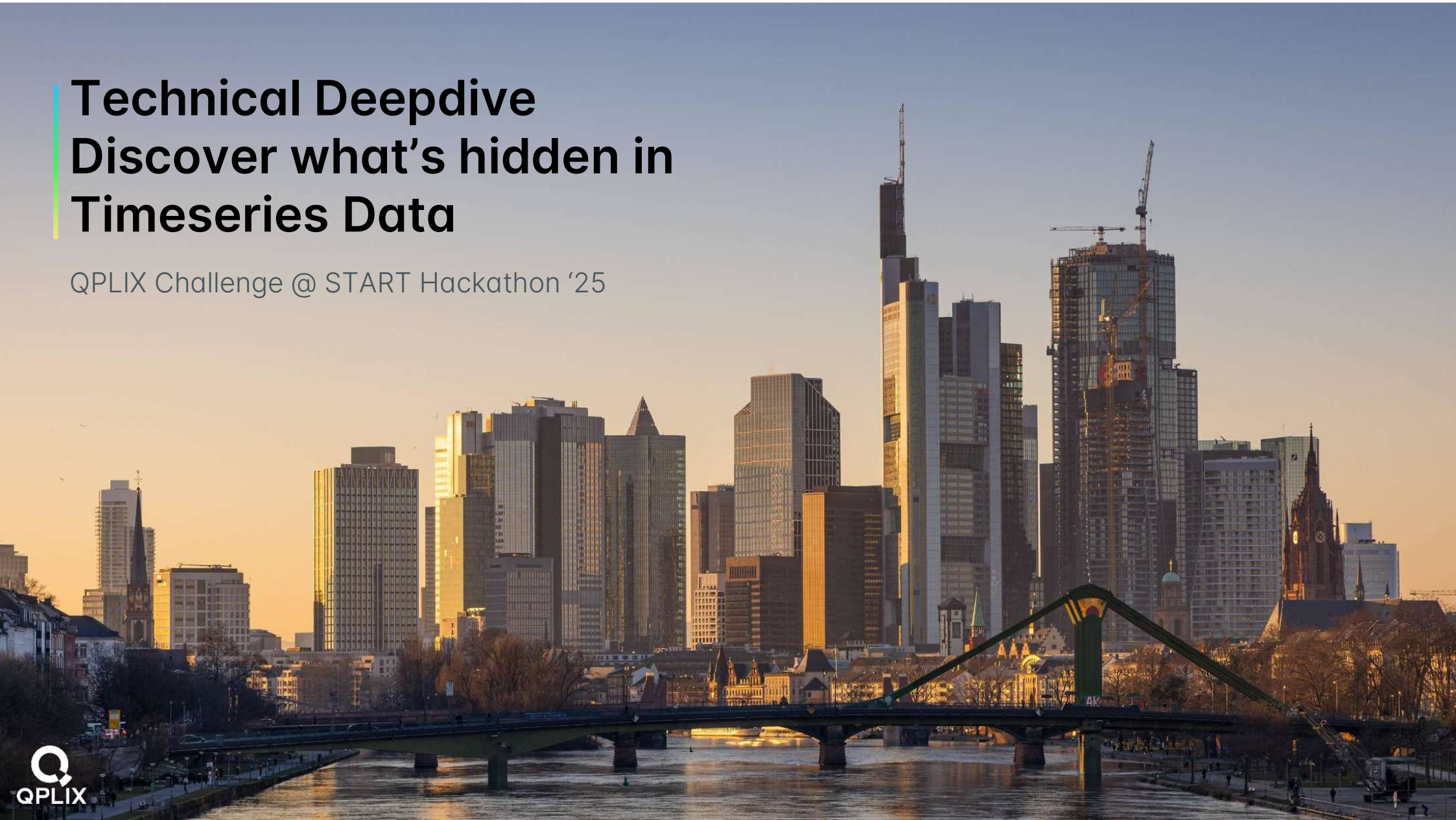


# Technical Deepdive

## Discover what's hidden in Timeseries Data

QPLIX Challenge @ START Hackathon '25



# QPLIX Material



Google Drive with  
Example Projects



Discord Server

# # Provided Example Projects

- Node.js
- Python
- C#

→ All include authentication workflow + querying for evaluation result

→ C# implementation also deserializes the result and outputs the values for the whole evaluation, including headers and different value formats

# # C# Implementation

- Understanding API Authentication Flow
- Navigating Hierarchical Result Structures
- Deserializing Complex Financial Data
- Building Reusable Display Components

# Architecture Overview - C#

## **Program.cs**

Entry point & orchestration

## **DataModels.cs**

Type definitions & contracts

## **MagicPrintFunction.cs**

Result visualization

## **Data Flow**

Authentication → API Call → Deserialization → Display

# Authentication Flow

## 1. F5 Bearer Token

```
httpClient.DefaultRequestHeaders.Authorization =  
    new AuthenticationHeaderValue("Bearer", F5Bearer);
```

Infrastructure-level authentication

## 2. User Credentials

```
var body = $"grant_type=password&username={QUserUsername}&password={QUserPassword}";
```

OAuth2 password grant flow

## 3. Q Bearer Token

```
var tokenData = await JsonSerializer.DeserializeAsync<TokenResponse>(...);  
var qBearer = tokenData?.AccessToken;
```

Application-level access token

## 4. Combined Authorization

```
var tokens = $"{{F5Bearer}, {qBearer}}";  
httpClient.DefaultRequestHeaders.Authorization =  
    new AuthenticationHeaderValue("Bearer", tokens);
```

Multi-layer security model

# Why Two Bearer Token

## F5 Bearer Token

- Infrastructure/load balancer authentication
- Network-level access control
- Environment-specific  
(smd43 vs smd44)

## Q Bearer Token

- User-specific authorization
- Obtained via username/password
- Controls access to specific evaluations
- Required for all API operations

# Data Models: The Contract

## QueryResultMatrix

Level 1

ResultLine Headers SubHeaders VisualizationHeaders

Top-level container for entire result set

## QueryResultLine

Level 2

Name Values SubLines Visualizations

Hierarchical node with recursive structure

## QueryResultValue

Level 3

Type RawValue Value SubValues

Typed value with formatting and metadata

# Understanding ResultValueType

## Simple Types

Amount Money Percentage Ratio Text Date Boolean

## Time Series Types

AmountTimeSeries MoneyTimeSeries PercentageTimeSeries RatioTimeSeries

## Complex Types

Classification MoneyExposure WeightedEnum Period

### ⚠ Important

Time series types require special handling - they contain Dictionary<string, decimal> in RawValue

# Navigating the Result Tree

```
// Top level result
result.ResultLine.Name           // Entity name
result.ResultLine.Values[3]      // 4th column value

// Access child results
result.ResultLine.SubLines[0]    // First child
result.ResultLine.SubLines[0].Values[3] // Child's value

// Headers provide context
result.Headers.ElementAt(3)       // Column header
result.SubHeaders.ElementAt(3).FirstOrDefault() // Sub-header
```

## Conceptual Structure

Matrix → ResultLine → Values[] & SubLines[]

# The Magic Print Function as an Example

Renders hierarchical financial data as formatted ASCII table

`GetMaxLengthPerColumn()`


Calculate column widths based on content + depth

`WriteTreeTable()`

Recursively render tree structure with proper indentation

`MaxDepth / MaxLengthNames()`

Pre-calculate formatting dimensions

 **Key Challenge**

Handling time series data stored as `JsonElement`

# API Endpoint Structure

Pattern

```
/qapi/v1/evaluation/preset/{presetId}/legalEntity/{entityId}
```

Example

```
/qapi/v1/evaluation/preset/691dd5953022610895c1aeff/legalEntity/5cb71a8b2c94de98b02aff19
```

**presetId**

Defines which evaluation template to run

```
691dd5953022610895c1aeff
```

**entityId**

Target legal entity for evaluation

```
5cb71a8b2c94de98b02aff19
```

Three alternative evaluation presets provided in comments

# Key Takeaways

## Technical Insights

- Dual authentication layer provides security depth
- Recursive data structures require recursive processing
- Type discrimination enables flexible value handling
- Pre-calculation optimizes display formatting

## Practical Advice

- Use `System.Text.Json` for modern APIs
- Keep `Newtonsoft.Json` for complex scenarios
- Plan for hierarchical result navigation
- Consider visualization requirements early

# Judging Criteria

**1. Insight Discovery (4 points)**

**2. Technical Execution (4 points)**

**3. Visualization & Communication (4 points)**

**4. Data Integration (2 points)**

**5. Presentation & Delivery (2 points)**

# Judging Criteria

## 1. Insight Discovery (4 points)

- "Wow factor": Does this reveal something genuinely surprising or non-obvious?
- Originality: Have you found a unique angle or perspective on the data?
- Depth: Goes beyond surface-level patterns to uncover meaningful relationships
- Breadth: Effectively leverages multiple data dimensions or asset classes

# Judging Criteria

## 2. Technical Execution (4 points)

- Analytical rigor: Sound methodology and appropriate techniques
- Code quality: Clean, well-structured, and documented implementation
- Data processing: Effective handling of complex, multi-dimensional datasets
- Technical innovation: Creative algorithms, novel approaches, or sophisticated methods
- Performance: Efficient execution even with large datasets
- Completeness & Reliability: Fully functional with intended features working correctly and graceful handling of edge cases

# Judging Criteria

## 3. Visualization & Communication (4 points)

- Clarity: Complex patterns made immediately understandable
- Design excellence: Polished, professional presentation
- Interactivity: Engaging interface that facilitates exploration and discovery
- Narrative flow: Compelling story that guides the audience through your findings
- Visual innovation: Novel or particularly effective ways of representing data

# Judging Criteria

## 4. Data Integration (2 points)

- Multi-source synthesis: Effective combination of market data, portfolio data, and/or external sources
- Cross-asset insights: Meaningful connections between different asset classes
- Temporal analysis: Insightful use of time-series patterns and historical context
- External enrichment: Integration with news, events, or other contextual data

# Judging Criteria

## 5. Presentation & Delivery (2 points)

- Professionalism: Delivered in a polished and confident manner
- Pitch: Just have a good pitch ;)

# Material



Google Drive with  
Example Projects



Discord Server