Software Architecture for Project and Task Management Tool

Key Features

1. **User Roles:**
   - **Admin: Can create projects, tasks, assign roles, add members, and manage all settings.**
   - **Members: Can view their assigned tasks/projects, update task status, but cannot modify or delete tasks/projects.**
2. **Task Management:**
   - **Tasks have three statuses: Begin, Ongoing, and Completed.**
   - **Only assigned members can update their task statuses.**
   - **Notifications are sent to the admin and assigned members when task statuses are updated.**
3. **Authentication:**
   - **Social media login (Google, Facebook) and email/password login.**
   - **Secure password storage using encryption.**
4. **Role-Based Access Control (RBAC):**
   - **Members only see their tasks and projects.**
   - **Permissions are restricted based on roles.**
5. **Notifications:**
   - **Real-time notifications using WebSocket or AWS SNS.**
6. **Tech Stack:**
   - **Frontend: React.js**
   - **Backend: Node.js with Express.js**
   - **Database: MongoDB**
   - **Hosting and Deployment: AWS**

Admin Capabilities:

- Create, update, and delete tasks and projects.
- Assign members to tasks and projects.
- Define roles and permissions for each member.

Member Capabilities:

- View assigned tasks and projects.
- Update the status of assigned tasks (Begin, Ongoing, Completed).
- Notifications sent to both admin and related members for status changes.
- No ability to delete or edit tasks/projects.

AWS Architecture

1. Frontend:
    - Amazon S3: Store and host the React frontend.
    - Amazon CloudFront: Content delivery for faster load times globally.
2. Backend:
    - AWS EC2 or AWS Elastic Beanstalk: Host Node.js backend.
    - Amazon API Gateway: API routing for scalable communication between frontend and backend.
    - AWS Lambda (Optional): For serverless functions like sending notifications.
3. Database:
    - Amazon DocumentDB: A managed MongoDB-compatible database.
4. Authentication:
    - Amazon Cognito: Handle user login/signup with social media integration.
5. Notifications:
    - Amazon Simple Notification Service (SNS): Send email or SMS notifications.

- **Amazon Simple Queue Service (SQS): Queue notifications for reliability.**

6. **Real-Time Communication:**
    - **AWS AppSync or WebSocket API: For real-time task updates and notifications.**
7. **Monitoring and Logging:**
    - **Amazon CloudWatch: Monitor performance and logs.**
    - **AWS X-Ray: Debugging and tracing requests.**

Database Design (MongoDB)

Collections

1. **Users: (payload)**

Json data

```
{
  "_id": "unique_user_id",
  "name": "User Name",
  "email": "user@example.com",
  "role": "Admin/Member",
  "password": "encrypted_password",
  "tasks": ["task_id_1", "task_id_2"],
  "projects": ["project_id_1"]
}
```

2. **Projects: (payload)**

Json data

```
{
  "_id": "unique_project_id",
  "name": "Project Name",
  "description": "Project description",
  "tasks": ["task_id_1", "task_id_2"],
  "members": ["user_id_1", "user_id_2"]
```

}

3. Tasks: (payload)

Json data

```
{
  "_id": "unique_task_id",
  "name": "Task Name",
  "description": "Task description",
  "status": "Begin/Ongoing/Completed",
  "assigned_to": "user_id",
  "project_id": "project_id",
  "notifications": [
    {
      "status": "Ongoing",
      "timestamp": "ISO_date_time"
    }
  ]
}
```

Frontend Flow

1. React Components:
   - Login/Signup: Handle authentication.
   - Dashboard: Display projects and tasks based on user roles.
   - Task Detail: Allow members to update task status.
   - Admin Panel: Enable task/project creation and role assignments.
2. State Management:
   - Use Redux or React Context API for state management.
3. Notifications:
   - Use React-Toastify or similar libraries for UI notifications.

Backend Design

1. **Node.js Modules:**
   - **Express.js: API endpoints.**
   - **Mongoose: MongoDB interaction.**
   - **jsonwebtoken: Authentication and authorization.**
   - **bcrypt: Encrypt passwords.**
2. **API Endpoints:**
   - **User Management:**
     - **POST /signup: Create new user.**
     - **POST /login: Authenticate user.**
   - **Projects:**
     - **POST /projects: Create project (Admin only).**
     - **GET /projects/:id: Fetch project details (Admin and assigned members).**
   - **Tasks:**
     - **POST /tasks: Create task (Admin only).**
     - **PATCH /tasks/:id/status: Update task status (Assigned members only).**
   - **Notifications:**
     - **GET /notifications: Fetch notifications for a user.**
3. **Middleware:**
   - **Authentication middleware for verifying JWT tokens.**
   - **Role-based access middleware.**

Development Phases

1. **Phase 1: Build foundational backend and frontend structure.**
2. **Phase 2: Implement authentication and RBAC.**
3. **Phase 3: Create and test notification mechanisms.**
4. **Phase 4: Deploy on AWS and integrate AWS services.**

Documentation Outline

1. Technical Architecture:
    - High-level and detailed architecture diagrams.
2. Database Schema:
    - Collections and sample documents.
3. API Reference:
    - Endpoints, request/response structure, and sample calls.
4. Deployment Guide:
    - AWS setup steps and CI/CD pipelines.
5. Timelines

   Max - 4 to 5 months