Acharya Prafulla Chandra College

New Barrackpur, Kolkata
West Bengal 700131
Phone: 033 2537 3297

Project report on *LMS Application*, a web based learning management application for the online delivery of educational courses and materials.

By Dwip Shekhar Mondal and Samay Sarkar, under the supervision of Ms. Debjani Bhattacharjee, department of computer science.

Department of computer science, Acharya prafulla chandra college, New Barrackpur, Kolkata.

General info

This is a major project submitted for the fulfilment of one the requirements for the award for the degree of Bachelor of Science (honours), Computer Science.
This project was done under guidance and supervision of Ms. Debjani Bhattacharjee, Faculty of Computer Science, Acharya Prafulla College .

Project info

This is a Full-stack web development project titled "**LMS Application**" aimed for building a learning management system **focusing on the requirements specified by the client/supervisor**.

Typically, a Full-stack web application uses a technology stack, i.e; a set of technologies that are stacked together to build any application. Popularly known as a technology infrastructure or solutions stack, technology stack has become essential for building easy-to-maintain, scalable web applications.

In this project we have used a technology stack consisting of **Nodejs, Express js, Ejs, HTML5, CSS3, Javascript, jQuery,** and other build-in or external libraries for building the Front-end (User-interface) and the Back-end of the application. **MySql** is used as a relational database for persistent data storage.

Keywords: ***learning management system (LMS), Web application, HTML5, CSS3, Nodejs, Express js, Ejs, javascript, MySql.***

Abstract

Learning management systems were designed to identify training and learning gaps, using analytical data and reporting. LMSs are focused on online learning delivery but support a range of uses, acting as a platform for online content, including courses, both asynchronous based and synchronous based. In the higher education space, an LMS may offer classroom management for instructor-led training or a flipped classroom.

Modern LMSs include intelligent algorithms to make automated recommendations for courses based on a user's skill profile as well as extract metadata from learning materials to make such recommendations even more accurate.

LMS, which is also referred as Course Management System (CMS) provides workspaces to **facilitate information sharing and communication among students and lecturers to participate in course activities**.

PROJECT DECLARATION

I/we, hereby declare with honesty that, the work in this report is original and has been done by us under the guidance of our supervisor(s) and also state that,

- The work has not been submitted to any other institute for any degree or diploma.
- The piece of work, or a part of the piece of work has not been submitted for more than one purpose (e.g. to satisfy the requirements in two different courses) without declaration.
- We have followed the guidelines provided by our supervisor and kept the advices given by them in mind when preparing this work/report for our assigned project.
- We have conformed to the norms and guidelines given in the Ethical Code of Conduct of this Institute
- I am/we are aware of the affiliated University's policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the University.
- I/we declare that I/we have not distributed/ shared/ copied any materials without the consent from the original source/ owner of the source material.
- Whenever we used any material (theoretical analysis, figures, statistical data, and text) from external sources, we have given them the due credit by referencing every external source in the references section of the report.
- All members of the group have read and checked that all parts of the work/report, irrespective of whether they are contributed by individual members or all members as a group, are original except for materials from external sources which are explicitly acknowledged as declared above.
- We are aware that all members of the group should be held responsible and liable to disciplinary actions, irrespective of whether he/she has signed the declaration and whether he/she has contributed, directly or indirectly, to the problematic contents.

Project team members

The below table contains info about the members involved in building the project and/or preparing the report/work containing detailed steps that were followed in preparing the final submission.

Project supervisor: Mrs. Debjani Bhattacharjee, department of computer science.

Serial no	Registration no	Name of member
1	1011911401022	Dwip Shekhar Mondal
2	1011911401035	Samay Sarkar

Acharya Prafulla Chandra College

New Barrackpur, Kolkata
West Bengal 700131

BONAFIDE CERTIFICATE

This is to certify that the Software Engineering project titled "**LMS Application**" (web based learning management system), is a bonafide work done by **Dwip Shekhar Mondal** and **Samay Sarkar** under guidance of **Ms. Debjani Bhattacharjee,** faculty of Computer Science in fulfillment of one of the requirements for the award of the degree of Bachelor of Science(honours) in Computer Science. This also certifies that all the team member(s) involved have successfully submitted the project, following the guidelines of authenticity provided by the institute and the affiliated University.

SIGNATURE

Prof. Joydeb Das Biswas
Head of Department, Computer Science
Acharya Prafulla Chandra College

SIGNATURE

Ms. Debjani Bhattacharjee (Supervisor)
Faculty of Computer Science
Acharya Prafulla Chandra College

Acknowledgement

We would like to express our sincerest and deepest sense of gratitude towards our guide/**supervisor Ms. Debjani Bhattacharjee**, faculty of Computer Science for the assistance, valuable guidance, and co-operation in carrying out / finalizing this project / report successfully. We were able to successfully develop and finalize the project with the help of the faculty members of our department (Department of computer science) of our institute and we are extremely grateful to all of them. We would also like to take the opportunity to thank the Head of our department **Prof. Joydeb Das Biswas** for providing the required facilities which immensely helped in developing and completing this project.

Sincerely,

1. Dwip Shekhar Mondal
2. Samay Sarkar

1. Introduction

1.1 Definition and features of learning management systems (LMS)

A learning management system (LMS) is a type of software application for the administration, documentation, tracking, reporting, automation, and delivery of educational courses, training programs, or learning and development programs. The learning management system concept emerged directly from e-Learning. Learning management systems make up the largest segment of the learning system market. The first introduction of the LMS was in the late 1990s. Learning management systems have faced a massive growth in usage due to the emphasis on remote learning during the COVID-19 pandemic.

An LMS delivers and manages all types of content, including video, courses, and documents. In the education and higher education markets, an LMS will include a variety of functionality that is similar to corporate but will have features such as rubrics, teacher and instructor-facilitated learning, a discussion board, and often the use of a syllabus.

Learning management systems may be used to create **professionally structured course content**. The teacher can add text, images, videos, pdfs, tables, links and text formatting, interactive tests, slideshows etc. Moreover, they can create different types of users, such as teachers, students, parents, visitors and editors (hierarchies). It helps control which content a **student can access, track studying progress and engage** students with contact tools. **Teachers can manage courses and modules**, enroll students or set up self-enrollment.

Through online learning management systems students can either **learn asynchronously** (on demand, self-paced) through course content such as pre-recorded videos, **PDF, PPT, SCORM** (Sharable Content Object Reference Model) or they can undertake synchronous learning through mediums such as Webinars, Online classrooms i.e online meetings through google meet, zoom, teams etc.

An LMS can enable **instructors to create automated assessments and assignments for learners**, which are accessible and submitted online. Most platforms allow a variety of different question types such as: one/multi-line answer; multiple choice answer; ordering; free text; matching; essay; true or false/yes or no; fill in the gaps; agreement scale and offline tasks.

Students' exchange of feedback both with teachers and their peers is possible through LMS. Teachers may create discussion groups or **provide an interactive comment section in the virtual classrooms** to allow students feedback, share their knowledge on topics and increase the interaction in course. Students' feedback is an instrument which helps teachers to improve their work, helps identify what to add or remove from a course, and ensures students feel comfortable and included.

1.2 Motivation and Objective of the project

LMS tools are created for making knowledge-sharing easier. The whole system was created to work across different web-based platforms. Technology has found its way in both education and the business world. With LMS tools, businesses receive a lot of benefits, as their employees can acquire knowledge without burning a hole in the pocket. On the education side of things, educational institutes are now in a position to work with students who are thousands of miles away. Plus, the functionality of these platforms makes the learning process easier than ever before.

This Project is developed with the aim to make an improvement upon the traditional offline learning process, by building an interface between the students and teachers and the institute, which **allows the students to keep track of their current academic information**, and **allows the teachers to better manage and organize their courses and classrooms and the study materials**, and overall reduces the various hardships or shortcoming of the traditional offline learning process. This allows better transparency between the students and instructors/teachers of the institute.

The development of this project is aimed to **provide a facility of virtual classrooms** for any course taught by any teacher where the teacher can post study materials that were taught in offline classrooms or in an online classroom i.e online meeting, this helps students who cannot attend all the classes, it also provides an interactive comment section where students as well as teachers can express their thoughts or opinions. This helps **increase transparency between teachers and students**.

1.3 Overview of the project LMS Application

Our project titled "LMS Application" is developed to make improvements upon the existing offline learning process and aims to make the process of learning through online mode more accessible and manageable by connecting students, teachers with the institution through an online interface that is LMS application. The required software and hardware for utilizing the interface is easily available and easy to work with. It provides students an interface for fast and secure access of available courses, classrooms, and study materials, and provides virtual classrooms for accessing course materials and information of classes with respect to the courses. It also provides teachers the ability to select courses to teach, create virtual classrooms for the courses, upload course materials and study materials. Once the users (students or teachers) register into the interface and login, they can securely access all the features of the application. The user data is protected by the application's authentication modules, it also provides methods for 2-step verification for each login for any user which adds another layer of security.

1.3.1 Functionalities provided by LMS Application

Below is a list of features our project/web application titled LMS Application provides.

- It **Provides two user roles that are student and teacher**, and provides different features based on the user type/role.
- **Provides a registration and login interface to each user role**, access to the features is restricted if the user requesting for access is not registered or logged into the LMS Application portal. This is essential for security of user data as well as for integrity and validity of application data in general.
- The LMS Application runs a stringent authentication system so it is less prone to attack from spammers and hackers.
- It also **provides a module for 2-step verification through a totp based solution for each login** for any user which adds another layer of security.
- Through the LMS Application that is by the institute, **a student of the institute can register and log into the interface** using **institute linked info** and other details which includes **email address, name, password** etc.
- Same as Students, **Teachers of the institute can register and log into the interface** using their institute linked info and other details like **email address,password, name** and others.
- **Provides a personalized Dashboard for student and teachers** where **students can see their current academic information such as current term, department info**, **courses** that the student is currently enrolled in, or the **classrooms in which the student is currently joined** or any **notifications from the joined classrooms** and other personalized information such as student same, display picture etc and **teachers** can see their respective personalized information such the **courses a teacher is currently teaching, classrooms created by the teacher or department information** into and other information such as **display picture, name** etc.
- Provides an interface for students to **access, enroll, leave courses** provided by the institute with respect to the student's current institute linked academic info such as department info and current term.
- Provides an interface for students to **access, join, leave any available classrooms** for any courses that the student is currently enrolled in. And students can also **access the classrooms, access materials, posts, comments** and are able to **interact with the teachers through the interactive comment section provided in the classrooms**.
- Provides an interface for students and teachers to **update** their **general information** such as **update email address, update password, upload or update display picture** etc.
- Provides an interface to the teachers to that allows **access courses available to teach** according to the institute linked info, allows teachers to **create, update, delete classrooms** for any courses the teacher is teaching, also **allows the teachers to post on classrooms** and **upload course contents, study materials** etc also allows the teachers to **interact with students through interactive comment section in each post** of the classrooms.
- It allows the students and teachers to **access their respective routine/schedule** with respect to the courses in which a student is enrolled for every user that is a registered student and with respect to the courses a teacher is currently teaching for every registered teacher.

1.3.2 Scope of improvements

Despite adding quite a number of features as specified in the software requirement specification, additional functionalities, outside of the scope of the SRS, can be developed and incorporated to enhance the capabilities and performance but due to the constraint of time and for other reasons we were unable to incorporate those developments into the web application. These developments can definitely be incorporated into the application in future or at the later stages of development to improve overall capabilities of this application. Some of these functionalities include

- Adding a calendar to build a reminder module for reminding the students about the upcoming classes, meetings, informing about the upcoming agendas etc.
- Building an online meeting platform so that we don't have to rely on external meeting platforms to conduct online classes, meetings etc.
- Effort can be given to develop a system which will take automatic attendance from the students in classes.
- To further enhance the classroom features, turning in assignment works can be made available to the student also.
- Classroom notification system can also be improved.
- From the back-end side of things load balancing can be incorporated to improve performance in hag traffic environments.
- Also a module can be developed to conduct online examinations for students.
- Also The front-end/ UI can be greatly improved by using some popular frameworks like Reactjs, Angular to provide better user experience.

2. Requirements engineering

2.1 Software requirement specifications

A software requirements specification (SRS) is a description of a software system to be developed. It is modelled after business requirements specification (CONOPS). The software requirements specification (SRS) lays out **functional** and **non-functional** requirements, and it may include a set of use cases that describe user interactions that the software must provide to the user for perfect interaction.Software requirements specification is a rigorous assessment of requirements before the more specific system design stages, and its goal is to reduce later redesign. It should also provide a realistic basis for estimating product costs, risks, and schedules.Used appropriately, software requirements specifications can help prevent software project failure.

The software requirements specification lists sufficient and necessary requirements for the project development. To derive the requirements, the developer needs to have a clear and thorough understanding of the products under development. This is achieved through detailed and continuous communications with the project team and customer throughout the software development process.

2.2 Functional requirements of LMS Application

- There should be two types of user groups i.e **teacher** and **student**, teachers will produce contents and materials and students will costume the contents produced by teachers.
- Both user types should be able to **register with their institute linked info and other details and log into the platform**.
- Students should be able to **access, enroll or leave courses** provided by the institute that is using the platform with respect to their current academic information.
- Students should be able to **access, join or leave any virtual classrooms created by teachers** for any courses that the student is currently enrolled in.

- Both students and teachers should be able to access **a personalized Dashboard for student and teachers** where **students can see their current academic information such as current term, department info**, **courses** that the student is currently enrolled in, or the **classrooms in which the student is currently joined** or any **notifications from the joined classrooms** and other personalized information such as student same, display picture etc and **teachers** can see their respective personalized information such the **courses a teacher is currently teaching, classrooms created by the teacher or department information** into and other information such as **display picture, name** etc.
- Students are able to **take admissions into different terms/ semesters**.
- Students are able to **see their class schedule with respect to the courses** that the user is currently enrolled in or is teaching.
- **Students should be able to get notifications** when a **teacher posts something in the classrooms via email** or other means.
- Students can access **posts, download class contents i.e study materials, and comment on posts** in the **virtual classrooms.**
- **Teachers can join a course to teach** or opt out of the course with respect to their institute linked information.
- **Teachers can create or delete virtual classrooms** for each course they opted for teaching.
- **Teachers can create or delete posts, to announce something, upload class contents, or to send online classroom meeting links**.
- Teachers can also **comment on posts to interact with the students** or can delete their comments.
- Both teachers and students should be able to **update their general credentials such as email address or password** and they can **upload or update their display picture**.
- The application should provide robust information **security matures** to **ensure the security of user data** as well as all other information.

2.3 Non-functional requirements of LMS Application

Reliability: The application would be reliable, provide and produce reliable and accurate information and should always be accessible to access to information.
Responsiveness: The r**esponse time should be as low as possible** following the recommended industry practices so that students and faculty should feel good while using the virtual classroom systems or the application as a whole.
Availability: **24 X 7 availability** should be there so that students as well as teachers can use it at any time according to their convenience **without any downtime** or issues.
Scalability: Number of users using the application will mainly depend on the server load, server processing capacity and its memory. It should **scale to the maximum number of users**.
Security: **HTTPS** enables access to web applications to secure access of confidential data (student information). Database Access There will be **no external access to the database**, except through the application's backend's **REST API** with **JWT and CSRF token enabled security measures**. Administrators of the system will have full database administration rights and teachers may have access to a copy of parts of the VCS database, for editing purposes. **2-step verification with totp should be incorporated** in order to add another layer of **login security** to mitigate unauthorized access to the platform or reduce spam attacks.

2.4 External interface requirements of LMS Application

2.4.1 User interfaces

The application GUI provides menus, buttons, panes, containers, grids allowing for easy control by a keyboard and a mouse. Below list contain the list of user interface requirements of the application.

- The **Login UI** enables **users to integrate with the contents of the application** and allows the **access of all the features provided by the application**. **2-step verification** is provided via a **totp** based solution for each login request for added security.

- The **Register UI** enables a new user to **register into the system** and allows the user to access the facilities provided by the application.
- The **Dashboard UI** allows the users i.e teachers and students to see their **current academic information such as current term, department info**, **courses** that the student is currently enrolled in, or the **classrooms in which the student is currently joined** or any **notifications from the joined classrooms** and other personalized information such as student same, display picture etc and **teachers** can see their respective personalized information such the **courses a teacher is currently teaching, classrooms created by the teacher or department information** into and other information such as **display picture, name** etc.
- The **Course UI** allows the users i.e teachers and students to **access information about available courses**, via **Course UI students can access, enrol or leave available courses** and **teachers can start to teach or opt out of a course**.
- The **Classes UI** allows users to **access info about the available virtual classrooms for any courses**. Via Classes UI **students can access, join or leave any available classroom with respect to the courses enrolled** and **teachers can Create a virtual classroom and access the available classrooms**.
- The **Admission UI** gives the **students an interface to get admission into different terms**/semesters (Access limited **with respect to the current academic calendar;** It is the responsibility of the site admin to control the access of this feature).
- The **Classroom UI** allows **teachers to create or delete a post, upload course materials, study materials, comment on post to interact with the students** and **remove the classroom**, and it allows the **students to access all the posts made by teacher** and and **access all the course/study materials** provided with the posts and allows the students to **post comments** on the post made by the teachers.
- The **Profile UI** allows the users to update their general credentials or information such as **updating email address** or **updating password** or **upload/update display picture**.

2.4.2 Hardware requirements

Additional hardware required for the development of the application without facing any issues. For the development of this project a **personal computer is required** with the following features, it recommended that the PC has **multi core CPU with clock speed of 2 ghz or above**, **RAM of 4GB or above**, it must have **secondary storage** (HDD/SSD) of **128GB or above**, it must also have a **monitor** of high definition resolution, it must also have **functional input devices** i.e **keyboard and mouse**.
Also **for deployment** of the application, a **linux enabled server/VPS** is required it must also have at least the following feature, it must have a **CPU** with good clock speed **1 ghz or better**, **1 GB of ram** or better, **at least 20 GB of secondary storage**, also the Operating system **must have Node js runtime and MySql database installed** for seamless deployment and for CI/CD tasks.

2.4.3 Software requirements

In development of this web application, a stack of various technologies/frameworks is required for building and deployment of the final product, below is a list of technologies used to build/construct the application.

- **Programming language of choice**: Javascript
- **Front-end** i.e **User Interfaces**: HTML5, CSS3, EJS, Javascript, Bootstrap, MDB.
- **Back-end and API**: Nodejs, Express.js, mysqljs (a mysql database driver for Nodejs)
- **Database**: MySql
- **Other libraries and packages**: bcryptjs, jsonwebtoken, express-session, nodemailer, otplib, csurf, cookie-parser, multer.

The technologies, libraries and packages specified above are the requirements needed in development of this application. Below is a short description of the technologies used in development of the application.

2.4.3.1 Javascript

JavaScript is a high-level, often **just-in-time compiled** language that conforms to the **ECMAScript standard**. It has **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)**.
JavaScript runs on the client side of the web, which can be used to design / program how the web pages behave on the occurrence of an event. JavaScript is an easy to learn and also powerful scripting language, widely used for controlling web page behavior.
JavaScript **can function as both** a **procedural** and an **object oriented language**. Objects are created programmatically in JavaScript, by attaching methods and properties to otherwise empty objects at run time, as opposed to the syntactic class definitions common in compiled languages like C++ and Java. **Once an object has been constructed it can be used as a blueprint (or prototype)** for creating similar objects.
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 **Nodejs**.

2.4.3.2 Nodejs

Node.js is an **open-source**, **cross-platform**, **back-end JavaScript runtime environment** that runs on **Google's V8 engine** and **executes JavaScript code outside a web browser**. Node.js lets developers use JavaScript to write command line tools and for **server-side scripting**; running scripts server-side to **produce dynamic web page content** before the page is sent to the user's web browser.
Node.js has an **event-driven architecture capable of asynchronous I/O**. These design choices aim to **optimise throughput and scalability in web applications** with many input/output operations, as well as for real-time Web applications e.g., real-time communication programs.

2.4.3.3 HTML5

HTML5 is a markup language used for **structuring and presenting content on the World Wide Web**. It is the fifth and final major HTML version that is a World Wide Web Consortium (W3C) recommendation. The current specification is known as the HTML Living Standard. **HTML is the World Wide Web's core markup languag**e. Originally, HTML was primarily designed as a language for semantically describing scientific documents. Its general design, however, has enabled it to be adapted, over the subsequent years, to describe a number of other types of documents and even applications.

2.4.3.4 CSS3

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 CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. **CSS is designed to enable the separation of presentation and content, including layout, colors**, and **fonts**. 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.

2.4.3.5 Ejs

EJS or Embedded Javascript Templating is a templating engine used by Node.js. Template engine helps to create an HTML template with minimal code. Also, it can inject data into an HTML template at the client side and produce the final HTML. EJS is a simple templating language which is used to generate HTML markup with plain JavaScript. It also helps to embed JavaScript to HTML pages.

2.4.3.6 Express.js

Express.js, or simply Express, is a **back end web application framework for Node.js**, released as free and open-source software under the MIT License. It is **designed for building web applications and APIs**. It has been called the de **facto standard server framework for Node.js**. Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications.

2.4.3.7 MySql

MySql is an open-source **relational database management system (RDBMS)**. A relational database organizes data into one or more **data tables** in which data may be **related to each other**; these relations help structure the data. SQL is a language programmers use to create, modify and extract data from the relational database, as well as control user access to the database. In addition to relational databases and SQL, an **RDBMS like MySQL** works with an operating system to implement a relational database in a computer's storage system, manages users, **allows for network access and facilitates testing database integrity and creation of backups**. MySQL has stand-alone clients that allow users to interact directly with a MySQL database using SQL, but more often, MySQL is used with other programs to implement applications that need relational database capability. The general architecture of an RDBMS is shown in **Figure 1**.

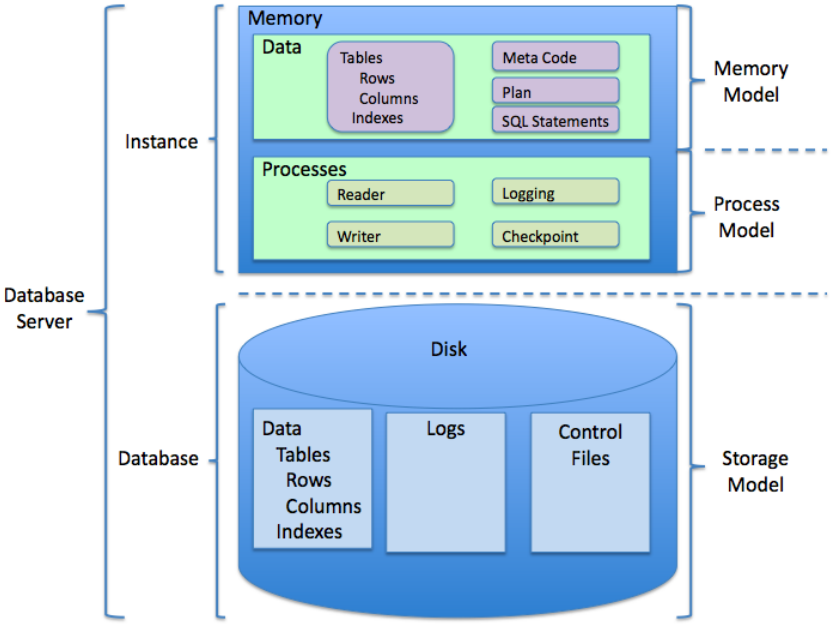

Figure 1: Database server architecture (RDBMS)

2.4.3.8 Bootstrap & MDB

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains HTML, CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. Bootstrap is an HTML, CSS & JS Library that focuses on simplifying the development of informative web pages (as opposed to web apps). MDB is a subset of bootstrap with UI components that are focused on Material design pattern.

2.4.3.9 Other libraries and packages

- **mysqljs**: A pure node.js JavaScript Client implementing the MySQL protocol. Used as a Nodejs driver for Mysql database.
- **bcryptjs**: Nodejs hashing library for **hashing and comparing bcrypt hashes**. Mainly used for hashing and comparing passwords.
- **jsonwebtoken**: This is a Nodejs cryptography module, used for **generating, signing and validating JSON Web Tokens** which is needed for API route protection.
- **express-session**: This is a built-in module in Express.js framework, it is **used for handling user sessions data**, used for generating and destroying user sessions, generating and validating user **session cookies**.
- **csurf**: This Nodejs module is used for **generating and validating CSRF tokens for each GET and POST request to the Express server.**
- **cookie-parser**: This is another built-in package in Express.js framework, it is **used for parsing cookies sent with each request to the Express server**.
- **multer**: This nodejs module is used for **parsing form-data** i.e **data-input that is sent with POST requests via forms**, it is also used for **parsing and handling files** sent with POST requests.
- **nodemailer**: This Nodejs module is used for **handling email services** through **SMTP** protocol.
- **otplib**: otplib is a pluggable **JavaScript One-Time Password (OTP) library** with support for **HOTP, TOTP** and **Google Authenticator**. It implements both **HOTP - RFC 4226** and **TOTP - RFC 6238**, and are tested against the test vectors provided in their respective RFC specifications.

2.4.4 Communication interface requirements

The communication architecture must follow the **client-server model**. Communication between the client and server should utilise a REST-compliant web service and must be served over **HTTP** or preferably HTTP Secure (**HTTPS**). The client-server communication must be stateless. A uniform interface must separate the client roles from the server roles. Communication the **QDR** (**queryable data repository**) must provide the user with the option of formatted **ASCII, CSV, JSON, or XML** structured data schema.

3. Analysis and Design

Design is a meaningful engineering representation of something that is to be built. Software design is a process through which the requirements are translated into a representation of the software. Design is the place where quality is fostered in software engineering. Design is the perfect way to accurately translate a customer's requirement into a finished software product. Design creates a representation or model, providing detail about software data structure, architecture, interfaces and components that are necessary to implement a system. This section discusses the design part of the project. There are many types of diagrams to represent the implementation of a software system. Some of the types of these diagrams are **Data flow diagram**, **UML diagram**, **Class diagram**, **Use case diagram** etc.

3.1 Introduction

In this phase of development, a logical system is built which fulfills the given requirements. Design phase of software deals with transforming the client's requirements into a logically working system. Normally ensign is performed in two steps: **Primary design phase** and **Secondary design phase**.
In the primary design phase the system is designed at block level, The blocks are created on the basis of analysis done in the problem identification phase. Different blocks are created for different functions, emphasis is put on minimizing the information flow between blocks. Thus, all activities which require more interaction are kept in one block. In the secondary design phase the detailed design of every block is performed.

3.2 Tasks involved in the application system design process

- A Software Requirement Specification (SRS), which specifies the software, hardware, functional, and network requirements of the system used in constructing the architecture of various modules of the application and it used for mapping the proposed feature of the project to high level diagrams such as Data-flow diagrams UML diagrams etc.
- Design various blocks for overall system processes. Gather, analyse, and validate the information. Define the requirements and prototypes for the new system. Evaluate the alternatives and prioritize the requirements. Evaluate the alternatives and prioritize the requirements.
- Includes the design of application, network, databases, user interfaces, and system interfaces.
- Transform the SRS into logical structure, which contains a detailed and complete set of specifications that can be implemented in a programming language.
- Create a contingency, training, maintenance, and operation plan.Review the proposed design. Ensure that the final design must meet the requirements stated in the SRS.

3.3 System Assumptions/Constraints/Dependencies/Risks

3.3.1 Assumptions

The largest assumption is that the existing LMS application will be extended to support the proposed new features. The existing architecture and system design will be used including all existing components and subsystems. It is certain that additional functionality will be added to the proposed solution.

3.3.2 Constraints

There are no hardware, software, or software technical constraints identified with this project. Financial constraints are not of concern for this exceptional case. Institutional constraints may exist due to the systems need for regional coordination, participation, and interoperability.

3.3.3 Dependencies

The current application that is developed is dependent on many third party systems (packages, libraries and APIs). These include: HTML5, CSS3, Nodejs, Express js, Ejs, javascript, MySql, and many external libraries and packages which includes: bcryptjs, jsonwebtoken, express-session, nodemailer, otplib, csurf, cookie-parser, multer etc.

3.3.4 Risks

Minimal risk is associated with the system design. This is primarily due to the fact that the existing system design and architecture will not be modified to meet the needs of the proposed solution. Financial risks are no of concern for this exceptional case of development. Ongoing maintenance of the system will also be of concern.

3.4 Design considerations

Features of the proposed system are the home page, the teacher login page, student login page,, register student page, student admission page, register teacher page, classes, courses , classroom, profile page for instructor/student; also security, delivery, interaction, reporting and record keeping will be put into place. The system will require the user to login to have access to the website to do the following tasks: view and download course materials, upload files, and other materials for users to see and also interact through comments in virtual classrooms.

The system will be designed and structured to cut down the level of disjointedness and ease the learning process. Most of the time, there is a break in the flow of study and information gathering because the student has to leave his/her current study location to look for the lecturer or get to the library to retrieve extra needed information.

A mailing and real-time support will make it possible for a student to contact a lecturer instantaneously from his/her very location, thus keeping the study session active and continuous. This will make the assimilation level higher since there are no breaks in communication. Course materials will also be made available for easy accessibility.

The system design will be structured with a mass communication section. This is to ensure easy dissemination of information to the entire student body. Also, upcoming events can be posted on the LMS to effectively notify the students of scheduled events.This LMS is indeed very feasible as it provides a larger avenue for learning in addition to rapid delivery of learning content on a scalable web-based platform. Once tested and implemented, the LMS would be updated and maintained at regular intervals.

3.5 Goals and Guidelines

The following goals must be addressed in the execution of the proposed solution.

- **Leverage Existing Architecture**: The proposed solution must leverage the current architecture and system design used by the current solution. This minimises negative impacts on usability, user experience, and financials. The proposed solution will simply extend the current application to support additional features, functionality, and use cases.
- **Development Environment**: The application development environment must remain consistent. This minimises negative impacts to interoperability and quality.
- **Ease of Use**: The new features must be easy to use and provide a strong user experience. New features cannot impact existing functionality from a user perspective.
- **Extensibility**: The proposed features must be extensible. Features can be enabled as needed or required by the users.
- **API Enabled**: The application must be API centric and support an open and published API architecture
- **RESTful Framework**: The application and underlying architecture must be a REST framework.

3.6 Operational environment

- Node.js and Express.js development
- Git Version Control
- Github repository
- MySql database.
- Express server.
- Linux enabled web server environment for deployment.

3.7 System Architecture and Architecture Design

This project makes an attempt to address difficulties of program analysis and generation of diagrams, which can depict the structure of a program in a better way. Today, UML is being considered as an industrial standard for software engineering design process. It essentially provides several diagramming tools that can express different aspects/characteristics of a program such as **Use case diagrams**, **Class diagrams**, **State diagrams**, **activity diagrams** etc.

3.7.1 Use case diagram of LMS Application

A use case is described in terms of a sequence of interactions between some actors and the system by which the system provides a service to the actors. Each use case then captures a piece of functional requirements for some users. All the use cases together describe the overall functional requirements of the system. The first step in requirements capture is to capture requirements as use cases. We have **two such use case diagrams** for the **teacher** , and for the **student**. **Figure 2 represents the Use case diagram for both the student and teacher's perspective**.

Figure 2: Use case diagram of LMS Application

3.7.2 Data Flow diagram of LMS Application

Data flow diagrams (**DFD**) are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. A data-flow diagram is **a way of representing a flow of data through a process or a system** (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. There are many levels of data-flow diagrams:

Level-0 DFD is also known as the fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood.

3.7.2.1 Level 0 DFD of LMS Application

The below figure that is **figure 3** shows the context level data flow diagram or Level 0 DFD of the from the proposed features of the application.

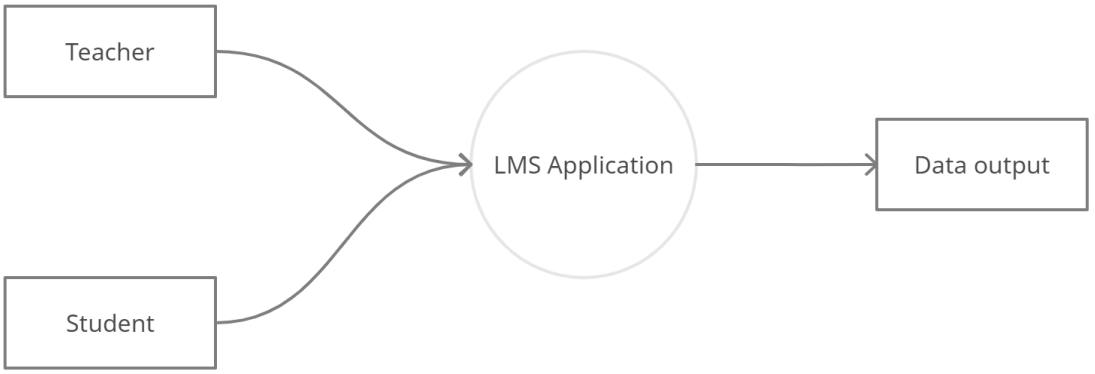

Figure 3: Level 0 DFD of LMS Application

3.7.2.2 Level 1 DFD of LMS Application

In 1-level DFD, a context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main objectives of the system and break down the high-level process of 0-level DFD into subprocesses. **Figure 4** shows the 1-level DFD of the system.

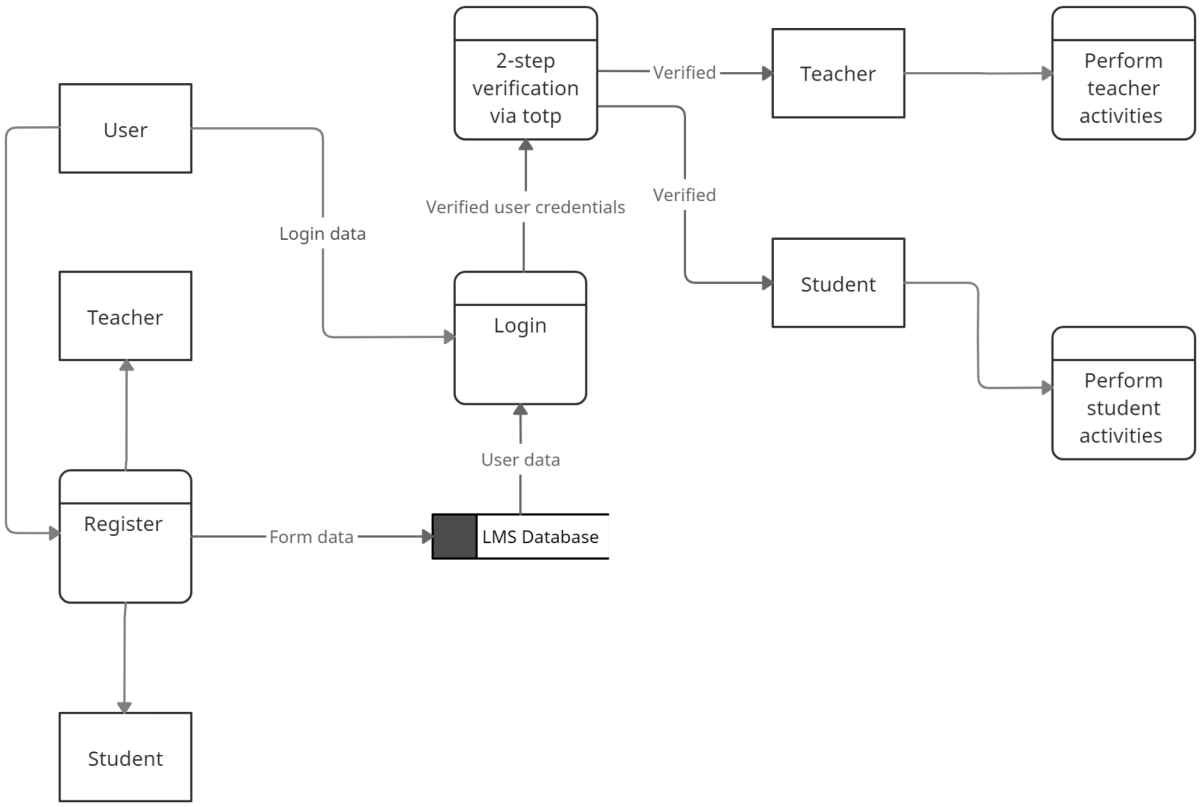

Figure 4: Level 1 Data flow diagram of LMS Application

3.7.3 Description of different modules of LMS Application

3.7.3.1 Registration module

In this module, either the teacher or students can register through their respective interface by providing the necessary information. Then each user profile will be maintained which can be edited by the user when desired. Each person will register only one time. **Details of each person along with their email address and hashed password is saved permanently in the database**. The detailed process of registration as described in **figure 5** is as follows

- Users **provide their required info** for registration **through their respective interface i.e registration form**.
- **Validate the data** that is provided by the user from **client side** and **server side** before processing further. **If the input is not of correct format, throw an error message and end/restart the process**.
- **If the input data is validated**, then proceed to the next step and **store the user information to the database** and **throw a success message and end the process**.

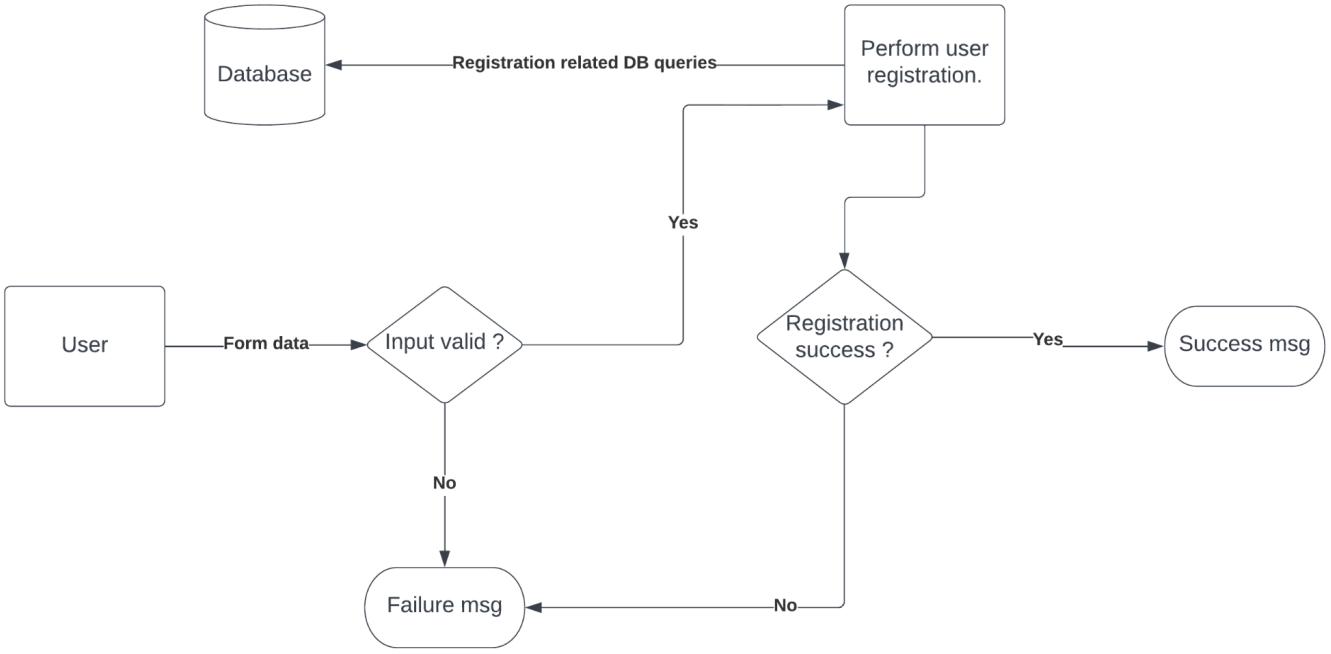

Figure: 5 Diagram of the registration module

3.7.3.2 API & website Route protection/authorization module

This module describes the process of application route protection and authorisation of routes to users. The **application utilizes** industry standard **JWT for API route protection** to **mitigate unauthorized access to sensitive routes** of web applications. The process of authentication as shown in **figure 6** is described below:

- After every successful login with 2-step verification, A **JWT** with respect to the requesting user is generated and signed against the backend server and it is stored as a **HTTP only cookie in the browser**.
- For every request an unique CSRF token is generated and sent with response and a CSRF Http only cookie is generated and stored in the browser.
- When a **user requests for a protected/private poute**, **fetch the json web token (JWT)** sent as an http only cookie from the **request object** ,if **JWT is not present** (sent) in the request, then **throw an 401 Unauthorized error message**.

- **If the JWT is present in the request** or sent with the request, then **validate the token against the backend server**. **If the token is a valid JWT with respect to the requesting user** then, **return success message**, **else throw a 401 Unauthorized message**.
- For **any POST request to a route after checking and validating the JWT**, **Check for the CSRF token** sent with the response, now **for every POST request, if the request body does not contain the CSRF token, then throw 401 unauthorized message**.
- If the **POST request contains the CSRF token** then **validate the CSRF against the CSRF cookie**, if the token is **verified**, then **proceed further else throw 401 unauthorized msg**.

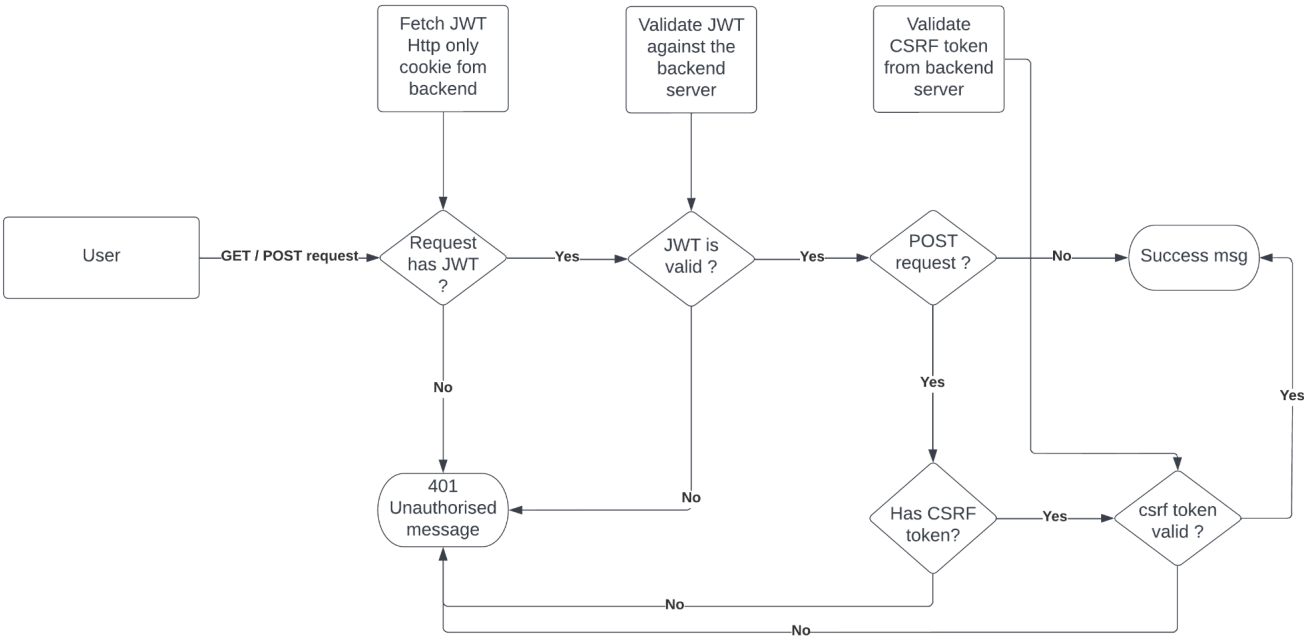

Figure 6: Diagram of route protection/authorisation module

3.7.3.3 Login/user authentication module

This module describes the process of login of the user and the process of user authentication to grant or limit the access of features or routes of the application to the users. Without login into the application, the users cannot use the full features of the application. The login process also involves the process of 2-step verification, without the success of both the process, the whole login process will not be complete. The detailed process as described in figure 7 is given below:

- The user i.e teacher or student **provides their registered email and password** for login verification.
- The **login info is validated** for correct format, if the format is not according to the expected or in malicious format, then return a failure message.
- If the login info is of correct format and is successfully validated then, proceed to **further check email password combination in the database for existing users, if user is not found the return 401 failure message**.
- If the **user credential is found to be correct** and is found in the database then **proceed further with the next step of verification involving totp based verification**.
- Using the totp library **generate a key string and store the key string in the database** with respect to the current user or **bind the key string with the requesting user**.
- Now **using the totp and qrcode library generate a scannable QR code** with **respect to the key string** and **send the QR code to the user via email service to the registered user's email** and at the same time redirect the user to the login verification page where the user enters the verification code get verified and get logged in.

- After the user receives the email, **the user scans the QR code with an authenticator app** like **Google authenticator** or **Microsoft authenticator** and **generates a 6-digit login verification code** that is regenerated within the interval of 30 seconds.
- The user enters the 6-digit code into the login verification page, a POST request is sent to the server for verification. If the request object doesn't contain the verification code then return 401 Unauthorized message (failure).
- If the request contains the 6-digit code then the process moves to the next step, where the backend fetches the current requesting user and gets the **key string** that was stored with respect to the user from the database.
- Now the **verification code is checked with the key string with respect to the current time delta** to check the validity of the verification code that was sent with the request. The check method provided by totp library is used to verify the code.
- **If the code that was sent is not correct** then the process throws a **401 unauthorized error message**.
- If the **code is found to be valid** then **generate and sign a JWT with respect to the requesting user** and **store the token as an HTTP only cookie in the browser**.
- Return success message.

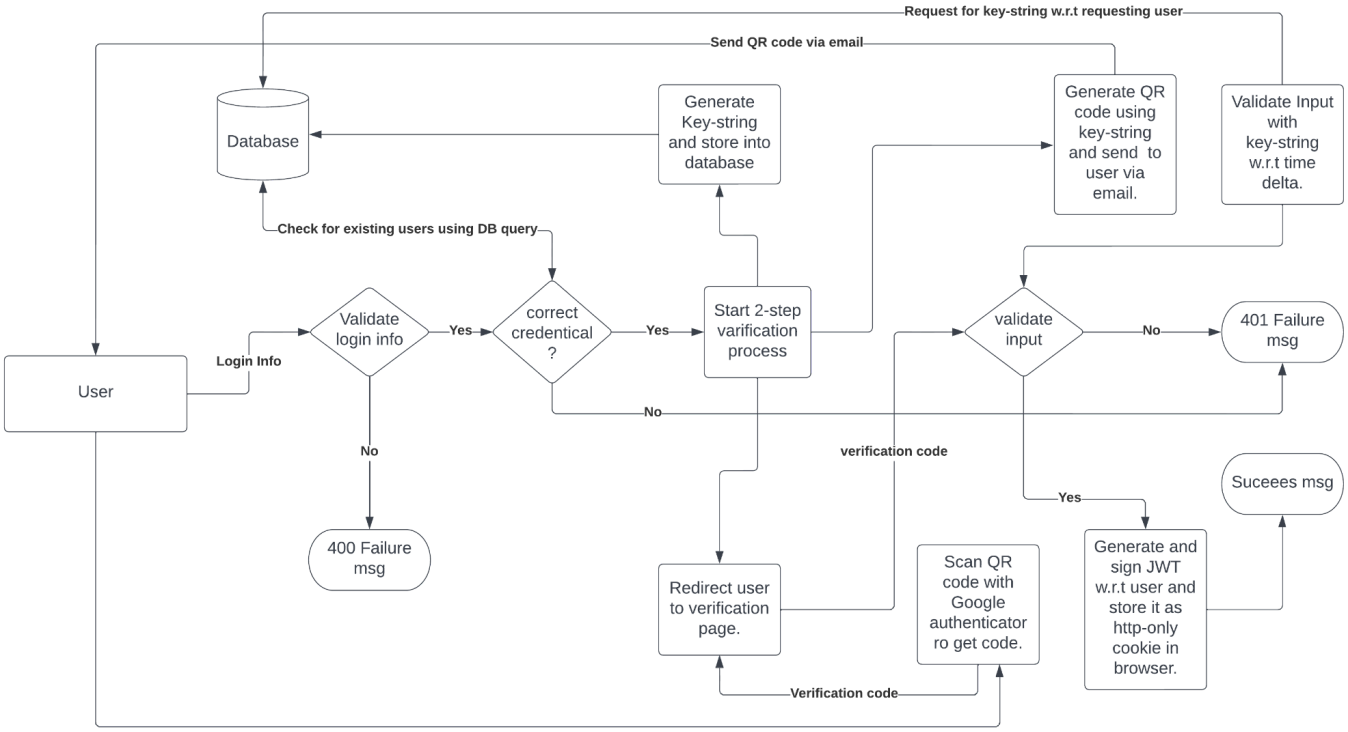

Figure 7: Diagram of login/user authentication module

3.7.3.4 General CRUD operations module

CRUD stands for **create, retrieve, update and delete** operations. This module describes the process of general CRUD operations that is available to perform with respect to the user. The features that fall under this category are as follows: accessing joining, leaving courses, accessing classrooms, or updating user information such as email and password, or creating or deleting classrooms, creating or deleting posts, or posting comments etc. All of the above processes/features fall under the general CRUD category. The detailed process is described in **figure 8** as shown below. The detailed process is is as follows:

- User requests with a **GET/POST request** to request for a resource from the server be it a website route or a API route using the **fetch** method via **JS**.
- Now, the request is checked if the request is authorized to access the route using route authorization module to check if the request contains the valid auth tokens required for the access of the route.

- If the request fails the authorization process, then return a failure message.
- If the request contains the required auth tokens then proceed to the next step of the process
- Now check if the request is a **POST request**, **if not** then the request must be a **read request**, and proceed to **perform the read process** by performing necessary database queries.
- If the read process is successful then return the success message.else throw a failure message.
- Now, if the request is a POST request, then the request must be for a process involving creating, updating, deleting records/info. To perform proceed to the next step first **check if the input data sent with the request is of valid format or not**.
- If the input is not of correct format or is malicious then return a failure msg.
- If the request input is successfully validated, then perform the general create, update or delete process by performing the necessary database queries through the backend methods constructed for the specific task.
- If the **CRUD** process is **successful** then, return a **success message** else **throw a failure message**.

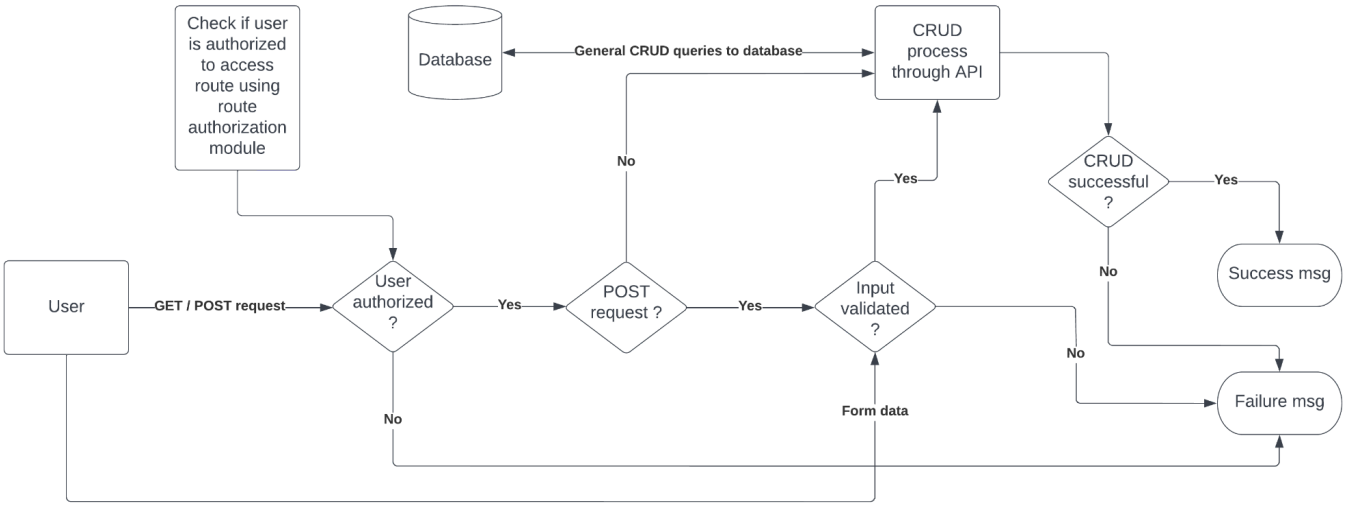

Figure 8: Diagram of general CRUD operations

3.7.4 Server side rendering of dynamic content

Server side rendering or simply **SSR** is a process of generating dynamic web pages (**DHTML**) from the server side in response to a URL request from the client side. Server-side rendering means using a server to generate HTML from JavaScript modules in response to a URL request. That's in contrast to client-side rendering, which uses the browser to create HTML using the DOM.It is an application's **ability to convert HTML files on the server into a fully rendered HTML page for the client**. The web browser submits a request for information from the server, which instantly responds by sending a fully rendered page to the client. A **server-side dynamic web page** is a web page whose construction is controlled by an application server processing **server-side scripts**. In server-side scripting, parameters determine how the assembly of every new web page proceeds, including the setting up of more client-side processing.

This web application i.e LMS application utilizes the server side rendering approach to deliver dynamically generated HTML pages in response to the request to any authorized URLS from the client on demand. This application uses **EJS** (**Embedded javascript templates**) template engine to generate the dynamic HTML pages.

The process as described in **figure 9** is as follows:

- User requests for a URL from server
- If the URL is found then, an HTML page is dynamically generated with respect to the aggregated resultant data from multiple database queries according to the request. And the final page is **compiled and rendered using the EJS render method**.

- The generated HTML page is sent to the browser and the browser shows the dynamic page to the client.

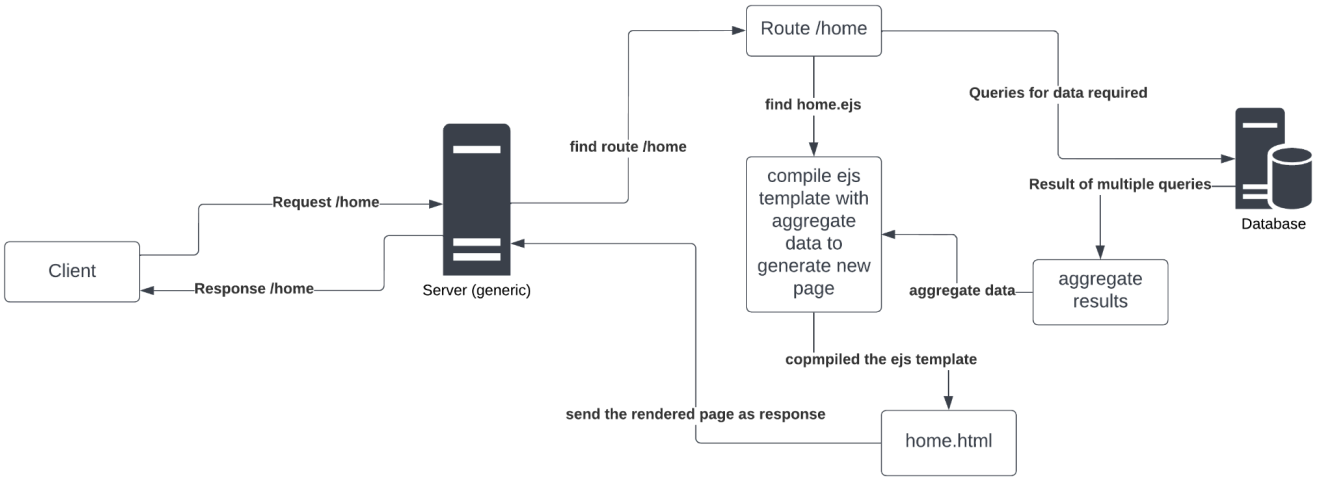

Figure 9: Dynamic Page generation through SSR using EJS

3.7.5 Database & database schema design

Designing the database or the database schema is one of the most crucial parts of the application system design process as it deals with the data generated from the application. In this application we have used MySql database as a persistent data storage.

MySql is an open-source **relational database management system (RDBMS)**. A relational database organizes data into one or more **data tables** in which data may be **related to each other**; these relations help structure the data. For the description of the entities types i.e tables and the relationships among the entities are clearly described in a diagram that is called the ER diagram.

3.7.5.1 ER diagram of LMS Application

ER Model stands for Entity Relationship Model is a high-level conceptual data model diagram. ER model helps to systematically analyze data requirements to produce a well-designed database. The ER Model represents real-world entities and the relationships between them. ER Modelling helps us to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modelling before implementing the database.

ER diagrams are related to data structure diagrams (DSDs), which focus on the relationships of elements within entities instead of relationships between entities themselves. ER diagrams also are often used in conjunction with data flow diagrams (DFDs), which map out the flow of information for processes or systems.

A database that is mapped to an ER diagram can be represented by a collection of tables in the relational system. An ER model is typically drawn at up to three levels of abstraction: 1. Conceptual ERD, 2. Logical ERD and 3. Physical ERD. While all the three levels of an ER model contain entities with attributes and relationships, they differ in the purposes they are created for and the specific groups of people/users they are meant to target. **The ER diagram of our application is described in figure 10.**

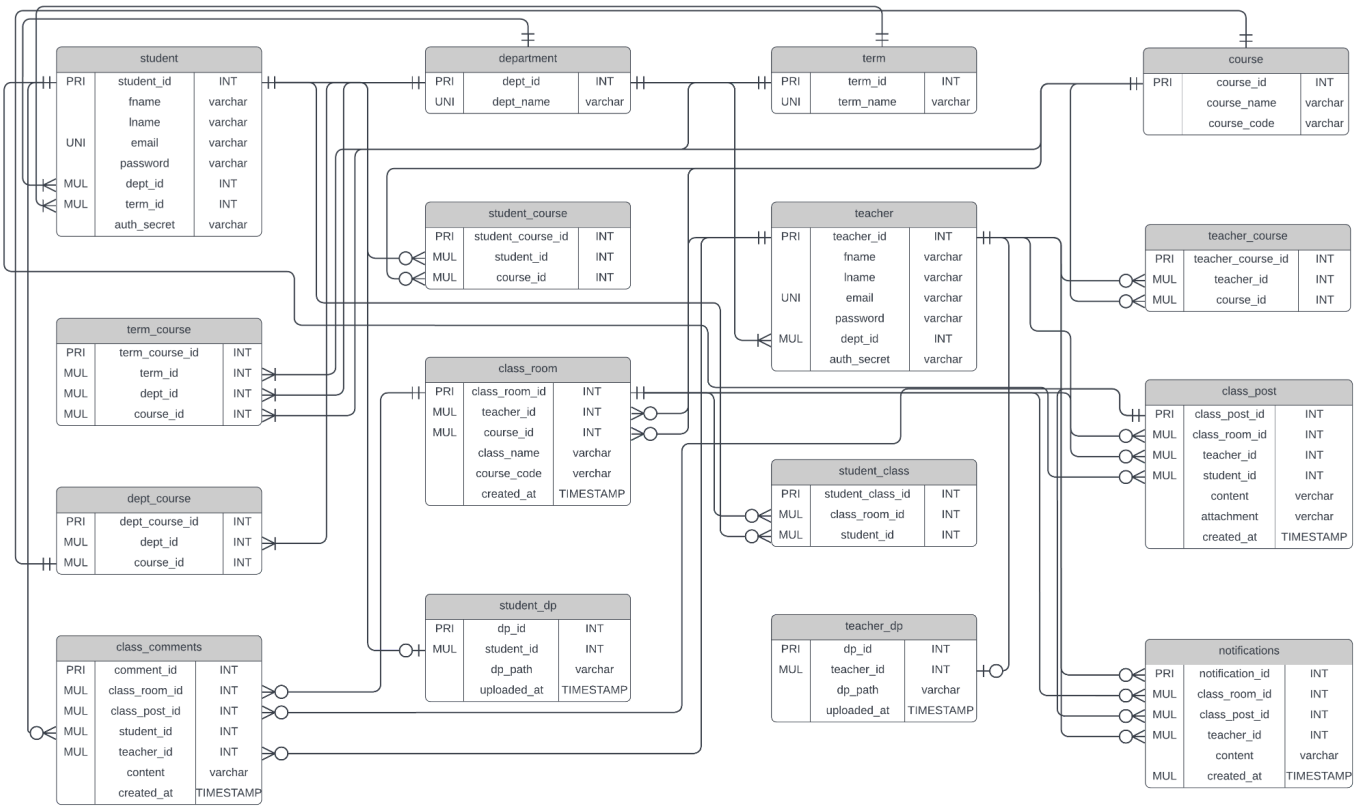

Figure 10: ERD of LMS application

The proposed database schema for this web application contains 16 different relations or (tables) each of which defines an Entity type. These relations are as follows: **student, department, term, course, term_course, student_course, teacher, taecher_course, dept_course, class_room, student_class, class_post, class_comments, student_dp, teacher_dp,** and **notifications**.
The schema for the database is as follows:

- **student** (**student_id**, fname, lname, email, password, **dept_id**, **term_id**, auth_secret)
- **teacher** (**teacher_id**, fname, lname, email, password, **dept_id**, auth_secret)
- **department** (**dept_id**, dept_name)
- **term** (**term_id**, term_name)
- **course** (**course_id**, course_name, course_code)
- **student_course** (**student_course_id**, **student_id**, **course_id**)
- **teacher_course** (**teacher_course_id** , **teacher_id**, **course_id**)
- **term_course** (**term_course_id**,**term_id**, **dept_id**, **course_id**)
- **dept_couse** (**dept_course_id**, **dept_id**, **course_id**)
- **class_room** (**class_room_id**, **teacher_id**, **course_id,** class_name, course_code, created_at)
- **student_class** (**student_class_id**, **student_id**, **class_room_id**)
- **class_post** (**class_post_id**, **class_room_id**, **teacher_id**, **student_id**, content, attachment, created_at)
- **class_comments** (**comment_id**, **class_room_id**, **class_post_id**, **student_id**, **teacher_id**, content, created_at)
- **student_dp** (**dp_id**, **student_id**, dp_path, uploaded_at)
- **teacher_dp** (**dp_id**, **teacher_id**, dp_path, uploaded_at)
- **notifications** (**notification_id**, **class_room_id**, **class_post_id**, **teacher_id**, content, created_at)

3.7.6 Interface Architecture

The current and proposed solutions utilise a services oriented architecture. The proposed system will utilise the existing interface architecture by implementing a **REST framework.**

3.7.6.1 Representational state transfer (REST)

Representational state transfer (**REST**) is a software architectural style that was created to guide the design and development of the architecture for the **World Wide Web**. REST defines a set of constraints for how the architecture of an Internet-scale distributed hypermedia system, such as the Web, should behave. The REST architectural style emphasises the **scalability** of **interactions between components, uniform interfaces, independent deployment of components, and the creation of a layered architecture to facilitate caching components to reduce user-perceived latency, enforce security, and encapsulate legacy systems**.

REST has been employed throughout the software industry and is **a widely accepted set of guidelines for creating stateless, reliable web APIs**. **A web API that obeys the REST constraints is informally described as RESTful**. RESTful web APIs are typically loosely based on **HTTP methods** to access resources via **URL-encoded parameters and the use of JSON or XML to transmit data**.

This is a style or framework for designing integrated applications or services over HTTP. The proposed solution will implement a true RESTful API for interapplication integration and regional coordination. This achieves the following results from an interface architecture perspective:

- Uniform interface.
- Client-server.
- Stateless
- Cacheable.
- Layered System.
- Code on demand.

3.7.6.2 Data Exchange over HTTP

JSON (**JavaScript Object Notation**) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence.

3.7.7 Detailed interface Design

All the third party applications and integrations will utilize the RESTful API that is designed in the application development phase. That means that a developer or third party, if they provide the proper security credentials, can access the application, database, and the published functions. REST model clients make standard requests over HTTP and it is recommended to validate the SSL certificate of the end point with which the client is communicating for secure encrypted communication over HTTP.

3.7.7.1 Resources

Clients make requests against resources, either in aggregate or specific resources. The general format of the requesting is:

- endpoint/version/namespace/resource[?query_parameters]
- endpoint/version/namespace/resource/resource_id

3.7.7.2 Operations

API requests are standard HTTP requests against publicised resources. **GET** queries for a list of a class of resources or the details of a specific resource. **POST** creates a new resource instance and will provide either a job or a resource instance in the response body.

3.7.7.3 Request headers

A request header is an **HTTP** header that can be used in an HTTP request to provide information about the request context, so that the server can tailor the response. For example, the Accept-* headers indicate the allowed and preferred formats of the response. Other headers can be used to supply authentication credentials (e.g. Authorization), to control caching, or to get information about the user agent or referrer, etc.
Headers can be grouped according to their contexts:

- **Request headers** contain more information about the resource to be fetched, or about the client requesting the resource.
- **Response headers** hold additional information about the response, like its location or about the server providing it.
- **Representation headers** contain information about the body of the resource, like its MIME type, or encoding/compression applied.
- **Payload headers** contain representation-independent information about payload data, including content length and the encoding used for transport.

In addition, **CORS** defines a subset of request headers as simple headers, request headers that are always considered authorized and are not explicitly listed in responses to preflight requests. **CORS** (Cross-Origin Resource Sharing) is a system, consisting of transmitting HTTP headers, that determines whether browsers block frontend JavaScript code from accessing responses for cross-origin requests.
Third parties may specify an "Accept" header to define whether you wish to receive responses as XML or JSON. The default response is XML. The values you may specify for "Accept" are:

- application/xml
- application/json

3.7.7.3 Response codes

API will respond with standard HTTP response codes appropriate to the result of the request. While the exact meaning of the code varies depending on the request.
HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

- Informational responses (100–199)
- Successful responses (200–299)
- Redirection messages (300–399)
- Client error responses (400–499)
- Server error responses (500–599)

A response code of **200** means the request was successful and details about the response can be found in the body of the response.
The response code **201** means the requested POST operation was successful and an object was created in the system.
The response code **202** means the requested operation has been accepted and the body contains information about an asynchronous job we can query to check on the progress of the request.
The response code **204** means the requested operation was successful and there is no response body.
The response code **307** means to repeat the request using the provided URI. Subsequent requests can use the old URI.

The response code **400** means the request was improperly formatted. You should verify that your request conforms to this specification and re-issue the request in a properly formatted manner.
The response code **404** means that the requested resource does not exist.
The response code **500** means the API failed to process the request because of an error inside the system.

3.7.7.4 Response entities

All **GET** methods respond with the **JSON or XML** of the resource(s) being requested. HEAD methods have no response entity. **POST** methods may respond with a **201 CREATED** or **202 ACCEPTED** response code depending on whether the creation is completed immediately or is an asynchronous operation. If the resource was created immediately, API should provide a JSON or XML entity that includes the new resource's unique ID. If the creation operation takes time, however, the response body will include a Job resource that can be tracked to completion.

4. System Development and Testing

4.1 Introduction

System Development or Construction consists of **all of the activities required to build and validate the new system** to the point at which it can be turned over for System Acceptance. Development efforts in this phase are based on the technical solution created during System Design, which, in turn, was based on the functional and operational requirements captured during System Requirements Analysis.

Included in this phase is the construction of all components of the system, including utilities required to adequately prepare and load the data. In addition, System Construction consists of **a series of tests of the system components**, with each set of tests being performed against a progressively larger grouping of components until the operation of the system in its entirety has been verified.

4.2 List of processes

This phase consists of the following processes:

- **Prepare for System development**, where the system development and testing environments are established, and where the Project Team is instructed in the processes and tools that will be used throughout this phase.
- **Refine System Standards**, where standards established in System Design are enhanced and adjusted as the team becomes more familiar with the project environment, or in response to changes in the strategic or technical direction of the project.
- **Build, Test, and Validate** (BTV), where individual system components and utilities are constructed and tested to ensure that they perform to the technical and functional specifications.
- **Conduct Integration and System Testing**, where logically related components of the system are assembled and tested as single units, and a complete end-to-end system test is performed.
- **Produce User and Training Materials**, where all Consumer-related documentation and training materials are developed.

4.3 Development

In development of the application, coding is undertaken once the design phase ios copm[plete and the design documents have been successfully reviewed. In the coding phase, every module specified in the system design document is coded and unit tested. During unit testing, each module is tested in isolation from other modules. That is, a module is tested independently as and when its coding is complete. Integration and testing of modules is carried out according to an integration plan. The integration plan, according to which different modules are integrated together, usually envisages integration of modules through a number of steps. During each integration step, a number of modules are added to the partially integrated system and the resultant system is tested.

4.3.1 Coding

The input to the coding phase is the design document produced at the end of the design phase. The objective of the coding phase is to transform the design of a system into code in a high-level language, and then to unit test this code.

During the development of the LMS application we chose Javascript along with Nodejs to be the primary programming language of choice for coding of the backend and frontend modules. Specifically, the backend modules of the application were coded and developed with **Nodejs** runtime with **Express Js** framework. For frontend HTML is used as markup language along with EJS for injecting javascript template code in the HTML.

4.3.1.1 Coding guidelines

During the development and Coding of the LMS application the standard guidelines were followed.

- **Rules for limiting the use of globals**: A significantly minority variables are declared as globals. Only the variables that were accessed by different modules throughout the application, were declared as global variables.
- **Standard headers for different modules**: For better understanding and maintenance of the code, It was decided that the header format of different module must contain the below informations that is:
    - Name of the module
    - Synopsis of the module about what the module does
    - Different functions supported in the module along with their input output parameters
- **Naming conventions for local variables, global variables, constants and functions:** Local variables were named using camel case lettering starting with small letter (**e.g. varName**) and in some cases snake case (**e.g sanke_case**)whereas Global variables were also named using the standard camelcase (**e.g. globalData**). Constants were formed using capital letters only (e.g. **CONST_DATA**).
- **Version control**: As an established version control standard, GIT is used for version control of the code base. Also Github is used as a repository for hosting the code base.

The codebase of this project i.e LMS Application can be found on this github repository.

4.4 Testing

The aim of program testing is to help identify all defects in a program. However, in practice, even after satisfactory completion of the testing phase, it is not possible to guarantee that a program is error free. This is because the input data domain of most programs is very large, and it is not practical to test the program exhaustively with respect to each value that the input can assume.

4.4.1 Testing terminology

As is true for any specialized domain, the area of software testing has come to be associated with its own set of terminologies. A few important terminologies that have been standardized by the IEEE Standard Glossary of Software Engineering Terminology [IEEE90]:

- A **mistake** is essentially any programmer action that later shows up as an incorrect result during program execution. A programmer may commit a mistake in almost any development activity. For example, during coding a programmer might commit the mistake of not initializing a certain variable, or might overlook the errors that might arise in some exceptional situations such as division by zero in an arithmetic operation. Both these mistakes can lead to an incorrect result.
- An **error** is the result of a mistake committed by a developer in any of the development activities. Among the extremely large variety of errors that can exist in a program. One example of an error is a call made to a wrong function.
- A software **bug** is an error, flaw or fault in the design, development, or operation of computer software that causes it to produce an incorrect or unexpected result, or to behave in unintended ways. The process of finding and correcting bugs is termed "debugging" and often uses formal techniques or tools to pinpoint bugs. Since the 1950s some computer systems have been designed to deter, detect or auto-correct various computer bugs during operations.

4.4.1.2 Black box testing

Black box testing is testing without the knowledge of the internal working of the item being tested. When black box testing is applied to software engineering, the tester selects valid and invalid input and what the expected outputs should be, but not how the program actually arrives at those outputs. Black box testing methods include equivalence partitioning, boundary value analysis, app-pairs testing, fuzz testing, model based testing, traceability matrix, exploratory testing and specification-based testing. This method of test design is applicable to all levels of software testing: Unit testing, integration testing, functional testing, system and acceptance testing etc.

4.4.1.3 White box testing

White box testing (glass-box testing) strategy deals with the internal data structures and algorithms. The tests written based on the white box testing strategies incorporate coverage of the code written, branch paths. Statements and internal logic of the code etc. These tests require programming skills to identify all paths through the software. Types of white box testing includes code coverage (creating tests to satisfy some criteria of code coverage), mutation testing fault injection methods, static testing.

4.4.1.4 Performance testing

Performance testing checks to see if the software can handle quantities of data or users. This is generally referred to as software scalability. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

4.4.2 Testing strategy and test plans for LMS Application

- Black box testing and white box testing were used as the main testing techniques to test the entire system for all the stages. Although both of these techniques have its advantages and disadvantages, when combined, they help to ensure through testing of the application.
- All the functions. Procedures are tested as a single individual unit first and after integrating individual units into one unit, integrated units were tested as a whole.
- When the function/procedure is tested for the first time, black box testing methodology is used. It is tested for expected outcomes from different user inputs. If the function did not deliver as expected, that is if it failed any test case, then white box testing is used.
- After testing for functionality of the functions/procedures, these functions/procedures were combined and tested using black box testing methodology. If the results show any discrepancy than expected output, then again they were tested for expected output using white box testing approach.
- Once the application was completed then it was tested again using test data through black box and white box approach.
- When preparing test data, more attention was paid to cover most edge cases.
- In case of data retrieved from a database it is assumed that the correct data is retrieved. When data is stored in the database, it is also tested. When deleting and updating the record in tables, it was also checked and tested whether the correct row was altered or updated.

4.4.3 Testing of different modules of LMS Application

4.4.3.1 Testing of Registration Interface

In this section some of the test cases and the test outputs were attached for the Registration interface of LMS Application.

User Interface: Registration
Operation: Register into the system

Action	Input	Expected output	Status
Input first name, last name, email, password, repeat password, select department and term for students and only department for teachers. And click Register btn. (Correct input format) (User role: teacher, student)	John Doe john@gmail.com John_1234 John_1234 CS 1	Output successful message: User registered successfully	**PASS**
Input invalid data i.e input data with incorrect format and click register btn. (invalid input format) (User role: teacher, student)	1John23 23Doe johngmail@gmail.com John_1234 John_1234 CS 1	Output failure message: bad name format	**PASS**
Type invalid email, while keeping other data in correct format (invalid input format) (User role: teacher, student)	John Doe john@gmail John_1234 John_1234 CS 1	Output failure message: bad name format	**PASS**
Keep one of the fields empty, while keeping other data in the correct format. (invalid input format) (User role: teacher, student)	John Doe John_1234 John_1234 CS 1	Output failure message: Please fill out all the fields.	**PASS**
Keep all the data format valid but input different passwords to confirm the password field. (invalid input format) (User role: teacher, student)	John Doe john@gmail.com John_1234 John_1235 CS 1	Output failure message: Both password must be same	**PASS**

4.4.3.2 Testing of Login Interface

In this section some of the test cases and the outputs were attached for the Login interface of LMS Application.

User Interface: Login
Operation: Log into the system

Action	Input	Expected output	Status
Input registered email address and password and click login button. (valid input format) (User role: teacher, student)	mondaldeep2000@gmail.com 1234	Login successful and redirected to the login verification page.	**PASS**
Input registered email address and password and click login button. (valid input format but wrong credentials) (User role: teacher, student)	mondaldeep2000@gmail.com 12345	Login failed and redirected and redirected to 401 error page.	**PASS**
Type invalid email, while keeping other data in correct format (invalid input format) (User role: teacher, student)	mondaldeep2000@= 1234	Login failed and redirected and redirected to 401 error page.	**PASS**

4.4.3.3 Testing of Login verification Interface

In this section some of the test cases and the outputs were attached for the Login verification interface of LMS Application.

User Interface: Login
Operation: Login verification

Generated from the login information, 255436 is the correct login verification code for this test case.

Action	Input	Expected output	Status
Input valid security code after scanning the received QR code to Google authenticator. (valid input format and correct credential) (User role: teacher, student)	255436	Login successful and redirected to the home page.	**PASS**
Input invalid security code after scanning the received QR code to Google authenticator. (valid input format and incorrect credential) (User role: teacher, student)	255736	Login failed and redirected and redirected to 401 error page.	**PASS**
Input random invalid security code. (invalid input format and/or incorrect credential) (User role: teacher, student)	64132qf	Login failed and redirected and redirected to 401 error page.	**PASS**

4.4.3.4 Testing of Home Interface

In this section some of the test cases and the outputs were attached for the Home interface of LMS Application.

User Interface: Home
Operation: Access home page and access features provided by the home interface.

Action	Input	Expected output	Status
Access the Homepage without logging into the platform. (Unauthorized user) (User role: teacher, student)	NONE	Redirect the user to the login page.	**PASS**
Access the Homepage after login and successful login verification. (Authorized user) (User role: teacher, student)	NONE	Home page is rendered w.r.t requesting user and user information.	**PASS**

4.4.3.5 Testing of Courses Interface

In this section some of the test cases and the outputs were attached for the courses interface of LMS Application.

User Interface: Courses
Operation: Access courses interface and access courses operations.

Action	Input	Expected output	Status
Access the courses page without logging into the platform. (Unauthorized user) (User role: teacher, student)	NONE	Redirect the user to the login page.	**PASS**
Access the Homepage after login and successful login verification. (Authorized user) (User role: teacher, student)	NONE	Courses page is rendered w.r.t requesting user and user information.	**PASS**
Enrol into a course that is not enrolled by the user. Click enroll btn for a specific course. (Authorized user and valid operation) (User role: teacher, student)	NONE	Success message: Enrolled into the course successfully.	**PASS**

Action	Input	Expected output	Status
Leave a course that is enrolled by the user. Click enroll btn for a specific course. Click opt out btn for a specific course. (Authorized user and valid operation) (User role: teacher, student)	NONE	Success message: Opt out of the course successfully.	**PASS**
Enroll into a course that is already enrolled by the user. Click enroll btn for a specific course. (Authorized user and invalid operation) (User role: teacher, student)	NONE	Failure message: Already enrolled into the course.	**PASS**
Leave from a course that is not enrolled by the user. Click opt out btn for a specific course. (Authorized user and invalid operation) (User role: teacher, student)	NONE	Failure message: Enrollment info regarding this course was not found.	**PASS**

4.4.3.6 Testing of Classes Interface

In this section some of the test cases and the outputs were attached for the classes interface of LMS Application.

User Interface: Classes
Operation: Access Classes interface and perform operations provided by classes interface.

Action	Input	Expected output	Status
Access the classes page without logging into the platform. (Unauthorized user) (User role: teacher, student)	NONE	Redirect the user to the login page.	**PASS**
Access the classes after login and successful login verification. (Authorized user) (User role: teacher, student)	NONE	Classes page is rendered w.r.t requesting user and user information.	**PASS**

Action	Input	Expected output	Status
For creating a classroom: Input class name and select course. (Authorized user valid input and Valid operation). (User role teacher)	C/C++ programming class term 1 CSCORE01 Programming in C/C++	Success message: Class created successfully	**PASS**
For creating a classroom: Input class name and but select a course for which a classroom already exists that was created by the same user. (Authorized user valid input and invalid operation). (User role teacher)	C/C++ programming class2 term 1 CSCORE01 Programming in C/C++	Failure message: Class for this course already exists.	**PASS**
For creating a classroom: Select a course but leave the class name field as blank. (Authorized user valid input and invalid operation). (User role teacher)	CSCORE02 Computer System Architecture	Failure message: Please fill out the form to continue.	**PASS**
Access a classroom by clicking the Go to class button in the classrooms created section. (Authorized user valid input and invalid operation). (User role teacher, student)	NONE	Success message: A dynamic page is rendered according to the requesting classrooms page and the user is redirected to the classrooms page.	**PASS**
Join a classroom by clicking the join button in the available classrooms section. (Authorized user valid operation). (User role student)	NONE	Success message: Class joined successfully	**PASS**
Join a classroom that was already joined by the same user by clicking the join button in the available classrooms section. (Authorized user invalid operation). (User role student)	NONE	Failure message: Student already joined the classroom.	**PASS**
Leave a classroom by clicking the leave button. (Authorized user valid operation). (User role student)	NONE	Success message: Class left successfully	**PASS**

Action	Input	Expected output	Status
Leave a classroom that was not joined by the same user by clicking the leave button in the available classrooms section. (Authorized user invalid operation). (User role student)	NONE	Failure message: Student has not joined this class.	**PASS**

4.4.3.7 Testing of Classroom Interface

In this section some of the test cases and the outputs were attached for the classes interface of LMS Application.

User Interface: Classroom
Operation: Access classroom interface and perform operation provided by classroom interface.

Action	Input	Expected output	Status
Access the classroom page without logging into the platform. (Unauthorized user) (User role: teacher, student)	NONE	Redirect the user to the login page.	**PASS**
Access the classroom page after login and successful login verification. (Authorized user) (User role: teacher, student)	NONE	Classroom page is rendered w.r.t classroom_id and requesting user and user information.	**PASS**
Create a post in a classroom: Input post title and attachment. (Authorized user and valid operation) (User role: teacher)	Test post 1 sample_attachment.pdf	New post created with attachment sample_attachment.pdf which can be downloaded by clicking on it.	**PASS**
Create a post in a classroom: Input post title but don't include any attachment. (Authorized user and valid operation) (User role: teacher)	Test post 2	New post created without any attachment.	**PASS**
Create a post in a classroom: Leave the fields blank. (Authorized user and invalid operation) (User role: teacher)	NONE	Failure msg: Fields cannot be empty.	**PASS**

Action	Input	Expected output	Status
Post a comment in a classroom: Give all the input. (Authorized user and valid operation) (User role: teacher, student)	Test comment 1	Success msg: Comment posted successfully.	**PASS**
Post a comment in a classroom: Leave the input field blank. (Authorized user and invalid operation) (User role: teacher, student)	NONE	Failure msg: Something went wrong..	**PASS**
Delete a comment in a classroom: (Authorized user and valid operation) (User role: teacher, student)	NONE	Success msg: Comment deleted successfully.	**PASS**
Leave from the classroom: By clicking the LEAVE CLASS button. (Authorized user and valid operation) (User role: student)	NONE	User successfully leaves from classroom.	**PASS**
Delete a classroom: By clicking the DELETE CLASS button. (Authorized user and valid operation) (User role: teacher)	NONE	Classroom is deleted successfully.	**PASS**

4.4.3.8 Testing of Admission Interface

In this section some of the test cases and the outputs were attached for the classes interface of LMS Application.

User Interface: Admission
Operation: Access admission interface and perform operations and actions provided by admission interface.

Action	Input	Expected output	Status
Access the admission page without logging into the platform. (Unauthorized user) (User role: student)	NONE	Redirect the user to the login page.	**PASS**

Action	Input	Expected output	Status
Access the admission page after successful login and login verification. (Authorized user) (User role: student)	NONE	Admission page is rendered w.r.t requesting user and user information.	**PASS**
Get admission into a different term by giving term info into the selection input field. (Authorized user and valid operation) (User role: student)	2	Success message: Admission successful	**PASS**
Get admission into a term in which the current user is already admitted. (Authorized user and invalid operation) (User role: student)	1	Failure message: Already admitted in this term.	**PASS**

4.4.3.9 Testing of Profile Interface

In this section some of the test cases and the outputs were attached for the classes interface of LMS Application.

User Interface: Classroom
Operation: Access classroom interface and perform operations provided by classroom interface.

Action	Input	Expected output	Status
Access the Profile page without logging into the platform. (Unauthorized user) (User role: student, teacher)	NONE	Redirect the user to the login page.	**PASS**
Access the Profile page after successful login and login verification. (Authorized user) (User role: student, teacher)	NONE	Profile page is rendered w.r.t requesting user and user information.	**PASS**

Action	Input	Expected output	Status
Change display picture by uploading a picture from local source through input upload form. (Authorized user and valid operation) (User role: student, teacher)	picture.png	The current display picture is changed and the uploaded picture will be shown.	**PASS**
Change email by giving input email through input form. The input email was not used before. (Authorized user and valid operation) (User role: student, teacher)	newemail@gmail.com	Success message: Email updated successfully.	**PASS**
Change email by giving input email through input form. The input email is used by the current user or another user. I.e the email is already taken. (Authorized user and invalid operation) (User role: student, teacher)	used.email@gmail.com	Failure message: Email is already taken or in use.	**PASS**
Change password by giving relevant input through input form(s). (Authorized user and valid operation) (User role: student, teacher)	newpassword@1234 newpassword@1234	Success message: Password updated successfully.	**PASS**
Change password by giving relevant input through input form(s). Give wrong input in repeat password form. (Authorized user and invalid operation) (User role: student, teacher)	newpassword@1234 newpassword_1234qwe	Failure message: Both fields must be the same.	**PASS**
Change email by giving input email through input form. But keep the input field empty. (Authorized user and invalid operation) (User role: student, teacher)	NONE	Failure message: Field cannot be empty.	**PASS**

4.4.4 Testing reports and considerations

The testing of the application was done in a development build using blackbox and white box testing approaches. During testing all the primary and secondary modules were thoroughly tested, some of the test cases and outcomes were described in **section 4.4.3** although many of the test cases were not shown. **The overall result of the tests were found to be satisfactory and no critical bugs were found (so far) in the development build**. Although no bugs were found during the testing, it is to be noted that testing can't possibly cover all the cases and extraordinary cases which may result in some race conditions or occurrence of new bugs. So the testing should have to be done again in future builds or after the addition of new features so that the occurrence of bugs keeps to be minimal. Some points to be considered is as follows:

- The testing process has to be done in regular intervals that is after addition of new features or when using a new build.
- The feedback from users (after deployment) have to be taken into account and if they find any new bugs the affected modules have to be investigated and tested again thoroughly this also applies to integration testing as well.

5. Project Planning & management

5.1 Introduction

Project management is the process of leading the work of a team to achieve all project goals within the given constraints. Project management involves the **planning** and **organization** of resources to move a specific task, event, or duty towards completion.

5.2 Project planning of LMS Application

Project planning is part of project management, which relates to the use of schedules such as Gantt charts to plan and subsequently report progress within the project environment. Project planning can be done manually or by the use of project management software. Project planning involves estimating several characteristics of a project and then planning the project activities based on these estimates made. Project planning involves estimating several characteristics of a project and then planning the project activities based on these estimates made.

Initially, the project scope is defined and the appropriate methods for completing the project are determined. Following this step, the durations for the various tasks necessary to complete the work are listed and grouped into a work breakdown structure. Project planning is often used to organize different areas of a project, including project plans, work loads and the management of teams and individuals. The logical dependencies between tasks are defined using an activity network diagram that enables identification of the critical path. The planning of a project generally consists of the steps that is: Estimation of cost, duration and effort, scheduling, staffing, risk management and other miscellaneous plans.

As this project falls under the category of an academic project the primary focus is given in project scheduling while project estimation and stuffing was not considered at the point of time.

5.3 Project scheduling

In project management, a schedule is a listing of a project's milestones, activities, and deliverables. Usually dependencies and resources are defined for each task, then start and finish dates are estimated from the resource allocation, budget, task duration, and scheduled events.

For development of this project a standard Gantt chart or timeline chart is used for scheduling of different tasks involved in development of the project. **Figure 11** shows an elementary Gantt chart for the development plan. The plan explains the tasks versus the time (in weeks) they will take to complete.

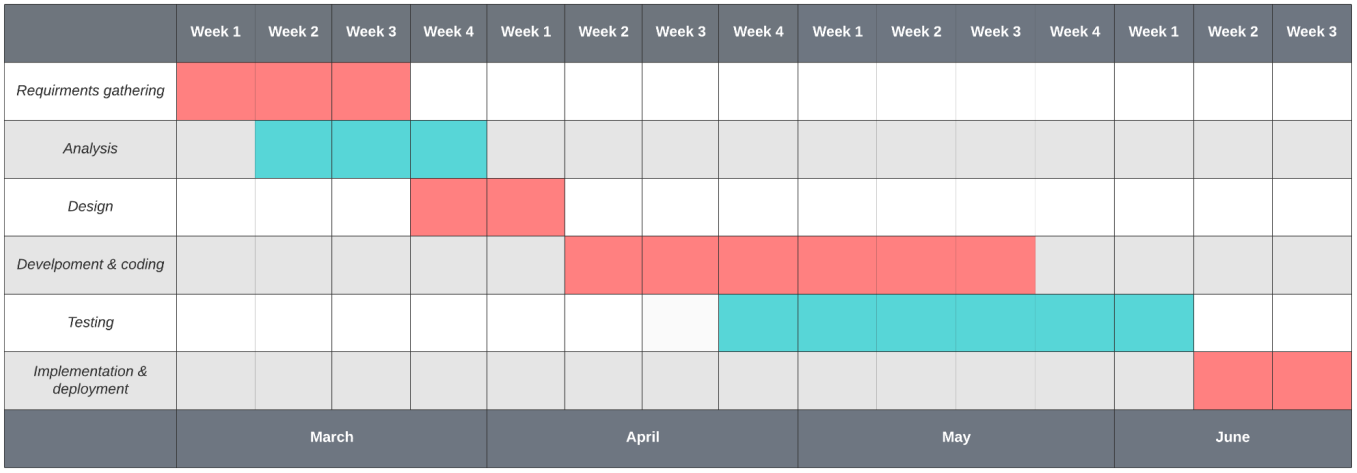

Figure 11: Gantt chart for project schedule