Week 4: CRUD API Implementation Assignment

Objective

By the end of this assignment, students will have implemented a **basic CRUD API** for one feature in their project, tested it using **Postman**, and documented any challenges encountered along with their solutions.

Assignment Instructions

1. **Backend: Implement Basic CRUD APIs**

Each student must implement CRUD operations for **atleast two feature in their project**. This could be users, tasks, products, or any other relevant resource.

1.1 Choose a Feature

- Implement **User Management** API (/users)
- Implement **Task Management** API (/tasks)
- Implement **Product Management** API (/products)
- Implement **Orders API** (/orders)

1.2 Setup the API Routes

Implement the following endpoints:

Method	Endpoint	Description
POST	/resource/	Create a new resource
GET	/resource/	Retrieve all resources
GET	/resource/{id}	Retrieve a single resource by ID
PUT	/resource/{id}	Update an existing resource
DELETE	/resource/{id}	Delete a resource

Example for Tasks API (tasks/views.py in Django REST Framework)

```python
from rest_framework import generics
from .models import Task
from .serializers import TaskSerializer

class TaskListCreate(generics.ListCreateAPIView):
    queryset = Task.objects.all()
    serializer_class = TaskSerializer

class TaskRetrieveUpdateDelete(generics.RetrieveUpdateDestroyAPIView):
    queryset = Task.objects.all()
    serializer_class = TaskSerializer
```

Example for Tasks API (Express.js)

```
const express = require("express");
const router = express.Router();
const Task = require("../models/Task");

// Create a task
router.post("/tasks", async (req, res) => {
    const task = new Task(req.body);
    await task.save();
    res.status(201).json(task);
});

// Get all tasks
router.get("/tasks", async (req, res) => {
    const tasks = await Task.find();
    res.json(tasks);
});

// Get a single task
router.get("/tasks/:id", async (req, res) => {
    const task = await Task.findById(req.params.id);
    res.json(task);
});

// Update a task
router.put("/tasks/:id", async (req, res) => {
    const task = await Task.findByIdAndUpdate(req.params.id, req.body, { new: true });
    res.json(task);
});

// Delete a task
router.delete("/tasks/:id", async (req, res) => {
    await Task.findByIdAndDelete(req.params.id);
    res.status(204).send();
});

module.exports = router;
```

Commit messages must be written for each successful step.

2. Test APIs Using Postman

After implementing the CRUD API, test each endpoint using **Postman**.

2.1 Steps for API Testing in Postman

1. Open Postman and create a **new request**.
2. Select the HTTP method (**GET, POST, PUT, DELETE**).
3. Enter the API URL (e.g., http://127.0.0.1:8000/tasks/).
4. If applicable, add a **request body** (for POST or PUT requests).
5. Click **Send** and check the response.

2.2 Expected API Responses

Method	Expected Status Code	Sample Response
POST	201 Created	{ "id": 1, "title": "New Task" }
GET	200 OK	[{ "id": 1, "title": "Task 1" }]
GET (by ID)	200 OK	{ "id": 1, "title": "Task 1" }
PUT	200 OK	{ "id": 1, "title": "Updated Task" }
DELETE	204 No Content	*No response body*

T**ake screenshots of Postman requests and responses for documentation.**

3. **Discuss and Resolve Issues Encountered**

Document and resolve any errors faced during implementation.

3.1 Common Errors and Fixes

Error	**Cause**	**Fix**
500 Internal Server Error	Database model issue	Check migrations & database schema
400 Bad Request	Missing fields in request	Ensure required fields are sent in request body
404 Not Found	Wrong ID in URL	Verify the correct ID exists in the database
401 Unauthorized	Authentication required	Implement proper authentication

3.2 Write-Up Requirement

Submit a brief write-up answering the following:

- **What challenges did you face while implementing CRUD operations?**
- **How did you debug errors?**
- **What improvements can be made?**

C**ommit code fixes with meaningful commit messages.**

Submission Requirements

1. **Push all commits** to a GitHub repository.
2. Submit a **link to the repository**.
3. Upload **Postman test screenshots**.
4. Submit the **write-up document** detailing challenges and resolutions.