## **Machine Test: Node.js + Express.js + MongoDB**

### **Tech Stack: Node.js, Express.js, MongoDB, JWT**

### **Requirements:**

* Use **Express.js** for routing
* Use **MongoDB** with mongoose
* Implement **JWT-based login/authentication**
* Implement **RBAC (Role-Based Access Control)**
* Use proper **MongoDB indexes**
* Use **aggregation** for analytics
* Follow proper project structure
* API-only backend

## **Modules to be Developed**

### **1. User Module**

#### **User Schema**

{

\_id: ObjectId,

name: String,

email: String (unique),

password: String (hashed),

role: String (enum: ['admin', 'author', 'reader']),

createdAt: Date,

updatedAt: Date

}

#### **APIs**

* POST /auth/register – Register a user
* POST /auth/login – Login and get JWT
* GET /users – Admin-only: View paginated user list

### **2. Blog Module**

#### **BlogPost Schema**

{

\_id: ObjectId,

title: String,

content: String,

tags: [String],

author: ObjectId (ref: User),

isPublished: Boolean,

createdAt: Date,

updatedAt: Date

}

#### **APIs**

* POST /blogs – Author only: Create a blog
* GET /blogs – Reader: View **published** blogs  
  + Filters:  
    - by tag: ?tag=javascript
    - by author: ?authorId=...
    - by title search: ?search=node
* PATCH /blogs/:id – Author only: Update own blog
* DELETE /blogs/:id – Author only: Delete own blog
* GET /blogs/my – Author only: Get own blogs

Blog listing must use **MongoDB Aggregation Pipeline**

### **3. Analytics Module (Aggregation-Based)**

GET /blogs/analytics – Admin only:

* Number of blogs per author
* Most used tags across all published blogs

**Use aggregation pipeline** for this.

### **Authentication & RBAC**

* Implement **JWT authentication**
* Protect routes using authMiddleware
* Restrict access via **role middleware**:  
  + Admin can manage users & see analytics
  + Author can manage own blogs
  + Reader can only read published blogs

## **Indexing Requirements**

* Unique index on email in User collection
* Compound index on isPublished, tags, title in Blog collection

## **Bonus (Optional)**

* Add bcrypt password hashing
* Add input validation middleware (express-validator)
* Use dotenv for config separation
* Add rate limiting for public routes

## 

## **Evaluation Criteria**

| **Criteria** | **Weight** |
| --- | --- |
| Authentication & JWT setup | 15% |
| RBAC implementation | 20% |
| Aggregation usage for filters/analytics | 20% |
| Database indexing | 10% |
| Clean project structure | 10% |
| CRUD functionality | 15% |
| Code quality, comments, modularity | 10% |