Date
Page No.

Advanced Web Technology

Assignment II

1) Prisma Orm

Prisma Orm is a modern ORM (object Relational Mapper) that lets us define models in a schema.prisma file and use JS to interact with database.

We used Prisma with MongoDB to define a Quote model and fetch/add records easily. It auto-generates types and gives a clean API (prisma.quote.findMany(), etc) making database work simpler.

2) Express.JS JSON API

Express.js is a lightweight backend framework for Node.js. We used it to create a REST API:-

→ "GET /api/quotes to fetch all quotes"

→ "POST /api/quotes to add new data"

Express makes routing easy and supports middleware for tasks like logging, error handling and parsing JSON bodies.

Date
Page No.

3) Frontend fetch + HTML

We made a simple index.html page inside the public/ folder.
Using fetch() API in JS, we sent a request to our Express backend and displayed the response (JSON) on the page.

```
const getQuotes = async() => {
  Const response = await fetch ('http://localhost:3030/
                    api/quotes');
  quotes = await response.json();
  return quotes;
}
```

4) Project folder Structure

I followed my folder structure :-

aot/assignments/project-1/

→ prisma/
→ public/
→ server.js
→ .env

Date
Page No.

5) MongoDB + .env Config

MongoDB is a NoSQL database that stores data in JSON-like documents. Prisma supports it with minor changes. We stored our database connection string in .env to keep credentials safe.

DATABASE_URL = "mongodb+srv://..."

This practice is used in professional projects to protect secrets.

6) MongoDB Replica set for Prisma

Prisma needs MongoDB to run in replica set mode (even on localhost) to support transactions. I modified the mongod.conf file like this:-

```
replication:
    replSetName: rs1
```

Then, I restarted MongoDB and run:

```
mongosh
>rs.initiate()
```

This made my MongoDB act as a primary node in a replica set. Only after this, Prisma could connect and perform DB operations.

Neon Neupane
Roll: 31