

# **Industrial Training Project Report**

**Submitted in partial fulfilment of the degree of**

**BTech**

**By**

**Aniket Saha (Lateral)**

**Julphikar Haque (Lateral)**

**Baishali Saha (Lateral)**

**Priya Shil (Lateral)**

**Second year student of**

**Siliguri Institute of Technology (SIT)**



**Under the supervision of**

**Mr. Mainak Deb**

**NATIONAL INSTITUTE FOR TECHNICAL  
TRAINING AND SKILL DEVELOPMENT(NITSD)**

---

I hereby forward the documentation prepared by **Group 4** (Aniket Saha, Baishali Saha, Julphikar Haque, Priya shil ) under the supervision of **Mr. Mainak Deb** Sir entitled **Learning Management System Using Django Framework** accepted as fulfilment of the requirement for the degree of bachelor of technology (B.TECH) in **computer Science and Engineering** from **Siliguri Institute of Technology (SIT)** affiliated to **Maulana Abul Kalam Azad University of Technology(MAKAUT)**.

---

---

**Mr. Mainak Deb**  
**(Technical Project Manager)**  
  
**Project Guide**  
  
**(NITSD)**

---

**Group 4**  
**Department of Computer Science**  
**and Engineering**  
**Siliguri Institute of Technology**

# Learning Management System

By

**Aniket Saha (Lateral)**

**Aniket Saha (Lateral)**

**Julphikar Haque (Lateral)**

**Baishali Saha (Lateral)**

**Priya Shil (Lateral)**

**UNDER THE GUIDANCE OF**

**Mr. Mainak Deb**

Project Guide

**(NITSD)**

THESE SUBMITTED IN FULFILLMENT OF THE  
REQUIREMENTS FOR THE DEGREE OF

**BTech**

IN

**COMPUTER SCIENCE AND ENGINEERING**

**SILIGURI INSTITUTE OF TECHNOLOGY**

**AFFILIATED TO**

**Maulana Abul Kalam Azad University of Technology**

**Address:** Hill Cart Road, Salbari, Sukna, West Bengal 734009

**Phone:**9434527272

**Emil id:** [mtsit@gmail.com](mailto:mtsit@gmail.com), **Website:** [sittechno.org](http://sittechno.org)

## **ACKNOWLEDGEMENT**

It is a great pleasure for me to acknowledge the assistance and participation of a large number of individuals to this attempt. Our project report has been structured under the valued suggestion, support and guidance of **Mr. Mainak Deb** Under his guidance we have accomplished the challenging task in a very short time.

Finally, we express our sincere thankfulness to our family members for inspiring me all throughout and always encouraging us.

**Group 4(LMS)**

**Department of Computer Science and Engineering**

## **TABLE OF CONTENTS**

### **Chapter 1: Introduction**

1.1: Introduction

### **Chapter 2: What We Used**

2.1: HTML

2.2: CSS

2.3: DJANGO

2.4: PYTHON

2.5: SQLite

### **Chapter 3: FEATURES**

3.1: Student

3.2: Teacher

3.3: Admin

### **Chapter 4: Functionality**

4.1: course.html

4.2: Assignments.html

4.3: Departments.html

4.4: Access.html

4.5: urls.py

4.6: views.py

4.7: settings.py

4.8: Database

### **Chapter 5: Functional Requirement of the System**

4.1: Hardware Requirement

4.2: Software Requirement

### **Chapter 6: Conclusion**

### **Chapter 7: References**

## ■ INTRODUCTION

---

**A Learning Management System (LMS)** is a software application that facilitates the management, delivery, and tracking of educational content and resources. It provides a centralized platform for educators to create courses, manage enrolments, deliver instructional materials, and assess student performance. **Django, a high-level Python web framework,** provides a robust foundation for developing custom LMS solutions due to its scalability, security features, and extensive ecosystem of libraries and plugins.

In this project, we'll leverage Django to build a basic LMS with key features such as user authentication, course creation, enrolment management, content delivery, and assessment tracking. By following a structured approach, we'll create a flexible and user-friendly platform that caters to the needs of both educators and learners

---

## ■ What We Used

**2.1: HTML:** HTML stands for Hypertext Markup Language. It allows the user to create and structure sections, paragraphs, headings, links, and blockquotes for web pages and applications. HTML is not a programming language, meaning it doesn't have the ability to create dynamic functionality. Instead, it makes it possible to organize and format documents, similarly to Microsoft Word. The average website includes several different HTML pages. For instance, a home page, an about page, and a contact page would all have separate HTML files. HTML documents are files that end with a .html or .html extension. A web

browser reads the HTML file and renders its content so that internet users can view it. All HTML pages have a series of HTML elements, consisting of a set of tags and attributes. HTML elements are the building blocks of a web page. A tag tells the web browser where an element begins and ends, whereas an attribute describes the characteristics of an element. To understand "HTML" from front to back, let's look at each word that makes up the abbreviation:

- **Hypertext:** text (often with embeds such as images, too) that is organized in order to connect related items
- **Markup:** a style guide for typesetting anything to be printed in hardcopy or soft copy format
- **Language:** a language that a computer system understands and uses to interpret commands.

**2.2: CSS:** stands for Cascading Style Sheets. It is a style sheet language used to describe the presentation of a document written in a markup language like HTML. CSS describes how elements should be displayed on the screen, in print, or spoken in a web page. It enables web developers to control the layout, colors, fonts, and other visual aspects of their web pages, ensuring consistency and aesthetic appeal across different devices and browsers. CSS operates by selecting HTML elements and applying various styles to them, either directly within HTML documents or externally in separate CSS files.

**2.3: DJANGO:** Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. Developed by Django Software Foundation, it follows the "don't repeat yourself" (DRY) principle and emphasizes reusability and pluggability of components, allowing developers to build web applications quickly and efficiently.

Key features of Django include:

1. Object-relational mapping (ORM): Django provides a convenient way to interact with databases through Python

code, abstracting away much of the complexity of SQL queries.

2. URL routing: URLs can be mapped to Python functions or classes called "views," allowing for clean and flexible URL configurations.
3. Template system: Django comes with a powerful template engine that allows developers to separate the design from the business logic of their applications.
4. Administration interface: Django automatically generates an admin interface for managing site content, which can be customized to fit specific needs.
5. Authentication and authorization: Django provides built-in support for user authentication, including user registration, login, and password management, as well as fine-grained access control through permissions and groups.
6. Security features: Django includes various security features out-of-the-box, such as protection against common web security vulnerabilities like SQL injection, cross-site scripting (XSS), and cross-site request forgery (CSRF).
7. Internationalization and localization: Django supports the creation of multilingual websites with built-in tools for translating text and formatting dates, numbers, and time zones according to user preferences.

Overall, Django is widely used for developing a wide range of web applications, from simple blogs and content management systems to complex enterprise-level platforms. Its robust features, active community, and extensive documentation make it a popular choice for developers seeking to build scalable, maintainable web applications in Python.

**2.4: PYTHON:** Python is a Programming language that is interpreted, object-oriented, and considered to be high-level too. Python is one of the easiest yet most useful programming languages which is widely used in the software industry. People use Python for Competitive Programming, Web Development,



and creating software. Due to its easiest syntax, it is recommended for beginners who are new to the software engineering field. Its demand is growing at a very rapid pace due to its vast use cases in Modern Technological fields like Data Science, Machine learning, and Automation Tasks. For many years now, it has been ranked among the top Programming language

**2.5: SQLite:** SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. It is a database, which is zero-configured, which means like other databases you do not need to configure it in your system.

SQLite engine is not a standalone process like other databases, you can link it statically or dynamically as per your requirement with your application. SQLite accesses its storage files directly.

## ■ **Features**

### **3.1: Student**

1. Student can easily get the study materials.
2. Can easily access the given assignments and task.
3. Student can participate in Quizzes.
4. Communicate with Teachers.

### **3.2: Teacher:**

1. Teacher can assign the assignments
2. Teacher provides the class materials
3. Teacher can also post announcement

### **3.3 Admin:**

- 1.Admin handle the admin section.
- 2.Assign the teacher for classes.
- 3.Provide class code and password.

## ■ Functionality of the Programme

### 4.1 Course .html:

```
{% extends 'index.html' %}
{% block title %} Courses | eLMS {% endblock title %}
{% block profile %}
{% if faculty %} {% url 'profile' faculty.faculty_id %}
{% else %} {% url 'profile' student.student_id %}
{% endif %}
{% endblock %}
{% block user %}
{% if faculty %} {{faculty.name}}
{% else %} {{student.name}}
{% endif %}
{% endblock user %}
{% block courses %}
{% url 'myCourses' %}
{% endblock courses %}
{% block allCourses %}
{% url 'courses' %}
{% endblock %}
{% block content %}
<div class="container">
  <!-- navigation links -->
  <div class="container shadow-sm">
    <nav style="--bs-breadcrumb-divider: url(&#34;data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' width='8' height='8'%3E%3Cpath d='M2.5 0L1
1.5 3.5 4 1 6.5 2.5 8 4-4-4z' fill='currentColor'/%3E%3C/svg%3E&#34;);"
aria-label="breadcrumb">
      <ol class="breadcrumb p-3">
        <li class="breadcrumb-item active animate__animated
animate__backInRight" aria-current="page">All courses</li>
      </ol>
    </nav>
  </div>
  <!-- navigation links end -->
  <div class="row gy-5">
    {% if courses %}
    {% for course in courses%}
    <!-- individual card starts -->
    <div class="col-sm-6 d-flex align-items-stretch justify-content-center
animate__animated animate__zoomIn">
      <div class="card shadow" style="width: 30rem !important;">
        <div class="class-header text-center text-light border-bottom bg-
dark rounded">
          <h6 class="my-2"> Dept. of {{course.department}}</h6>
        </div>
```

```

        <div class="card-body d-flex flex-column align-items-center">
            <h5 class="card-title fw-bold">{{course.department}}-
{{course.code}} : {{course.name}}</h5>
            {% if course.faculty %}
            <p class="card-text fw-bold">Course Teacher : {{
course.faculty.name}}</p>
            {% else %}
            <p> Teacher not assigned</p>
            {% endif %}
            {% if not faculty %}
            {% comment %} if student {% endcomment %}
            {% if course not in enrolled%}
            <a href="{% url 'access' course.code %}" class="btn btn-
outline-dark"><span class="px-2">Access</span></a>
            {% else %}
            <a href="{% url 'course' course.code %}" class="btn btn-
outline-dark"><span
                class="px-3">Enter</span></a>
            {% endif %}
            {% endif %}
        </div>
    </div>
</div>
<!-- individual card ends -->
{% endfor %}
</div>
{% else %}
<div class="text-center h4 text-secondary">No courses to show</div>
{% endif %}
</div>
{% endblock %}

```

## 4.2 Assignments.html:

```

{% extends 'index.html' %}
{% load static %}
{% block title %} {{ course.name }} | eLMS {% endblock title %}
{% block profile %}
{% url 'profile' faculty.faculty_id %}
{% endblock %}
{% block user %}
{{faculty.name}}
{% endblock user %}
{% block courses %}
{% url 'facultyCourses' %}
{% endblock courses %}
{% block allCourses %}

```

```

{% url 'courses' %}
{% endblock %}
{% block content %}
<!-- navigation links -->
<div class="container shadow-sm mb-4">
    <nav style="--bs-breadcrumb-divider: url(&#34;data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' width='8' height='8'%3E%3Cpath d='M2.5 0L1
1.5 3.5 4 1 6.5 2.5 8 4-4-4z' fill='currentColor'/%3E%3C/svg%3E&#34;);"
aria-label="breadcrumb">
        <ol class="breadcrumb p-3">
            <li class="breadcrumb-item fw-bold"><a style="color: rgb(10, 10, 48)"
href="{% url 'facultyCourses' %}">My
                courses</a>
            </li>
            <li class="breadcrumb-item fw-bold"><a style="color: rgb(10, 10, 48)"
href="{% url 'faculty' course.code %}">{{ course.name }}</a></li>
            <li class="breadcrumb-item active animate__animated
animate__backInRight" aria-current="page">Assignment</li>
        </ol>
    </nav>
</div>
<!-- navigation links end -->
<div class="container">
    <div class="row justify-content-center">
        <div class="col-sm-12 col-lg-8 shadow-lg rounded p-4">
            <div class="text-center">
                <div class="fw-bold text-start mb-3">Create Assignment</div>
                <div class="form-group">
                    <form action="" method="POST" enctype="multipart/form-data">
                        {% csrf_token %}
                        {% for field in form %}
                            <div class="mb-3">
                                {{ field.label_tag }}
                                {{ field }}
                                <span style="color:red">{{ field.errors }}</span>
                            </div>
                        {% endfor %}
                        <p class="fw-bold" id="fileError"></p>
                        <!-- submit button -->
                        <div class="d-grid col-12 mx-auto">
                            <button class="btn btn-primary my-2"
type="submit">Post</button>
                        </div>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

```

```

</div>
<script>
    let fileError = document.getElementById('fileError');
    // File format & size validation
    document.getElementById('file').addEventListener('change', function () {
        var file = this.files[0];
        var fileType = file.type;
        var validTypes = ['image/gif', 'image/jpeg', 'image/png',
        'application/pdf', 'application/msword', 'application/vnd.openxmlformats-officedocument.wordprocessingml.document'];
        if (!validTypes.includes(fileType)) {
            fileError.innerHTML = "Only pdf, png, jpg, jpeg, gif and word files are allowed";
            fileError.classList.add("text-danger", "fw-bold", "border", "border-danger", "p-2", "rounded");
            this.value = '';
        }
        else if (file.size > 10000000) {
            fileError.innerHTML = "File size should be less than 10MB";
            fileError.classList.add("text-danger", "fw-bold", "border", "border-danger", "p-2", "rounded");
            this.value = '';
        }
        else {
            fileError.innerHTML = "";
            fileError.classList.remove("text-danger", "fw-bold", "border", "border-danger", "p-2", "rounded");
        }
    });
</script>
{% endblock %}

```

## 4.3 Departments.html:

```

{% extends 'index.html' %}
{% block title %} Departments | eLMS {% endblock title %}
{% block profile %}
{% if faculty %} {% url 'profile' faculty.faculty_id %}
{% else %} {% url 'profile' student.student_id %}
{% endif %}
{% endblock %}
{% block user %}
{% if faculty %} {{faculty.name}}
{% else %} {{student.name}}
{% endif %}
{% endblock user %}

```

```

{% block courses %}
{% url 'myCourses' %}
{% endblock courses %}
{% block allCourses %}
{% url 'courses' %}
{% endblock %}
{% block departments %}
{% url 'departments' %}
{% endblock %}
{% block content %}
<div class="container">
  <!-- navigation links -->
  <div class="container shadow-sm">
    <nav style="--bs-breadcrumb-divider: url(&#34;data:image/svg+xml,%3Csvg
xmlns='http://www.w3.org/2000/svg' width='8' height='8'%3E%3Cpath d='M2.5 0L1
1.5 3.5 4 1 6.5 2.5 8 4-4-4z' fill='currentColor'/%3E%3C/svg%3E&#34;);"
aria-label="breadcrumb">
      <ol class="breadcrumb p-3">
        <li class="breadcrumb-item active animate__animated
animate__backInRight" aria-current="page">All Departments</li>
      </ol>
    </nav>
  </div>
  <!-- navigation links end -->
  {% if deps %}
  <span class="text-muted fst-italic">Showing all departments</span>
  <table class="table align-middle mb-0 bg-white text-center my-3 shadow
rounded">
    <thead class="bg-light">
      <tr>
        <th>Name</th>
        <th>Dept. ID</th>
        <th>No. of Students</th>
        <th>Total Faculty</th>
        <th>Total Courses</th>
      </tr>
    </thead>
    <tbody>
      {% for dep in deps%}
      <tr>
        <td>
          {{dep.name}}
        </td>
        <td>
          {{dep.department_id}}
        </td>
        <td>
          {{dep.student_count}}

```

```

        </td>
        <td>
            {{dep.faculty_count}}
        </td>
        <td>
            {{dep.course_count}}
        </td>
    </tr>
{% endfor %}
</tbody>
</table>

```

## 4.4 Access.html:

```

{% extends 'index.html' %}
{% block title %} {{ course.name }} | eLMS {% endblock title %}
{% block profile %}
{% url 'profile' student.student_id %}
{% endblock %}
{% block user %}
{{student.name}}
{% endblock user %}
{% block allCourses %}
{% url 'courses' %}
{% endblock %}
{% block courses %}
{% url 'myCourses' %}
{% endblock %}
{% block content %}
{% if messages %}
{% for message in messages %}
<div class="alert alert-danger alert-dismissible fade show text-center"
role="alert">
    <span class="fw-bold"> {{ message }}</span>
    <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>
</div>
{% endfor %}
{% endif %}

<div class="container-fluid d-flex" style="min-height: 70vh;">
    <div class="row justify-content-center m-3 align-self-center w-100">
        <div class="col-lg-4 col-sm-12 col-md-6 border p-3 rounded">
            <form action="" method="post">
                {% csrf_token %}

```

```

        <div class="mb-3">
            <label for="key" class="form-label">Enter the access key:
</label>
            <input type="number" class="form-control" id="key" name="key"
min="0" placeholder="e.g. 1234">
        </div>
        <button type="submit" class="btn btn-sm btn-
primary">Submit</button>
    </form>
</div>
</div>
</div>
{% endblock %}

```

## 4:5 urls.py:-

```

from django.urls import path
from . import views

urlpatterns = [
    path('', views.std_login, name='std_login'),
    path('my/', views.myCourses, name='myCourses'),
    path('facultyCourses/', views.facultyCourses, name='facultyCourses'),
    path('login/', views.std_login, name='std_login'),
    path('logout/', views.std_logout, name='std_logout'),
    path('my/<int:code>/', views.course_page, name='course'),
    path('profile/<str:id>/', views.profile, name='profile'),
    path('facultyProfile/<str:id>/', views.profile, name='profile_faculty'),
    path('faculty/<int:code>/', views.course_page_faculty, name='faculty'),
    path('addAnnouncement/<int:code>/',
        views.addAnnouncement, name='addAnnouncement'),
    path('announcement/<int:code>/<int:id>/',
        views.deleteAnnouncement, name='deleteAnnouncement'),
    path('edit/<int:code>/<int:id>/',
        views.editAnnouncement, name='editAnnouncement'),
    path('update/<int:code>/<int:id>/',
        views.updateAnnouncement, name='updateAnnouncement'),
    path('addAssignment/<int:code>/', views.addAssignment,
name='addAssignment'),
    path('assignment/<int:code>/<int:id>/',
        views.assignmentPage, name='assignmentPage'),
    path('assignments/<int:code>/', views.allAssignments,
name='allAssignments'),
    path('student-assignments/<int:code>/',
        views.allAssignmentsSTD, name='student-assignments'),
    path('addSubmission/<int:code>/<int:id>/',
        views.addSubmission, name='addSubmission'),

```



```

path('submission/<int:code>/<int:id>/',
     views.viewSubmission, name='submission'),
path('gradeSubmission/<int:code>/<int:id>/<int:sub_id>',
     views.gradeSubmission, name='gradeSubmission'),
path('course-material/<int:code>/',
     views.addCourseMaterial, name='addCourseMaterial'),
path('course-material/<int:code>/<int:id>/',
     views.deleteCourseMaterial, name='deleteCourseMaterial'),
path('courses/', views.courses, name='courses'),
path('departments/', views.departments, name='departments'),
path('access/<int:code>/', views.access, name='access'),
path('changePasswordPrompt/', views.changePasswordPrompt,
     name='changePasswordPrompt'),
path('changePhotoPrompt/', views.changePhotoPrompt,
name='changePhotoPrompt'),
path('changePassword/', views.changePassword, name='changePassword'),
path('changePasswordFaculty/', views.changePasswordFaculty,
     name='changePasswordFaculty'),
path('changePhoto/', views.changePhoto, name='changePhoto'),
path('changePhotoFaculty/', views.changePhotoFaculty,
     name='changePhotoFaculty'),
path('search/', views.search, name='search'),
path('error/', views.error, name='error'),
]

```

## 4:6 views.py:

```

from django.shortcuts import redirect, render
from discussion.models import FacultyDiscussion, StudentDiscussion
from main.models import Student, Faculty, Course
from main.views import is_faculty_authorised, is_student_authorised
from itertools import chain
from .forms import StudentDiscussionForm, FacultyDiscussionForm

# Create your views here.

''' We have two different user models.
    That's why we are filtering the discussions based on the user type and
    then combining them.'''

def context_list(course):
    try:
        studentDis = StudentDiscussion.objects.filter(course=course)
        facultyDis = FacultyDiscussion.objects.filter(course=course)

```

```

        discussions = list(chain(studentDis, facultyDis))
        discussions.sort(key=lambda x: x.sent_at, reverse=True)

        for dis in discussions:
            if dis.__class__.__name__ == 'StudentDiscussion':
                dis.author = Student.objects.get(student_id=dis.sent_by_id)
            else:
                dis.author = Faculty.objects.get(faculty_id=dis.sent_by_id)
        except:

            discussions = []

        return discussions

def discussion(request, code):
    if is_student_authorised(request, code):
        course = Course.objects.get(code=code)
        student =
Student.objects.get(student_id=request.session['student_id'])
        discussions = context_list(course)
        form = StudentDiscussionForm()
        context = {
            'course': course,
            'student': student,
            'discussions': discussions,
            'form': form,
        }
        return render(request, 'discussion/discussion.html', context)

    elif is_faculty_authorised(request, code):
        course = Course.objects.get(code=code)
        faculty =
Faculty.objects.get(faculty_id=request.session['faculty_id'])
        discussions = context_list(course)
        form = FacultyDiscussionForm()
        context = {
            'course': course,
            'faculty': faculty,
            'discussions': discussions,
            'form': form,
        }
        return render(request, 'discussion/discussion.html', context)
    else:
        return redirect('std_login')

def send(request, code, std_id):
    if is_student_authorised(request, code):

```

```

        if request.method == 'POST':
            form = StudentDiscussionForm(request.POST)
            if form.is_valid():
                content = form.cleaned_data['content']
                course = Course.objects.get(code=code)
                try:
                    student = Student.objects.get(student_id=std_id)
                except:
                    return redirect('discussion', code=code)
                StudentDiscussion.objects.create(
                    content=content, course=course, sent_by=student)
                return redirect('discussion', code=code)
            else:
                return redirect('discussion', code=code)
        else:
            return redirect('discussion', code=code)
    else:
        return render(request, 'std_login.html')

def send_fac(request, code, fac_id):
    if is_faculty_authorised(request, code):
        if request.method == 'POST':
            form = FacultyDiscussionForm(request.POST)
            if form.is_valid():
                content = form.cleaned_data['content']
                course = Course.objects.get(code=code)
                try:
                    faculty = Faculty.objects.get(faculty_id=fac_id)
                except:
                    return redirect('discussion', code=code)
                FacultyDiscussion.objects.create(
                    content=content, course=course, sent_by=faculty)
                return redirect('discussion', code=code)
            else:
                return redirect('discussion', code=code)
        else:
            return redirect('discussion', code=code)
    else:
        return render(request, 'std_login.html')

```

## 4.7 Setings.py:

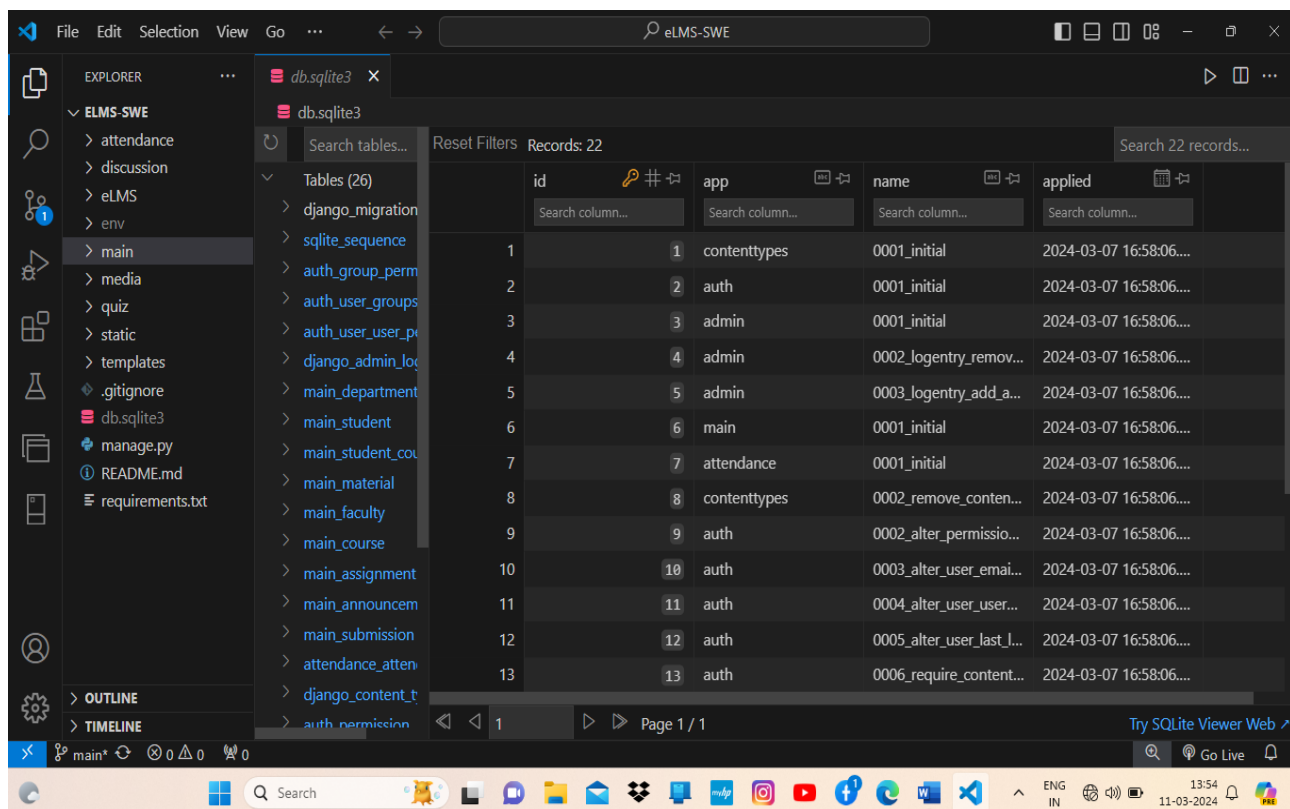
```
STATICFILES_DIRS = [
    BASE_DIR / 'static',
]
MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

# Default primary key field type
# https://docs.djangoproject.com/en/4.0/ref/settings/#default-auto-field

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

SESSION_EXPIRE_AT_BROWSER_CLOSE = True
```

## 4:8 Database part:



The screenshot shows the VS Code interface with the 'db.sqlite3' database viewer open. The Explorer sidebar on the left shows the project structure, with 'main' selected. The database viewer displays a table with 22 records, showing columns: id, app, name, and applied. The table lists various database entries including migrations, authentication, and content types.

id	app	name	applied
1	contenttypes	0001_initial	2024-03-07 16:58:06...
2	auth	0001_initial	2024-03-07 16:58:06...
3	admin	0001_initial	2024-03-07 16:58:06...
4	admin	0002_logentry_remov...	2024-03-07 16:58:06...
5	admin	0003_logentry_add_a...	2024-03-07 16:58:06...
6	main	0001_initial	2024-03-07 16:58:06...
7	attendance	0001_initial	2024-03-07 16:58:06...
8	contenttypes	0002_remove_conten...	2024-03-07 16:58:06...
9	auth	0002_alter_permissio...	2024-03-07 16:58:06...
10	auth	0003_alter_user_emai...	2024-03-07 16:58:06...
11	auth	0004_alter_user_user...	2024-03-07 16:58:06...
12	auth	0005_alter_user_last_l...	2024-03-07 16:58:06...
13	auth	0006_require_content...	2024-03-07 16:58:06...

## ■ Functionality Requirement of the System

Being a web-based solution the first and foremost thing that starts acquiring importance in this project is the way the complete package needs to be configured. Web-based solutions by virtue of their designs are mostly thin client solutions (unless they are heavy on DHTML). To run this kind of a solution properly it is necessary that the Server configurations are properly worked out. It is the server that will have to ultimately scale up as and when the numbers of users start increasing.

### 4.1 HARDWARE REQUIREMENTS

The minimum Hardware requirements for the application to run smoothly should have the following configuration:

<b>Processor</b>	Intel Core i3
<b>RAM</b>	4GB or more
<b>HDD</b>	3GB or more

### 4.2 SOFTWARE SPECIFICATIONS

The minimum software requirements are as follows:

<b>Operating System</b>	Windows 7,8 and upwards
<b>Language Used</b>	Python, HTML, CSS, SQLite
<b>Working IDE</b>	Visual Studio Code, XAMPP

## ■ **CONCLUSION**

Developing a Learning Management System (LMS) using Django offers numerous advantages. Django provides a robust and efficient framework for building web applications, making it well-suited for creating complex systems like an LMS. Here's a simple conclusion highlighting the key points: In conclusion, building a Learning Management System (LMS) using Django provides a powerful platform for educational institutions and organizations to manage their courses, users, and content effectively. By leveraging Django's features such as its ORM for database interactions, built-in authentication and authorization system, and customizable admin interface, developers can rapidly develop a feature-rich LMS tailored to the specific needs of learners and educators. Additionally, Django's scalability and extensibility ensure that the LMS can grow and adapt to evolving requirements over time. Overall, Django simplifies the development process, resulting in a reliable and efficient LMS solution that enhances the teaching and learning experience for all stakeholders.

## ■ **Reference**

1. <https://www.geeksforgeeks.org/college-management-system-using-django-python-project>
2. [https://www.youtube.com/watch?v=5BDgKJFZMI8&list=PLu0W\\_9lI9ah7DDtYtflgwMwpT3xmjXY9&index=1&t=6s](https://www.youtube.com/watch?v=5BDgKJFZMI8&list=PLu0W_9lI9ah7DDtYtflgwMwpT3xmjXY9&index=1&t=6s)
3. <https://www.youtube.com/watch?v=HRLIEgwYSHc&t=635s>