Role-Based Access Control (RBAC) - Teaching Slides

Session Overview: Role-Based Access Control (RBAC)

- What is RBAC and why do we need it?
- Adding roles to users
- Creating role-based middleware
- Securing admin-only routes
- Demo and practical assignment

What is RBAC?

- RBAC stands for Role-Based Access Control
- Assigns different roles to users (e.g., 'user', 'admin')
- Controls access to API routes based on roles
- Prevents unauthorized access to sensitive actions

Why Use RBAC?

- Protect critical operations (like deleting products)
- Prevent users from accessing other users' data
- Clean and scalable permission system
- Common in production-grade APIs

Adding Role Field to User Model

```js
const userSchema = new mongoose.Schema({
  name: String,
  email: String,
  password: String,
  role: {
    type: String,
    enum: ['user', 'admin'],
    default: 'user'
  }
});
```

Middleware to Check Role

```js
const checkRole = (role) => {
  return (req, res, next) => {
    if (req.user.role !== role) {
      return res.status(403).json({ error: 'Access denied' });
```

```
    }
    next();
  };
};
module.exports = checkRole;
```

Example: Admin-Only Route

```js
const authenticateUser = require('./authMiddleware');
const checkRole = require('./checkRole');

router.delete('/products/:id', authenticateUser, checkRole('admin'), deleteProduct);
```

Example: Admin Creating a Product

- Only users with role 'admin' should access:

```http
POST /products
Authorization: Bearer <admin-token>
```

Example: Normal User Trying Same

- A user with role 'user' will get:

```json
{
  "error": "Access denied"
}
```

Design Best Practices

- Define roles early in your user model
- Store roles in tokens (optional)
- Create reusable `checkRole()` middleware
- Keep role-checking logic out of controllers

Assignment

- Add `role` to user schema
- Add `checkRole('admin')` to product create/delete routes
- Try login with admin and user accounts
- Use Postman to test access control

Summary

- RBAC lets you control access based on user roles
- Middleware is used to restrict route access
- Protect your API from unauthorized actions
- Makes your app scalable and secure