**Logging Research – Project J**

**Is Logging available out of the box?**

Frameworks such as **Serilog, NLog, or Microsoft.Extensions.Logging** provide standard logging functionalities without extensive setup. They support basic logging (DEBUG, INFO, WARN, ERROR) with minimal configuration, allowing for immediate implementation in our app.

**Centralized Log Management and Integration**:

The framework of choice should support integration with centralized log solutions, in our case this could be **Azure Monitor**. This helps in log aggregation, search, and visualization. All 3 of the above-mentioned logging frameworks support Azure Monitor.

**Centralized Logging vs. Local Logging**:

We need to decide whether to implement centralized logging (**Azure Monitor**) or rely on local logs stored within each service instance.

Centralized logging makes monitoring and debugging easier but often incurs additional setup, maintenance, and costs.

**I believe that we should begin our project using local logging and expand to get insights from Azure Monitor later.**

**Security and Data Compliance**:

We will have to look for support for secure logging practices, like sensitive data masking and log encryption. This is crucial if logs include user data or other sensitive information.

The framework we decide on should help us filter or anonymize sensitive information.

**Will we need Alerts/tracking?**

Check if our selected framework integrates with error tracking systems or alerting platforms (**Azure Monitor**) to notify the team of critical issues.

**Format and Structure of Logs**:

Deciding on a structured logging format (JSON vs. plain text) is another consideration. Structured logs (like JSON) are easier for log aggregation tools to parse but may increase complexity.

Consistency in log structure is crucial, so the team may need to standardize what information is included in each log entry (e.g., timestamp, user ID, error codes).