# PCS Software- Final‑Round Interview & Hands‑On API Integration Exercise (C# / React)

Nhu Fenske<NhuFenske@PCSSoft.com>

## ​Panel

* Mark Hill, CEO
* Danielle Villegas, Chief Product Officer
* Alan Shaw, Chief Customer Officer
* Lori Marwill, Chief Financial Officer
* Julie Cummingham- VP of HR
* Peter Rhode, Sr. Director of Strategic Partnerships & Product Integration
* Brandy Minx, QA Manager and Release Coordinator
* Adam Masterson, Senior Director of Application Development
* Yoba Cabrera, Application Architect

This panel interview will revolve around a brief, hands‑on API Integration exercise designed to showcase both your technical depth and your cross‑functional collaboration style.

1. Business Context

PCS is expanding its Open API framework. Your task is to outline a thin‑slice release of an ELD (Hours‑of‑Service) connector that will feed both our Carrier and Shipper applications.

Our core stack (for reference): C# (.NET 8), React + TypeScript, PostgreSQL, Azure, Bitbucket, plus a legacy VB.NET / SQL Server data‑sync.

Preferred coding languages for this exercise: C# and React.

1. Your Exercise

Design—and be prepared to walk us through—an approach that:

Ingests HOS events from multiple third‑party ELD vendors.

Normalizes those events into our current Open API 3.1 format.

Exposes a secure REST endpoint implemented in C# (.NET 8 Web API).

Demonstrates a lightweight React client that calls the endpoint and displays a driver’s current HOS status.

Addresses how you would integrate or migrate any needed data from our legacy VB.NET / SQL Server system.

The solution does not need to be fully production‑ready, but the code and artifacts you provide should illustrate your engineering approach, design rationale, and collaboration mindset.

1. Pre‑Work Deliverables

Please email the materials below no later than 24 hours before your panel (a PDF or shared link is fine).

### Deliverable

What We Expect

#### Solution overview (1–2 pages)

* High‑level architecture diagram (ingestion → normalization → persistence → API → UI)
* Key C# layers/services & data‑flow description
* React component tree & state flow

#### Representative code snippets

* openapi.yaml fragment for GET /drivers/{id}/hos
* C# controller or minimal‑API sample returning stub data
* React <HosStatusCard> with loading / success / error states

#### Data‑handling note

How you’d validate, reconcile, and back‑fill inconsistent or missing HOS data from different ELD vendors.

#### Risks / open questions

Top unknowns (e.g., vendor SLAs, rate limits, UI latency, legacy data gaps).