❖ **Project Description:**

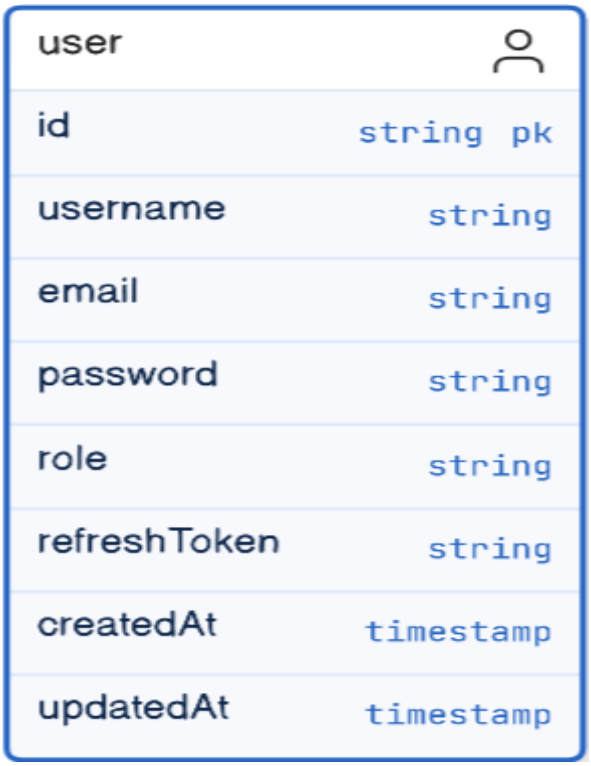
Alright, let's dive into the key features of my project! I promise I'll keep it interesting — no boring tech talk here! First, let me walk you through what makes this project stand out.

❖ **Key features of this backend project:**

1. Optimized Code
2. Better Performance
3. Well Structured folders and files
4. Impressive API response and error handling
5. Deployed links are available to use
6. Simplified Response Handling, Error Management, and Schema Validation

❖ **User Model:**

On the basis of below user model I have implemented the **Authentication System** where user can **Register, Login** and **Logout securely.**



❖ Roll-Based Access Control(RBAC) API Design:

This is the API design for this project. I will also share the API documentation using postman, as I am going to **deploy** this node.js based backend on **render.com.**

The API is design following the REST architectural style. The API is implemented using Node.js and Express.js. It used JSON for data exchange and follows standard HTTP request methods such as GET, POST, PUT and DELETE.

❖ List of API endpoints and their functionalities:

1. api/v1/auth/register (POST)
   - Create the new user (admin, moderator and user) account
2. api/v1/auth/login (POST)
   - Log in using credentials and generate JWT tokens (Refresh and Access token)
3. api/v1/auth/logout (POST)
   - To logout the user and remove the refresh token from the database
4. api/v1/auth/refresh-token (GET)
   - To generate the new access token
5. api/v1/auth/checkSession (POST)
   - To verify the current session of user

❖ **Login credentials for different roles:**

Email Id	Password
admin@gmail.com	12345678
user@gmail.com	12345678
moderator@gmail.com	12345678

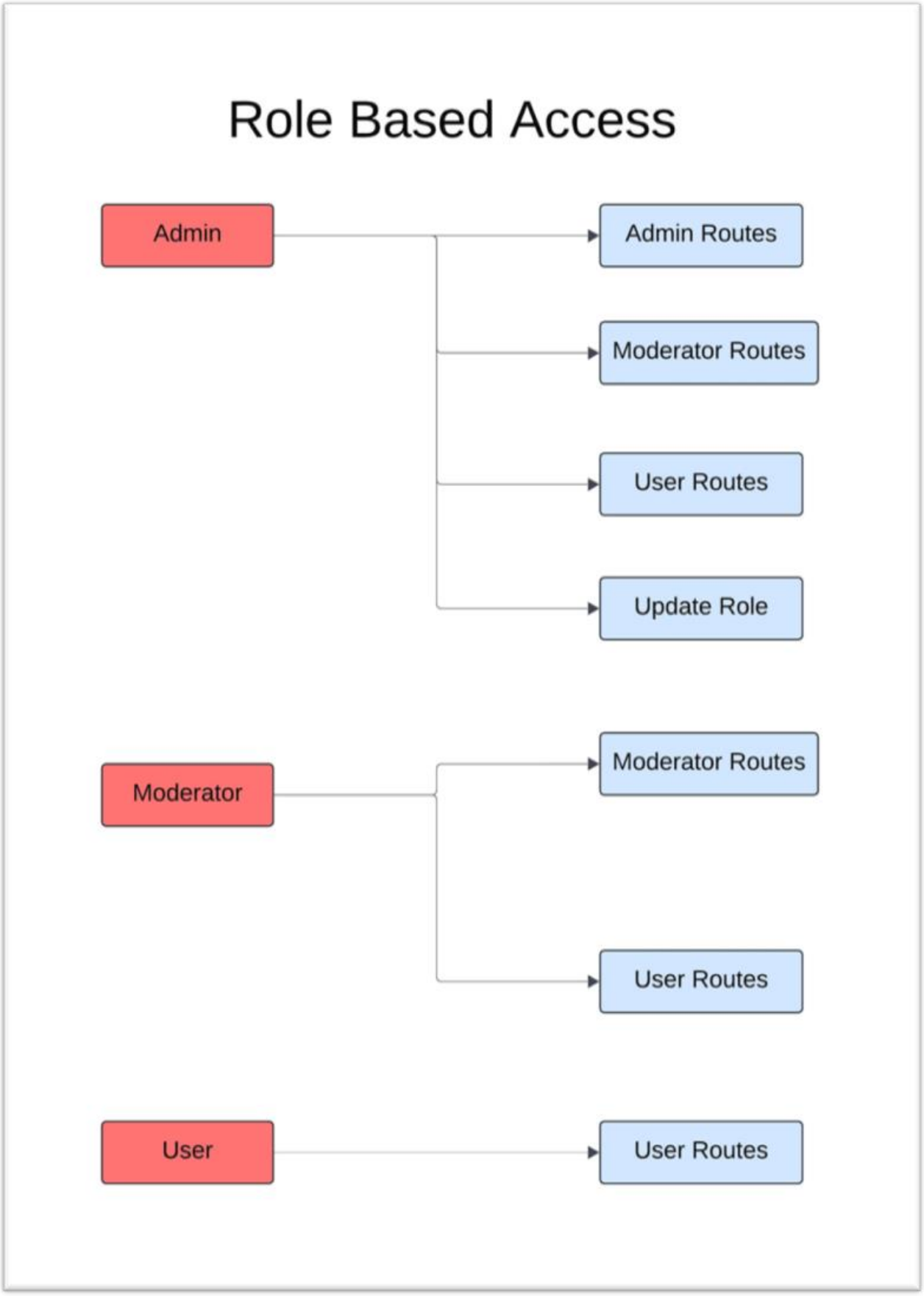
❖ **List of API endpoints and their functionalities:**

This API implements **Role-Based Access Control (RBAC)** where users are assigned different roles (**admin**, **moderator**, and **user**). Each role has specific access permissions to various routes in the system.

1. api/v1/access/update-user-role (POST)
   - Only admin can access this role and update the user role to admin, moderator and user
2. api/v1/access/admin (GET)
   - Access to admin-specific resources
3. api/v1/access/moderator
   - Access to moderator-specific resources
4. api/v1/access/user
   - Access to user-specific resources

❖ **Summary of this project:**

1. **Admin** has the highest level of access and can assign roles.
2. **Moderators** have more access than users but less than admins.
3. **Users** can only access basic user routes.


Role Based Access
Admin
Admin Routes
Moderator Routes
User Routes
Update Role
Moderator
Moderator Routes
User Routes
User
User Routes