Full Stack Engineer – Technical Assessment

RetailFixIt (Web + Azure + AI-Enabled Platform)

Purpose

This assessment evaluates a candidate's ability to design, implement, and reason about a **production-grade, AI-enabled web platform** built on Azure.

The role requires demonstration of:

- Strong frontend architecture judgment (SPA, performance, UX clarity)
- Clean API and backend design
- Event-driven Azure architecture thinking
- Safe, observable AI integration
- Production readiness (monitoring, scaling, failure handling)
- Security, RBAC, and multi-tenant isolation
- Clear engineering tradeoffs and communication

This is not an academic design exercise. We are evaluating your ability to design and ship a realistic, operable system.

Time Expectation

Use documentation, references, and tools as you would in real work.

We value clarity and sound engineering decisions over volume of code.

Assessment Structure

1. System Design & Architecture (Conceptual)
2. Hands-On Implementation (Practical)
3. Engineering Reasoning & Operational Policies (Written)
4. Platform Walkthrough (Video Recording)

Part 1: System Design & Architecture

Scenario

RetailFixIt coordinates service jobs between customers and ~1,000 vendors.

The company is building a **web-based operations platform** used by:

- Dispatchers (job orchestration)
- Vendor managers
- Admins
- Support agents

Modernization goals:

- Responsive, low-latency dashboard (SPA)
- Event-driven Azure backend

- Strong RBAC and tenant isolation
- Real-time updates across users
- AI-assisted job dispatch and insights
- Production-grade observability

Your Task

Design a **web-first, Azure-native system with AI integration** that supports:

- Real-time job dashboards
- Job assignment and lifecycle tracking
- AI-assisted vendor recommendations
- AI-generated job summaries
- Event-driven dispatch workflows
- Multi-tenant data access
- Secure RBAC enforcement
- Production reliability and monitoring

Required AI Component

Include at least one AI-enabled workflow, such as:

- AI vendor recommendation engine
- AI job complexity / escalation risk scoring
- AI-generated summary from raw customer text

- AI-assisted search over job history

You must address:

- Where AI runs (Azure OpenAI, Azure ML, containerized model)
- Synchronous vs asynchronous inference
- Latency management (timeouts, caching, circuit breakers)
- Human override model
- Logging and auditability of AI outputs
- Failure fallback behavior

We expect clear thinking about AI system boundaries and safety controls — not just API calls.

Design Requirements

Your design should include:

Data Model

- Job
- Vendor
- Assignment
- AIRecommendation
- AuditLog
- UserRole / Tenant

Architecture Components

- Web frontend (React/Vue/etc.)
- API layer
- Event backbone (Service Bus / Event Grid)
- Data store (Cosmos DB or Azure SQL)
- Cache (Redis)
- Real-time channel (SignalR)
- AI service integration
- Auth & RBAC (Azure AD / Entra ID)

Event Flow Example

JobCreated →

Persisted →

AIRecommendationRequested →

AIRecommendationGenerated →

Dispatcher reviews →

JobAssigned →

SignalR broadcast →

Audit logged

Deliverables

Provide:

- 1–2 page architecture write-up and/or diagram including:
    - Azure services used and justification

- AI integration design
- Event flow lifecycle
- Multi-tenant and RBAC model
- Observability strategy
- Scaling approach
- Key tradeoffs and risks

Evaluation Criteria

- Strong full-stack architectural clarity
- Clean separation of concerns
- Thoughtful AI safety and integration
- Real-world production awareness
- Clear reasoning and communication

Part 2: Hands-On Implementation

Implement a minimal but production-minded vertical slice.

You must demonstrate:

- Frontend
- Backend
- Event workflow

- AI integration
- Observability considerations

Required Features

Frontend (Web SPA)

- Job dashboard (paginated + filterable)
- Job detail page
- Assign job workflow
- Display AI recommendation or summary
- Real-time updates (SignalR/WebSocket)
- Optimistic UI updates
- Loading and error states

Backend

- REST or gRPC API
- JobCreated endpoint
- Assignment endpoint
- AI integration service
- Event publishing
- RBAC enforcement
- Audit logging

AI Integration

Include:

- Real AI call (or realistic mock)
- Timeout handling
- Retry or fallback strategy
- Logging of:
    - Request
    - Model response
    - Confidence / metadata (if applicable)
- Override tracking (human vs AI decision)

Event-Driven Workflow

Demonstrate at least one event flow:

JobCreated → AI → JobAssigned → Real-time update

Use:

- Service Bus or simulated event queue
- Clean JSON message schema

Observability

Describe or implement:

- Request tracing
- AI latency metrics
- Event lag metrics
- API error rates
- Override rate (AI vs human)

Infrastructure

Provide either:

- IaC (Bicep / Terraform), OR
- Clear documentation of Azure deployment setup

Include scaling assumptions.

Deliverables

Repository must include:

- Frontend code
- Backend code
- README with:
    - Architecture overview
    - Setup instructions
    - Example event flow
    - AI integration explanation

    - Performance considerations

Optional but strong signal:

- Docker support
- Structured logging
- Clean domain modeling

Part 3: Engineering Reasoning

Answer in 1–3 paragraphs each:

1. AI Autonomy & Governance
2. Performance & Latency Strategy
3. Failure Modes & Degradation
4. Multi-Tenancy & RBAC

Part 4: Platform Walkthrough (Required)

Provide a **Loom or screen-recorded walkthrough (5–10 minutes)** covering:

1. Architecture overview (high-level explanation)
2. Key frontend flows (dashboard → job detail → assignment)
3. AI feature demonstration
4. Event-driven flow explanation

5. How failures are handled (AI timeout, backend error, etc.)
6. Brief explanation of tradeoffs and design decisions

The walkthrough should demonstrate:

- Your understanding of the system you built
- How the pieces connect
- Why you made certain design choices
- Where you would improve the system next

This recording is an important part of the evaluation. Clear communication and reasoning matter.

Optional Bonus

- AI evaluation strategy (offline test harness)
- A/B rollout strategy for AI dispatch
- Chaos testing plan
- Cost optimization for high-read dashboards
- Feature flag strategy