📌 Backend Developer (Intern) – Project Submission

About the Submission

The Assignment was to test my backend knowledge, so I have not focused that much on frontend. In this assignment I have used MVC pattern, to deal with controller, middleware and routes. For Routes I have separately used routes folder and used express to route controller.

Assignment Overview

A landing page will occur, where there will be Get Started button, user will click it, if user is already registered, then sign in, otherwise sign up. Currently no page forwarding for forgot password. SignIn have only email validation and sign up have all mandatory field validation.
In SignUp at the top there is section to chooose their role, without selecting role, user is unable to move forward.
If user SignIn : he wil see the notes he wrote, he can manipulate notes like update and delete and can also add new notes.
If Admin Sign In : he will see all notes of every user, he can update, delete notes and can also create new note for other user who Is already registered, he cannot create note for that user, which is not registered yet.

Core Features to Implement

✅ Backend (Primary Focus)

- User registration & login APIs with password hashing and JWT authentication
  Done : password encryption is done when user send form data to backend and backend encrypt password and with store encrypt password to DB with other information.
  JWT authentication will work when any request is done by user, It will verify access token, provided when user sign in.

- Role-based access (user vs admin)
  Done : user can choose it while signing up.

- CRUD APIs for a secondary entity (e.g., tasks, notes, or products)
  Done : when they sign in they can perform CRUD operation on their notes. While admin can perform CRUD operation on every user notes.

- API versioning, error handling, validation
  Done : Every possible error is handled and sended to frontend using status code and json, so that it can be notify to frontend user's

- API documentation (Swagger/Postman)
  Done : Currently I have deployed my backend server on this domain :
  https://prime-trade-back.vercel.app
  for local it is : `http://localhost:5000`

  1st API : /signin (method : POST)
  This API takes email and password as payload,
  ex : {"email":'harishnigam21@gmail.com','password':'Harish@18'}

  2nd API : /signup (method : POST)
  This API takes role,firstname,middlename(optional),lastname,gender,mobileno, email, password as payload
  ex :
  {'role':'user','first_name':'Harish','middle_name':'','last_name':'Nigam','gender':'male','mobile_no':'8962008472','email':'harishnigam21@gmail.com','password':'Harish@18'}

  3rd API : /getNotes (method : GET)
  This API is GET request so no payload is attached, It cross checks the valid user using access token.
  After validation if it is user then it gives all notes wrote by him as response or if it is Admin then it will notes of all existing users.

  4th API : /postNotes (method : POST)
  payload depends on role
  → if it is user, it will take note details as payload
  ex: {'postDetails':{title:'NewTile',body:'Your Note'}}, why only title and body, because in whole note there only this fields. Email id will fetch using access token and created_at date will be assigned by backend.
  → if it is admin, it will take note details and email as payload
  ex: {'postDetails':{title:'NewTile',body:'Your Note'},email:'harishnigam21@gmail.com'}, why email because here admin can create notes, but only for those user who are registered.

  5th API : /updateNotes (method : PUT)
  payload depends on role
  → if it is user, it will take note details and id as payload
  ex: {'postDetails':{title:'NewTile',body:'Your Note'},id:'id of note'}, why id because it will find note matching with this ID and then update it.
  → if it is admin, it will take note details, id and email as payload
  ex: {'postDetails':{title:'NewTile',body:'Your Note'},email:'harishnigam21@gmail.com',id:'note id'}, why email because here admin can update notes, but only for those user who are registered and id to find that note.

  6th API : /deleteNotes (method : DELETE)
  here on this API, note id is given as payload
  ex: {'id':'note id'}

- Database schema (Postgres/MySQL/MongoDB)
  Done : to connect with DB, prisma ORM is used on MySQL
  down below is schema that contains Prisma connections and models used in this assignment :

```
generator client {
provider      = "prisma-client-js"
output        = "../prisma/app/generated/prisma/client"
binaryTargets = ["native", "rhel-openssl-3.0.x"]
}
datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}
model Users {
id              Int     @id @unique(map: "id_UNIQUE") @default(autoincrement())
role            String  @db.VarChar(45)
first_name      String  @db.VarChar(128)
middle_name     String? @db.VarChar(128)
last_name       String  @db.VarChar(128)
gender          String  @db.VarChar(45)
mobile_no       String  @unique(map: "mobile_no_UNIQUE") @db.VarChar(20)
email           String  @unique(map: "email_UNIQUE") @db.VarChar(128)
password        String  @db.VarChar(512)
reference_token String  @unique(map: "reference_token_UNIQUE") @db.VarChar(512)
}

model Notes {
id         Int    @id @unique(map: "id_UNIQUE") @default(autoincrement())
email      String @db.VarChar(128)
created_at String @db.VarChar(45)
title      String @db.Text
body       String @db.Text
}
```

✅ **Basic Frontend (Supportive)**

- frontend has been designed very simple and user friendly.

✅ **Security & Scalability**

- Secure JWT token handling : Done
- Input sanitization & validation : Done
- Scalable project structure for new modules : Done
- Optional: caching (Redis), logging, or Docker deployment

Deliverables

1. Backend project hosted in GitHub with README.md setup : Done
2. Working APIs for authentication & CRUD : Done
3. Basic frontend UI that connects to your APIs : Done
4. API documentation (Swagger/Postman collection) : In this document
5. .env file is encluded here, as this will help in setupping project
6. Short scalability note (e.g., microservices, caching, load balancing)
