````markdown name=project-requirements-todo-task-manager.md
# Basic Project Requirements: Todo Task Manager

## Overview
Build a Todo Task Manager application with:
- **Frontend**: React
- **Backend/API**: Java (Spring Boot recommended)
- **Database**: AWS DynamoDB
- **Serverless Infrastructure**: AWS Lambda for business logic, AWS API Gateway for API endpoints

---

## Functional Requirements

### Core Features
1. **User Authentication**
   - Registration, login, logout
   - JWT or Cognito integration for auth

2. **Task CRUD Operations**
   - Create, Read, Update, Delete tasks
   - Each task: title, description, due date, status (pending/completed), priority

3. **Task List & Filtering**
   - View all tasks in a list
   - Filter by status, due date, priority

4. **Mark Task as Complete**
   - Toggle task completion status

5. **Delete Completed Tasks**
   - Bulk delete option for completed tasks

6. **Task Details View**
   - View/edit details for a specific task

---

## Non-Functional Requirements

- **Responsive UI**: Works on desktop and mobile
- **Security**: Secure API endpoints, data validation
- **Scalability**: Use serverless for scaling
````

- **Performance**: Optimized DynamoDB queries
- **Error Handling**: User-friendly error messages

Technical Requirements

Frontend (React)
- React (with Hooks)
- Axios or Fetch for API requests
- Redux (optional, for state management)
- Routing (React Router)
- Basic styling (CSS/Styled Components/Material UI)

Backend (Java + AWS)
- Java (Spring Boot or AWS Lambda Java runtime)
- AWS Lambda functions for CRUD logic
- AWS API Gateway for RESTful API endpoints
- DynamoDB SDK for Java integration
- Security: JWT verification middleware (if not using Cognito)

AWS Infrastructure
- DynamoDB Table for tasks
  - Partition key: userId
  - Sort key: taskId
- API Gateway configuration for routing
- AWS Lambda permissions for DynamoDB access
- IAM roles for Lambda execution

Development & Deployment

- **Local Development**: Use AWS SAM/Serverless Framework for Lambda testing
- **CI/CD**: Deploy Lambda/API Gateway/DynamoDB via AWS CloudFormation or Terraform
- **Frontend Hosting**: AWS S3 + CloudFront (optional)

Documentation & Testing

- API documentation (Swagger/OpenAPI)
- Unit tests for Lambda functions

- End-to-end tests for frontend

Optional Features

- User profile/settings
- Reminders/notifications
- Task categories/tags
- Sharing/collaboration

Deliverables

- Source code for React frontend
- Java Lambda functions (API)
- Infrastructure as code (CloudFormation/Terraform)
- Deployment guide and setup instructions

````
````