A PROJECT REPORT
ON

Learning Management System

The Study Hall College

Affiliated To

UNIVERSITY Of LUCKNOW

Submitted by

Sachin Kumar Yadav

Course
Bachelors in Computer Application

2112410580019

2023-24

CERTIFICATE

ACKNOWLEDGMENT

I would like to express my special thanks to our mentor **Mr. Carlos** for his time and efforts. Your useful advice and suggestions were really helpful to me during the projects's completion. In this aspect, I am eternally grateful to you. I would also like to express my special thanks of gratitude to my **HOD Dr. Neha Mahendra** and **Mr. Rohit Dixit** for guiding us.

Also I would like to take this opportunity to thank my family members and supporters, without them it could not have been done effectively in such a short period of time. A project is a bridge between theoretical and practical learning and with this thinking I worked on the project and made it successful due to timely support and efforts of all who helped me.

Sachin Kumar Yadav

PREFACE

There is a rapid growth in the field of information technology and this has severely influenced the whole world. Thus, websites are coming up very frequently these days to make the people relax with their complexities.

The project has been made by my own effort that I have learned in our project period.

My project is **"Learning Management System"** and name is **'Study Hall College** The purpose of this project can be defined as "a suite of services designed to deliver, track, report on and administer learning content, student progress and student interactions". The importance of these systems is growing as far as term such as 'Internet-based Educational Systems', 'Educational Service Providers' and 'Distance-learning' or 'On-demand Education' are becoming popular now a days.

Learning Management System is a Web-based system for training programs and information sharing between individuals giving them the flexibility to access it from their workplace or home. Authorized individuals have 24/7 access to this unique system through a unique User ID and Password.

Table of contents

The format of the report is given herewith and should be strictly followed.

Text Formatting Instructions:

i. One and half (1.5'') line spacing should be used for typing the general text. The general text shall be **justified** and typed in the Font style 'Times New Roman' and Font size **12.**

ii. **Subheading** shall be typed in the Font style 'Times New Roman' and Font size 14 and bold.

iii. **Heading** shall be typed in the Font style 'Times New Roman' and Font size 16 and bold.***Page specification: left margin=1.5", top=1.5", bottom=1", right=1"***

Note: *Cover Page as per format (colored).*

1. INTRODUCTION AND FEATURES OF TECHNOLOGY

a) HTML:

The **Hyper Text Markup Language** or **HTML** is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages.HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page.HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, quotes another items.HTML elements are delineated by tags, written using angle brackets. Tags such as **<img/>** and **<input/>** directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browser do not display the HTML tags but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. The inclusion of CSS defines the look and layout of content. The World Wide Web Consortium(W3C),former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.A form of HTML, known as HTML5, is used to display video and audio, primarily using the **<canvas>** element in collaboration with JavaScript.

b) CSS:

Cascading Style Sheets(CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as HTML or XML (including XML dialects such as SVG, Math ML or XHTML). CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

CSS is designed to enable the separation of content and presentation, including layout, colors and font. This separation can improve content accessibility; provide more flexibility and control in the specification of presentation characteristics; enable multiple web pages to share formatting by specifying the relevant CSS in a

separate .css file, which reduces complexity and repetition in the structural content; and enable the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on braille-based tactile devices.CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more them one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium(W3C).

Internet media type (MIME type) text / css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

c) JAVASCRPIT:

JavaScript, often abbreviated as **JS,** is a programming language that is one core technologies of the Word Wide Web, alongside HTML and CSS. As of 2022, 98% of websites use JavaScript on the client side for webpage behavior, often incorporating third- party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on user, devices.

JavaScript is a high-level, often just-in-time compiled language that conforms to the ECMA Script standard. It was dynamic typing, prototype-based object-orientation, and first-class functions. It is multi-paradigm, supporting event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates, regular expressions, standard data structures, and the Document Object Model (DOM).

The ECMAScript standard does not include any input/output (I/O), such as networking, storage, or graphics facilities. In practice, the web browser or other runtime system provide JavaScript APIs for I/O. JavaScript engines were originally used only in web browsers, but are now core components of some servers and a variety of applications. The most popular runtime system for this usage is Node js.

JavaScript is the most popular programming language in the world. It is the language for HTML, for the web, for computers, servers, laptop, tablets, smart phone, and more. You can use JavaScript to:

- Change HTML elements
- Delete HTML elements

- Create new HTML elements

d) PYTHON WITH DJANGO:

Django is a **Python-based web application framework** that is free and open source. A framework is simply a collection of modules that facilitate development. They are grouped together and allow to build apps or websites from scratch rather than starting from scratch.

Django is a Python framework that makes it easier to create web sites using Python. Django emphasizes reusability of components, also referred to as DRY (Don't Repeat Yourself) and comes with ready-to-use features like login system, database connection and CRUD operations(Create Read Update Delete).

Django follows the MVT design pattern(Model View Template):

- Model – The data we want to present, usually data from a database.
- View – A request handler that returns the relevant template and content – based on the request from the user.
- Template – A text file (like an HTML file) containing the layout of the web page, with logic on how to display the data.

Django is a Python coding tool that adds functionality and speeds up the process. Django includes Python code that has already been written and is ready to be used in a project. Django is a "framework" because it includes a fully functional set of classes, libraries and modules that allow developers to do almost anything they need to create robust websites and apps.

Python is the foundation and Django is built on top of it because it is written in Python. Django is the foundation for our websites or app.

1.2 FEATURES OF TECHNOLOGY

Scripting Language

JavaScript is a lightweight scripting language made for client-side execution on the browser. Since it is not designed as a general-purpose language and is specially engineered for web applications, the set of libraries is also geared primarily towards web applications.

Interpreter Based

Javascript is an interpreted language instead of a compiled one. In that sense, it is closer to languages like Ruby and Python. The browser interprets JavaScript's source code, line by line and runs it. In contrast, a compiles language needs to be compiled into a byte-code code executable. Java and C++ are examples of compiled languages.

Event Handling

An event is an action or an occurrence in a system communicates about said occurrence so that you can respond to it somehow. For example, a user clicks on a button, and the system tells you to respond to the button click event with an action, say an information box. JavaScript enables you to handle events and even generate custom events.

Light Weight

JavaScript isn't a compiled language, so it doesn't get converted to byte-code beforehand. However, it does follow a paradigm called Just-In-Time(JIT) Compilation. Meaning it gets converted to byte code just as it's about to run. This enables JS to be lightweight. Even less powerful devices are capable of running JavaScript.

Features of Django:

Rapid Development

Django was created with the goal of creating a framework that would allow developers to build web applications in less time. The project implementation phase takes a long time, but Django makes it happen quickly.

Versatile

Django is a versatile framework that can be used to create applications in a variety of domains. Companies are now using Django to create a variety of applications, such as content management systems, social networking sites, and scientific computing platforms, among others.

Open Source

Django is a web application framework that is free and open source. It is freely available to the public. The source code is available for download from the public repository. The total cost of developing application is reduced by using open source.

Portable

Since Django is a Python-based web framework it runs on almost all platforms. Almost all Linux flavors, Windows Editions and Mac versions support Django. Django is also officially supported by many web-hosting server platforms that even provide detailed documentation on hosting Django Applications on their servers.

2 . INTRODUCTION OF PROJECT

A **Learning Management System(LMS)** is a software application for the administration, documentation, tracking, reporting, automation and delivery of educational courses, training programs, materials or learning and development programs. The learning management system concept emerged directly from e-learning. Typically, an LMS provides an instructor with a way to create and deliver content, monitor student participation and assess student performance. It might also provide students with interactive features such as live classes and discussion forums.

LMS are beneficial to a range of organizations, including higher education institutions and companies. They are primarily used for knowledge management and effective learning. An LMS is like our own online university where we can create, upload, store and assign online courses for learners to access and take on PCs, laptops, tablets or smartphones in a browser.

Benefits of LMS

Some of the key benefits include:

- Centralized learning
- Cost-effectiveness
- Time savings
- Consistent and standardized training
- Tracking and reporting
- Improved employee performance and productivity
- Scalability
- Encouraging a learning culture

3. OBJECTIVE AND SCOPE OF THE PROJECT

3.1 Objective:

The main objective of this system is to enhance the quality of learning process. Meet the learning style or needs of students. Improve the efficiency and effectiveness. Improve user-accessibility and time flexibility to engage learners in the learning process. There are plenty of reasons why we might want to use an LMS to train and engage students. Consider the following objectives of a learning management system.

- Access to learning material
- Streamline Training Process
- Engage students
- Track, Assess and Report
- Reduce Costs
- Improve Efficiency
- Instant Feedback

3.2 Scope of the project:

This system has gained recognition worldwide due to the plethora of tools that come handy with online courses. Development of information and communication technology has supported the growth of learning management software and its propagation. LMS system makes education easier as there is an availability of multimedia sources alongside textual materials for the students. This makes learning fun and engaging thereby making a progressive e-learning environment.

Apart from students, even those who have been placed can get certificate courses in specialized areas related to their field without the need to quit their jobs. It’s easier as well as an economical way to acquire education.

4. THEORETICAL BACKGROUND DEFINITION OF PROBLEM

In this section we shall discuss the limitation and drawback of the existing system that forced us to take up this project. Really that work was very typical to manage the daily errors free records and adding or removing any node from server. This problem produces a need to change the existing system. Some of these shortcomings are being discussed below: -

- **Low Functionality**

With the existing system, the biggest problem was the low functionality. The problem faced hampered the work. For small task like adding any new node to server or deleting a node or keeping daily record we have to appoint minimum two or three employee.

- **Erroneous Input and Output**

In the existing system, humans performed all the tasks. As in the human tendency, error is also a possibility. Therefore, the inputs entered by the person who is working in the company, in the registers may not be absolutely foolproof and may be erroneous. As a result of wrong input, the output reports etc. will also be wrong which would in turn affect the performance.

- **Portability Problem**

System that existed previously was manual. As a result, the system was less portable. One has to carry the loads of many registers to take the data from one place to another. A big problem was that the system was less flexible and if we wanted to calculate yearly or monthly maintenance report or efficiency report, then it was a big headache.

- **Security-**

Security concerns were also one of the motives of the Company for the need of software. In the registers, the data is not secure as anybody can tamper with the data written in the registers. While in this

software, just a password makes it absolutely secure from the reach of unauthorized persons.

- **Data Redundancy**

In the case of manual system, the registers are maintained in which, a lot of data is written.

- **Processing Speed**

In manual system maintaining a register and performing the necessary calculation has proved to be a troublesome job, which takes a lot of time and may affect the performance of the Company. But with this software we can have all the tasks performed in a fraction of second by a single click thus making the troublesome job much easier.

- **Manual Errors**

When a number of tough tasks are prepared by the humans like preparation of reports, performing long calculation then some human error are obvious due to a number of factors like mental strain, tiredness etc. But as we all know that computer never get tired irrespective of the amount of work it has to do. So this software can nullify the probability of manual error that improve the performance.

5. FEASIBILITY STUDY

5.1 Feasibility Study

Feasibility study is the step of preliminary study of the system development life cycle. Things are always easy at the beginning in any software process. In fact nothing is in feasible with unlimited time and resources. But it is not the fact. So, practically we have to do in limited resources in a restricted time margin. So for the system to be feasible, following points we have to consider.

The feasibility study is conducted to check whether the candidate system is feasible. The system which is selected to be the best against the criteria is there after designed and developed. The feasibility study takes into consideration, the risks involved in the project development beforehand. Therefore in this phase we have to do feasibility study which is the test of the website according to its work ability, impact on the organization, ability to meet user need and effective use of resources. We do the feasibility study for website to analyze the risks, costs and benefits relating to economics, technology and user organization. There are several types of feasibility depending on the aspect they cover. Important of these includes:

1. Technical Feasibility:

This is an important outcome of preliminary investigation. It comprise of following questions:-

- Can the work of project bed one with the current equipment, existing software and available man power resource?
- If Technology is required what are the possibilities that it can be developed?

2. Economic Feasibility:

It deals with question related to the economy. It comprise of the following questions:-

- Are there sufficient benefits in creating the system to make the cost acceptable?
- Are the costs of not creating the system so great that the project must be undertaken?

3. Legal Feasibility:

It deals with the question related to the legal issues. It comprise of the following questions: -

- Contract Signing
- Software License agreement
- Issues related to cyber laws.
- Legal issues relating to the man power contract.

4. Operational Feasibility:

The operational feasibility consists of the following activity:

- Will the system be useful if it is developed & implemented?
- Will there be resistance from employee?

5. Social & Behavioral Feasibility:

It deals with the various issues related to the human behavior like: -

- Whether the user be able to adapt a new change or not?
- Whether the ambiance we are providing suits the user or not?

6. SYSTEM PLANNING

System planning for a Learning Management System (LMS) involves a structured approach to defining the objectives, scope, requirements, and overall strategy for the development and implementation of the system. Here are key steps and considerations for the system planning phase:

6.1 Define Objectives and Goals:

Clearly articulate the objectives of the Learning Management System. Identify the primary goals, such as improving educational outcomes, enhancing training effectiveness, or streamlining administrative processes.

6.2 Conduct Needs Assessment:

Perform a thorough needs analysis to identify the specific requirements of the users (students, instructors, administrators). Consider factors such as the type of content to be delivered, desired features, and technical infrastructure.

6.3 Define Scope and Boundaries:

Clearly outline the scope of the LMS, specifying what functionalities will be included and any limitations. Consider integration with other systems, scalability, and the targeted user base.

6.4 Identify Stakeholders:

Identify and involve key stakeholders, including educators, IT professionals, administrators, and learners. Gather input and feedback to ensure that the LMS meets the diverse needs of its users.

6.5 Set Budget and Resource Constraints:

Define the budget for the LMS project, taking into account development costs, maintenance, and potential ongoing expenses. Identify resource constraints, including human resources, technology infrastructure, and time frames.

6.6 Technology Infrastructure:

Assess the existing technology infrastructure to determine compatibility and integration requirements. Consider factors such as hardware, software, network capabilities, and security protocols.

6.7 Compliance and Standards:

Ensure compliance with relevant educational standards (e.g., SCORM, API) and data protection regulations. Adhering to industry standards facilitates interoperability and ensures that the LMS aligns with broader educational frameworks.

6.8 User Requirements and Experience:

Identify user requirements by considering the needs and preferences of learners, instructors, and administrators. Prioritize features that enhance the user experience, such as intuitive interfaces, mobile compatibility, and personalized learning paths.

6.9 Content Management:

Determine how learning content will be managed, including content creation, duration, and version control. Consider support for various content types (text, multimedia, assessments) and the ability to update content easily.

6.10 Training and Support:

Plan for user training and support mechanisms. Develop documentation, tutorials, and training programs to ensure that users can effectively navigate and utilize the LMS. Consider ongoing support for technical issues and user inquiries.

6.11 Security and Privacy:

Develop a comprehensive security plan to safeguard user data, ensure system integrity, and protect against unauthorized access. Adhere to privacy regulations and establish protocols for data encryption, access control, and regular security audits.

6.12 Timeline and Milestones:

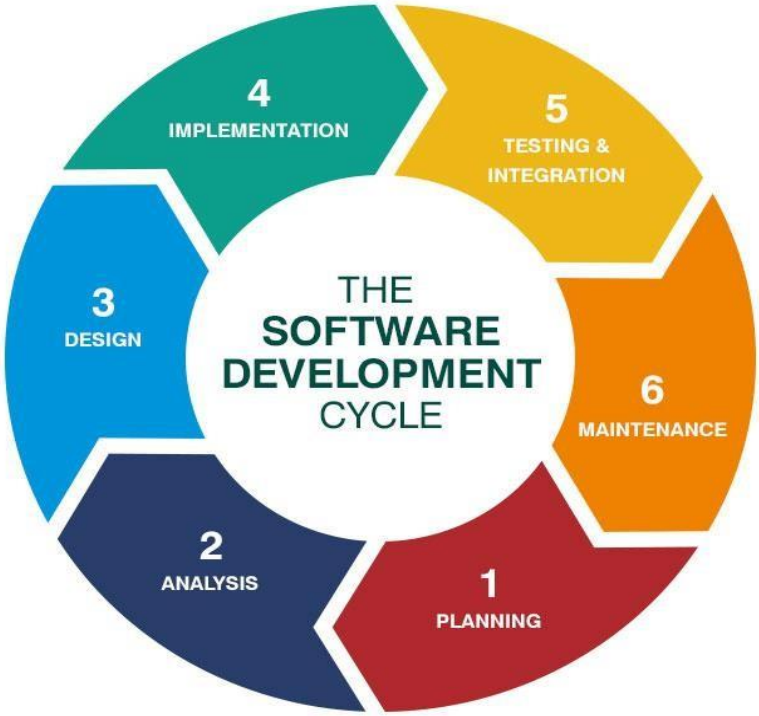
Create a realistic timeline with well-defined milestones. Consider the phased implementation of the LMS, with opportunities for testing, feedback, and adjustments throughout the development process.

6.13 Evaluation and Feedback Mechanisms:

Establish mechanisms for ongoing evaluation and feedback from users. This can include pilot testing, surveys, and regular reviews to assess the effectiveness of the LMS and identify areas for improvement.
By systematically addressing these considerations during the system planning phase, organizations can lay a solid foundation for the successful development and implementation of a Learning Management System.

Maintenance: -

Merely developing the system is not important but also maintenance is important. The company that has built the system provides for some time free of cost maintenance to the client and after that period it is usually a paid service.



7. METHODOLOGY ADOPTED, SYSTEM IMPLEMENTATION

7.1 Methodology adopted

The methodology adopted for developing a Learning Management System (LMS) typically follows a structured and iterative process that encompasses various stages from planning to implementation. Below is a generalized methodology that organizations might adopt for the development of an LMS:

Iterative Enhancement Model

- This model has the same phases as the waterfall model, but with fewer restrictions. Generally the phases occur in the same order as in the waterfall model, but they may be conducted in several cycles.
- Useable product is released at the end of the each cycle, with each release providing additional functionality. Customers and developers specify as many requirements as possible and prepare a SRS document. Developers and customers then prioritize these requirements. Developers implement the specified requirements in one or more cycles of design, implementation and test based on the defined priorities.

  The procedure itself consists of the initialization step, the iteration step, and the Project Control List. The initialization step creates a base version of the system. The goal for this initial implementation is to create a product to which the user can react. It should offer a sampling of the key aspects of the problem and provide a solution that is simple enough to understand and implement easily. To guide the iteration process, a project control list is created that contains a record of all tasks that need to be performed. It includes such items as new features to be implemented and areas of redesign of the existing solution.

  The control list is constantly being revised as a result of the analysis phase.

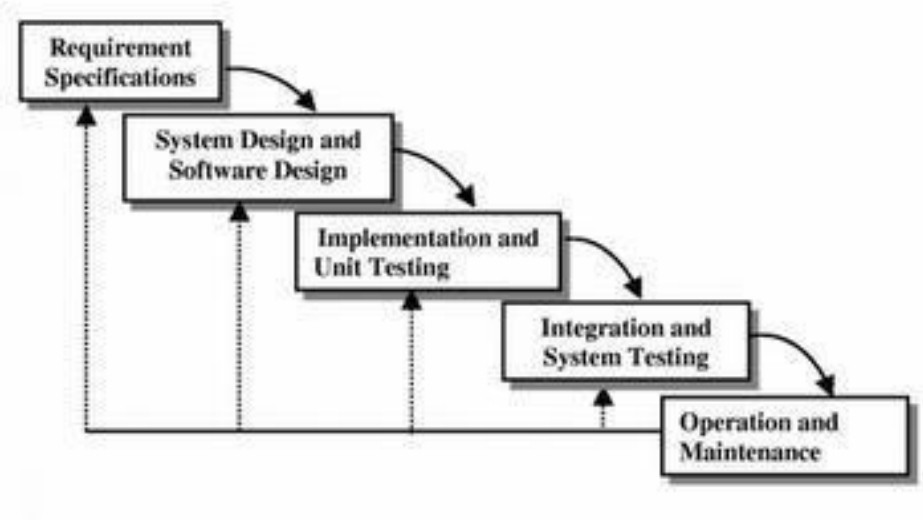
  The iteration involves the redesign and implementation of iteration is to be simple, straightforward, and modular, supporting redesign at that stage or as a task added to the project control list. The level of design detail is not dictated by the iterative approach. In a light-weight iterative project the code may represent the major source of documentation of the system; however, in a critical iterative project a formal Software Design Document may be used. The analysis of iteration is based upon user feedback, and the program analysis facilities available. It involves

analysis of the structure, modularity, usability, reliability, efficiency, & achievement of goals. The project control list is modified in light of the analysis results.

PHASES:

Incremental development slices the system functionality into increments (portions). In each increment, a slice of functionality is delivered through cross- discipline work, from the requirements to the deployment. The unified process groups increments/iterations into phases: inception, elaboration, construction, and transition.

- Inception identifies project scope, requirements (functional and non-functional) and risks at a high level but in enough detail that work can be estimated.
- Elaboration delivers a working architecture that mitigates the top risks and fulfills the non- functional requirements.
- Construction incrementally fills-in the architecture with production-ready code produced from analysis, design, implementation, and testing of the functional requirements.
- Transition delivers the system into the production operating environment.



Waterfall Model

7.2 SYSTEM IMPLEMENTATION

In LMS system implementation, key steps include:

1. Data Migration: Transfer existing data to the new system, ensuring accuracy and integrity.
2. Software Installation: Deploy the LMS software across the organization's infrastructure, considering scalability and compatibility.
3. User Training: Provide comprehensive training to administrators, instructors, and learners to ensure effective system utilization.
4. Testing: Conduct thorough testing to identify and rectify any issues before full deployment. This includes functionality, security, and performance testing.
5. Rollout Plan: Develop a well-defined plan for gradually introducing the LMS to users, minimizing disruptions and providing support.
6. Integration: Integrate the LMS with other systems if necessary, such as HR systems or content repositories.
7. User Support: Establish ongoing support mechanisms for users, addressing queries and issues post-implementation.
8. Monitoring and Evaluation: Continuously monitor system performance and gather feedback for further improvements .Effective communication and collaboration between project teams, IT staff, and end-users are critical during the implementation phase.

8. HARDWARE AND SOFTWARE REQUIREMENTS

A requirements specification for a software system is a complete description of the behavior of a system to be developed and it includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements.

Non-functional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).Requirements are a sub-field of software engineering that deals with the elicitation, analysis, specification, and validation of requirements for software.

The software requirement specification document enlists all necessary requirements for project development. To derive the requirements, we need to have clear and thorough understanding of the products to be developed. This is prepared after detailed communications with project team and the customer.

SERVER-SIDE HARDWARE REQUIREMENT:

- AMD Athlon 64 with processor speed 2.8 or more
- 256 DDR Ram
- 40 GB Hard disks
- Network Interface card
- CD-Drive

SERVER-SIDE SOFTWARE REQUIREMENT:

- Windows
- Python 3.7 and PyCharm IDE 2021.3

- SQLite

CLIENT-SIDE HARDWARE REQUIREMENT:

- Processor Dual core-based computer
- 2 GB Minimum RAM
- 20 GB HDD
- 100 Mbps LAN
- Web Browser

TO DEVELOP THIS PROJECT THE VARIOUS SOFTWARE RESOURCES ARE USED.

- Front End - HTML-CSS &Bootstrap
- Back End - SQLite
- Web Server - Apache Server
- Technology - Python technology
- Code-Behind Language - Python
- IDE - PyCharm

9. SYSTEM DESIGN

9.1 Data base table structure-

A database is an organized collection of data. Instead of having all the data in a list with a random order, a database provides a structure to organize the data. One of the most common data structures is a database table. A database table consist of rows and columns. A database table consists of rows and columns. A database table is also called a two dimensional array. An array is like a list of values, and each value is identified by a specific index. A two-dimensional array uses two indices, which correspond to the rows and columns of a table.

In database terminology, each row is called a record. A record is also called an object or an entity. In other words, a database table is a collection of records. The records in a table are the objects. A field corresponds to a column in the table and represents a single value for each record. A field is also called an attribute. In other words, a record is a collection of related attributes that make up a single database entry.

In my project, I used SQLite database. SQLite is a database engine written in the C programming language. It is a lightweight and compact database that does not require any kind of server to run. It is easily integrated into any kind of mobile application. SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private.

Why use SQLite?

- SQLite is an open-source software. The software does not require any license after installation.
- SQLite is serverless as it does not need a different server process or system to operate.
- SQLite facilitates us to work on multiple databases on the same session simultaneously, thus making it flexible.
- SQLite is a cross-platform DBMS that can run on all platforms, including MACOS, Windows etc.
- SQLite does not require any configuration. It needs no setup or administration.

9.2 Data Flow Diagram-

Introduction

A data flow diagram (DFD) is a visual representation of the information flow through a process or system. DFDs help you better understand process or system operation to discover potential problems,

improve efficiency, and develop better processes. They range from simple overviews to complex, granular displays of a process or system.

Types of DFD

Data Flow Diagrams are either Logical or Physical.

- **Logical DFD-** Thi9s type of DFD concentrates on the system process and flow of data in the system. For example in a Banking software system, how data is moved between different entities.
- **Physical DFD-** This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

DFD Components

DFD can represent Source, destination, storage and flow of data using the following set of components-

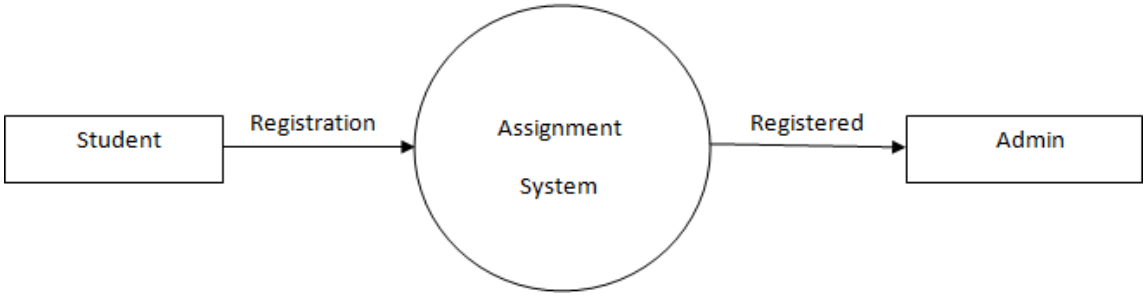
- **Entities-** Entities are source and destination of information data. Entities are represented by a rectangle with their respective names.
- **Process**- Activities and action taken on the data are represented by Circle or Round-edged rectangles.
- **Data Storage-** There are two variants of data storage – it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
- **Data Flow-** Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

DFD levels

A data flow diagram can dive into progressively more detail by using levels and layers, zeroing in on a particular piece. DFD levels are numbered 0,1 or 2 and occasionally go to even Level 3 or beyond. The necessary level of detail depends on the scope of what you are trying to accomplish.

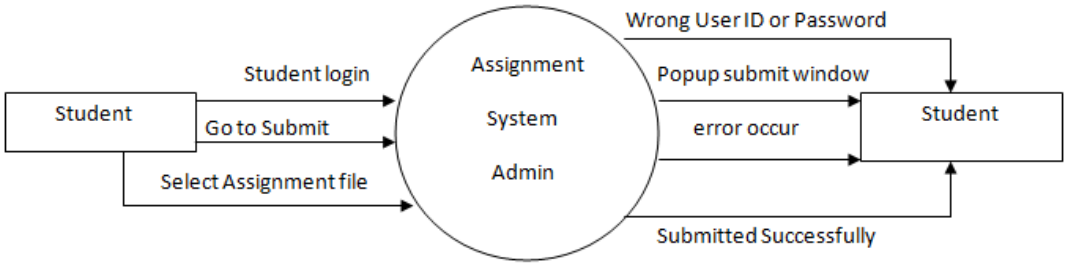
DFD 0 level

DFD Level 0 is also called a Context Diagram. It's a basic overview of the whole system or process being analyzed or modeled. It's designed to be an at-a-glance view, showing the system as a single high-level process, with its relationship to external entities. It should be easily understood by a wide audience, including stakeholders, business analysts, data analysts and developers.



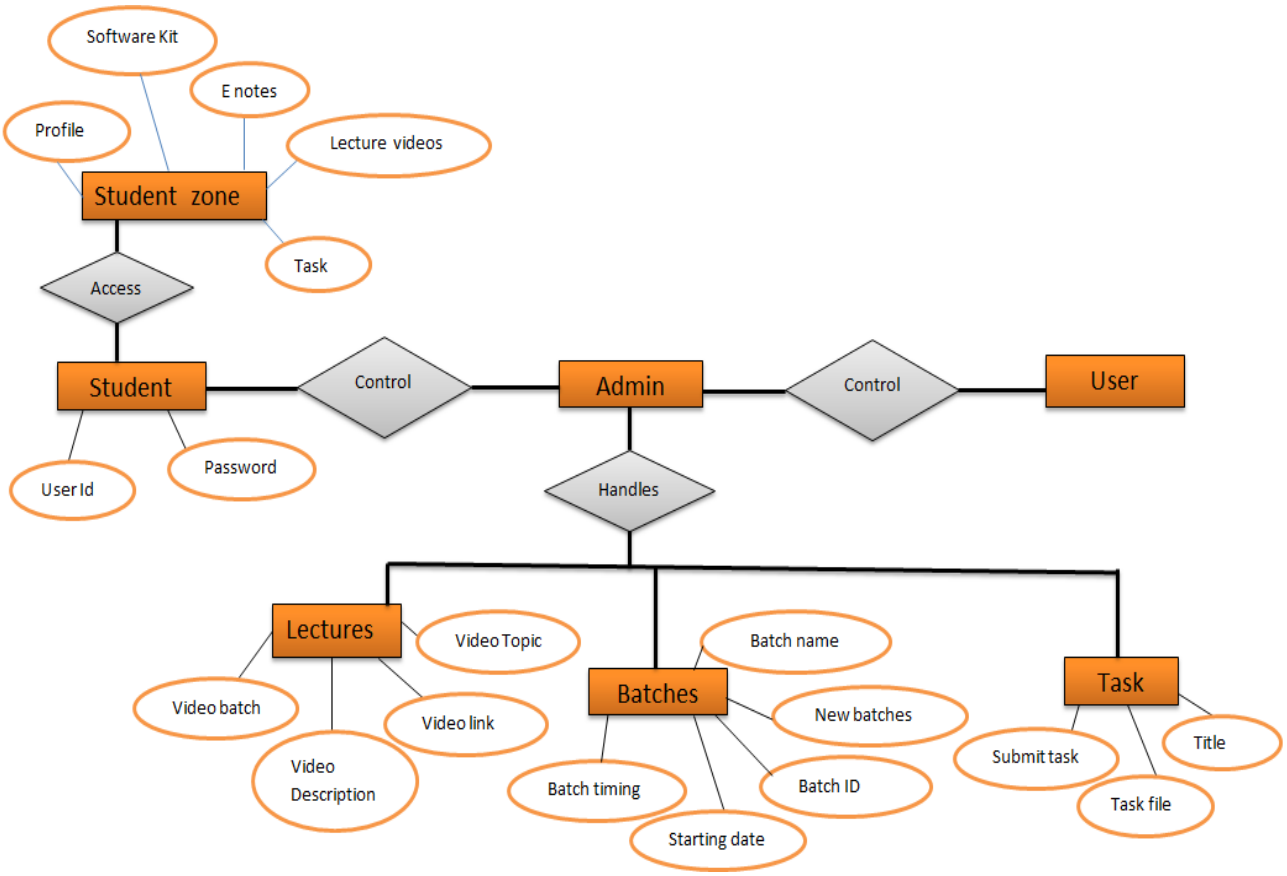
DFD 1 level

DFD Level 1 provides a more detailed breakout of pieces of the Context Level Diagram. You will highlight the main functions carried out by the system, as you break down the high-level process of the Context Diagram into its sub processes.



9.3 ER Diagram-

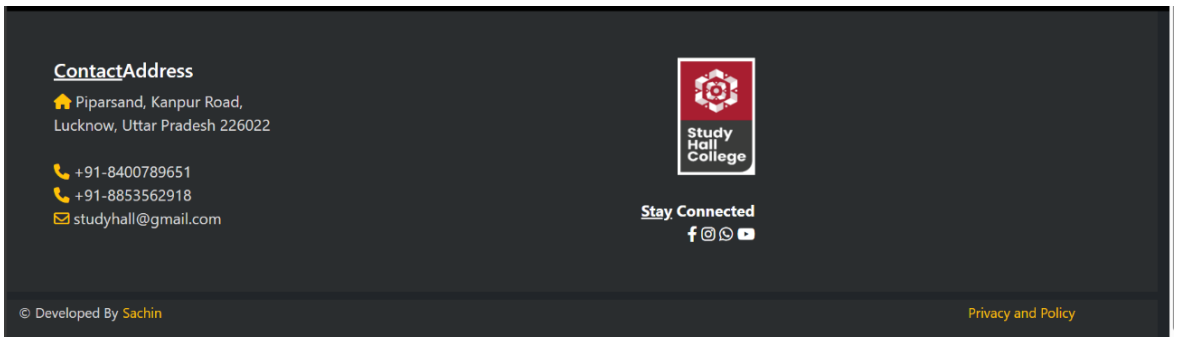
An Entity Relationship (ER) Diagram is a type of flowchart that illustrates how "entities" such as people, objects or concepts relate to each other within a system. ER Diagrams are most often used to design or debug relational databases in the fields of software engineering, business information systems, education and research. Also known as ERDs or ER Models, they use a defined set of symbols such as rectangles, diamonds, ovals and connecting lines to depict the interconnectedness of entities, relationships and their attributes. They mirror grammatical structure, with entities as nouns and relationships as verbs.



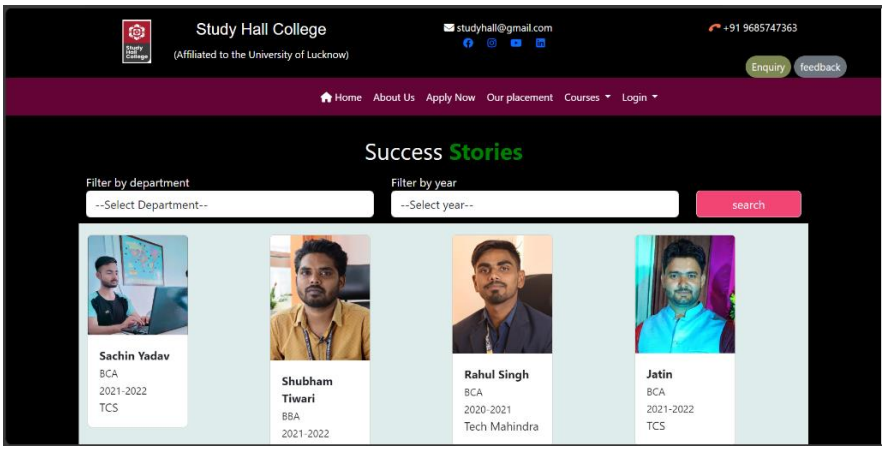
10. SNAP SHOTS

- Home page of the website:

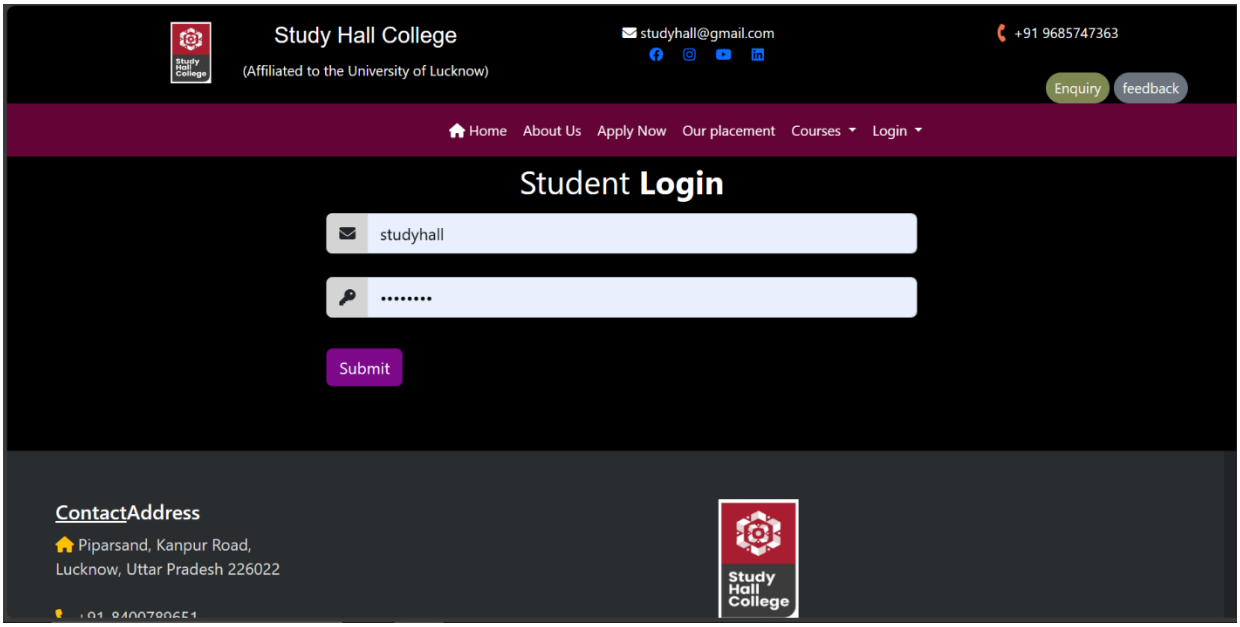
- Footer of the website:



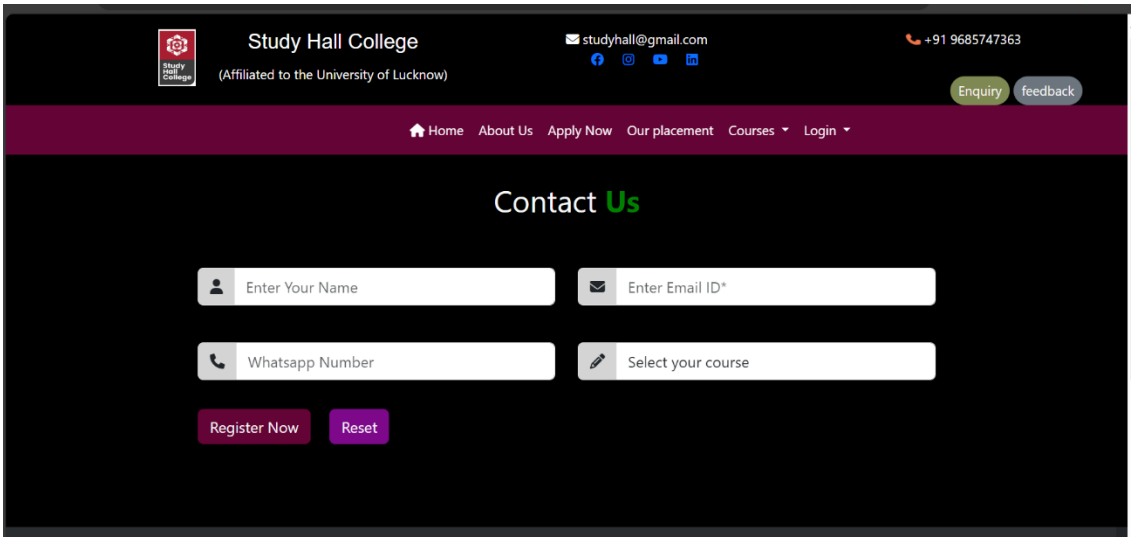
- Our placement :



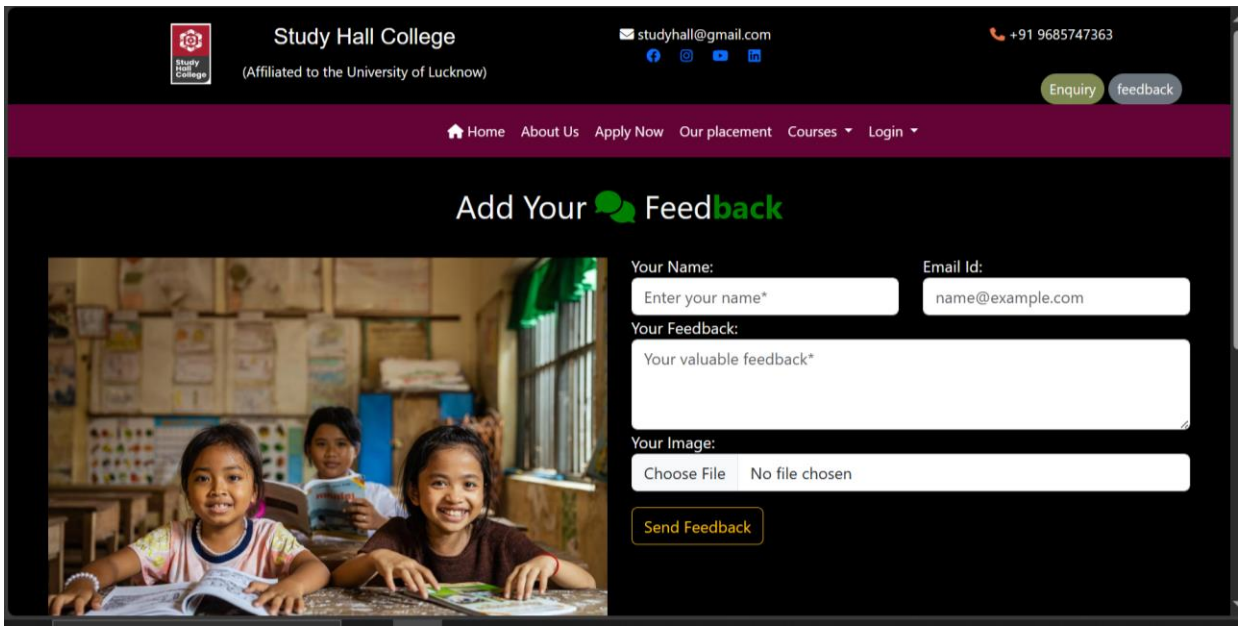
- Login form:



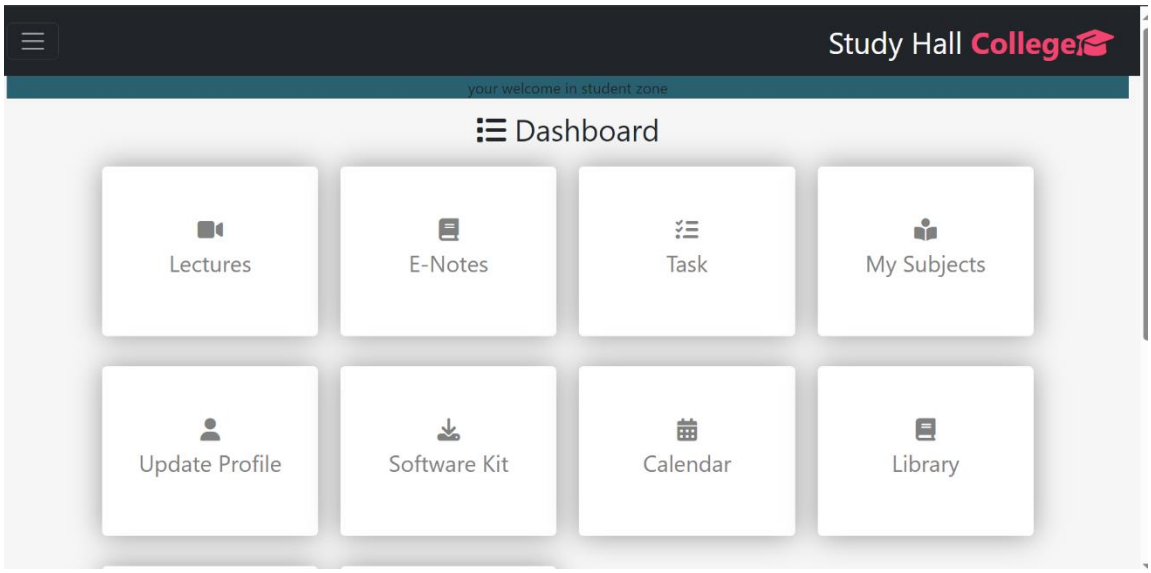
- Contact us form for any enquiry :



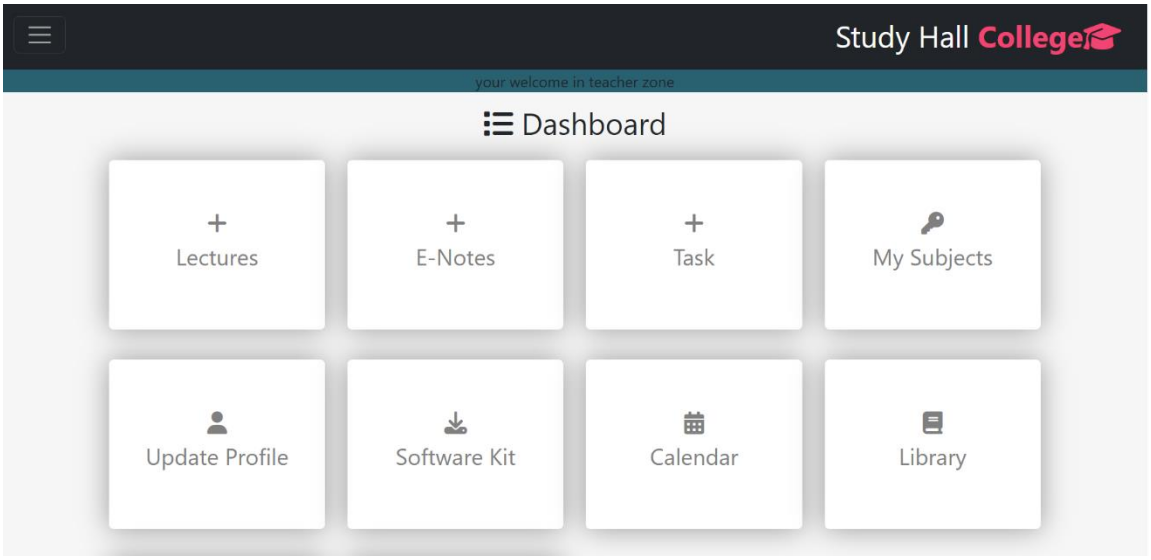
Feedback form:

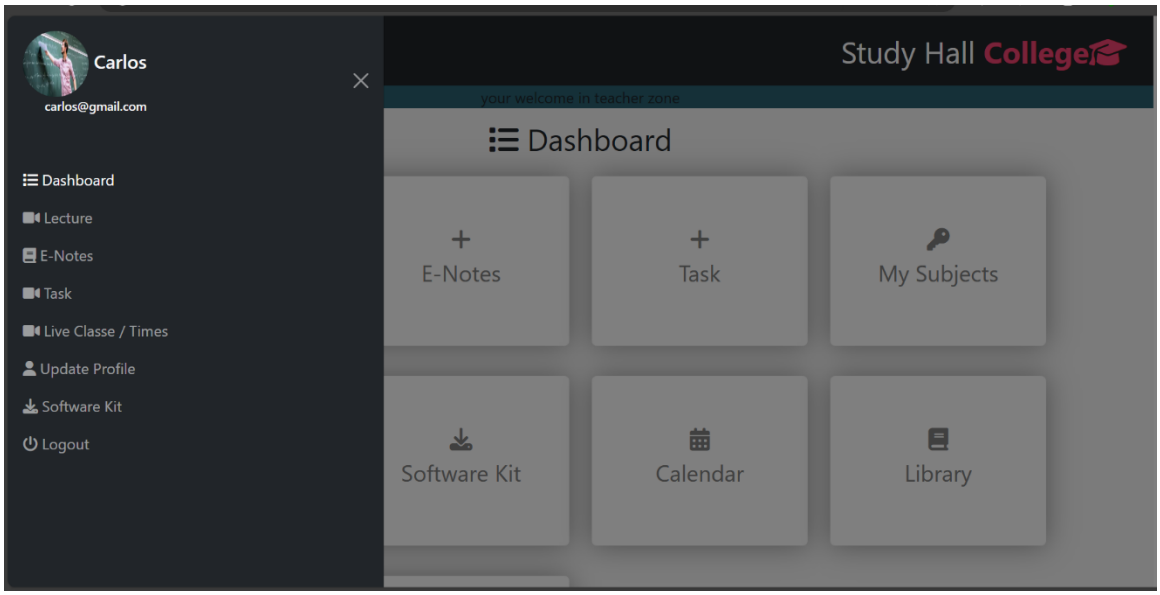


Student Dashboard.

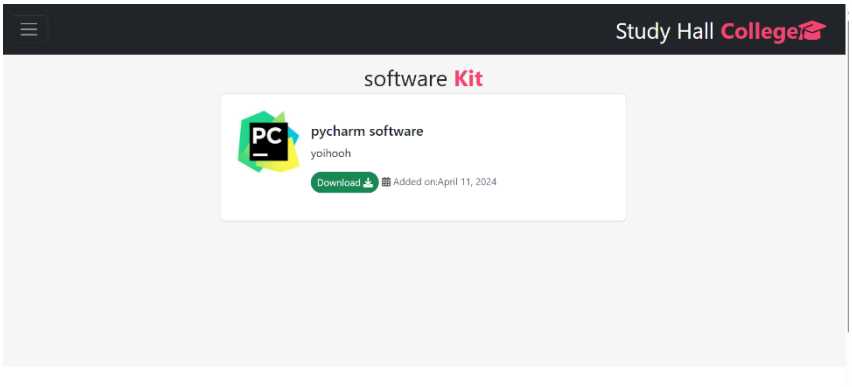


Teacher Dashboard

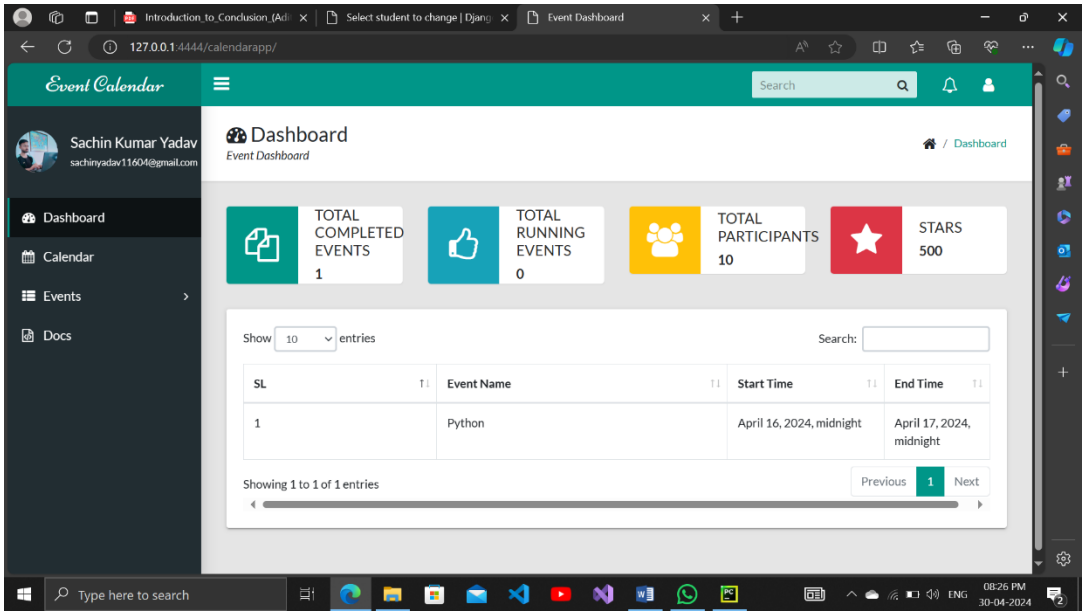




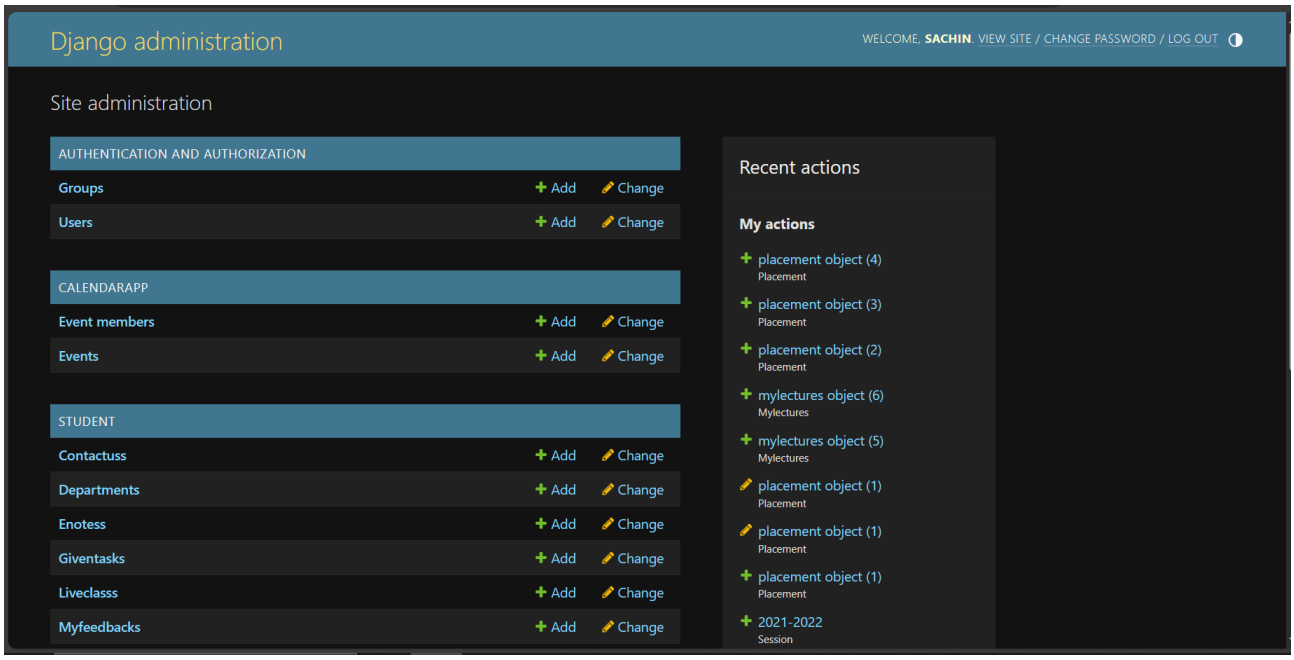
Software kit

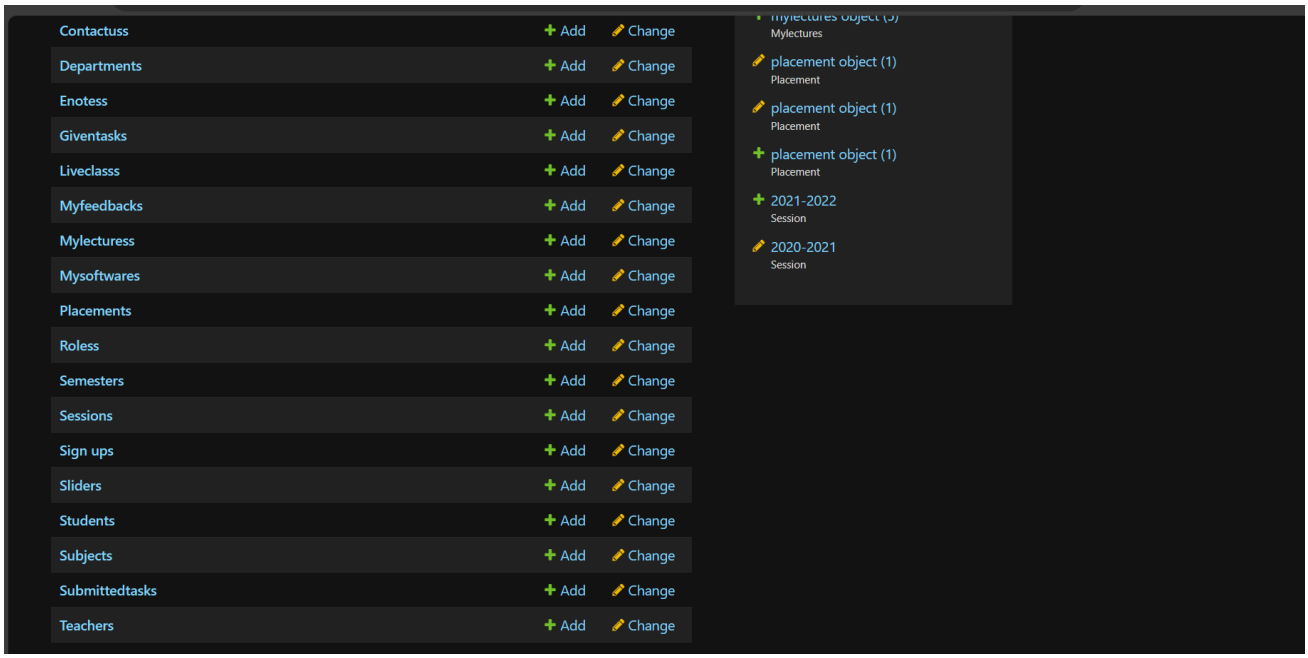


Calendar



Database Tables





11. TESTING

Software testing is the act of examining the artifacts and the behavior of the software under test by validation and verification. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but not necessarily limited to:

- Analyzing the product requirements for completeness and correctness in various contexts like industry perspective, business perspective, feasibility and viability of implementation, usability, performance, security, infrastructure considerations, etc.
- Reviewing the product architecture and the overall design of the product
- Working with product developers on improvement in coding techniques, design patterns, tests that can be written as part of code based on various techniques like boundary conditions, etc.
- Executing a program or application with the intent of examining behavior
- Reviewing the deployment infrastructure and associated scripts and automation
- Take part in production activities by using monitoring and observe ability techniques

Software testing can provide objective, independent information about the quality of software and risk of its failure to users or sponsors.

Testing Approaches

Following are main types of tests software undergoes:

1. Black Box Testing

Black-box testing (also known as functional testing) treats the software as a "black box," examining functionality without any knowledge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing, and specification-based testing.

Specification-based testing aims to test the functionality of software according to the applicable requirements.[This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the

expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case or leaves some parts of the program untested.

This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

2.White Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) verifies the internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing, an internal perspective of the system (the source code), as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g., in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration, and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Unit testing refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

Techniques used in white-box testing include:

- **API testing** – testing of the application using public and private APIs (application programming interfaces)
- **Code coverage** – creating tests to satisfy some criteria of code coverage (for example, the test designer can create tests to cause all statements in the program to be executed at least once)
- **Fault injection methods** – intentionally introducing faults to gauge the efficacy of testing strategies
- **Mutation testing methods**
- **Static testing methods**

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for:

- Function coverage, which reports on functions executed
- Statement coverage, which reports on the number of lines executed to complete the test
- Decision coverage, which reports on whether both the True and the False branch of a given test has been executed.

12. IMPLEMENTATION

Implementing a Learning Management System (LMS) involves several key steps. Here's a general outline:

1. Define Requirements:

- Identify the target audience (students, teachers, administrators).
- List the essential features such as course management, user authentication, content creation, assessments, etc.

2. Choose Technology Stack:

- Select a suitable programming language (e.g., Python, Java, PHP).
- Choose a web framework if you're building a web-based LMS.
- Consider using a database for data storage (e.g., MySQL, PostgreSQL).

3. Database Design:

- Create a database schema to store information about users, courses, assessments, etc.
- Establish relationships between different entities.

4. User Authentication:

- Implement a secure user authentication system to manage user accounts and roles.

5. Course Management:

- Develop features for creating, editing, and deleting courses.
- Allow instructors to upload course materials, videos, and other resources.

6. Content Delivery:

- Build a system for organizing and delivering course content efficiently.
- Consider supporting various file formats for content upload.

7. Assessment and Grading:

- Implement features for creating quizzes, assignments, and exams.
- Include a grading system and provide feedback to users.

8. User Interface (UI):

- Design an intuitive and user-friendly interface for easy navigation.
- Ensure responsiveness for various devices.

9. Notifications:

- Include a notification system for updates on courses, assessments, and other important events.

10. Reporting and Analytics:
    - Develop reporting tools for tracking user progress and performance.
    - Implement analytics to gather insights into user behavior.
11. Security:
    - Implement security measures to protect user data and ensure secure transactions.
    - Regularly update and patch the system for any security vulnerabilities.
12. Testing:
    - Conduct thorough testing to identify and fix bugs.
    - Perform usability testing to ensure a positive user experience.
13. Deployment:
    - Deploy the LMS on a server or a cloud platform.
    - Configure necessary settings for production.
14. Training and Documentation:
    - Provide training materials for users and administrators.
    - Document the system architecture and codebase for future reference.

Frontend Code:

index.html:

```
{% extends "userbase.html" %}
{% load static %}
{% block userblock %}


<div class="row mt-1">
            <div class="col-sm-6">
               <!-- carousel -->
               <div id="carouselExampleInterval" class="carousel slide" data-bs-ride="carousel">
                  <div class="carousel-inner">

                     <div class="carousel-item active" data-bs-interval="2000">
```

```
                    <img src="/{{sactive.slider_pic}}" class="d-block w-100" alt="image" style="max-height:
350px;">
                  </div>
                    {% for i in sdata%}

                  <div class="carousel-item" data-bs-interval="2000">
                    <img src="/{{i.slider_pic}}" class="d-block w-100" alt="image" style="max-height:
350px;">
                  </div>
                  {% endfor %}
                </div>
                <button class="carousel-control-prev" type="button" data-bs-
target="#carouselExampleInterval" data-bs-slide="prev">
                  <span class="carousel-control-prev-icon" aria-hidden="true"></span>
                  <span class="visually-hidden">Previous</span>
                </button>
                <button class="carousel-control-next" type="button" data-bs-
target="#carouselExampleInterval" data-bs-slide="next">
                  <span class="carousel-control-next-icon" aria-hidden="true"></span>
                  <span class="visually-hidden">Next</span>
                </button>
              </div>
              <!-- end crousel -->
            </div>
            <div class="col-sm-6 text-center">

              <div>
                <b style="font-size: 22px; color:white;">Announcement</b>
              </div>
```

```
            <marquee direction="up" scrollamount="5" scrolldelay="100" onmouseover=stop()
onmouseleave=start()>
                <div class="text-light">
                    <p>this is an example </p>
                    <p>this is an example </p>
                    <p>this is an example </p>
                    <p>this is an example </p>
                    <p>this is an example </p>
                    <p>this is an example </p>
                    <p>this is an example </p>

                </div>
            </marquee>
        </div>
</div>

{% endblock %}
```

studentlogin.html

```
{% extends 'userbase.html'%}
{% load static %}
{% block userblock %}
 <!--
<style>

   .input-group
     {
      height:40px;
      margin-bottom:15px;
```

```
    }
  .input-group-text
  {
   background:#d4d4d4;
  }
  .btn
  {

  margin-bottom:3%
  }

</style>
 -->
<div class="text-center fs-2 ">Student <b>Login</b></div>
<div class="row">
<div class="col-sm-6 mx-auto">
   <form method="post">
      {% csrf_token %}
      <div class="row">
          <div class="my-2">
             <div class="input-group">
               <i class="input-group-text py-0">
                  <i class="fa-solid fa-envelope"></i>
               </i>
               <input type="email" name="email" required placeholder="Enter Email ID*" class="form-
control"/>
               </div>
                  </div>
          <div>
               <div class="input-group">
```

```
                <i class="input-group-text py-0">
                    <i class="fa-solid fa-key"></i>
                </i>
                <input type="password" name="passwd" required placeholder="Enter Password*" class="form-
control"/>
                </div>
                    </div>
            </div>
        <div class="my-3">
            <button type="submit" value="login" class=" btn text-light"
style="background:#7d088a;">Submit</button>
        </div>

    </form>
</div>
</div>

{% endblock %}
```

feedback.html:

```
{% extends 'userbase.html' %}
{% load static %}
{% block userblock %}
<!--feedback form start-->
<div class="text-center fs-2 py-4 text-light">Add Your
    <i class="fa-solid fa-comments" style="color:green;"></i>
    Feed<b style="color:green;">back</b>
</div>
<div class="row mb-5">
  <div class="col-sm-12">
```

```
<div class="row mx-3">
 <div class="col-lg-6 col-md-12 col-sm-12">
  <img src="{% static 'images/educ.jpg' %}" class="img-fluid mt-1" style="max-height:400px;"/>
 </div>

 <div class="col-lg-6 col-md-12 col-sm-12">
    <form method="post" enctype="multipart/form-data">
        {% csrf_token %}
  <div class="row">
    <div class="col-lg-6 col-md-12 col-sm-12 ">
     Your Name:
     <input type="text" name="fname" class="form-control fs-6" placeholder="Enter your name*">
    </div>
    <div class="col-lg-6 col-md-12 col-sm-12 ">
     Email Id:
     <input type="email" name="femail" class="form-control "  placeholder="name@example.com">
    </div>
    <div class="col-lg-12">
     Your Feedback:
     <textarea class="form-control" name="fmsg" placeholder="Your valuable feedback*"
style="height:90px;"></textarea>
    </div>
    <div class="col-lg-12">
       Your Image:
       <input class="form-control" name="fp" type="file">
    </div>
     <div>
       <input type="submit" value="Send Feedback" class=" btn btn-outline-warning mt-3"/>
     </div>
   </div>
```

```
        </form>
      </div>
    </div>
        </div>
</div>

<!--feedback form end-->
{% endblock %}
```

contact.html:

```
{% extends 'userbase.html'%}
{% load static %}
{% block userblock %}

<div class="text-center text-light fs-2 mt-4 ">Contact<b class="txt-mycolor3"> Us</b></div>
<div class="row">
    <div class="col-sm-10 py-4 mx-auto">
        <div class="row mx-auto">
            <div class="col-sm-10 mx-auto">
                <form method="post" enctype="multipart/form-data">
                    {% csrf_token %}

                    <div class="row ">
                        <div class="col-lg-6 col-md-6 col-sm-12 pt-4">

                         <div class="input-group">
                            <i class="input-group-text">
                                <i class="fa-solid fa-user"></i>
                            </i>
```

```
                <input type="text" name="name" required placeholder="Enter Your Name" class="form-
control"/>
                </div>
                </div>

                <div class="col-lg-6 col-md-6 col-sm-12 pt-4">

                <div class="input-group">
            <i class="input-group-text py-0">
                <i class="fa-solid fa-envelope"></i>
            </i>
            <input type="email" name="email" required placeholder="Enter Email ID*" class="form-
control"/>
            </div>
                </div>
                <div class="col-lg-6 col-md-6 col-sm-12 pt-4">

            <div class="input-group">
            <i class="input-group-text py-0">
                <i class="fa-solid fa-phone"></i>
            </i>
            <input type="number" name="mobile" required placeholder="Whatsapp Number" class="form-
control"/>
            </div>
                </div>
                <div class="col-lg-6 col-md-6 col-sm-12 pt-4">
            <div class="input-group">
            <i class="input-group-text py-0">
                <i class="fa-solid fa-pencil"></i>
            </i>
```

```
                <select name="course" class="form-control">
                    <option>Select your course</option>
                    {% for i in course %}
                    <option value="{{i.department_name}}">{{i.department_name}}</option>
                    {% endfor %}
                </select>
                </div>
                    </div>

                <div class="col-lg-6 col-md-6 col-sm-12">
                    <input type="submit" value="Register Now" class=" btn bg-mycolor2 text-light my-3"
style="background:#7d088a;"/>
                <button type="reset" value="Reset" class=" btn text-light my-3 ms-3"
style="background:#7d088a;">Reset</button>
                </div>
            </div>
            </form>
          </div>
       </div>
    </div>
</div>
{% endblock %}
```

teacherlogin.html:

```
{% extends 'userbase.html'%}
{% load static %}
{% block userblock %}

<div class="text-center fs-2 text-light">Teacher <b>Login</b></div>
<div class="row">
```

```
<div class="col-sm-6 mx-auto">
   <form method="post">
      {% csrf_token %}
      <div class="row">
         <div class="my-2">
            <div class="input-group">
               <i class="input-group-text py-0">
                  <i class="fa-solid fa-envelope"></i>
               </i>
               <input type="email" name="email" required placeholder="Enter Email ID*" class="form-
control"/>
               </div>
         </div>
         <div>
               Password
               <div class="input-group">
               <i class="input-group-text py-0">
                  <i class="fa-solid fa-key"></i>
               </i>
               <input type="password" name="passwd" required placeholder="Enter Password*" class="form-
control"/>
               </div>
                  </div>
         </div>
      <div class="my-3">
         <button type="submit" value="login" class=" btn text-light"
style="background:#7d088a;">Submit</button>
      </div>

   </form>
```

```
</div>
</div>

{% endblock %}
```

Backend Code:

student/views.py :

```
from django.shortcuts import render
from django.http import HttpResponse
from .models import *

# Create your views here.
def index(request):

    return render(request,'student/index.html')

def lectures(request):
    department_id=request.session['department_id']
    semester_id=request.session.get('semester_id')
    cdata=mylectures.objects.filter(semester=semester_id,department=department_id).order_by('-id')
    md={"cdata":cdata}
    return render(request,'student/lectures.html',md)
def lecturesvideo(request):
    department_id=request.session.get('department_id')
    subject_id=request.GET.get('subject_id')
    vdata=mylectures.objects.filter(subject=subject_id,department=department_id)
    md={"vdata":vdata}
    return render(request,'student/lecturesvideo.html',md)
def enote(request):
    department_id=request.session.get('department_id')
```

```
    semester_id=request.session.get('semester_id')
    ndata = enotes.objects.filter(department=department_id,semester=semester_id)
    md = {"ndata": ndata}
    return render(request, 'student/notes.html', md)
def library(request):
    return render(request,'student/library.html')
def softwarekit(request):
    x=mysoftware.objects.all().order_by('-id')
    md={"sdata":x}
    return render(request,'student/softwarekit.html',md)
def mytask(request):
    x=giventask.objects.all().order_by('-id')
    md={"tdata":x}
    return render(request,'student/tasks.html',md)
def uprofile(request):
    user=request.session.get('user')
    udata=student.objects.filter(email=user)
    upd=student.objects.get(email=user)
    oldpasswd=request.POST.get('opasswd')
    md={"udata":udata}
    if request.method=="POST":
        if upd.passwd == oldpasswd:
            if request.POST.get('name'):
                upd.name = request.POST.get('name')
                upd.save()
                return HttpResponse("<script>alert('Your profile is updated
successfully...');location.href='/student/uprofile/'</script>")
            if request.POST.get('mobile'):
                upd.mobile = request.POST.get('mobile')
                upd.save()
```

```
            return HttpResponse("<script>alert('Your profile is updated
successfully...');location.href='/student/uprofile/'</script>")
        if request.POST.get('passwd'):
            upd.passwd = request.POST.get('passwd')
            upd.save()
            return HttpResponse("<script>alert('Your profile is updated
successfully...');location.href='/student/uprofile/'</script>")
        if request.FILES['fu']:
            upd.profile= request.FILES['fu']
            upd.save()
            return HttpResponse("<script>alert('Your profile is updated
successfully...');location.href='/student/uprofile/'</script>")
    else:
        return HttpResponse("<script>alert('wrong password....');location.href='/student/uprofile/'</script>")

#signup(name=name,mobile=mobile,passwd=passwd,college=college,course=course,profile=picture,email=u
ser).save()
    #return HttpResponse("<script>alert('Your profile is updated
successfully...');location.href='/student/uprofile/'</script>")
  return render(request,'student/uprofile.html',md)
def stask(request):
  user=request.session.get('user')
  if request.method=="POST":
    tid=request.POST.get('tid')
    title = request.POST.get('title')
    answer_file=request.FILES['fu']
    x=submittedtask.objects.filter(tid=tid,userid=user).count()
    if x==1:
        return HttpResponse("<script>alert('This task is already
submitted...');location.href='/student/tasks'</script>")
```

```
        else:
            submittedtask(title=title,tid=tid,answer_file=answer_file,userid=user).save()
            return  HttpResponse("<script>alert('Your task has been submitted
successfully...');location.href='/student/tasks'</script>")
    return render(request,'student/stask.html')
def myliveclasses(request):
    department_id = request.session.get('department_id')
    semester_id = request.session.get('semester_id')
    data=liveclass.objects.filter(department=department_id,semester=semester_id)
    md={"data":data}
    return render(request,'student/liveclasses.html',md)
def stask(request):
    user=request.session.get('user')
    if request.method=="POST":
        tid=request.POST.get('tid')
        title = request.POST.get('title')
        answer_file=request.FILES['fu']
        x=submittedtask.objects.filter(tid=tid,userid=user).count()
        if x==1:
            return HttpResponse("<script>alert('This task is already
submitted...');location.href='/student/tasks'</script>")
        else:
            submittedtask(title=title,tid=tid,answer_file=answer_file,userid=user).save()
            return  HttpResponse("<script>alert('Your task has been submitted
successfully...');location.href='/student/tasks'</script>")
    return render(request,'student/stask.html')
def logout(request):
    user=request.session.get('user')
    if user:
        del request.session['user']
```

```
        del request.session['userpic']
        del request.session['username']
        return HttpResponse("<script>location.href='/user/index/'</script>")
    return render(request,'student/logout.html')
```

user/views.py:

```
from django.shortcuts import render
from django.http import HttpResponse
from student.models import *

# Create your views here.
def index(request):
    slider_active=slider.objects.all().order_by('-id')[0]
    slider_pic=slider.objects.all().order_by('-id')[1:]
    mydict={"sactive":slider_active,"sdata":slider_pic}
    return render(request,'user/index.html',mydict)

def about(request):
    return render(request,"user/about.html")
def feedback(request):
    return render(request,'user/feedback.html')
def contact(request):
    course=department.objects.all().order_by('id')
    cdata={"course":course}
    if request.method=="POST":
        a=request.POST.get('name')
        b=request.POST.get('email')
        c=request.POST.get('mobile')
        d=request.POST.get('course')
        signUp(name=a,email=b,mobile=c,course=d).save()
```

```
    return render(request,'user/contact.html',cdata)


def feedback(request):
    if request.method=="POST":
        a=request.POST.get('fname')
        b=request.POST.get('femail')
        c=request.POST.get('fmsg')
        # d=request.FILES['fp']
        myfeedback(name=a,email=b,message=c).save()
        return HttpResponse("<script>alert('Thanks for giving me feedback');
location.href='/user/feedback/'</script>")
    return render(request,"user/feedback.html")


def studentlogin(request):
    if request.method=="POST":
        email=request.POST.get('email')
        passwd = request.POST.get('passwd')
        x=student.objects.filter(passwd=passwd,email=email).count()
        if x==1:
            request.session['user']=email
            y=student.objects.filter(email=email,passwd=passwd)
            request.session['userpic']=str(y[0].pic)
            request.session['username'] = str(y[0].name)
            request.session['department'] = str(y[0].department)
            request.session['department_id'] = str(y[0].department.id)
            request.session['semester'] = str(y[0].semester)
            request.session['semester_id'] = str(y[0].semester.id)

            return HttpResponse("<script>location.href='/student/index/'</script>")
        else:
```

```
            return HttpResponse("<script>alert('your username or password is
incorrect...');location.href='/user/studentlogin/'</script>")
    return render(request,'user/studentlogin.html')
def successstory(request):
    return render(request,'user/successstory.html')

def teacherlogin(request):
    if request.method=="POST":
        email=request.POST.get('email')
        passwd = request.POST.get('passwd')
        x=teacher.objects.filter(passwd=passwd,email=email).count()
        if x==1:
            request.session['user']=email
            y=teacher.objects.filter(email=email,passwd=passwd)
            request.session['userpic']=str(y[0].pic)
            request.session['username'] = str(y[0].name)
            request.session['department'] = str(y[0].department)
            request.session['department_id'] = str(y[0].department.id)

            return HttpResponse("<script>location.href='/teacher/index/'</script>")
        else:
            return HttpResponse("<script>alert('your username or password is
incorrect...');location.href='/user/teacherlogin/'</script>")
    return render(request,'user/teacherlogin.html')

def successstory(request):
    depart = request.GET.get('department')
    year = request.GET.get('year')

    ddata = department.objects.all().order_by('-id')
```

```
sdata = session.objects.all().order_by('-id')
pdata = placement.objects.all().order_by('-id')

if year is not int and depart is int :
    year=sdata[0]
    if depart is not None and year is not None:
        pdata = placement.objects.filter(department=depart, session=year)
    else:
        pdata = placement.objects.all().order_by('-id')
elif depart is not int and year is int:
    depart=ddata[0]

    if depart is not None and year is not None:
        pdata = placement.objects.filter(department=depart, session=year)
    else:
        pdata = placement.objects.all().order_by('-id')

# if depart is not None and year is None:
#     pdata = placement.objects.filter(department=depart)
# elif depart is None and year is not None:
#     pdata = placement.objects.filter(session=year)
# elif depart is not None and year is not None:
#     pdata = placement.objects.filter(department=depart, session=year)
# else:
#     pdata = placement.objects.all().order_by('-id')


md={"cdata":ddata,"sdata":sdata,"pdata":pdata}
return render(request,"user/successstory.html",md)
```

Database Code:

student/model.py:

```
from django.db import models

# Create your models here.

class department(models.Model):
    department_name=models.CharField(max_length=100,null=True)
    def __str__(self):
        return self.department_name

class semester(models.Model):
    semester=models.CharField(max_length=50,null=True)
    def __str__(self):
        return self.semester

class roles(models.Model):
    roles=models.CharField(max_length=100,null=True)
    def __str__(self):
        return self.roles


class session(models.Model):
    session=models.CharField(max_length=100,null=True)
    def __str__(self):
        return self.session

class student(models.Model):
    student_ID=models.CharField(max_length=15,null=True)
    name=models.CharField(max_length=200, null=True)
```

```
    email=models.EmailField(primary_key=True,max_length=200)
    admit_year=models.DateField(null=True)
    passwd=models.CharField(max_length=100,null=True)
    mobile=models.IntegerField(null=True)
    pic=models.ImageField(upload_to='static/profie/',null=True)
    role=models.CharField(max_length=100,null=True,default='Student')
    department=models.ForeignKey(department,on_delete=models.CASCADE)
    semester=models.ForeignKey(semester,on_delete=models.CASCADE)
    def __str__(self):
        return self.name

class teacher(models.Model):
    teacher_Id=models.CharField(max_length=15,null=True)
    name=models.CharField(max_length=100,null=True)
    email=models.EmailField(primary_key=True,max_length=150)
    passwd=models.CharField(max_length=100,null=True)
    department=models.ForeignKey(department,on_delete=models.CASCADE,null=True)
    role = models.ForeignKey(roles, on_delete=models.CASCADE,null=True)
    mobile=models.IntegerField(null=True)
    joining_date=models.DateField(null=True)
    pic=models.ImageField(upload_to='static/profie/',null=True)
    address=models.TextField(null=True)
    salary=models.IntegerField(null=True)
    def __str__(self):
        return self.name

class subject(models.Model):
    subject_code=models.CharField(max_length=100,null=True)
    subject_name=models.CharField(max_length=200,null=True)
    department=models.ForeignKey(department,on_delete=models.CASCADE)
```

```
    semester=models.ForeignKey(semester,on_delete=models.CASCADE)
    teacher=models.ForeignKey(teacher,on_delete=models.CASCADE)
    date=models.DateField(null=True)
    def __str__(self):
        return self.subject_name

class contactus(models.Model):
    name=models.CharField(max_length=200,null=True)
    email=models.CharField(max_length=100,null=True)
    mobile=models.CharField(max_length=25,null=True)
    message=models.TextField(null=True)
    def __str__(self):
        return self.name

class myfeedback(models.Model):
    name=models.CharField(max_length=200,null=True)
    email=models.CharField(max_length=200,null=True)
    message=models.TextField(null=True)


class placement(models.Model):
    student_picture=models.ImageField(upload_to='static/placement',null=True)
    student_name=models.CharField(max_length=100,null=True)
    department=models.ForeignKey(department,on_delete=models.CASCADE,null=True)
    session = models.ForeignKey(session, on_delete=models.CASCADE,null=True)
    company_name=models.CharField(max_length=200,null=True)


class mylectures(models.Model):
    subject = models.ForeignKey(subject, on_delete=models.CASCADE, null=True)
```

```
    department = models.ForeignKey(department, on_delete=models.CASCADE, null=True)
    semester = models.ForeignKey(semester, on_delete=models.CASCADE, null=True)
    vlink = models.CharField(max_length=300, null=True)
    thumbnail = models.ImageField(upload_to='static/video', null=True)
    video_description = models.TextField(null=True)
    added_date = models.DateField(null=True)


class enotes(models.Model):
    subject = models.ForeignKey(subject, on_delete=models.CASCADE, null=True)
    note_pic = models.ImageField(upload_to='static/enotes', null=True)
    notes_pdf = models.FileField(upload_to='static/pdf', null=True)
    department = models.ForeignKey(department, on_delete=models.CASCADE, null=True)
    semester = models.ForeignKey(semester, on_delete=models.CASCADE, null=True)
    added_date = models.DateField(null=True)


class giventask(models.Model):
    department = models.ForeignKey(department, on_delete=models.CASCADE, null=True)
    semester = models.ForeignKey(semester, on_delete=models.CASCADE, null=True)
    subject = models.ForeignKey(subject, on_delete=models.CASCADE, null=True)
    task_file = models.FileField(upload_to='static/task', null=True)
    added_date = models.DateField(null=True)

class submittedtask(models.Model):
    subject = models.ForeignKey(subject, on_delete=models.CASCADE, null=True)
    answer_file = models.FileField(upload_to='static/submittedtask', null=True)
    taskid = models.CharField(max_length=20, null=True)
    userid = models.CharField(max_length=200, null=True)
    marks = models.CharField(max_length=200, null=True)
```

```
    submit_date = models.DateField(null=True)
    marks_date = models.DateField(null=True)


class slider(models.Model):
    slider_pic = models.ImageField(upload_to='static/slider/', null=True)

class signUp(models.Model):
    name=models.CharField(max_length=100, null=True)
    email=models.EmailField(max_length=150,null=True)
    mobile=models.IntegerField(null=True)
    course=models.CharField(max_length=100,null=True)

class mysoftware(models.Model):
    software_title=models.CharField(max_length=100,null=True)
    software_description=models.TextField(null=True)
    link=models.TextField(null=True)
    software_picture=models.ImageField(upload_to='static/softwarepic/',null=True)
    software_date=models.DateField()

class liveclass(models.Model):
    department = models.ForeignKey(department, on_delete=models.CASCADE, null=True)
    semester = models.ForeignKey(semester, on_delete=models.CASCADE, null=True)
    subject = models.ForeignKey(subject, on_delete=models.CASCADE, null=True)
    link = models.TextField(null=True)
    teacher=models.ForeignKey(teacher,on_delete=models.CASCADE,null=True)
    date=models.DateField(null=True)
    time=models.TimeField(null=True)
```

student/urls.py:

```
from django.urls import path
from . import views

urlpatterns=[
    path('index/', views.index),
    path('', views.index),
    path('lectures/', views.lectures),
    path('lecturesvideo/',views.lecturesvideo),
    path('notes/',views.enote),
    path('library/',views.library),
    path('softwarekit/',views.softwarekit),
    path('tasks/',views.mytask),
    path('uprofile/',views.uprofile),
    path('liveclasses/', views.myliveclasses),
    path('stask/',views.stask),
    path('logout/', views.logout),

]
```

teacher/urls.py:

```
from django.urls import path
from . import views

urlpatterns=[
    path('index/', views.index),
    path('', views.index),
    path('lecturedepartment/', views.lectures),
    path('lectures/',views.lecturesvideo),
    path('notes/',views.enote),
```

```
    path('library/',views.library),
    path('softwarekit/',views.softwarekit),
    path('tasks/',views.mytask),
    path('uprofile/',views.uprofile),
    path('liveclasses/', views.myliveclasses),
    path('stask/',views.stask),
]
```

13. Limitations of Project

- Payment method is not included.
- There is no doubt clear option in this project.
- When task is submitted by the students, teacher can only show the marks but can’t show the checked task to the students so that students are unable to check their errors.
- There is no job portal option.

14. FUTURE SCOPE

The future scope of Learning Management Systems (LMS) is likely to be influenced by emerging trends in education, technology, and workforce development. Here are some potential areas of growth and development for LMS:

11.1 Personalized Learning:

LMS platforms are expected to increasingly focus on personalized learning experiences. Adaptive learning technologies will cater to individual learner needs, offering customized content, assessments, and pacing.

11.2 Microlearning and Mobile Learning:

The future of LMS is likely to include greater support for microlearning, delivering content in small, easily digestible segments. Additionally, LMS platforms are expected to be optimized for mobile learning, allowing users to access educational materials on various devices.

11.3 AI and Machine Learning Integration:

The integration of artificial intelligence (AI) and machine learning (ML) will play a significant role in LMS. These technologies can facilitate adaptive learning, automate assessments, and provide intelligent insights into learner progress.

11.4 Social Learning and Collaboration:

LMS platforms will likely emphasize social learning features, enabling learners to collaborate, share insights, and engage in discussions. Social elements can enhance the sense of community and foster a collaborative learning environment.

11.5 Competency-Based Learning:

Future LMS platforms may focus on competency-based learning, where learners progress based on their mastery of specific skills or competencies rather than fixed timeframes. This approach aligns with personalized learning and competency development.

11.6 Continuous Learning in the Workplace:

LMS will likely play a crucial role in supporting continuous learning in the workplace. Integration with performance management systems and a focus on upskilling and reskilling will be essential for workforce development.

The future of LMS will be shaped by advancements in educational technology, changing pedagogical approaches, and the evolving needs of learners and organizations. Keeping pace with these trends will be essential for LMS developers and educators looking to provide effective and future-ready learning solutions.

15. REFERENCES

Search Engine:

www.google.com

Online Material:

www.techpile.in

www.javatpoint.com

www.tutorialspoint.com

www.W3Schools.com

www.djangoproject.com

www.python.org