Created by: Bryan Loo Chun Wai

Advanced Web Development

Final project assignment

Student details

Name: Bryan Loo Chun Wai

Description: E-learning application

Table of Contents

Section 1 - Introduction

1.1) Overview

The rapid evolution of digital technologies have fundamentally transformed the landscape of education. In the contemporary digital era, online education platforms have transcended their status as mere supplementary tools, evolving into indispensable resources for learning. With the relentless advancements in technology, E-learning applications have garnered significant importance in both academic and corporate training environments. These platforms offer learners an unprecedented degree of flexibility and interactivity, enabling them to acquire new skills and knowledge without the constraints of geographical limitations.

The rise of the internet has democratized access to education, allowing individuals from diverse backgrounds to pursue learning opportunities that were previously out of reach. This shift has been particularly beneficial for non-traditional students, including working professionals, parents, and individuals in remote areas. E-learning applications have become essential in providing tailored educational experiences that cater to the unique needs of these learners.

This project focuses on the meticulous design and robust implementation of a sophisticated and scalable E-learning web application, leveraging the power of Django, Django REST Framework (DRF), Web Sockets, Celery, and Redis. This application is engineered to provide a secure and efficient learning environment, ensuring stringent authentication protocols, granular role-based access control, comprehensive course management functionalities, real-time chat capabilities, and a structured feedback mechanism.

The primary aim of this project is to create an intuitive and feature-rich platform that empowers students to enrol in diverse courses, interact seamlessly with teachers, provide valuable feedback, and engage in meaningful discussions. Concurrently, teachers are equipped with the tools to create and manage courses, upload a wide array of study materials, and maintain effective communication with their students. This report meticulously documents the entire system development lifecycle (SDLC), from the initial planning and conceptualization stages to the detailed implementation, rigorous testing, and insightful exploration of future improvements.

1.2) Objectives

The objectives of this project are meticulously defined to ensure the creation of a high-quality, functional, and scalable E-learning platform:

1) Develop a Secure and Scalable E-learning Platform
    - To construct a robust platform that can accommodate a growing number of users and courses, while maintaining the highest standards of security and data integrity. This includes implementing encryption protocols, secure data storage, and regular security audits to protect user information.

2) Implement Role-Based Authentication for Students and Teachers
    - To establish a secure and efficient authentication system that differentiates between student and teacher roles, granting appropriate access and permissions. This involves creating distinct user interfaces and functionalities tailored to each role, ensuring that users can only access features relevant to their responsibilities.

3) Provide Functionalities for Course Management, Including Creation, Enrolment, and Material Uploads
    - To develop a comprehensive course management system that enables teachers to create and manage courses, and students to enrol and access course materials seamlessly. This includes features such as course categorization, progress tracking, and the ability to upload various types of learning materials.

4) Enable Real-Time Communication Between Students and Teachers Through Web Sockets
    - To integrate real-time chat functionality, fostering immediate and effective communication between students and teachers. This feature is crucial for enhancing the learning experience, allowing for quick clarification of doubts and fostering a sense of community among users.

5) Develop a RESTful API to Facilitate Interaction Between Different Application Components
    - To create a well-documented and efficient API that enables seamless interaction between the frontend and backend components of the application. This API will serve as the backbone of the application, allowing for easy integration with third-party services and mobile applications.

6) Implement Unit Testing to Ensure System Reliability
    - To conduct thorough unit testing to validate the functionality and reliability of the application's core components. This includes writing test cases for all major functionalities and ensuring that the application behaves as expected under various conditions.

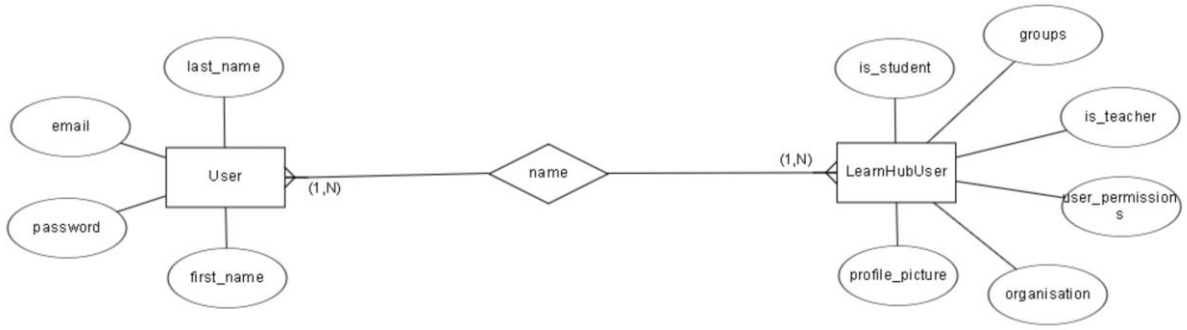
Section 2 - System Design and Architecture

2.1) Database Schema

The application adheres to a relational database model, utilizing Django's Object-Relational Mapper (ORM) for efficient database interactions. The key database entities and their relationships are meticulously designed to ensure data integrity and efficient retrieval.

2.1.1) Key Database Entities

ER Diagram for User and LearnHubUser models:



1) **User**

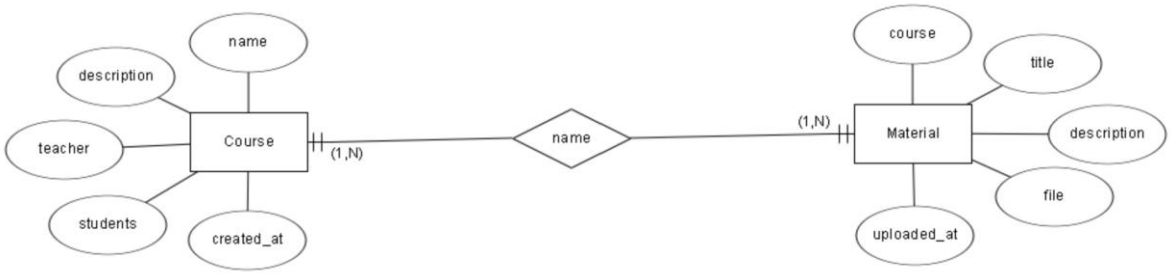
Description: This entity stores fundamental authentication details, including username, email, and password, providing the foundation for user identification and access control. The User model is extended to include additional fields that may be relevant for the application, such as profile pictures and user preferences.

2) **LearnHubUser**

Column Name	Data Type	Constraints
ID	Serial	Primary Key
User_id	Integer	Unique, Not null, References auth_user(id) On Delete Cascade
Is_student	Boolean	Default False
Is_teacher	Boolean	Default False
Profile picture	Varchar(255)	Nullable
Organisation	Varchar(255)	Nullable

Description: This entity extends the Django User model, enabling the differentiation between students and teachers through a 'role' field, facilitating role-based access control. This design allows for easy management of user permissions and ensures that the application can scale to accommodate various user types in the future.

ER Diagram for Course and Material models:



3) Course

Column name	Data type	Constraints
Id	Serial	Primary key
Name	Varchar(255)	Not null
Description	Text	Not null
Teacher_id	Integer	Not null, References auth_user(id) On Delete Cascade
Created_at	Timestamp	Default Current_timestamp

Description: This entity represents the various courses available on the platform, establishing a crucial link between teachers and enrolled students, and storing crucial course metadata such as course title, description, and creation date. Each course can also have associated tags or categories to facilitate easier searching and filtering.

4) Material

Column name	Data type	Constraints
Id	Serial	Primary key
Course_id	Integer	Not null, References Course(id) On Delete Cascade
Title	Varchar(255)	Not null
Description	Text	Nullable
File	Varchar(255)	Not null
Uploaded_at	Timestamp	Default Current_timestamp

Description: This entity stores uploaded course materials, encompassing a wide range of file types, including PDFs and images, and associating them with specific courses. The design allows for version control of materials, enabling teachers to update resources while keeping track of previous versions.

5) **Feedback**

Column name	Data type	Constraints
Id	Serial	Primary key
Course_id	Integer	Not null, References Course(id) On Delete Cascade
Student_id	Integer	Not null, References auth_user(id) On Delete Cascade
Comment	Text	Not null
Created_at	Timestamp	Default Current_timestamp

Description: This entity allows students to provide valuable feedback on their enrolled courses, capturing their experiences and suggestions. Feedback can be structured to include ratings, comments, and suggestions for improvement, providing teachers with actionable insights.

6) **ChatMessage**

Column name	Data type	Constraints
Id	Serial	Primary key
Room_name	Varchar(255)	Not null
Sender	Varchar(255)	Not null
Message	Text	Not null
Timestamp	Timestamp	Default Current_timestamp

Description: This entity manages real-time messages exchanged between users, storing message content, sender, receiver, and timestamp. The chat system is designed to support both one-on-one and group conversations, enhancing collaboration among students and teachers.

7) **EnrollmentNotification**

Column name	Data type	Constraints
Id	Serial	Primary key
Course_id	Integer	Not null, References Course(id) On Delete Cascade
Student_id	Integer	Not null, References auth_user(id) On Delete Cascade
Teacher_id	Integer	Not null, references auth_user(id) On Delete Cascade
Created_at	Timestamp	Default Current_timestamp
Read	Boolean	Default False

Description: This entity notifies teachers when a student enrolls in their course, ensuring timely awareness of student enrollment. Notifications can be customized to include relevant details about the student and the course.

8) **ResourceNotification**

Column name	Data type	Constraints
Id	Serial	Primary key
Resource_id	Integer	Not null, References Material(id) On Delete Cascade
Student_id	Integer	Not null, References auth_user(id) On Delete Cascade
Created_at	Timestamp	Default Current_timestamp
Read	Boolean	Default False

Description: This entity notifies students when new learning materials are uploaded to their enrolled courses, keeping them informed of updates. This feature is essential for maintaining engagement and ensuring that students have access to the latest resources.

Conclusion

The meticulously defined relationships between these entities ensure efficient data retrieval and management, facilitating seamless interaction between different components of the application. For example, a "Course" has a one-to-many relationship with "Material," meaning one course can have multiple materials. This relational design allows for complex queries and efficient data management.

2.2) User Roles and Permissions

The system employs role-based access control (RBAC) to provide differentiated access for students and teachers, ensuring that each user has the appropriate permissions based on their role.

1) Students:

- Can enrol in courses, gaining access to course materials and participating in learning activities.
- Can post status updates, sharing their progress and insights with the community. This feature fosters a sense of belonging and encourages peer-to-peer interaction.
- Can leave feedback on courses, providing valuable input for course improvement. Feedback mechanisms are designed to be user-friendly, encouraging students to share their thoughts openly.
- Can participate in real-time chats, engaging in discussions and seeking assistance from teachers and peers. The chat system is designed to be intuitive, allowing for easy navigation and interaction.

2) Teachers:

- Can create and manage courses, designing and delivering educational content. This includes the ability to set prerequisites, define learning objectives, and structure course materials effectively.
- Can upload learning materials, providing students with comprehensive resources. Teachers can also organize materials into modules or units, enhancing the learning experience.
- Can monitor student enrolments.
- Can communicate with students via real-time chat, providing immediate support and guidance. The chat system is designed to facilitate quick responses and foster a collaborative learning environment.
- Can remove students from courses for administrative purposes. This functionality is essential for maintaining course integrity and ensuring that students meet the necessary requirements.

2.3) Application Architecture

The application follows the Model-View-Controller (MVC) architectural pattern, promoting modular development and maintainability.

1) Frontend
   - The frontend utilizes Django templates and Bootstrap to create a responsive and user-friendly interface, ensuring a seamless user experience across various devices. The design is focused on accessibility, ensuring that all users can navigate the platform easily.
2) Backend
   - Django serves as the backend framework, managing business logic, database interactions, and API endpoints, providing a robust and efficient foundation for the application. The backend is designed to handle high traffic loads, ensuring that the application remains responsive even during peak usage.
3) Database
   - SQLite is employed as the persistent storage solution, offering reliability, scalability, and data integrity. The choice of SQLite allows for advanced querying capabilities and support for complex data types.
4) Web Sockets
   - Django Channels is used to implement Web Sockets, enabling real-time chat functionality and fostering immediate communication. This technology allows

for bidirectional communication between the server and clients, enhancing the interactivity of the platform.

5) Celery & Redis

- Celery and Redis are utilized for asynchronous background tasks, such as sending notifications and processing resource uploads, ensuring optimal performance and responsiveness. This architecture allows for the decoupling of long-running tasks from the main application flow, improving user experience.

This architecture ensures a scalable, maintainable, and performant E-learning platform, capable of accommodating future growth and enhancements. The modular design allows for easy updates and the addition of new features without disrupting existing functionalities.

Section 3 - Implementation

3.1) User Authentication and Registration

The authentication system is designed to ensure secure user registration and login, leveraging Django's built-in authentication framework.

1) User Registration

The registration process includes robust username and password validation, with strength checks to ensure secure account creation. Users are required to provide a valid email address, which is verified through a confirmation link sent to their inbox. This step is crucial for preventing spam accounts and ensuring that users have access to their accounts.

2) Login & Logout Mechanism

Secure authentication is implemented using Django's session management, providing a reliable and efficient login and logout experience. Users can choose to remain logged in for convenience, with the option to log out at any time. The system also includes measures to prevent brute-force attacks, such as account lockout after multiple failed login attempts.

3) Role Assignment

Users are assigned roles as either students or teachers during registration, restricting access to specific functionalities based on their role. This role assignment is crucial for maintaining the integrity of the application and ensuring that users can only access features relevant to their responsibilities.

3.2) Course Management

The course management system provides comprehensive functionalities for teachers and students.

1) Teachers
    - Teachers can create and manage courses, define course content, and organize learning materials. The course creation process includes setting learning objectives, defining prerequisites, and structuring the course into modules or units. Teachers can also set start and end dates for courses, allowing for time-bound learning experiences.
2) Students
    - Students can enroll in courses from the available list, gaining access to course materials and participating in learning activities. The enrollment process is designed to be straightforward, with clear instructions and prompts to guide students through the process.
3) Material Uploads
    - Teachers can upload and manage various course materials, including PDFs and images, providing students with comprehensive resources. The system supports multiple file formats and includes features for organizing materials into folders or categories for easy access.
4) Resource Access
    - Students can access uploaded resources within their enrolled courses, ensuring easy access to learning materials. The platform includes a search functionality that allows students to quickly find specific resources based on keywords or tags.

3.3) Feedback and Notifications

The feedback and notification system enhances communication and engagement.

1) Feedback Submission
    - Students can leave feedback about their enrolled courses, providing valuable input for course improvement. The feedback form is designed to be user-friendly, allowing students to rate their experience and provide comments easily.
2) Teacher Notifications

- Teachers receive notifications when new feedback is submitted, enabling them to address student concerns and suggestions. This feature is essential for fostering a responsive learning environment where students feel heard and valued.

3) Automatic Alerts

- Automatic alerts are triggered when students enroll in courses or new materials are uploaded, keeping users informed of updates. Notifications can be customized based on user preferences, allowing for a personalized experience.

3.4) Real-Time Communication (WebSockets)

The chat system enables instant student-teacher communication using Django Channels.

1) Live Chat Rooms
   - Live chat rooms allow students and teachers to discuss course materials and engage in real-time conversations. The chat system supports both public and private chat rooms, enabling group discussions as well as one-on-one interactions.

2) Message Storage

   - Messages are stored for future reference, ensuring that conversations are preserved and accessible. This feature is crucial for maintaining a record of discussions and allowing users to revisit important information.

3) Unread Message Notifications

   - Automatic notifications are provided for unread messages, ensuring that users are aware of new communications. This feature enhances engagement and encourages timely responses to inquiries.

3.5) REST API Development

A RESTful API is developed to facilitate interactions between the frontend and backend.

- /api/users/: This endpoint retrieves user information, providing access to user details. It supports various query parameters to filter and sort user data based on specific criteria.
- /api/courses/: This endpoint fetches available courses, enabling users to browse and enroll in courses. The API supports pagination and filtering options, allowing users to find courses that match their interests.
- /api/learnhub/feedback/: This endpoint submits and retrieves feedback, facilitating the collection and management of student feedback. The API allows for bulk retrieval of feedback data, enabling teachers to analyze trends and patterns.

The API is designed to be comprehensive and well-documented, ensuring seamless integration between the frontend and backend components. The documentation includes examples of requests and responses, making it easy for developers to understand how to interact with the API.

3.6) Serializer, API and validators

Serialization is a fundamental component of the Django REST Framework (DRF), enabling the conversion of complex data structures—such as Django model instances—into JSON format for API responses. This facilitates seamless communication between the backend and frontend, ensuring that data can be easily transmitted, stored, and interpreted by client applications.

In this application, serializers play a pivotal role in data representation, transformation, and validation. By using Django's built-in serializer classes, the application ensures structured and efficient handling of data. The key serializers implemented include:

- **UserSerializer**

  Converts user model instances into JSON format, allowing for seamless API interaction when retrieving or modifying user data.

- **CourseSerializer & CourseDetailSerializer**

  These serializers structure course-related data, including course details, student enrollments, and associated materials. CourseDetailSerializer provides an in-depth representation of a course, including nested relationships where necessary.

- **FeedbackSerializer**

  Standardizes feedback submissions, ensuring they follow a predefined format while preventing malicious input or improper data submissions. This enhances security and data integrity.

- **ChatMessageSerializer**

  Facilitates real-time messaging by converting chat messages into JSON format, enabling quick and efficient data exchange in live chat functionalities.

RESTful API Implementation

The application follows **RESTful principles**, ensuring modularity, scalability, and ease of access to resources. REST (Representational State Transfer) principles help in designing a stateless and resource-oriented API structure, allowing different parts of the application to interact seamlessly.

Key API endpoints include:

- **/api/users/** - Retrieves and manages user-related data, such as profile details and authentication information.
- **/api/courses/** - Provides a structured list of available courses, including course descriptions, enrolled students, and related content.
- **/api/feedback/** - Enables users to submit and retrieve feedback, ensuring a structured mechanism for user engagement and quality improvement.

Each endpoint is designed to follow standard HTTP methods (GET, POST, PUT, DELETE), ensuring a consistent approach to data management.

Data Validation and Security Measures

To maintain data integrity and enhance security, various validators have been implemented within the application. These validators prevent improper or malicious data entry, ensuring smooth operation and a positive user experience. Examples of key validation mechanisms include:

- **Username Validator**: Restricts usernames to alphanumeric characters only, preventing the use of special characters that could introduce security risks or database inconsistencies.
- **Password Strength Validator**: Enforces strong password policies by requiring at least one digit, one letter, and a minimum password length of 8 characters, thereby enhancing account security.
- **File Type Validator**: Restricts file uploads to specific formats such as PDF, DOCX, PNG, and other approved types. This helps prevent the upload of potentially harmful or unsupported files.

Conclusion

By leveraging **serialization**, **RESTful API design**, and **rigorous validation mechanisms**, this application ensures efficient data handling, robust security, and an optimal user experience. These measures collectively enhance the reliability and maintainability of the system, providing a solid foundation for future scalability and feature expansion.

Section 4 - Testing and Deployment

4.1) Unit Testing

Unit tests are a cornerstone of ensuring system reliability and maintainability. In this project, unit tests were meticulously implemented to validate critical functionalities.

- Authentication Tests:
    - These tests verify the correctness of login, registration, and logout functionalities. They include checks for valid and invalid credentials, password strength, and session management.
    - Tests also confirm that role assignment is correctly implemented during registration. This ensures that users are granted the appropriate permissions based on their roles.
- Course Management Tests:

- These tests ensure the correct creation and management of courses. They validate the functionality of adding, editing, and deleting courses.
- Tests also verify student enrollment and access to course materials. This includes checking that students can only access materials for courses they are enrolled in.
- Tests that teachers can correctly upload and manage materials. This ensures that the material upload process is functioning as intended.

4.2) Deployment

- Security Measures:
    - HTTPS encryption is implemented using SSL certificates, ensuring secure communication between clients and the server. This is essential for protecting user data during transmission.
    - CSRF (Cross-Site Request Forgery) protection is enabled to prevent malicious attacks. This ensures that users cannot be tricked into performing actions without their consent.
    - Security groups and IAM (Identity and Access Management) roles are configured to restrict access to resources. This ensures that only authorized users can access sensitive data and functionalities.
    - Regular security audits are performed to identify and address potential vulnerabilities. This proactive approach helps maintain the integrity of the application.

4.3) Installation Guide

For developers and administrators, a detailed installation guide is provided.

1. Install Dependencies:
    - cd elearning
    - pip install -r requirements.txt
2. Install Redis:
    - https://redis.io/lp/try1/?utm_campaign=gg_s_core_apac_en_brand_acq_21161918358&utm_source=google&utm_medium=cpc&utm_content=redis_exact&utm_term=&gad_source=1&gclid=Cj0KCQiAlbW-BhCMARIsADnwasob0pJjxKk8MG1x7cQ3ewC7ZrZgURUS-3RhFrLz22eDvVpbed4TJ5AaAmDEEALw_wcB
3. Apply Database Migrations:
    - python manage.py migrate
4. Start the Server:
    - python manage.py runserver

5. Start Redis and Celery Workers:
    - Start Redis: redis-server

Demo Credentials:

- Student:
    - Username: student1
    - Password: password123
- Teacher:
    - Username: teacher1
    - Password: password123

Section 5 - Evaluation and Future Improvements

5.1) Strengths

- Secure Authentication System: The application provides a robust and secure authentication system, protecting user data and ensuring authorized access. This is achieved through industry-standard practices, including password hashing and secure session management.
- Role-Based Access Control: RBAC ensures that users have appropriate permissions based on their roles, enhancing security and usability. This design allows for easy management of user permissions and ensures that users can only access features relevant to their responsibilities.
- Real-Time Communication: WebSockets enable instant communication between students and teachers, fostering a collaborative learning environment. This feature enhances engagement and allows for quick resolution of questions and concerns.
- RESTful API: The well-designed API facilitates seamless data retrieval and interaction between different components. This API is essential for integrating with third-party services and mobile applications, allowing for a more comprehensive learning experience.
- Scalable Architecture: The application's architecture is designed for scalability, allowing it to handle a growing number of users and courses. This design ensures that the application can adapt to increasing demand without compromising performance.

5.2) Challenges and Limitations

- Performance Bottlenecks in High-Load Scenarios: While the architecture is designed for scalability, performance bottlenecks may arise in extremely high-load scenarios. This could be addressed through further optimization of database queries and caching strategies.
- Limited Multimedia Support: While the application supports various file types, advanced multimedia features, such as integrated video lectures, could be further enhanced. Future iterations could explore the integration of video hosting services to provide a richer learning experience.
- Mobile Responsiveness: While the website is responsive, a dedicated mobile application could provide a better user experience for mobile users. Developing native mobile applications for iOS and Android could enhance accessibility and engagement.
- Accessibility: Further improvements can be made to improve accessibility for users with disabilities. This includes implementing features such as screen reader compatibility and keyboard navigation support.

5.3) Future Enhancements

- AI-Driven Course Recommendations: Implement AI algorithms to provide personalized course recommendations based on user preferences and learning history. This feature could enhance user engagement and help students discover relevant courses.
- Mobile App Integration: Develop dedicated mobile applications for iOS and Android, providing a seamless mobile learning experience. This would allow users to access course materials and participate in discussions on the go.
- Gamification Features: Incorporate gamification elements, such as badges, points, and leaderboards, to enhance user engagement and motivation. This approach could encourage students to complete courses and participate actively in discussions.
- Multimedia Content Support: Enhance multimedia support by integrating video lectures, interactive simulations, and other rich media content. This would provide a more engaging learning experience and cater to different learning styles.
- Advanced Analytics: Implement advanced analytics tools to track user progress, identify learning patterns, and provide insights for course improvement. This data could be invaluable for teachers in refining their course content and delivery methods.
- Integration with Third-Party Tools: Integrate with third-party tools, such as learning management systems (LMS) and video conferencing platforms. This would allow for a more comprehensive learning experience and facilitate collaboration among users.
- Improved Search Functionality: Improve the search functionality to allow users to easily find courses and materials. This could include implementing advanced filtering options and search algorithms to enhance discoverability.

- Internationalization: Add support for multiple languages. This would make the platform accessible to a broader audience and cater to users from different linguistic backgrounds.

Section 6 - Conclusion

6.1) Conclusion

This report has detailed the design, development, and implementation of an advanced E-learning platform. The system successfully integrates authentication, course management, feedback, and real-time communication, providing a robust and feature-rich learning environment. The application is designed to be secure, scalable, and user-friendly, ensuring that both students and teachers can effectively engage with the platform.

Future enhancements will further improve user engagement, functionality, and scalability, ensuring that the platform remains a valuable resource for online education. By continuously iterating on the design and incorporating user feedback, the application can evolve to meet the changing needs of learners and educators alike.

Section 7 – References

7.1) References

- Django Documentation: https://docs.djangoproject.com/
- Django REST Framework Docs: https://www.django-rest-framework.org/
- Django Channels for WebSockets: https://channels.readthedocs.io/
- Celery for Task Management: https://docs.celeryq.dev/en/stable/
- Redis Documentation: https://redis.io/docs/
- AWS Documentation: https://aws.amazon.com/documentation/
- Bootstrap Documentation: https://getbootstrap.com/docs/5.3/getting-started/introduction/

Section 8 – Full Code Implementation Scripts

8.1) Full code implementation scripts

```
#=============================================================================
Asgi.py
#=============================================================================
# import libraries and modules
import os
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "elearning.settings")

from django.core.asgi import get_asgi_application
from channels.auth import AuthMiddlewareStack
from channels.routing import ProtocolTypeRouter, URLRouter
from learnhub.routing import websocket_urlpatterns

# define application for ASGI file
application = ProtocolTypeRouter({
  "http": get_asgi_application(),
  "websocket": AuthMiddlewareStack(
    URLRouter(websocket_urlpatterns)
  ),
})

#=============================================================================
Routing.py
#=============================================================================
# import libraries and modules
import learnhub.routing
from channels.auth import AuthMiddlewareStack
from channels.routing import ProtocolTypeRouter, URLRouter

application = ProtocolTypeRouter({
  'websocket': AuthMiddlewareStack(
    URLRouter(
      learnhub.routing.websocket_urlpatterns
    )
  ),
})

#=============================================================================
Settings.py
#=============================================================================
# import libraries and modules
import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY = "django-insecure-*pd*=9&9*n8y5mj!qf&zi0wv5(@=e_k@(7!s=99+70te3h%()d"
DEBUG = True

ALLOWED_HOSTS = ['localhost', '127.0.0.1']
```

```
INSTALLED_APPS = [
    "learnhub",
    "rest_framework",
    "django_bootstrap5",
    "channels",
    "daphne",
    "django.contrib.admin",
    "django.contrib.auth",
    "django.contrib.contenttypes",
    "django.contrib.sessions",
    "django.contrib.messages",
    "django.contrib.staticfiles",
]

MIDDLEWARE = [
    "django.middleware.security.SecurityMiddleware",
    "django.contrib.sessions.middleware.SessionMiddleware",
    "django.middleware.common.CommonMiddleware",
    "django.middleware.csrf.CsrfViewMiddleware",
    "django.contrib.auth.middleware.AuthenticationMiddleware",
    "django.contrib.messages.middleware.MessageMiddleware",
    "django.middleware.clickjacking.XFrameOptionsMiddleware",
]

ROOT_URLCONF = "elearning.urls"

TEMPLATES = [
    {
        "BACKEND": "django.template.backends.django.DjangoTemplates",
        "DIRS": [],
        "APP_DIRS": True,
        "OPTIONS": {
            "context_processors": [
                "django.template.context_processors.debug",
                "django.template.context_processors.request",
                "django.contrib.auth.context_processors.auth",
                "django.contrib.messages.context_processors.messages",
            ],
        },
    },
]

WSGI_APPLICATION = "elearning.wsgi.application"
ASGI_APPLICATION = "elearning.asgi.application"

DATABASES = {
    "default": {
        "ENGINE": "django.db.backends.sqlite3",
        "NAME": os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
AUTH_PASSWORD_VALIDATORS = [
  {
    "NAME": "django.contrib.auth.password_validation.UserAttributeSimilarityValidator",
  },
  {
    "NAME": "django.contrib.auth.password_validation.MinimumLengthValidator",
  },
  {
    "NAME": "django.contrib.auth.password_validation.CommonPasswordValidator",
  },
  {
    "NAME": "django.contrib.auth.password_validation.NumericPasswordValidator",
  },
]

LANGUAGE_CODE = "en-us"
TIME_ZONE = "UTC"
USE_I18N = True
USE_L10N = True
USE_TZ = True
STATIC_URL = "/static/"
DEFAULT_AUTO_FIELD = "django.db.models.BigAutoField"

# set up for websockets
CHANNEL_LAYERS = {
  'default': {
    'BACKEND': 'channels_redis.core.RedisChannelLayer',
    'CONFIG': {
      "hosts": [('127.0.0.1', 6379)],
    },
  },
}

#=========================================================================
Urls.py
#=========================================================================
# import libraries and modules
from django.contrib import admin
from django.urls import include, path
from django.views.generic import RedirectView
from django.conf.urls.static import static
from django.conf import settings

# define url patterns
urlpatterns = [
  path("", RedirectView.as_view(url="/learnhub/login/", permanent=False)),
  path('', include('learnhub.urls')),
  path("admin/", admin.site.urls),
] + static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

```
#=============================================================================
Wsgi.py
#=============================================================================
# import libraries and modules
import os
from django.core.wsgi import get_wsgi_application

# set up os from elearning settings file
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "elearning.settings")

# define application using WSGI
application = get_wsgi_application()

#=============================================================================
Add_material.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Add Resource</title></head>
<h1 class="resourceTitle">Add resource for: {{ course.name }}</h1>
<form method="POST" enctype="multipart/form-data">
  {% csrf_token %}
  {% for field in form %}
    <div class="form-group">
      <label class="form-label">{{ field.label }}</label>
      {{ field }}
      {% if field.errors %}
        <span style="color: red;">{{ field.errors }}</span>
      {% endif %}
    </div>
  {% endfor %}
  <button type="submit">Upload</button>
</form>
<a href="{% url 'course_detail' course.id %}">Back to Course</a>
{% endblock %}

#=============================================================================
Admin_panel.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Admin Panel</title></head>
<div class="segment">
  <h1 class="pageTitle">Admin Panel</h1>
  {% if messages %}
    {% for message in messages %}
      <p>{{ message }}</p>
    {% endfor %}
  {% endif %}
  {% if students %}
    <table border="1">
```

```
        <thead>
          <tr>
            <th>ID</th>
            <th>Username</th>
            <th>Email</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {% for student in students %}
          <tr>
            <td>{{ student.id }}</td>
            <td>{{ student.username }}</td>
            <td>{{ student.email }}</td>
            <td>
              <!-- Redirects to confirm delete page -->
              <a href="{% url 'delete_student' student.id %}"
                 style="color: white; background-color: red; padding: 5px 10px; text-decoration:
none;">
                 Delete
              </a>
            </td>
          </tr>
          {% endfor %}
        </tbody>
      </table>
    {% else %}
      <p>No students found.</p>
    {% endif %}
</div>
{% endblock %}

#=========================================================================
Base.html
#=========================================================================
{% include "./header.html" %}
{% include "./title.html" %}
<div class="container-fluid">
  <div  class="col-8" class="content">
    {% block content %}Didn't render content{% endblock %}
  </div>
</div>

#=========================================================================
Browse_profiles.html
#=========================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Browse Profiles</title></head>
<h1>Browse Profiles</h1>
<p>Welcome to the Browse Profiles page!</p>
```

```
<!-- Search Form -->
<form method="GET" action="{% url 'browse_profiles' %}">
  <input type="text" name="q" placeholder="Search profiles..." value="{{ query }}">
  <button type="submit">Search</button>
</form>
<!-- Display Search Results -->
{% if users %}
  <ul>
    {% for user in users %}
    <li>
      <a class="enrolledList" href="{% url 'user_detail' user.pk %}">{{ user.username }}</a>
    </li>
    {% endfor %}
  </ul>
{% else %}
  <p>No profiles found.</p>
{% endif %}
{% endblock %}

#================================================================================
Chat.html
#================================================================================
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"/>
  <title>Chat Rooms</title>
  <h1 class="chatsPage">Chats page</h1>
</head>
<body>
  <style>
    body{
      background-color: black;
    }
    .chatTitle{
      color:white;
    }
    .chatList{
      color:white;
      text-decoration: none;
      list-style: none;
    }
    .chatsPage{
      color: white;
      text-align: center;
      padding: 50px;
    }
  </style>
    <h2 class="chatTitle">Enter a chat room:</h2>
    <input id="room-name-input" type="text" size="100"><br><br>
    <input id="room-name-submit" type="button" value="Enter"><br><br>
```

```
    <script>
      document.querySelector('#room-name-input').focus();
      document.querySelector('#room-name-input').onkeyup = function(e) {
        if (e.keyCode === 13) {  // Enter key
          document.querySelector('#room-name-submit').click();
        }
      };
      document.querySelector('#room-name-submit').onclick = function(e) {
        var roomName = document.querySelector('#room-name-input').value;
        window.location.href = "/chat/" + roomName + "/";
      };
    </script>
    <a href="{% url 'home' %}">Back to home</a><br><br>
    <h2 class="chatTitle">Past chats:</h2>
    <ul>
      {% for chat in chatMessage %}
        <li class="chatList">{{ chat.room_name }}</li>
      {% empty %}
        <p>No past chats</p>
      {% endfor %}
    </ul>
</body>
</html>

#=============================================================================
Confirm_delete.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Confirm Deletion</title></head>
<h1>Confirm Deletion</h1>
<p>Are you sure you want to delete <strong>{{ object.username }}</strong>?</p>
<form method="post">
  {% csrf_token %}
  <button type="submit" style="color: white; background-color: red; padding: 5px
10px;">Confirm Delete</button>
</form>
<a href="{% url 'admin_panel' %}" style="padding: 5px 10px; background-color: grey; color:
white; text-decoration: none;">Cancel</a>
{% endblock %}

#=============================================================================
Course_detail.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>{{ course.name }} - Course Details</title></head>
<h1 class="courseTitle">{{ course.name }}</h1>
<p class="courseDescription">{{ course.description }}</p>
<!-- Add Resource Button (Visible Only to Teachers) -->
```

```
{% if user == course.teacher %}
  <a class="addResourceLink" href="{% url 'add_material' course.id %}">Add Resource</a>
{% endif %}
<div class="courseContent">
  <h2>Course Resources:</h2>
  <ul class="resource-list">
    {% for material in course.materials.all %}
      <li class="resource-item">
        <div class="resource-content">
          <span class="resource-title"><strong>{{ material.title }}</strong></span>
          <p class="resource-description">{{ material.description }}</p>
        </div>
        <span class="resource-uploaded">{{ material.uploaded_at }}</span>
        <p class="resource-link"><a href="{{ material.file.url }}" download>Download</a></p>
      </li>
    {% empty %}
      <p>No resources added yet.</p>
    {% endfor %}
  </ul>
  <h2>Enrolled Students:</h2>
  {% if user.is_authenticated and user.learnhubuser.is_student %}
  <!-- Join Course Button -->
  <form action="{% url 'join_course' course.id %}" method="POST">
    {% csrf_token %}
    <button type="submit">Join Course</button>
  </form>
  {% else %}
  {% endif %}
  <ul>
    {% for student in course.students.all %}
      <li class="enrolledList">{{ student.username }}</li>
    {% empty %}
      <p class="enrolledList">No students enrolled in this course yet.</p>
    {% endfor %}
  </ul>
</div>
{% endblock %}

#================================================================================
Courses.html
#================================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Courses Page</title></head>
<div class="segment">
  <h1 class="pageTitle">Courses</h1>
  <p>Welcome to the Courses page!</p>
  <ul>
    {% for course in courses %}
      <li><a href="{% url 'course_detail' course.id %}">{{ course.name }}</a></li>
    {% empty %}
```

```
        <p>No courses available.</p>
    {% endfor %}
  </ul>
</div>
{% endblock %}

#=============================================================================
Create_course.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Create course page</title></head>
<div class="segment">
  <h1 class="pageTitle">Create Course</h1>
  <form method="POST">
    {% csrf_token %}
    {% if form %}
      {% for field in form %}
        <div>
          <label>{{ field.label }}</label><br>
          {{ field }}
          {% if field.errors %}
            <span style="color: red;">{{ field.errors }}</span>
          {% endif %}
        </div>
        <br>
      {% endfor %}
    {% else %}
      <p style="color: red;">Error: Form is not available!</p>
    {% endif %}
    <button type="submit">Create Course</button>
  </form>
</div>
{% endblock %}

#=============================================================================
Feedback.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Feedback Page</title></head>
<div class="segment">
  {% if messages %}
    {% for message in messages %}
      <p style="color: green;">{{ message }}</p>
    {% endfor %}
  {% endif %}
  {% if user.is_authenticated and user.learnhubuser.is_student %}
  <form method="post" class="pageTitle">
    {% csrf_token %}
    <strong><label for="course">Select Course:</label></strong><br>
```

```
      <select name="course" required>
        {% for course in courses %}
          <option value="{{ course.id }}">{{ course.name }}</option>
        {% endfor %}
      </select><br><br><br>
      <strong><label for="comment">Feedback:</label></strong><br>
      <textarea name="comment" rows="3" cols="50" required></textarea><br><br>
      <button type="submit">submit</button><br><br>
    </form>
    {% endif %}
    <h2>Course feedback</h2>
    <ul class="resource-list">
      {% for feedback in feedbacks %}
        <li class="resource-item">
          <div class="resource-content">
            <span class="resource-title"><strong>{{ feedback.course.name }}</strong></span>
            <p class="resource-description">{{ feedback.comment }}</p>
          </div>
          <span class="resource-uploaded">{{ feedback.created_at }}</span>
        </li>
      {% empty %}
        <p>No feedback added yet.</p>
      {% endfor %}
    </ul>
  </div>
  {% endblock %}

  #=============================================================================
  Header.html
  #=============================================================================
  {% load django_bootstrap5 %}
  <!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8">
    <style>
      /* general html tag styling */
      body{
        background-color:#121212;
        margin: 0;
        padding: 0;
        width: 100vw;
      }
      h1, h2, p, label, strong{
        color: #EAEAEA;
      }
      table{
        background-color: white;
        width: 100vw;
      }
      thead{
```

```
    background-color: black;
    color: white;
}
input, select, textarea {
    width: 60%;
    padding: 5px;
    background-color: white;
    flex: 1;
    padding: 8px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button{
    background-color: #8A2BE2;
    color: #EAEAEA;
}

/* custom styling */
.scroll-text {
    position: relative;
    transform: translateY(0px);
    transition: transform 0.2s ease-out;
}
.banner{
    width:100vw;
    height: auto;
}
.segment{
    padding-left: 50px;
}
.pageTitle{
    padding-top: 50px;
}
.enrolledList .chatList{
    color:white;
}
.courseTitle .courseDescription .courseContent{
    padding-left: 50px;
    padding-top: 50px;
}

/* home page introductory segment styling */
.introductory{
    background-color: black;
    width:100vw;
    height:600px;
}
.introductorytext{
    text-align: center;
    padding-top: 50px;
}
```

```
.header {
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px 20px;
}
.welcome {
  margin-left: auto;
}

/* navigation bar styling for unauthorized users*/
.navbar-nav {
  list-style: none;
  padding: 0;
  margin: 0;
  display: flex;
}
.nav-center {
  flex: 3;
  display: flex;
  justify-content:end;
}
.navbar-nav .nav-item {
  margin: 0 10px;
}
.auth-links {
  flex: 1;
}
.auth-links .navbar-nav {
  justify-content: flex-start;
}

/* navigation bar styling */
.navbar {
  background-color: #8A2BE2;
}
.navbar-nav .nav-link {
  color: #EAEAEA;
  padding: 10px 15px;
  text-decoration: none;
  transition: all 0.3s ease-in-out;
}
.navbar-nav .nav-link:hover {
  color: #00D1FF !important;
  text-decoration: none;
  box-shadow: 0px 0px 10px rgba(0, 209, 255, 0.5);
}
.navbar-nav .nav-item {
  margin-right: 10px;
}
.navbar-nav .nav-link.disabled {
```

```
    color: gray !important;
}

/* list of updates segment styling */
.update-item {
    display: flex;
    align-items: center;
    border: 1px solid #ddd;
    padding: 15px;
    margin-bottom: 10px;
    border-radius: 5px;
    background-color: black;
    width:500px;
}
.update-image {
    max-width: 150px;
    height: auto;
    margin-right: 20px;
    border-radius: 3px;
}
.update-text {
    flex: 1;
}
.update-text p {
    margin: 5px 0;
}

/* list of resources segment styling */
.resourceTitle{
    padding: 50px;
}
.resource-list {
    list-style: none;
    padding: 0;
}
.addResourceLink{
    padding-left: 50px;
}
.resource-item {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    align-items: flex-start;
    border: 2px solid black;
    padding: 10px;
    margin: 10px 0;
    background-color: black;
}
.resource-content {
    display: flex;
    flex-direction: column;
```

```
    }
    .resource-title {
      font-weight: bold;
      text-align: left;
    }
    .resource-description {
      font-size: 14px;
      color: white;
      margin: 5px 0;
    }
    .resource-uploaded {
      font-size: 12px;
      color: white;
      text-align: right;
    }
    .resource-link {
      padding-top: 5px;
    }
    .resource-link a {
      text-decoration: none;
      color: red;
      font-weight: bold;
    }
    .resource-link a:hover {
      text-decoration: underline;
    }

    /* list of notifications segment styling */
    .notification-list {
      list-style: none;
      padding: 0;
    }
    .notification-item {
      border: 2px solid white;
      padding: 10px;
      margin: 10px 0;
      background-color: #f9f9f9;
    }
    .notification-item p {
      margin: 0;
    }
    .notification-item a {
      text-decoration: none;
      color: white;
      font-weight: bold;
    }
    .notification-item a:hover {
      text-decoration: underline;
    }
    .notifications-container {
      display: flex;
```

```
    justify-content: space-between;
    gap: 20px;
    width: 2010px;
  }
  .notification-section {
    flex: 1;
    max-height: 400px;
    overflow-y: auto;
    background: #1a1a1a;
    color: white;
    padding: 15px;
    border-radius: 10px;
  }
  .notification-item {
    background-color: black;
    padding: 10px;
    border-radius: 5px;
    margin-bottom: 10px;
  }
  .notification-item button, .update-item button {
    background-color: #8A2BE2;
    color: white;
    border: none;
    padding: 5px 10px;
    border-radius: 5px;
    cursor: pointer;
  }

  /* login page styling */
  .login-page {
    background-image: url('../../../images/banner.png');
    background-size: cover;
    height: 100vh;
    width:100vw;;
    display: flex;
    align-items: center;
    justify-content: flex-end;
    padding-right: 5%;
  }
  .login-container {
    background: black;
    padding: 20px 40px;
    box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
    width: 600px;
    border-radius: 8px;
  }
  .form-group {
    display: flex;
    align-items: center;
    margin-bottom: 15px;
  }
```

```
.form-label {
  width: 100px;
  font-weight: bold;
  text-align: right;
  margin-right: 10px;
}
.input-field {
  flex: 1;
  padding: 8px;
  border: 1px solid #ccc;
  border-radius: 5px;
}
.submit-btn {
  width: 100%;
  padding: 10px;
  background-color: #8A2BE2;
  color: white;
  border: none;
  border-radius: 5px;
  font-size: 16px;
  cursor: pointer;
}

/* registration page styling */
.register-page {
  background-image: url('../../../images/banner.png');
  background-size: cover;
  height: 100vh;
  width: 100vw;
  display: flex;
  align-items: center;
  justify-content: flex-end;
  padding-right: 5%;
}
.register-container {
  background: black;
  padding: 20px 40px;
  border-radius: 8px;
  box-shadow: 0px 4px 10px rgba(0, 0, 0, 0.2);
  width: 600px;
}
.form-group {
  display: flex;
  align-items: center;
  margin-bottom: 15px;
}
.form-label {
  width: 120px;
  font-weight: bold;
  text-align: right;
  margin-right: 10px;
```

```
    }
    .submit-btn {
      width: 100%;
      padding: 10px;
      background-color: #8A2BE2 ;
      color: white;
      border: none;
      border-radius: 5px;
      font-size: 16px;
      cursor: pointer;
    }
    .form-group {
      display: flex;
      justify-content: space-between;
      align-items:flex-start;
      margin-bottom: 10px;
      width: 100%;
    }
    .form-label {
      font-weight: bold;
    }
  </style>
</head>
<body>

#================================================================================
Home.html
#================================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Home Page</title></head>
<script>
  window.addEventListener("scroll", function () {
    let scrollY = window.scrollY;
    document.querySelectorAll(".scroll-text").forEach((el) => {
      el.style.transform = `translateY(${scrollY * 0.2}px)`;
    });
  });
</script>
<img src="../../../images/banner.png" alt="Logo" class="banner">
<div class="introductory">
  {% if user.is_authenticated %}
    <h1 class="introductorytext scroll-text">Welcome {{ user.username }} !</h1>
  {% endif %}
  <p class="introductorytext scroll-text">
    Skill Hub, the ultimate e-learning hub for aspiring web developers. Whether you're a
complete beginner or a coding <br><br>
    enthusiast leveling up your skills, we've designed an interactive, hands-on learning
experience to take you from <br><br>
    theory to real-world application.Build. Innovate. Dominate. Dive into cutting-edge
frontend, backend, and full-stack <br><br>
```

```
    development courses, explore real-world projects, and sharpen your skills with expert
guidance—all in a sleek, futuristic <br><br>
    learning environment built for the next generation of developers. Ready to code your
future? Let's get started.
  </p>
</div>
<div class="notifications-container">
  <!-- Enrollment Notifications for Teachers -->
  {% if user.is_authenticated and user.learnhubuser.is_teacher %}
    <div class="notification-section">
      <h2>New Enrollments</h2>
      {% for notification in enrollment_notifications %}
        <div class="notification-item">
          <p><strong>{{ notification.student.username }}</strong> enrolled in <strong>{{
notification.course.name }}</strong></p>
          <form method="post" action="{% url 'mark_enrollment_as_read' notification.id %}">
            {% csrf_token %}
            <button type="submit">Mark as Read</button>
          </form>
        </div>
      {% empty %}
        <p>No new enrollments.</p>
      {% endfor %}
    </div>
  {% endif %}
  <!-- New Resources Notifications for Students -->
  {% if user.is_authenticated and user.learnhubuser.is_student %}
    <div class="notification-section">
      <h2>New Course Resources</h2>
      {% for notification in resource_notifications %}
        <div class="notification-item">
          <p>A new resource <strong>{{ notification.resource.title }}</strong> was added to
<strong>{{ notification.resource.course.name }}</strong>.</p>
          <p>Click <a href="{{ notification.resource.file.url }}" download>here</a> to
download</p>
          <form method="post" action="{% url 'mark_resource_as_read' notification.id %}">
            {% csrf_token %}
            <button type="submit">Mark as Read</button>
          </form>
        </div>
      {% empty %}
        <p>No new resources available.</p>
      {% endfor %}
    </div>
  {% endif %}
  <!-- Status Updates -->
  <div class="notification-section">
    <h2>Unread Updates</h2>
    {% for update in updates %}
      <div class="update-item">
        {% if update.image %}
```

```
            <img src="{{ update.image.url }}" alt="Status Image" class="update-image">
          {% endif %}
          <div class="update-text">
            <strong>Created by:</strong> <p>{{ update.user.username }}</p>
            <strong>Date and time:</strong> <p>{{ update.created_at|date:"F j, Y, g:i a" }}</p>
            <strong>Details:</strong> <p>{{ update.content }}</p>
          </div>
        </div>
      {% empty %}
        <p>No unread updates.</p>
      {% endfor %}
    </div>
</div>
<div class="segment">
  {% if user.is_authenticated %}
    <h2>Any new posts?</h2>
    <form method="post" enctype="multipart/form-data">
      {% csrf_token %}
      {{ form.as_p }}
      <button type="submit">Post</button>
    </form>
  {% else %}
    <p><a href="{% url 'login' %}">Log in</a> to post a status update.</p>
  {% endif %}
</div>
{% endblock %}

#================================================================================
Login.html
#================================================================================
{% extends "./base.html" %}
{% load django_bootstrap5 %}
{% block content %}
<div class="login-page">
  <div class="login-container">
    <h1>Login</h1>
    <form id="login_form" method="post" action="/learnhub/login/">
      {% csrf_token %}
      <div class="form-group">
        <label for="username" class="form-label">Username:</label>
        <input type="text" id="username" name="username" size="50" class="input-field" />
      </div>
      <div class="form-group">
        <label for="password" class="form-label">Password:</label>
        <input type="password" id="password" name="password" size="50" class="input-
field" />
      </div>
      <button type="submit" class="submit-btn">Submit</button>
    </form>
  </div>
</div>
```

```
{% endblock %}

#=============================================================================
Profile.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>{{ profile_user.user.username }}'s Profile</title></head>
<div class="segment">
  <h1 class="pageTitle">{{ profile_user.user.username }}'s Profile</h1>
  <!-- Profile Picture -->
  {% if profile_user.profile_picture %}
    <p><strong>Profile Picture:</strong></p>
    <img src="{{ profile_user.profile_picture.url }}" alt="Profile Picture" width="150">
  {% else %}
    <p>No profile picture available.</p>
  {% endif %}
  <!-- Basic User Information -->
  <p><strong>Username:</strong> {{ profile_user.user.username }}</p>
  <p><strong>Email:</strong> {{ profile_user.user.email }}</p>
  <p><strong>First Name:</strong> {{ profile_user.user.first_name }}</p>
  <p><strong>Last Name:</strong> {{ profile_user.user.last_name }}</p>
  <!-- Role -->
  <p><strong>Role:</strong>
    {% if profile_user.is_student %}
      Student
    {% elif profile_user.is_teacher %}
      Teacher
    {% else %}
      Not specified
    {% endif %}
  </p>
  <!-- Organisation -->
  {% if profile_user.organisation %}
    <p><strong>Organisation:</strong> {{ profile_user.organisation }}</p>
  {% endif %}
</div>
{% endblock %}

#=============================================================================
Register.html
#=============================================================================
{% extends "./base.html" %}
{% load django_bootstrap5 %}
{% block content %}
<div class="register-page">
  <div class="register-container">
    <h1>Register</h1>
    {% if registered %}
      <strong>Thank you for registering! Please <a href="{% url 'login' %}">log
in</a>.</strong>
```

```
    {% else %}
      <strong>Register here</strong><br /><br />
      <form id="user_form" method="post" action="{% url 'register' %}"
enctype="multipart/form-data">
        {% csrf_token %}
        {% for field in registration_form %}
        <div class="form-group">
          <label for="{{ field.id_for_label }}" class="form-label">{{ field.label }}:</label>
          {{ field }}
        </div>
        {% endfor %}
        <button type="submit" class="submit-btn">Register</button>
      </form>
    {% endif %}
  </div>
</div>
{% endblock %}

#================================================================================
Room.html
#================================================================================
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8"/>
  <title>Chat Room</title>
  <style>
    body {
      background-color: black;
      color: white;
      font-family: Arial, sans-serif;
    }
    .chat-container {
      display: flex;
      flex-direction: column;
      max-width: 800px;
      margin: auto;
      padding: 20px;
    }
    #chat-log {
      height: 400px;
      overflow-y: auto;
      border: 1px solid #ccc;
      padding: 10px;
      background-color: #222;
      border-radius: 5px;
    }
    .chat-message {
      padding: 8px;
      border-radius: 5px;
      margin-bottom: 5px;
```

```
    }
    .chat-input {
      display: flex;
      margin-top: 10px;
    }
    #chat-message-input {
      flex: 1;
      padding: 10px;
      border: none;
      border-radius: 5px;
      font-size: 16px;
    }
    #chat-message-submit {
      background-color: #4caf50;
      color: white;
      border: none;
      padding: 10px 20px;
      cursor: pointer;
      border-radius: 5px;
      font-size: 16px;
      margin-left: 5px;
    }
  </style>
</head>
<body>
  <div class="chat-container">
    <h2>Chat Room: {{ room_name }}</h2>
    <!-- Chat log (Scrollable) -->
    <div id="chat-log">
      {% for message in messages %}
        <div class="chat-message {% if message.sender == user.username %}user-
message{% else %}other-message{% endif %}">
          <strong>{{ message.sender }}:</strong> {{ message.message }}
        </div>
      {% endfor %}
    </div>
    <!-- Chat Input -->
    <div class="chat-input">
      <input id="chat-message-input" type="text" placeholder="Type your message...">
      <input id="chat-message-submit" type="button" value="Send">
    </div><br><br>
    <a href="{% url 'chat' %}" class="return">Return to chats</a>
  </div>
  {{ room_name|json_script:"room-name" }}
  <script>
    const username = "{{ user.username|escapejs }}";
    const roomName = JSON.parse(document.getElementById('room-name').textContent);
    const chatSocket = new WebSocket(
      (window.location.protocol === "https:" ? "wss://" : "ws://") + window.location.host +
"/ws/" + roomName + "/"
    );
```

```
    chatSocket.onmessage = function(e) {
      const data = JSON.parse(e.data);
      const chatLog = document.querySelector('#chat-log');
      const messageDiv = document.createElement('div');
      messageDiv.classList.add('chat-message');
      if (data.sender === username) {
        messageDiv.classList.add('user-message');
      } else {
        messageDiv.classList.add('other-message');
      }
      messageDiv.innerHTML = `<strong>${data.sender}:</strong> ${data.message}`;
      chatLog.appendChild(messageDiv);
      chatLog.scrollTop = chatLog.scrollHeight;
    };
    chatSocket.onclose = function(e) {
      console.error('Chat socket closed unexpectedly');
    };
    document.querySelector('#chat-message-input').focus();
    document.querySelector('#chat-message-input').onkeyup = function(e) {
      if (e.keyCode === 13) {
        document.querySelector('#chat-message-submit').click();
      }
    };
    document.querySelector('#chat-message-submit').onclick = function(e) {
      const messageInputDom = document.querySelector('#chat-message-input');
      const message = messageInputDom.value.trim();
      if (message !== "") {
        chatSocket.send(JSON.stringify({
          'message': message,
          'sender': username
        }));
        messageInputDom.value = '';
      }
    };
  </script>
</body>
</html>

#================================================================================
Title.html
#================================================================================
<nav class="navbar navbar-expand-md sticky-top">
  <div class="w-100 d-flex justify-content-between align-items-center">
    <div class="auth-links">
      {% if not user.is_authenticated %}
        <ul class="navbar-nav">
          <li class="nav-item active"><a class="nav-link" href="{% url 'register'
%}">Register</a></li>
          <li class="nav-item active"><a class="nav-link" href="{% url 'login'
%}">Login</a></li>
        </ul>
```

```
        {% endif %}
      </div>
      <div class="nav-center">
        <ul class="navbar-nav" id="navigation">
          {% if user.is_authenticated %}
            <li class="nav-item active"><a class="nav-link" href="{% url 'home'
%}">Home</a></li>
            {% if user.is_authenticated and user.learnhubuser.is_teacher %}
            <li class="nav-item active"><a class="nav-link" href="{% url 'admin_panel'
%}">Admin Panel</a></li>
            {% else %}
            {% endif %}
            <li class="nav-item active"><a class="nav-link" href="{% url 'chat'
%}">Chat</a></li>
            <li class="nav-item active"><a class="nav-link" href="{% url 'profile'
request.user.username %}">Profile</a></li>
            <li class="nav-item active">
              {% if courses.exists %}
              <a class="nav-link" href="{% url 'course_detail' courses.first.id %}">Course
Details</a>
              {% else %}
              <a class="nav-link" href="#">No Courses Available</a>
              {% endif %}
            </li>
            <li class="nav-item active"><a class="nav-link" href="{% url 'courses'
%}">Courses</a></li>
            {% if user.is_authenticated and user.learnhubuser.is_teacher %}
            <li class="nav-item active">
              <a class="nav-link" href="{% url 'create_course' %}">Create Course</a>
            </li>
            {% else %}
            {% endif %}
            <li class="nav-item active">
              <a class="nav-link" href="{% url 'feedback' %}">Feedback</a>
            </li>
            {% if user.is_authenticated and user.learnhubuser.is_teacher %}
            <li class="nav-item active">
              <a class="nav-link" href="{% url 'browse_profiles' %}">Browse Profiles</a>
            </li>
            {% else %}
            {% endif %}
            <li class="nav-item active"><a class="nav-link" href="{% url 'updates'
%}">Updates</a></li>
            <li class="nav-item active"><a class="nav-link" href="{% url 'logout'
%}">Logout</a></li>
          {% endif %}
        </ul>
      </div>
    </div>
</nav>
```

```
#=============================================================================
Updates.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>Updates</title></head>
<div class="segment">
  <h1 class="pageTitle">All Updates</h1>
  {% for update in updates %}
    <div class="update-item">
      {% if update.image %}
        <img src="{{ update.image.url }}" alt="Status Image" class="update-image">
      {% endif %}
      <div class="update-text">
        <strong>Created by:</strong> <p>{{ update.user.username }}</p>
        <strong>Date and time:</strong> <p>{{ update.created_at|date:"F j, Y, g:i a" }}</p>
        <strong>Details:</strong> <p>{{ update.content }}</p>
      </div>
      {% if user.is_authenticated %}
        {% if not update.read %}
        <form method="post" action="{% url 'mark_as_read' update.id %}">
          {% csrf_token %}
          <button type="submit">Mark as Read</button>
        </form>
        {% else %}
        {% endif %}
      {% endif %}
    </div>
  {% empty %}
    <p>No updates yet.</p>
  {% endfor %}
</div>
{% endblock %}

#=============================================================================
User_detail.html
#=============================================================================
{% extends "./base.html" %}
{% block content %}
<head><title>{{ user_detail.username }} profile page</title></head>
<div class="segment">
  <h1 class="pageTitle">{{ user_detail.username }}'s profile</h1>
  {% if learnhub_user.profile_picture %}
  <p><strong>Profile Picture:</strong></p>
    <img src="{{ learnhub_user.profile_picture.url }}" alt="Profile Picture" width="150">
  {% else %}
  <p>No profile picture available.</p>
  {% endif %}
  <p><strong>Username:</strong> {{ user_detail.username }}</p>
  <p><strong>Email:</strong> {{ user_detail.email }}</p>
  <p><strong>First Name:</strong> {{ user_detail.first_name }}</p>
```

```
  <p><strong>Last Name:</strong> {{ user_detail.last_name }}</p>
  {% if learnhub_user %}
    <p><strong>Organisation:</strong> {{ learnhub_user.organisation }}</p>
    <p><strong>Role:</strong>
      {% if learnhub_user.is_student %} Student {% endif %}
      {% if learnhub_user.is_teacher %} Teacher {% endif %}
    </p>
  {% else %}
    <p>This user does not have a LearnHub profile.</p>
  {% endif %}
</div>
{% endblock %}

#=============================================================================
Api.py
#=============================================================================
# import libraries and modules
from rest_framework import generics, permissions, status
from rest_framework.response import Response
from rest_framework.views import APIView
from django.shortcuts import get_object_or_404
from django.contrib.auth.models import User
from .models import Course, Material, Feedback, ChatMessage
from .serializers import (
  UserSerializer,
  CourseSerializer,
  CourseDetailSerializer,
  FeedbackSerializer,
  MaterialSerializer,
  ChatMessageSerializer,
)

# define class for retrieving and sending information
class CourseListViewAPI(generics.ListCreateAPIView):
  queryset = Course.objects.all()
  serializer_class = CourseSerializer
  permission_classes = [permissions.IsAuthenticated]

  def perform_create(self, serializer):
    if not self.request.user.learnhubuser.is_teacher:
      return Response({"error": "Only teachers can create courses."},
status=status.HTTP_403_FORBIDDEN)
    serializer.save(teacher=self.request.user)

class CourseDetailViewAPI(generics.RetrieveUpdateDestroyAPIView):
  queryset = Course.objects.all()
  serializer_class = CourseDetailSerializer
  permission_classes = [permissions.IsAuthenticated]

  def update(self, request, *args, **kwargs):
    course = self.get_object()
```

```
        if request.user != course.teacher:
            return Response({"error": "Only the course teacher can update this course."},
status=status.HTTP_403_FORBIDDEN)
        return super().update(request, *args, **kwargs)

    def destroy(self, request, *args, **kwargs):
        course = self.get_object()
        if request.user != course.teacher:
            return Response({"error": "Only the course teacher can delete this course."},
status=status.HTTP_403_FORBIDDEN)
        return super().destroy(request, *args, **kwargs)

class JoinCourseViewAPI(APIView):
    permission_classes = [permissions.IsAuthenticated]

    def post(self, request, course_id):
        course = get_object_or_404(Course, id=course_id)
        if request.user in course.students.all():
            return Response({"message": "Already enrolled in this course."},
status=status.HTTP_400_BAD_REQUEST)
        course.students.add(request.user)
        return Response({"message": "Successfully joined the course."},
status=status.HTTP_200_OK)

class AddMaterialViewAPI(generics.CreateAPIView):
    queryset = Material.objects.all()
    serializer_class = MaterialSerializer
    permission_classes = [permissions.IsAuthenticated]

    def perform_create(self, serializer):
        course_id = self.request.data.get("course")
        course = get_object_or_404(Course, id=course_id)

        if self.request.user != course.teacher:
            return Response({"error": "Only the course teacher can add materials."},
status=status.HTTP_403_FORBIDDEN)

        serializer.save(course=course)

class UserListViewAPI(generics.ListAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
    permission_classes = [permissions.IsAuthenticated]

class UserDetailViewAPI(generics.RetrieveAPIView):
    queryset = User.objects.all()
    serializer_class = UserSerializer
    permission_classes = [permissions.IsAuthenticated]

class FeedbackViewAPI(generics.ListCreateAPIView):
    queryset = Feedback.objects.all()
```

```
  serializer_class = FeedbackSerializer
  permission_classes = [permissions.IsAuthenticated]

  def perform_create(self, serializer):
    if not self.request.user.learnhubuser.is_student:
      return Response({"error": "Only students can submit feedback."},
status=status.HTTP_403_FORBIDDEN)
    serializer.save(student=self.request.user)

class ChatMessageListViewAPI(generics.ListCreateAPIView):
  queryset = ChatMessage.objects.all()
  serializer_class = ChatMessageSerializer
  permission_classes = [permissions.IsAuthenticated]

  def get_queryset(self):
    room_name = self.kwargs["room_name"]
    return ChatMessage.objects.filter(room_name=room_name).order_by("timestamp")

  def perform_create(self, serializer):
    serializer.save(sender=self.request.user)

#=============================================================================
Apps.py
#=============================================================================
# import libraries and modules
from django.apps import AppConfig

# define LearnhubConfig class
class LearnhubConfig(AppConfig):
  default_auto_field = "django.db.models.BigAutoField"
  name = "learnhub"

#=============================================================================
Consumers.py
#=============================================================================
# import libraries and modules
import json
from channels.generic.websocket import AsyncWebsocketConsumer
from .models import ChatMessage
from asgiref.sync import sync_to_async

# define ChatConsumer class
class ChatConsumer(AsyncWebsocketConsumer):
  async def connect(self):
    self.room_name = self.scope['url_route']['kwargs']['room_name']
    self.room_group_name = f"chat_{self.room_name}"
    await self.channel_layer.group_add(
      self.room_group_name,
      self.channel_name
    )
    await self.accept()
```

```
        history = await self.get_chat_history(self.room_name)
        for message in history:
            await self.send(text_data=json.dumps({
                'sender': message['sender'],
                'message': message['message']
            }))

    @sync_to_async
    def get_chat_history(self, room_name):
        return list(ChatMessage.objects.filter(room_name=room_name).values('sender',
'message').order_by('timestamp')[:50])  # Limit to 50 messages

    async def disconnect(self, close_code):
        # Leave the room group
        await self.channel_layer.group_discard(
            self.room_group_name,
            self.channel_name
        )

    async def receive(self, text_data):
        text_data_json = json.loads(text_data)
        message = text_data_json.get('message')
        sender = text_data_json.get('sender', 'Anonymous')
        await self.save_message(self.room_name, sender, message)
        await self.channel_layer.group_send(
            self.room_group_name,
            {
                'type': 'chat_message',
                'message': message,
                'sender': sender,
            }
        )

    async def chat_message(self, event):
        message = event['message']
        sender = event['sender']
        await self.send(text_data=json.dumps({
            'type': 'chat_message',
            'sender': sender,
            'message': message
        }))

    @sync_to_async
    def save_message(self, room_name, sender, message):
        ChatMessage.objects.create(room_name=room_name, sender=sender,
message=message)

#======================================================================
Forms.py
#======================================================================
# import libraries and modules
```

```
from django import forms
from django.core.exceptions import ValidationError
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from .models import LearnHubUser, Course, Material
from django import forms
from django.forms import ModelForm
from .models import *
from django.contrib.auth.models import User
from django.contrib.auth.forms import UserCreationForm
from django import forms
from django.contrib.auth.forms import UserCreationForm
from django.contrib.auth.models import User
from .models import LearnHubUser
from .models import Course

# define validators functions to ensure data integrity and reliability
def validate_username(value):
  if not value.isalnum():
    raise ValidationError("Username should only contain letters and numbers.")

def validate_password_strength(value):
  if len(value) < 8:
    raise ValidationError("Password must be at least 8 characters long.")
  if not any(char.isdigit() for char in value):
    raise ValidationError("Password must contain at least one digit.")
  if not any(char.isalpha() for char in value):
    raise ValidationError("Password must contain at least one letter.")

# define form class for the different pages
class LoginForm(forms.Form):
  username = forms.CharField(
    max_length=150,
    required=True,
    validators=[validate_username],
    widget=forms.TextInput(attrs={"placeholder": "Username"})
  )
  password = forms.CharField(
    widget=forms.PasswordInput(attrs={"placeholder": "Password"}),
    required=True
  )

class RegistrationForm(UserCreationForm):
  email = forms.EmailField(required=True)
  first_name = forms.CharField(max_length=150, required=True)
  last_name = forms.CharField(max_length=150, required=True)
  is_student = forms.BooleanField(required=False)
  is_teacher = forms.BooleanField(required=False)
  profile_picture = forms.ImageField(required=False)
  organisation = forms.CharField(max_length=256, required=False)
```

```
    username = forms.CharField(
        max_length=150,
        required=True,
        validators=[validate_username]
    )
    password1 = forms.CharField(
        widget=forms.PasswordInput(),
        validators=[validate_password_strength]
    )
    password2 = forms.CharField(
        widget=forms.PasswordInput(),
        validators=[validate_password_strength]
    )

    class Meta:
        model = User
        fields = ("username", "email", "first_name", "last_name", "password1", "password2")

    def clean_email(self):
        """ Ensure the email is unique. """
        email = self.cleaned_data.get("email")
        if User.objects.filter(email=email).exists():
            raise ValidationError("Email is already in use.")
        return email

    def save(self, commit=True):
        user = super().save(commit=False)
        user.email = self.cleaned_data["email"]

        if commit:
            user.save()
            LearnHubUser.objects.create(
                user=user,
                is_student=self.cleaned_data.get("is_student", False),
                is_teacher=self.cleaned_data.get("is_teacher", False),
                profile_picture=self.cleaned_data.get("profile_picture", None),
                organisation=self.cleaned_data.get("organisation", "")
            )
        return user

class CourseForm(forms.ModelForm):
    name = forms.CharField(
        max_length=255,
        required=True,
        error_messages={"required": "Course name is required."}
    )
    description = forms.CharField(
        widget=forms.Textarea,
        min_length=10,
        error_messages={"min_length": "Description should be at least 10 characters long."}
    )
```

```
    class Meta:
        model = Course
        fields = ["name", "description"]

ALLOWED_FILE_TYPES = ["pdf", "docx", "pptx", "txt", "jpg", "png", "mp4", "zip"]

def validate_file_extension(value):
    """ Ensure uploaded file is of an allowed type. """
    ext = value.name.split(".")[-1].lower()
    if ext not in ALLOWED_FILE_TYPES:
        raise ValidationError(f"Invalid file type. Allowed formats: {', '.join(ALLOWED_FILE_TYPES)}")

class MaterialForm(forms.ModelForm):
    title = forms.CharField(max_length=255, required=True)
    description = forms.CharField(widget=forms.Textarea, required=False)
    file = forms.FileField(validators=[validate_file_extension])

    class Meta:
        model = Material
        fields = ["title", "description", "file"]

class StatusUpdateForm(forms.ModelForm):
    class Meta:
        model = StatusUpdate
        fields = ['content', 'image']

    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.fields["image"].required = False

#======================================================================
Models.py
#======================================================================
# import libraries and modules
from django.db import models
from django.contrib.auth.models import Group, Permission
from django.contrib.auth.models import User

# define classes for the different models to store data for storage
class LearnHubUser(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    is_student = models.BooleanField(default=False)
    is_teacher = models.BooleanField(default=False)
    profile_picture = models.ImageField(upload_to='profiles/', blank=True, null=True)
    groups = models.ManyToManyField(Group, related_name="learnhub_users")
    user_permissions = models.ManyToManyField(Permission,
related_name="learnhub_user_permissions")
    organisation = models.CharField(max_length=256, null=True, blank=True)

    def __unicode__(self):
```

```
        return self.user.username

class Course(models.Model):
    name = models.CharField(max_length=255)
    description = models.TextField()
    teacher = models.ForeignKey(User, on_delete=models.CASCADE, related_name='courses')
    students = models.ManyToManyField(User, related_name='enrolled_courses', blank=True)
    created_at = models.DateTimeField(auto_now_add=True)

class Material(models.Model):
    course = models.ForeignKey(Course, on_delete=models.CASCADE,
related_name='materials')
    title = models.CharField(max_length=255)
    description = models.TextField(blank=True)
    file = models.FileField(upload_to='materials/')
    uploaded_at = models.DateTimeField(auto_now_add=True)

class Feedback(models.Model):
    course = models.ForeignKey(Course, on_delete=models.CASCADE,
related_name='feedbacks')
    student = models.ForeignKey(User, on_delete=models.CASCADE)
    comment = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)

class ChatRoom(models.Model):
    name = models.CharField(max_length=255, unique=True)

    def __str__(self):
        return self.name

class Message(models.Model):
    room = models.ForeignKey(ChatRoom, on_delete=models.CASCADE)
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.user.username}: {self.content[:50]}..."

class StatusUpdate(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    content = models.TextField()
    image = models.ImageField(upload_to="status_images/", blank=True, null=True)
    created_at = models.DateTimeField(auto_now_add=True)
    read = models.BooleanField(default=False)  # New field for read status

    def __str__(self):
        return f"{self.user.username}: {self.content[:50]}"

class Alert(models.Model):
```

```
    teacher = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='notifications')
    student = models.ForeignKey(User, on_delete=models.CASCADE,
related_name='student_notifications')
    course = models.ForeignKey('Course', on_delete=models.CASCADE)
    message = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)
    read = models.BooleanField(default=False)

    def __str__(self):
        return f"{self.student.username} enrolled in {self.course.name}"

class ChatMessage(models.Model):
    room_name = models.CharField(max_length=255)
    sender = models.CharField(max_length=255)
    message = models.TextField()
    timestamp = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return f"{self.sender} in {self.room_name}: {self.message[:50]}"

class EnrollmentNotification(models.Model):
    course = models.ForeignKey(Course, on_delete=models.CASCADE)
    student = models.ForeignKey(User, on_delete=models.CASCADE)
    teacher = models.ForeignKey(User, on_delete=models.CASCADE,
related_name="enrollment_notifications")
    created_at = models.DateTimeField(auto_now_add=True)
    read = models.BooleanField(default=False)

    def __str__(self):
        return f"{self.student.username} enrolled in {self.course.name}"

class ResourceNotification(models.Model):
    resource = models.ForeignKey(Material, on_delete=models.CASCADE)
    student = models.ForeignKey(User, on_delete=models.CASCADE)
    created_at = models.DateTimeField(auto_now_add=True)
    read = models.BooleanField(default=False)

    def __str__(self):
        return f"New resource '{self.resource.title}' for {self.student.username}"

#=======================================================================
Routing.py
#=======================================================================
# import libraries and modules
from django.urls import re_path
from . import consumers

# set up websocket url patterns
websocket_urlpatterns = [
    re_path(r"^ws/(?P<room_name>\w+)/$", consumers.ChatConsumer.as_asgi()),
```

```
]

#=========================================================================
Serializers.py
#=========================================================================
# import libraries and modules
from rest_framework import serializers
from django.contrib.auth.models import User
from .models import Course, Material, Feedback, ChatMessage

# define new serializer classes
class UserSerializer(serializers.ModelSerializer):

  class Meta:
    model = User
    fields = ["id", "username", "email", "first_name", "last_name"]

class CourseSerializer(serializers.ModelSerializer):

  teacher = UserSerializer(read_only=True)

  class Meta:
    model = Course
    fields = ["id", "name", "description", "teacher", "students"]
    extra_kwargs = {"students": {"read_only": True}}

class CourseDetailSerializer(serializers.ModelSerializer):

  teacher = UserSerializer(read_only=True)
  students = UserSerializer(many=True, read_only=True)
  materials = serializers.SerializerMethodField()

  class Meta:
    model = Course
    fields = ["id", "name", "description", "teacher", "students", "materials"]

  def get_materials(self, obj):
    return MaterialSerializer(obj.materials.all(), many=True).data

class MaterialSerializer(serializers.ModelSerializer):

  course = serializers.PrimaryKeyRelatedField(queryset=Course.objects.all())

  class Meta:
    model = Material
    fields = ["id", "course", "title", "description", "file", "uploaded_at"]

class FeedbackSerializer(serializers.ModelSerializer):

  student = UserSerializer(read_only=True)
  course = serializers.PrimaryKeyRelatedField(queryset=Course.objects.all())
```

```
    class Meta:
        model = Feedback
        fields = ["id", "student", "course", "comment", "created_at"]

class ChatMessageSerializer(serializers.ModelSerializer):

    sender = UserSerializer(read_only=True)

    class Meta:
        model = ChatMessage
        fields = ["id", "room_name", "sender", "message", "timestamp"]

#=========================================================================
Tests.py
#=========================================================================
# import libraries and modules
from django.test import TestCase
from rest_framework.test import APIClient
from learnhub.forms import RegistrationForm
from learnhub.models import Course
from django.urls import reverse

# define class to write unit tests
class LearnHubTests(TestCase):

    def test_login(self):
        """Set up test users, courses, and API client."""
        self.client = APIClient()
        self.client.login(username="admin2", password="p@55w0rd")

    def test_failed_login(self):
        """Set up test users, courses, and API client."""
        self.client.login(username=" ")

    def test_valid_registration(self):
            """Test valid form submission."""
            form_data = {
                "username": "newuser",
                "email": "new@example.com",
                "first_name": "John",
                "last_name": "Doe",
                "password1": "StrongPass123",
                "password2": "StrongPass123",
                "is_student": True,
                "organisation": "University"
            }
            form = RegistrationForm(data=form_data)
            self.assertTrue(form.is_valid())

    def test_failed_registration(self):
```

```
        """Test valid form submission."""
        form_data = {
            "username": "newuser",
            "email": "new@example.com",
            "first_name": "John",
            "last_name": "Doe",
            "password1": "password",
            "password2": "password",
            "is_student": True,
            "organisation": "University"
        }
        form = RegistrationForm(data=form_data)
        self.assertTrue(form.is_valid())

    def test_user_logout(self):
        """Test user logout."""
        self.client.logout()
        response = self.client.get(reverse('home'))
        self.assertEqual(response.status_code, 302)  # Redirect to login

    def test_failed_create_course(self):
        """Ensure only teachers can create courses."""
        self.client.login(username="Jane", password="p@55w0rd")
        response = self.client.post(reverse('create_course'), {"name": " ", "description": " "})
        self.assertEqual(response.status_code, 201)
        self.assertEqual(Course.objects.count(), 2)

    def test_non_admin_cannot_access_admin_panel(self):
        """Ensure a normal user is redirected when trying to access the admin panel."""
        self.client.login(username="Jane", password="p@55w0rd")
        response = self.client.get(reverse("admin_panel"))
        self.assertEqual(response.status_code, 302)

#========================================================================
Urls.py
#========================================================================
# import libraries and modules
from django.urls import path
from .views import AdminPanelView, ChatView, ProfileView, CourseDetailView,
FeedbackView, BrowseProfilesView, UserDetailView
from .views import CourseCreateView, CourseListView, MarkEnrollmentAsReadView,
MarkResourceAsReadView, DeleteStudentView
from .views import AdminPanelView, HomePageView, UpdateListView, AddMaterialView,
MarkAsReadView
from .views import register, user_login, user_logout, join_course
from django.contrib.auth.decorators import login_required
from . import views
from .api import (
    CourseListViewAPI,
    CourseDetailViewAPI,
    AddMaterialViewAPI,
```

```
    UserListViewAPI,
    UserDetailViewAPI,
    FeedbackViewAPI,
    ChatMessageListViewAPI
)

urlpatterns = [
    # urls for registration, login and logout
    path('learnhub/register/', register, name='register'),
    path('learnhub/login/', user_login, name='login'),
    path('learnhub/logout/', user_logout, name='logout'),

    # urls for home page
    path('learnhub/home/', login_required(login_url='login')(HomePageView.as_view()),
name='home'),

    # urls for live chat rooms
    path('learnhub/chat/', login_required(login_url='login')(ChatView.as_view()), name='chat'),
    path('chat/<str:room_name>/', views.room, name='room'),

    # urls for profile related pages
    path('learnhub/profile/<str:username>/', ProfileView.as_view(), name='profile'),
    path('learnhub/user/<int:pk>/', UserDetailView.as_view(), name='user_detail'),
    path('learnhub/browse_profiles/',
login_required(login_url='login')(BrowseProfilesView.as_view()), name='browse_profiles'),

    # urls for course related pages
    path('courses/<int:course_id>/join/', join_course, name='join_course'),
    path('learnhub/courses/<int:pk>/',
login_required(login_url='login')(CourseDetailView.as_view()), name='course_detail'),
    path('learnhub/courses/', login_required(login_url='login')(CourseListView.as_view()),
name='courses'),
    path('learnhub/create_course/',
login_required(login_url='login')(CourseCreateView.as_view()), name='create_course'),

    # urls for feedback page
    path('learnhub/feedback/', login_required(login_url='login')(FeedbackView.as_view()),
name='feedback'),

    # urls for user account management
    path('admin_panel/', login_required(login_url='login')(AdminPanelView.as_view()),
name='admin_panel'),
    path('admin_panel/delete/<int:pk>/',
login_required(login_url='login')(DeleteStudentView.as_view()), name='delete_student'),

    # urls for update related pages
    path('updates/', UpdateListView.as_view(), name='updates'),
    path("update/read/<int:update_id>/", MarkAsReadView.as_view(), name="mark_as_read"),

    # urls for enrolment related pages
```

```
    path('learnhub/courses/<int:course_id>/add_material/', AddMaterialView.as_view(),
name='add_material'),

    # urls for marking successful enrolment into a course
    path("enrollment/read/<int:notification_id>/", MarkEnrollmentAsReadView.as_view(),
name="mark_enrollment_as_read"),

    # urls for marking newly uploaded resources as read
    path("resource/read/<int:notification_id>/", MarkResourceAsReadView.as_view(),
name="mark_resource_as_read"),

    # API endpoints for Courses
    path('api/courses/', CourseListViewAPI.as_view(), name='api_courses'),
    path('api/courses/<int:pk>/', CourseDetailViewAPI.as_view(), name='api_course_detail'),

    # API endpoints for Users
    path('api/users/', UserListViewAPI.as_view(), name='api_users'),
    path('api/users/<int:pk>/', UserDetailViewAPI.as_view(), name='api_user_detail'),

    # API endpoints for Feedback
    path('api/learnhub/feedback/', FeedbackViewAPI.as_view(), name='api_feedback'),

    #e API endpoints for AddMaterial
    path('api/learnhub/courses/<int:course_id>/add_material/', AddMaterialViewAPI.as_view(),
name='api_add_material'),

    # API endpoints for Chat message list
    path('api/learnhub/chat/<str:room_name>/', ChatMessageListViewAPI.as_view(),
name='api_chat_message_list'),
]

#=========================================================================
Views.py
#=========================================================================
# import libraries and modules
from django.contrib.auth import login, logout, authenticate
from django.contrib.auth.decorators import login_required
from django.contrib import messages
from django.views.generic import ListView, DeleteView, FormView, TemplateView, DetailView
from django.views import View
from django.urls import reverse, reverse_lazy
from django.shortcuts import get_object_or_404, render, redirect
from django.http import HttpResponse, HttpResponseRedirect
from .forms import RegistrationForm, MaterialForm, StatusUpdateForm, CourseForm
from .models import LearnHubUser, StatusUpdate, ChatMessage, Alert, Course,
EnrollmentNotification, ResourceNotification, User, Course, Feedback

# user authentication views
def user_login(request):
    if request.method == 'POST':
        username = request.POST.get('username')
```

```
      password = request.POST.get('password')
      user = authenticate(username=username, password=password)
      if user:
        if user.is_active:
          login(request, user)
          return HttpResponseRedirect('/learnhub/home/')
        else:
          return HttpResponse("Your account is disabled.")
      else:
        return HttpResponse("Invalid login details supplied.")
    return render(request, 'learnhub/login.html')

def register(request):
    registered = False
    if request.method == 'POST':
      user_form = RegistrationForm(request.POST, request.FILES)
      if user_form.is_valid():
        user_form.save()
        registered = True
        return redirect('login')
      else:
        print(user_form.errors)
    else:
      user_form = RegistrationForm()
    return render(request, 'learnhub/register.html', {
      'registration_form': user_form,
      'registered': registered
    })

@login_required
def user_logout(request):
    logout(request)
    return HttpResponseRedirect('/learnhub/login/')

# home page view
class HomePageView(FormView):
    template_name = "learnhub/home.html"
    form_class = StatusUpdateForm

    def form_valid(self, form):
      if self.request.user.is_authenticated:
        status = form.save(commit=False)
        status.user = self.request.user
        status.save()
        messages.success(self.request, "Your status update has been posted successfully!")
        return redirect("home")
      else:
        return redirect("login")

    def get_context_data(self, **kwargs):
      context = super().get_context_data(**kwargs)
```

```
        context["updates"] = StatusUpdate.objects.filter(read=False)

        if self.request.user.is_authenticated and self.request.user.learnhubuser.is_teacher:
            context["enrollment_notifications"] = EnrollmentNotification.objects.filter(
                teacher=self.request.user, read=False
            )
        elif self.request.user.learnhubuser.is_student:
            context["resource_notifications"] = ResourceNotification.objects.filter(
                student=self.request.user, read=False
            )
        else:
            context["enrollment_notifications"] = []
            context["resource_notifications"] = []
        return context

# live chat views
class ChatView(ListView):
    model = ChatMessage
    template_name = "learnhub/chat.html"
    context_object_name = 'chatMessage'

def room(request, room_name):
    messages =
ChatMessage.objects.filter(room_name=room_name).order_by('timestamp')[:50]
    return render(request, 'learnhub/room.html', {'room_name': room_name, 'messages':
messages})

# profile related views
class ProfileView(TemplateView):
    template_name = "learnhub/profile.html"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        username = self.kwargs.get("username")
        context["profile_user"] = get_object_or_404(LearnHubUser, user__username=username)
        return context

class BrowseProfilesView(TemplateView):
    template_name = "learnhub/browse_profiles.html"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        query = self.request.GET.get('q', '')
        if query:
            users = User.objects.filter(username__icontains=query)
        else:
            users = User.objects.all()
        context['users'] = users
        context['query'] = query
        return context
```

```
class UserDetailView(DetailView):
    model = User
    template_name = "learnhub/user_detail.html"
    context_object_name = "user_detail"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        try:
            context["learnhub_user"] = self.object.learnhubuser
        except LearnHubUser.DoesNotExist:
            context["learnhub_user"] = None
        return context

# course related views
class CourseCreateView(TemplateView):
    template_name = "learnhub/create_course.html"

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context["form"] = CourseForm()
        return context

    def post(self, request, *args, **kwargs):
        form = CourseForm(request.POST)
        if form.is_valid():
            course = form.save(commit=False)
            course.teacher = request.user
            course.save()
            print("Course created successfully!")
            return redirect("courses")

        print("Error: Unable to create course!")
        return self.render_to_response({"form": form})

class CourseListView(ListView):
    model = Course
    template_name = 'learnhub/courses.html'
    context_object_name = 'courses'

class CourseDetailView(DetailView):
    model = Course
    template_name = 'learnhub/course_detail.html'
    context_object_name = 'course'

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context["courses"] = Course.objects.all()
        return context

def join_course(request, course_id):
    course = get_object_or_404(Course, id=course_id)
```

```
    if request.user in course.students.all():
        messages.info(request, "You are already enrolled in this course.")
    else:
        course.students.add(request.user)
        messages.success(request, "Successfully joined the course!")
        EnrollmentNotification.objects.create(
            course=course,
            student=request.user,
            teacher=course.teacher
        )
    return redirect('course_detail', pk=course_id)

class AddMaterialView(View):
    template_name = "learnhub/add_material.html"

    def dispatch(self, request, *args, **kwargs):
        self.course = get_object_or_404(Course, id=self.kwargs["course_id"])
        if request.user != self.course.teacher:
            messages.error(request, "You are not authorized to add materials to this course.")
            return redirect("course_detail", course_id=self.course.id)

        return super().dispatch(request, *args, **kwargs)

    def get(self, request, *args, **kwargs):
        form = MaterialForm()
        return render(request, self.template_name, {"form": form, "course": self.course})

    def post(self, request, *args, **kwargs):
        form = MaterialForm(request.POST, request.FILES)
        if form.is_valid():
            material = form.save(commit=False)
            material.course = self.course
            material.save()

            for student in self.course.students.all():
                ResourceNotification.objects.create(resource=material, student=student)

            messages.success(request, "Resource added successfully!")
            return redirect(reverse("course_detail", kwargs={"pk": self.course.id}))
        return render(request, self.template_name, {"form": form, "course": self.course})

# user management views
class AdminPanelView(ListView):
    model = User
    template_name = "learnhub/admin_panel.html"
    context_object_name = "students"

    def get_queryset(self):
        return User.objects.filter(learnhubuser__is_student=True)
```

```
class DeleteStudentView(DeleteView):
    model = User
    template_name = "learnhub/confirm_delete.html"
    success_url = reverse_lazy("admin_panel")

    def delete(self, request, *args, **kwargs):
        messages.success(self.request, "Student deleted successfully.")
        return super().delete(request, *args, **kwargs)

# mark as read views to update list of notifications
class MarkAsReadView(View):
    def post(self, request, update_id):
        update = get_object_or_404(StatusUpdate, id=update_id)
        update.read = True
        update.save()
        messages.success(request, "Update marked as read!")
        return HttpResponseRedirect(reverse("updates"))

class MarkEnrollmentAsReadView(View):
    def post(self, request, notification_id):
        notification = get_object_or_404(EnrollmentNotification, id=notification_id)
        if request.user == notification.teacher:
            notification.read = True
            notification.save()
        return HttpResponseRedirect(reverse("home"))

class MarkResourceAsReadView(View):
    def post(self, request, notification_id):
        notification = get_object_or_404(ResourceNotification, id=notification_id)
        if request.user == notification.student:
            notification.read = True
            notification.save()
        return HttpResponseRedirect(reverse("home"))

# feeback related views
class FeedbackView(TemplateView):
    template_name = 'learnhub/feedback.html'

    def dispatch(self, request, *args, **kwargs):
        user = request.user

        # Ensure the user is a student
        if not hasattr(user, 'learnhubuser'):
            return redirect("home")

        return super().dispatch(request, *args, **kwargs)

    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        user = self.request.user
        context["courses"] = user.enrolled_courses.all()
```

```
        context["feedbacks"] = Feedback.objects.filter(student=user)
        return context

    def post(self, request, *args, **kwargs):
        user = request.user
        courses = user.enrolled_courses.all()
        course_id = request.POST.get("course")
        comment = request.POST.get("comment")
        if course_id and comment:
            try:
                course = Course.objects.get(id=course_id)
                if course in courses:
                    Feedback.objects.create(course=course, student=user, comment=comment)
                    messages.success(request, "Feedback submitted successfully!")
            except Course.DoesNotExist:
                messages.error(request, "Invalid course selection.")
        return redirect("feedback")

# updates page view
class UpdateListView(ListView):
    model = StatusUpdate
    template_name = "learnhub/updates.html"
    context_object_name = "updates"
    ordering = ["-created_at"]

#======================================================================
Manage.py
#======================================================================
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys

def main():
    """Run administrative tasks."""
    os.environ.setdefault("DJANGO_SETTINGS_MODULE", "elearning.settings")
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)

if __name__ == "__main__":
    main()
```