BSc Computer Science
CM3035 - Advanced Web Development
Final Coursework: Build an eLearning app

Introduction

During the course so far, we have developed several applications using Django. This has covered single page applications, database schemas, appropriate model design, forms and templates, RESTful webservices, Celery, Django Channels, Web Sockets, Authentication and many other topics.

For this assignment you are tasked with developing an eLearning application using all knowledge you have gained on the course so far.

This assignment is worth 50% of the total mark for this module.

Task

To implement the eLearning web application. The minimum application requirements are:

Your application should allow users to create new password secured accounts.

You should have two different types of users i.e. students and teachers. Note: each type should have different permissions e.g. a teacher can have access to several student records, but this is not a possible for a student account.

You should collect and store an appropriate amount of information about each user (e.g. username, real name, photo etc.).

Each user should have a "home" page that shows their user information and any other interesting data such as registered courses, upcoming deadlines, etc. It should also display user status updates. These home pages should be discoverable and visible to other users.

Students should be able to post status updates to their "home" page.

Students should also be able to leave feedback for a particular course.

Teachers should be able to search for students and other teachers.

Teachers should create the courses and upload course material – images, pdfs, etc.

Teachers should view their courses and see a list of students enrolled on their course.

Students should see a list of available courses and select the courses they want to enrol to.

Additionally, users should be able to use a portion of the application that makes use of real time communication by using web sockets. Some examples may be real-time text chat, shared whiteboard between students and teachers etc. Other functionality could also utilise audio streaming and file transfers. You are free to use web sockets as you wish but you must include 1 web sockets app in your application.

An appropriate REST interface for User data should be provided.

The application should include functionality that makes appropriate use of Web Sockets.

You are free to design the application layout as you wish. Each functional module (student list, courses, search results, etc...) may be a separate page or you may choose to make the application a Single Page Application.

Deliverables

D1. A django application that implements an eLearning web application and fulfils the functional requirements. The Application should include some users (i.e. students and teachers) for demonstration purposes.

D2. A report (4000-6000 words) describing the application and the reasoning for its design and functionality. The report should explain how your application meets the requirements (see below for criteria R1-R5). Explain the logic of your approach, why is your code arranged as it is? Explain the design of your application, covering all design and implementation decisions you have made. The end of the report should critically evaluate your application. Discuss your design and implementation, did it work well? what parts of the application could be better? What would you change if you attempted the project again? This report should also include a brief section on how to run the unit tests. Finally include a section on how to unzip, install requirements and run your application:

- A list of all packages and the versions used for your implementation.
- Your development environment i.e. the operating system and python version
- Instruction for logging into the django-admin site i.e. username and password
- Login credentials for teacher and students
- Include how to run the unit tests

D3: Include a video of your web application showing and verbally highlighting the main functionalities and your achievements. You can upload the video in .mp4 format or use the alternative link – see submission page. Your video should show how you:

- install the app using the requirements.txt
- talk about the database design and normalisation
- test the app by running the tests
- launch the app and logging in – showing e.g. feedback status updates for a course
- launch redis server and in a second browser to login the second user to initiate a chat between students or a student and teacher.

This should not be longer than 10 minutes. We recommend that you capture the video in mp4 format using software such as OBS.

D4. Bonus points will be given to those who deploy their app using AWS, Digital ocean, etc. You should supply details in your report i.e. app address and login details.

Requirements

We will assess your work based on the following requirements and criteria:

R1: The application should implement the following:

a) Users to create accounts
b) Users to log in and log out
c) Teachers to search for students and other teachers
d) Teachers to add new courses
e) Students to enrol themselves on a course
f) Students to leave feedback for a course
g) Users to chat in real time
h) Teachers to remove / block students
i) Users to add status updates to their home page
j) Teachers to add files (such as teaching materials to their account and these are accessible via their course home page
k) When a student enrols on a course, the teacher should be notified
l) When new material is added to a course the student should be notified

R2: The application should also use:

a) correct use of models and migrations
b) correct use of form, validators and serialisation
c) correct use of django-rest-framework
d) correct use of URL routing
e) appropriate use of unit testing

R3: The application should implement an appropriate database model to model accounts, the stored data and the relationships between accounts

R4: The application should implement appropriate code for a REST interface that allows users to access their data

R5: The application should implement appropriate tests for the server-side code

Code style and technique

Your code should be written according to the following style and technique guidelines:

C1: Code is clearly organised into appropriate files (i.e. view code is placed in an appropriate view.py or api.py file, models are placed in an appropriate models.py file)

C2: Appropriate comments are included to ensure the code is clear and readable

C3: Code is laid out clearly with consistent indenting, ideally following python pep8 standard

C4: Code is organised into appropriate functions with clear, limited purpose

C5: Functions, classes and variables have meaningful names, with a consistent naming style

C6: Appropriate tests to cover the API functionality are provided.

Submission

You should write a brief report, record a video demo and submit your source code. The submission should contain the following items and information:

S1: Deliverables D1 compressed in standard .ZIP format.

S2: Deliverables D2 and D4 in .PDF format.

S3: Deliverable D3 in .mp4 format.

S4: Deliverable D3 - alternative link – use of YouTube or similar and submit the link. **Make sure your video remains unlisted.**

Marking Criteria

The application will be graded on whether it is technically correct and implements the API as requested. Code should be clear and easy to follow. The application should be well organised - for instance - it should make correct use of models, API, view and serialiser files. A good application will include a suite of tests that ensure that application correctly implements the API that is described.

See separate rubric file for details.

Criteria	Marks available	CumTotalPoints	%
Code			
Application loads as required - using a valid requirements.txt file	4	4	5.7
Application implements all functionality specified	5	9	12.9
Database/Model design is appropriate for the application	3	12	17.1
Application frontend design is appropriate for the application	4	16	22.9
Application makes use of functionality of Django as described in class (topics 1-10) - All topics including swagger etc	4	20	28.6
Unit testing is included	6	26	37.1
Code is clean with good syntax and comments.	3	29	41.4
Code is functional (ie free of errors, reproducibility of results is reasonable)	3	32	45.7
Code is modular and well organised (i.e. naming conventions, correct use of functions and classes, no huge code blocks)	4	36	51.4
Advanced techniques not yet covered in the course used (e.g alternatives to bootstrap, frontend frameworks, graph database	5	41	58.6
Report			58.6
Report is clearly written	3	44	62.9
Report explains how the application requirements have been met	3	47	67.1
Application of the techniques taught (best practice use of django and django-rest-framework , asynchrnous web sockets evid	4	51	72.9
Report show evidence of critical evaluation of work/approach in relation to state of the art in web development	4	55	78.6
Report includes necessary run info such as OS, Python version and login credentials	3	58	82.9
Misc			82.9
Submission contains a video presentation with student clearly highlighting their achievements	5	63	90
Bonus points if you deploy your own app using AWS, Digital ocean, etc.	7	70	100