**Instructions: All questions are mandatory. Each question carries 10 marks Max Marks : 30**

**Q1. Scenario:**

You're tasked with developing a customer feedback submission feature for an e-commerce platform built with Django. After users complete a purchase, they're redirected to a feedback form to provide their opinions on the shopping experience.

**Task:**

**Define Feedback Model:**

Create a Django model to store feedback data in your app's models.py file. The model should include fields for rating the overall shopping experience, comments from the user, and an optional email for follow-up.

**Create Feedback Form:**

Define a form using Django's ModelForm to collect feedback from users. The form should be based on the Feedback model defined earlier.

**Implement Feedback Submission View:**

Create a view to handle feedback form submissions. When users submit feedback, ensure that the form data is processed securely and saved to the database using the Feedback model.

**Implement POST-Redirect Pattern:**

Implement the POST-Redirect pattern to handle form submissions efficiently. After processing the form data securely, redirect users to a confirmation page to acknowledge that their feedback has been received.

**Create Confirmation Page:**

Design a confirmation page using Django templates to display a thank-you message to users for submitting their feedback. Include any additional information or calls to action as deemed appropriate.

**Implement Error Handling:**

Handle errors gracefully during the feedback submission process. If form submission fails or encounters errors, display informative error messages to users and provide guidance on resolving issues encountered during the feedback submission process.

**Q2. Scenario:**

You're tasked with developing an online quiz platform for a university's computer science department. The platform should allow students to take quizzes on various topics and provide instant feedback on their performance.

**Task:**

**Design Django Models:**

Define Django models to represent quizzes, questions, and multiple-choice answers. Each quiz should have a title, description, and a set of questions with associated answers.

**Create Quiz Taking Interface:**

Develop Django views and templates to display quizzes and questions one at a time. Allow users to select answers using HTML forms.

**Handle Submission of Answers:**

Process user-submitted answers and calculate scores based on correctness. Provide instant feedback to users after each question submission.

**Display Quiz Results:**

Create a Django view and template to display quiz results, showing the user's score, correct and incorrect answers, and a summary of their performance.

**Q3. Scenario:**

You're developing a web application for an online bookstore, and you want to enhance the user experience by implementing a feature that remembers users' preferred language using cookies.

**Task:**

**Cookie Implementation:**

Design a Django view that sets a cookie to store the user's preferred language selection when they choose a language option on the website. Ensure that the cookie is set with an appropriate expiry date to maintain the user's language preference across multiple sessions.

**Language Selection Interface:**

Develop a user interface component, such as a dropdown menu or radio buttons, to allow users to select their preferred language. Ensure that when a user selects a language, it triggers the setting of the corresponding cookie to store their language preference.

**Cookie Retrieval:**

Implement logic to retrieve the user's preferred language from the cookie when they visit the website. Use the stored language preference to dynamically display content in the user's chosen language.