Full Stack Development (21CD71) Assignment – 20 Marks

Instructions:

- Each student is assigned a specific question based on their USN.
- Solve your assigned question and submit it following the GitHub Submission Guidelines provided.
- If you solve additional questions outside your assigned USN, you will receive bonus marks.
- Ensure that you submit the project along with a detailed report (PDF format) attached to your GitHub repository.

Submission Details:

- **Deadline:** 21/FEB/2025
- **Mode of Submission:** Google Form Link: Google Forms
- **Evaluation Criteria:** Code functionality, Report quality, and Correctness of Implementation.

Roll Number-wise Assignment Mapping:

USN – 21SECD01-21SECD06

Q1. Develop a Django-based Student Management System with the following features:

- The system should allow the admin to add, update, and delete student records using Django Admin.
- Each student should have fields like `name`, `roll number`, `course`, `email`, and `date of birth`.
- Use Django's generic `ListView` to display all students and `DetailView` to show a particular student's details.
- Implement URL configuration that allows navigation between student list and detail views using `reverse_lazy()`.

USN – 21SECD7-21SECD12

Q2. Build an Event Registration System using Django where:

- Users can register for events by filling out a form.
- The form should include fields such as `name`, `email`, `phone number`, and `event name`.
- Store the registered participants in a database, and use Django Admin to manage event registrations.
- Display the list of registered participants using Django's `ListView`, and allow event organizers (admin) to approve or reject a registration.

- Implement `reverse_lazy()` to redirect users to a success page after submission.

USN – 21SECD13-21SECD18

Q3. Create a User Feedback System with the following features:

- Users should be able to submit feedback using a Django ModelForm.
- The form should contain `name`, `email`, `subject`, and `message` fields.
- Implement custom validation to ensure:
    - The email provided is from a valid domain (e.g., only allow `@example.com` emails).
    - The feedback message contains at least 50 characters.
- Store the submitted feedback in a database and display all feedback entries on an admin panel.
- Use CSRF protection to secure the feedback submission process.

USN – 21SECD19-21SECD24

Q4. Implement a Secure Job Application Form using Django where:

- Applicants can submit their `name`, `email`, `phone number`, `resume (PDF only)`, and a cover letter.
- The application form should be implemented using Django ModelForms and should validate:
    - The resume file is in PDF format only.
    - The phone number contains exactly 10 digits.
- Store all applications in a database and allow the admin to view, shortlist, or reject applications.
- Implement CSRF protection for the form submission.

USN – 21SECD25-21SECD30

Q5. Create a Multi-Page Blogging System with the following features:

- Users can write blog posts containing a `title`, `author`, `content`, and `published date`.
- Use Django's generic `CreateView`, `ListView`, and `DetailView` to manage blogs.
- Implement multiple URL configurations:
    - `/blogs/` → List all blog posts
    - `/blogs/<int:id>/` → Display a single blog post
    - `/blogs/new/` → Allow users to create a new blog post
- Use `reverse_lazy()` to redirect users to the blog list after successfully posting an article.

USN – 21SECD31-21SECD36

Q6. Develop a Contact Form System with URL Configuration and Custom Validation:

- Implement a contact form where users can submit inquiries.
- The form should have `name`, `email`, `phone number`, and `message` fields.
- Validate that the phone number contains exactly 10 digits and email belongs to a corporate domain (`@company.com`).
- Store the contact messages in a database and allow the admin to view them.
- Implement multiple URL configurations to separate different functionalities.

USN – 21SECD37-21SECD42

Q7. Implement an E-Commerce Product Management System where:

- Products have fields like `name`, `description`, `price`, `stock`, and `category`.
- Each product should belong to a category, and the category should have a `One-to-Many` relationship with products.
- Use Django Admin to manage product listings and categories.
- Display a list of products on the website using Django's `ListView`.
- Use Django migrations to evolve the database schema, adding a `discount` field to the `Product` model after initial development.

USN – 21SECD43-21SECD48

Q8. Develop a Student-Teacher Relationship Management System with the following:

- Each student is assigned to a teacher using a `ForeignKey` relationship (One-to-Many).
- Each teacher can have multiple students, but a student can only have one assigned teacher.
- Store student and teacher data in a database and manage them through Django Admin.
- Use `ListView` and `DetailView` to display student and teacher profiles.

USN – 21SECD50-21SECD55

Q9. Build a Full-Stack Django Web Application for Customer Complaints:

- Customers can submit complaints through a form.
- The form should include `name`, `email`, `product`, `issue description`, and `priority level (High, Medium, Low)`.
- The data should be stored in a database and managed via Django Admin.
- Display all complaints on an admin page, allowing the admin to change the status (`Pending`, `Resolved`).

- Implement `reverse_lazy()` to redirect users to a success page after submission.

USN – 21SECD56-21SECD59 and 21SECD91-94, 21SECV01

Q10. Develop an Online Course Enrollment System with Django:

- Create models for `Course` and `Student`.
- Implement a `Many-to-Many` relationship where students can enroll in multiple courses, and a course can have multiple students.
- Use Django Admin to manage students and courses.
- Implement a form where students can enroll in courses by selecting from available options.
- Use Django’s `ListView` to display all available courses and enrolled students.
- After successful enrollment, use `reverse_lazy()` to redirect students to a confirmation page.

GitHub Submission Guidelines:

Instructions for Assignment Submission:

1. Complete the assigned question as per your USN.
2. Refer to the assignment document for question mapping.
3. Solve the question in Django, ensuring correct implementation of models, admin interface, forms, generic views, and other required functionalities.
4. Upload your code to GitHub in a public repository and attach the repository link in this form.
5. Prepare a detailed report (PDF format) containing the following:
6. Step-by-step procedure of how you solved the question.
7. Screenshots of the implemented code, web pages, and output results.
8. Attach the report PDF to your GitHub repository. Ensure the PDF is inside the repository for easy access

Important Notes:

- Ensure the GitHub repository is public so that it can be accessed for evaluation.
- Submissions without a report or incorrect links will not be considered.
- Bonus marks will be awarded for solving additional questions outside your assigned range.
- Deadline: 21/FEB/2025 – Late submissions will not be entertained.
- Double-check your form before submission! Incomplete or incorrect entries may affect your evaluation.

Bonus Marks: If you attempt and successfully complete **any additional question** outside your assigned range, **you will receive bonus marks**.

Happy coding!