.86	70.77	74.0	66.0	72.4	82.86	70.77	74.0	66.0	72.4	72.4
Science	58.57	76.15	76.0	78.0	73.16	58.57	76.15	76.0	78.0	73.16	73.16

Co-Scholastic Areas	Grade
Social Skills	A
Work Skills	B
Art And Craft	A
Physical Education	B

Class Teacher's Comments :

Pooja is a spontaneous learner and shows keen interest in group activities. He shows great enthusiasm in class discussions and is able to read familiar words with ease. He has a quest to learn new words and is able to write simple sentences. He takes pride in completing his assignments on time. He grasps the grammar concepts taught in class.

Grade Boundaries :

A+ : 100 - 91, A : 90 - 81, B+ : 80 - 71, B : 70 - 61, C : 60 - 51, D : 50 - 41, E : 40 - 31, F : 30 - 0

Layout: Term End Report

[7] Future Scope & Further Enhancement

The future scope and further enhancements of the One Spot Education system are vast, providing opportunities for continuous growth and improvement. Some potential future scope and enhancements include:

1. **Artificial Intelligence Integration:**
   - Implement AI-based adaptive learning algorithms to personalize courses for each student based on their learning preferences and progress.
   - Utilize AI-powered chatbots for instant user support and answering common queries.
2. **Virtual Reality and Augmented Reality (VR/AR) Expansion:**
   - Expand VR/AR learning experiences across various subjects to provide immersive and interactive education.
   - Develop virtual laboratories for practical subjects like science, engineering, and medicine.
3. **Blockchain-Based Credentialing:**
   - Integrate blockchain technology to issue secure and verifiable certificates, diplomas, and badges upon course completion.
   - Establish a decentralized credentialing system to enhance the credibility of achievements.
4. **Enhanced Data Analytics and Reporting:**
   - Develop advanced analytics dashboards with real-time insights into user engagement, course performance, and learning trends.
   - Introduce predictive analytics to identify at-risk students and provide early interventions.
5. **Mobile App Offline Mode and Mobile-First Approach:**
   - Improve the mobile application to support offline access to course materials, allowing students to study without an internet connection.
   - Adopt a mobile-first approach to optimize the user experience on smartphones and tablets.
6. **Continuous Learning and Professional Development:**
   - Offer continuous learning resources and courses for professionals to upgrade their skills and stay competitive in their industries.
   - Partner with industry experts and organizations to provide certifications and skill development programs.
7. **Multilingual Support and Internationalization:**
   - Translate course content and the user interface into multiple languages to cater to a global audience.
   - Customize the platform to comply with the educational standards and regulations of different countries.
8. **Social Learning Enhancements:**
   - Foster more extensive social learning communities with discussion forums, study groups, and collaborative projects.
   - Encourage peer-to-peer learning and knowledge sharing among students and educators.
9. **Enhanced Security and Privacy Measures:**
   - Implement advanced security protocols to protect user data, prevent cyber threats, and ensure compliance with data privacy regulations.
   - Conduct regular security audits and penetration testing to identify and address potential vulnerabilities.

10. Integration with External Learning Platforms and Resources:

- Collaborate with reputable educational content providers, open educational resources (OER), and Massive Open Online Course (MOOC) platforms to expand course offerings.
- Facilitate seamless transfer of credits and certifications from other recognized educational institutions.

11. Continuous User Feedback and Improvement:

- Conduct user surveys and feedback loops to gather insights for continuous improvement of the platform's features and usability.
- Regularly update and optimize the system based on user preferences and needs.

Conclusion:-

In conclusion, the One Spot Education system presents a comprehensive and cutting-edge solution to revolutionize the traditional education landscape. By leveraging the power of technology and advanced features, the platform aims to digitize and modernize the learning experience for students, educators, and institutions alike. Through its user-friendly interface, interactive live classes, and personalized learning paths, One Spot Education endeavors to foster a dynamic and engaging educational environment. The emphasis on security mechanisms ensures the safety and privacy of user data, building trust and confidence in the platform. As we continually strive for improvement, the future scope of the system includes integrating emerging technologies like AI, VR/AR, and blockchain, further enhancing the learning process and making education accessible to a global audience. With a commitment to excellence and a vision for continuous development, One Spot Education aspires to shape the future of education, empowering learners to explore their full potential and embrace a lifetime of knowledge acquisition and growth. As the educational landscape continues to evolve, One Spot Education remains dedicated to pushing the boundaries of online education, creating a collaborative and innovative environment that inspires lifelong learning and empowers individuals to thrive in an ever-changing world.

[8] Bibliography

BOOKS

- **Data Flow Diagram –** Ref - Object Oriented Analysis and Design (MCS-219),Block 2:Modeling,UNIT 8: Functional Modeling
- **ER Diagram –** Ref - **Modeling the System Architecture** Unit 2, Block 1 of MCS 213 (Software Engineering), Study Material by IGNOU
- Ref - **Software Engineering** MCS 213 , Study Material by IGNOU
- Ref - **Class Diagram-** Unit 2: Structural Modeling using UML, Block 1 of MCS 219 (Object Oriented Analysis and Design), Study Material by IGNOU

Articles and Documents

- Ref Cloudways– Tips to Write Web Development Proposal Template in 2022by Najam Ahmed
- Ref - MCSP – 232 project guidelines Programme Guide For Master Of Computer Applications (Online) (Programme Code: Mcaol)
- Ref – Laravel Documentation : https://laravel.com/docs/10.x

Websites

- Ref – W3school - https://www.w3schools.com/
- Ref – Tutorial Point- https://www.tutorialspoint.com/
- Ref - Real Time Chat With Laravel Broadcast, Pusher and Vuejs : https://www.udemy.com/course/laravel-real-time-chat-with/
- Ref: FreeCodeCamp - https://www.freecodecamp.org/learn/
- Ref- Teach Yourself in SQL Server Database Development-Techmedia.Ref - RDBMS -Bipin C. Desai.

Glossary

1. **API (Application Programming Interface):** A set of rules and protocols for building and interacting with software applications, allowing different software systems to communicate with each other.

2. **CRUD (Create, Read, Update, Delete):** Basic operations performed on database records. These operations form the foundation of managing persistent data in an application.

3. **CSRF (Cross-Site Request Forgery):** A type of security attack in which an attacker tricks a user into performing actions on a web application in which they're authenticated, without their consent.

4. **CSS (Cascading Style Sheets):** A style sheet language used for describing the presentation of a document written in HTML or XML. CSS defines how elements should be rendered on screen, on paper, in speech, or on other media.

5. **DBMS (Database Management System):** Software that uses a standard method to store and organize data. It allows users to create, read, update, and delete data in a database.

6. **DFD (Data Flow Diagram):** A graphical representation of the data flow through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into detail about the system.

7. **DRY (Don't Repeat Yourself):** A principle of software development aimed at reducing the repetition of code. By ensuring that "every piece of knowledge must have a single, unambiguous, authoritative representation within a system," DRY helps maintain consistency and ease of maintenance.

8. **Eloquent ORM (Object-Relational Mapping):** Laravel's ORM, which provides a simple Active Record implementation for working with the database. It allows developers to interact with database records using a model-based approach.

9. **Gantt Chart:** A type of bar chart that represents a project schedule. It illustrates the start and finish dates of the various elements of a project, providing a visual timeline.

10. **HTTP (HyperText Transfer Protocol):** The foundation of any data exchange on the Web and a protocol used for transmitting hypermedia documents, such as HTML. It is designed for communication between web browsers and web servers.

11. **JSON (JavaScript Object Notation):** A lightweight data-interchange format that's easy for humans to read and write, and easy for machines to parse and generate. It's commonly used for transmitting data in web applications.

12. **MVC (Model-View-Controller):** An architectural pattern commonly used for developing user interfaces that divides an application into three interconnected components. This separation helps manage complex applications by separating the internal representations of information from the ways that information is presented and accepted by the user.

13. **ORM (Object-Relational Mapping):** A programming technique for converting data between incompatible type systems in object-oriented programming languages. It creates a "virtual object database" that can be used from within the programming language.

14. **PERT (Program Evaluation Review Technique):** A statistical tool used in project management, which is designed to analyze and represent the tasks involved in completing a given project. It helps in scheduling, organizing, and coordinating tasks within a project.

15. **PHP (Hypertext Preprocessor):** A popular general-purpose scripting language that is especially suited to web development. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group.

16. **REST (Representational State Transfer):** An architectural style for designing networked applications. It relies on a stateless, client-server, cacheable communications protocol — and in virtually all cases, the HTTP protocol is used.

17. **SaaS (Software as a Service):** A software distribution model in which applications are hosted by a vendor or service provider and made available to customers over the internet.

18. **SQL (Structured Query Language):** A domain-specific language used in programming and designed for managing data held in a relational database management system.

19. **TDD (Test-Driven Development):** A software development process that relies on the repetition of a very short development cycle: first, the developer writes an initially failing automated test case that defines a desired improvement or new function, then produces the minimum amount of code to pass that test, and finally refactors the new code to acceptable standards.

20. **UI (User Interface):** The space where interactions between humans and machines occur. The goal of this interaction is effective operation and control of the machine from the human end, while the machine simultaneously provides feedback that aids the operators' decision-making process.

21. **UML (Unified Modeling Language):** A standardized modeling language that provides a general-purpose, developmental, modeling language in the field of software engineering. It is intended to provide a standard way to visualize the design of a system.

22. **XSS (Cross-Site Scripting):** A type of security vulnerability typically found in web applications. XSS attacks enable attackers to inject client-side scripts into web pages viewed by other users.

END OF PROJECT DOCUMENT