✅ Technical Test Overview

Title: Full-Stack Task Management App with Azure Integration

🎯 Objective

Build a **Task Management Web Application** where users can:

- Register and authenticate
- Create, update, delete, and list tasks
- See task status updates in real-time
- Use filtering and sorting

🔧 Tech Stack Requirements

- **Frontend:** React + TypeScript + Bootstrap (or Tailwind)
- **Backend:** Azure Functions in C#
- **Database:** Azure SQL Server
- **Authentication:** OAuth2 or OpenID Connect (e.g., Microsoft Identity Platform)
- **API Design:** RESTful + Swagger documentation
- **DevOps:** GitHub repo, Git Flow branching, CI/CD-ready structure

---

🛠️ Features to Implement

1. User Self Registration & Login

- Implement with OAuth2/OpenID Connect (Microsoft or Google)
- Token-based auth (JWT) on frontend and backend

2. Task Management

- CRUD operations (Create/Read/Update/Delete)
- Task fields: title, description, dueDate, status, createdBy, assignedTo
- Task statuses: Pending, In Progress, Done

3. Search and Filtering

- Server-side filtering via Azure Function API
- Optional: use Azure Search (bonus)

4. Database Design

- Use Azure SQL: proper schema, indexes, and stored procedures for performance

5. Backend

- Azure Functions in C#
- Apply DI, Repository Pattern, and Factory Pattern
- Unit tests using xUnit + Moq

6. Frontend

- React with Context or Redux
- Axios or Fetch API
- Error handling, form validation, and clean UX
- Mobile responsive using Bootstrap or Tailwind CSS

7. CI/CD Ready

- Folder structure that supports:
    - Azure DevOps / GitHub Actions pipelines
    - Test automation
    - Infrastructure-as-Code (optional bonus)

📁 **GitHub Submission Requirements**

Your GitHub repository should contain:

bash

CopyEdit

/client/ ← React frontend

/api/ ← Azure Functions backend

/sql/ ← DB schema + Stored Procedures

/docs/ ← README + architecture decisions + Swagger

/tests/ ← xUnit/Moq unit tests

.github/workflows/ ← Optional: CI/CD config

📄 README Instructions

README must include:

1. **How to run locally**:
    - React app: npm install && npm run dev
    - Azure Functions: func start
    - SQL setup: instructions to create DB and run schema
2. **Auth setup** using Microsoft Identity or another provider
3. **Postman collection or Swagger URL**
4. **Architecture decisions & patterns used**
5. **What you would improve if given more time**

✅ Evaluation Criteria

Area	Criteria
Architecture	Clean separation of concerns, SOLID principles
Backend	Proper use of Azure Functions, C#, DI, async/await, security
Database	T-SQL best practices, indexes, stored procedures
Frontend	React hooks, clean components, form validation, responsive UI
Authentication	OAuth2/OpenID integration, JWT token handling
Unit Testing	Coverage and test structure using xUnit + Moq
Documentation	Clear, professional README with architecture notes
Code Quality	Naming conventions, comments, modularity, error handling
Bonus (Not Required)	Use of Azure Search, Azure API Manager, Redis, Power BI, etc.