****Real-World Backend Engineering Tasks****

Authentication & Authorization

1. **User Sign-Up & Sign-In**: Implement secure user registration and login using JWT.
   Tools: Node.js, Express.js, MongoDB, JSON Web Tokens (JWT), bcrypt
2. **Password Hashing**: Use bcrypt to securely hash and verify passwords.
   Tools: bcrypt, Node.js
3. **Role-Based Access Control (RBAC)**: Restrict access to resources based on user roles.
   Tools: Node.js, Express.js, Middleware functions
4. **OAuth 2.0 Integration**: Allow users to authenticate via third-party providers like Google or Facebook.
   Tools: Passport.js, OAuth 2.0, Node.js
5. **Two-Factor Authentication (2FA)**: Add an extra layer of security using OTPs or authenticator apps.
   Tools: Node.js, speakeasy, nodemailer

Rate Limiting & Throttling

6. **Basic Rate Limiter**: Limit the number of requests per user/IP per time window.
   Tools: Express-rate-limit, Node.js
7. **Token Bucket Algorithm**: Implement a scalable rate limiter using the token bucket approach.
   Tools: Custom middleware, Node.js
8. **Leaky Bucket Algorithm**: Control data flow to prevent burst traffic.
   Tools: Custom middleware, Node.js
9. **Sliding Window Log**: Maintain a log of request timestamps for precise rate limiting.
   Tools: Redis, Node.js
10. **Distributed Rate Limiting**: Use Redis or similar tools to manage rate limits across multiple servers.
    Tools: Redis, Node.js, Express.js

Caching & Performance

11. **In-Memory Caching**: Use tools like Redis or Memcached to cache frequent queries.
    Tools: Redis, Memcached, Node.js
12. **HTTP Caching Headers**: Implement ETag, Cache-Control, and Last-Modified headers.
    Tools: Express.js, Node.js
13. **Database Query Optimization**: Analyze and optimize slow database queries.
    Tools: MongoDB Profiler, Mongoose

14. **Lazy Loading**: Load data only when necessary to improve performance.
    Tools: Node.js, Mongoose
15. **Pagination**: Implement efficient data pagination for large datasets.
    Tools: Mongoose, Express.js

API Design & RESTful Services

16. **CRUD Operations**: Build Create, Read, Update, Delete endpoints for a resource.
    Tools: Node.js, Express.js, MongoDB
17. **Pagination & Filtering**: Allow clients to paginate and filter data results.
    Tools: Express.js, Mongoose
18. **Sorting & Searching**: Implement sorting and search functionality in APIs.
    Tools: MongoDB, Mongoose
19. **Versioning**: Manage multiple API versions gracefully.
    Tools: Express.js, Node.js
20. **Rate-Limited APIs**: Combine API development with rate limiting mechanisms.
    Tools: Express-rate-limit, Node.js

Real-Time Data & WebSockets

21. **Chat Application**: Build a real-time chat app using WebSockets.
    Tools: Socket.IO, Node.js
22. **Live Notifications**: Implement server-sent events (SSE) for real-time updates.
    Tools: Node.js, EventSource API
23. **Collaborative Editing**: Allow multiple users to edit documents simultaneously.
    Tools: Socket.IO, Node.js
24. **Presence Indicators**: Show online/offline status of users in real-time.
    Tools: Socket.IO, Node.js
25. **Real-Time Analytics Dashboard**: Display live metrics and data visualizations.
    Tools: Socket.IO, Node.js, Chart.js

Microservices & Distributed Systems

26. **Service Discovery**: Implement a mechanism for services to discover each other.
    Tools: Consul, etcd, Node.js
27. **Load Balancing**: Distribute incoming traffic across multiple service instances.
    Tools: Nginx, HAProxy, Node.js
28. **Circuit Breaker Pattern**: Prevent cascading failures in microservices.
    Tools: opossum (Node.js library)

29. **API Gateway**: Aggregate multiple services under a single entry point.
    Tools: Express.js, Kong, Node.js
30. **Event-Driven Architecture**: Use message brokers like Kafka or RabbitMQ for communication.
    Tools: Kafka, RabbitMQ, Node.js

E-Commerce & Transactions

31. **Shopping Cart**: Build a persistent shopping cart system.
    Tools: Node.js, Express.js, MongoDB
32. **Order Management**: Handle order placement, tracking, and history.
    Tools: Node.js, MongoDB
33. **Payment Integration**: Integrate with payment gateways like Stripe or PayPal.
    Tools: Stripe API, PayPal SDK, Node.js
34. **Inventory Management**: Track product stock levels and availability.
    Tools: MongoDB, Node.js
35. **Discount & Coupon System**: Apply promotional codes and discounts at checkout.
    Tools: Node.js, MongoDB

Logging, Monitoring & Alerts

36. **Structured Logging**: Implement consistent and searchable logs.
    Tools: Winston, Bunyan, Node.js
37. **Error Tracking**: Capture and report application errors.
    Tools: Sentry, Loggly, Node.js
38. **Performance Monitoring**: Monitor application metrics and performance.
    Tools: New Relic, AppDynamics, Node.js
39. **Health Checks**: Create endpoints to check service health.
    Tools: Express.js, Node.js
40. **Alerting System**: Notify stakeholders of critical issues via email or messaging platforms.
    Tools: PagerDuty, Slack API, Node.js

Security Best Practices

41. **Input Validation**: Prevent SQL injection and XSS attacks.
    Tools: express-validator, Node.js
42. **Rate Limiting**: Protect against brute-force attacks.
    Tools: Express-rate-limit, Node.js

43. **Data Encryption**: Encrypt sensitive data at rest and in transit.
    Tools: Node.js crypto module, HTTPS
44. **Secure Headers**: Implement HTTP security headers like Content-Security-Policy.
    Tools: Helmet.js, Node.js
45. **Audit Logging**: Keep track of user actions for compliance and debugging.
    Tools: Winston, MongoDB, Node.js

Testing & CI/CD

46. **Unit Testing**: Write tests for individual functions and modules.
    Tools: Mocha, Chai, Jest, Node.js
47. **Integration Testing**: Test interactions between different parts of the application.
    Tools:

Tools to Master Along the Way:

- **Languages:** JavaScript (Node.js), Python, Go
- **Databases:** MongoDB, PostgreSQL, Redis
- **Security:** JWT, OAuth, bcrypt
- **Cloud & Infra:** AWS (S3, EC2), Docker, Kubernetes (basic)
- **Testing:** Jest, Mocha, Postman
- **Frameworks:** Express.js, FastAPI, Flask
- **Monitoring:** Prometheus, Grafana, Sentry

Top 50 Practical Backend Tasks

1. **Rate Limiter**
   Build a middleware to prevent abuse using token-bucket or sliding window algorithms.
2. **Authentication System (JWT based)**
   Sign up, Sign in, and secure access to protected routes.
3. **Authorization Middleware**
   Restrict routes to users with specific roles (admin, user).
4. **Session Management**
   Store user sessions in Redis with expiration and logout support.
5. **Password Hashing with Bcrypt**
   Store secure passwords and authenticate safely.

6. **Refresh Token Mechanism**
   Implement a secure refresh token system to renew JWT tokens.
7. **Email Verification Flow**
   Send OTP or magic link to validate user email during signup.
8. **Password Reset with Token Expiry**
   Allow users to reset their password via secure email links.
9. **Logging Middleware**
   Log all incoming requests with status, time, and payload.
10. **Pagination & Filtering API**
    Build scalable endpoints that support page, limit, sort, and filters.
11. **File Upload API (with AWS S3)**
    Upload and store media files securely in cloud.
12. **Image Resizing/Compression API**
    Resize user profile pictures or thumbnails on upload.
13. **Webhook Listener (e.g., Stripe, GitHub)**
    Receive and validate events from external services.
14. **CRUD API for MongoDB/PostgreSQL**
    RESTful endpoints to manage resources with proper status codes.
15. **Transactional Operations**
    Ensure atomicity using Mongoose transactions or SQL transactions.
16. **Realtime Chat Server (Socket.IO or WebSocket)**
    Create a basic messaging system with room management.
17. **Search Endpoint with Fuzzy Matching**
    Implement a search?q= route using regex or full-text search.
18. **Queue System (with Bull or RabbitMQ)**
    Process heavy tasks like emails, invoices asynchronously.
19. **Scheduled Jobs (e.g., cron with Node-cron)**
    Run tasks every X minutes — data cleanups, reminder emails, etc.
20. **Database Seeder & Migration System**
    Script to initialize the database with test or production data.
21. **Multi-Tenant SaaS Architecture**
    Build structure to handle multiple organizations with separate data.
22. **Role-Based Access Control (RBAC)**
    Define user roles and assign permissions for routes/actions.
23. **API Rate Limiting by IP/User**
    Add Redis-based rate limit tied to req.ip or userId.
24. **Product/Service Recommendation Engine (basic)**
    Based on user interactions, return smart suggestions.

25. **E-Commerce Cart Logic**
    Implement add-to-cart, remove, update quantity and price.

26. **Payment Gateway Integration (Stripe/PayPal)**
    Accept and verify payments with webhook response validation.

27. **Nested Commenting System**
    Build a comment-reply hierarchy with tree or linked-list model.

28. **REST to GraphQL Migration**
    Learn to write schemas, resolvers, and mutations.

29. **GitHub OAuth Login Flow**
    Integrate GitHub OAuth for social login with callback verification.

30. **Real-Time Notification System (Sockets + Redis)**
    Show live alerts, message updates, or status changes.

31. **Audit Trail System**
    Log who made what change, when, and how — for admin dashboard.

32. **File Download with Signed URLs**
    Allow secure, time-limited file download using AWS S3 signed URLs.

33. **Content Moderation Queue**
    Admin interface to approve/reject user-generated content.

34. **Internal Metrics API**
    Return analytics like number of active users, recent signups, etc.

35. **Environment-Based Config Management**
    Use .env files and dotenv to separate dev/prod settings.

36. **Error Monitoring with Sentry**
    Track and debug real errors in production.

37. **Unit + Integration Testing (Jest)**
    Write tests for controllers, services, and API endpoints.

38. **Service Health Check Route (/healthz)**
    Return service status, DB connection, memory info, etc.

39. **Global Exception Handler**
    Catch and structure all errors through centralized middleware.

40. **Auto-Scaling Sim with Load Testing**
    Use tools like k6 or artillery to simulate heavy traffic.

41. **Two-Factor Authentication (2FA)**
    Add OTP/QR-based 2FA to secure login.

42. **Upload Profile Photo with Preview + Crop**
    Implement preview + crop with frontend and backend.

43. **Soft Delete Implementation**
    Add isDeleted flag instead of actually deleting a record.

44. **Email Queue & Retry Logic**
    Add exponential backoff and retries for failed notifications.

45. **Data Caching with Redis**
    Cache frequently fetched data and invalidate on updates.

46. **API Versioning**
    Structure endpoints using /v1, /v2, etc. for long-term support.

47. **CSV Upload + Processing**
    Accept CSVs and convert to database records with error handling.

48. **Export to Excel or CSV Feature**
    Download filtered data from backend as a downloadable file.

49. **Rate Your Purchase API**
    Enable 1-5 star reviews and calculate average dynamically.

50. **User Location Tracking with Geo Queries**
    Use MongoDB's $near to find nearby users/resources.