Ayulla's Software Engineering Tryout Task (ASETT)

Ayulla Limited

GENERAL INSTRUCTIONS

- You will have 2 weeks to complete this task from the time you receive it. You will receive a final data and time of submission from your respective recruiter.
- You may submit the task before the due date and not necessarily on the due date.
- Only two resubmissions are allowed. Every new submission will render the previous one obsolete.
- Your code will be judged by black box software testers who may have little or no knowledge of the language you used in the task. You must include a README file that has execution instructions.
- If you have a database that must be populated for your system to work, you must include the steps for populating the database and sample data for your database.
- You may or may not include notes on your implementation that describe steps you took in accomplishing the task
- Add comments to your files as needed appropriately.

As part of Ayulla's Software Engineering Tryout Task (ASETT), you are required to implement a basic but scalable Learning Management System (LMS) for a small university. Your system is expected to be abstract, hence it is important that you skip implementation of minor details. The LMS must be built to the specifications of this document, although you can implement extra features if you have fulfilled the requirements. **The full completion of this task is not necessary**, although significant progress towards completion should be demonstrated.

The following list outlines the required coding restrictions for the system:

- A database should be created along with a corresponding schema showing the relationship between the various users and courses. The database must be fully decomposed if it's relational.
- The database must contain at least 10,000 students,
- The frontend should utilize Django's template tags to reduce the amount of redundant html code. Demonstration of some custom template tags is encouraged, but not required.
- Url configurations should be easily maintainable, so slight changes to the paths will not break the code in the views.
- All implemented views and models must have corresponding unit tests.
- An appropriate admin site should be implemented through which users and courses can be added and monitored.
- A reasonable understanding of class-based views must be demonstrated in the implementation.
- The entire project must be packaged with the appropriate python virtual environment.

USER:

A user refers to an entity who will use the system under design to achieve a certain goal. The user is the most important part of the system as they will add data, modify data, destroy (remove) data, and retrieve data from the system. The kind of data a user may interact with and the process of this interaction is solely dependent on the permission of user.

Users must have a role assigned to them. The role of a user determines the degree of their permission. They are three major roles in the system: admin, instructor, and student. **It is important to implement all three major roles.** The system also has an additional minor role, that of a teacher's assistant. **You may implement the minor role if you have time.** The attributes and permission of these roles are described below:

- **Admin:** The admin has the greatest privilege, but has certain restrictions. The admin can create a new course, and add students, one or more instructors, teaching assistants, and observers. After creating a course, an admin does not have access to add contents to the course. **The admin should not be able to create assignments, modify assignment, create discussions, create a quiz, take a quiz or change any other content in the course.** ***The admin must be able to view the contents of any course.*** The admin cannot remove another admin but can remove other users with other roles.

- **Instructor:** The instructor is the course moderator. They can add or remove assignments, create or delete quizzes, upload or remove class files (PowerPoints and recommended readings), conduct roll call, grade student works, add or change grades for the students, and provide comments on submitted work. Instructor should be able to post announcements and create a new discussion thread. **The instructor is not able to remove any other users in the system.** An instructor can have multiple classes assigned to them.

- **Student:** The student is a major component of the system. The students respond to activities. They take quizzes published by the instructor, submit assignments, participate in discussions, and view their grades. The student can only view courses to which they are registered to.

- **Teaching Assistant:** A teaching assistant can carry out similar duties as the instructor, although with minor restrictions. The teacher assistant is not able to post any assignments or quizzes, but can grade them on the teacher's behalf.

COURSES

A course refers to an entity which can be accessed or modified by the users of the LMS. Each course must have a unique identifier, and must have a many to many relationship with the instructors, students and teaching assistants. There can be multiple redundant courses with different instructors, which can be defined as different sections of the same class.

A course has multiple attributes. The most important attributes include: Grade, Assignment, Quiz, and Discussion.

You will have to implement at least 2 of these attributes. You must decide on which one to implement, however, you should consider all factors and functions related to each attribute before choosing it.

USER INTERFACE AND USER EXPERIENCE

You will not be provided with any UI/UX frameworks and/or wireframes. Your creativity is judged here. You should be able to display your data to the different users as well as have buttons, controls, and forms that the users can use in their interaction with the system. You can exhibit any kind of skill set in this section without restrictions. However, you must have at least an abstract user interface that can allow a user to communicate with the system.

Do not spend most of your time working on a UI. Although it is important, your skill is mainly judged on your implementation of the backend of the system.