Asynch Technical Screening presentation for hirefraction.com

By Drew Iorio

Agenda

- Assignment
- High Level Overview
- Architectural Design & Choices
- Tools Used
- Demo
- Code Review

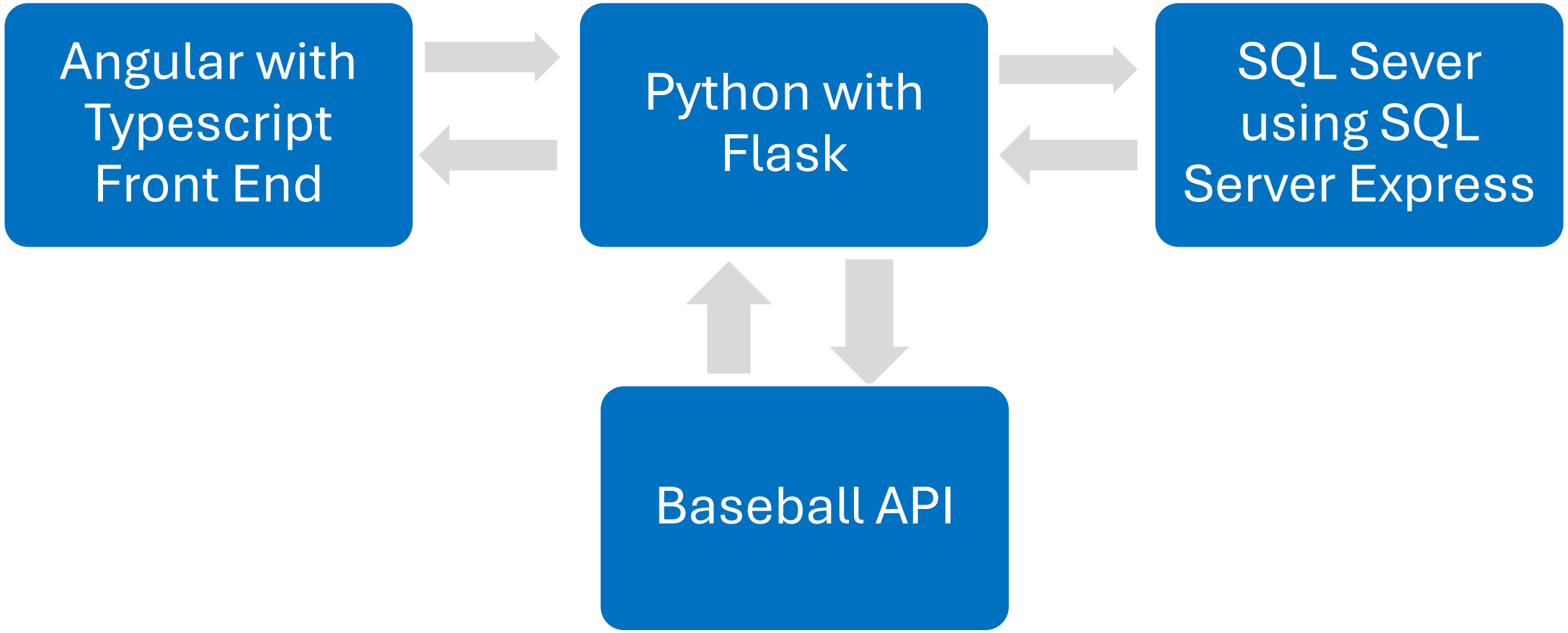
Assignment

1. **Front End (Angular)**
   - Initialize React app.
  - Create components: `PlayerList`, `PlayerDetail`, `EditPlayer`.
 - Fetch player data from the server.
 - Display player list with stats.
 - Handle click events to show player details.
 - Implement edit functionality.

2. **Server Side (Python)**
- Set up routes: `/players`, `/players/:id`.
- Fetch data from the baseball API.
- Correct missing data on ranking.
- Implement CRUD operations.

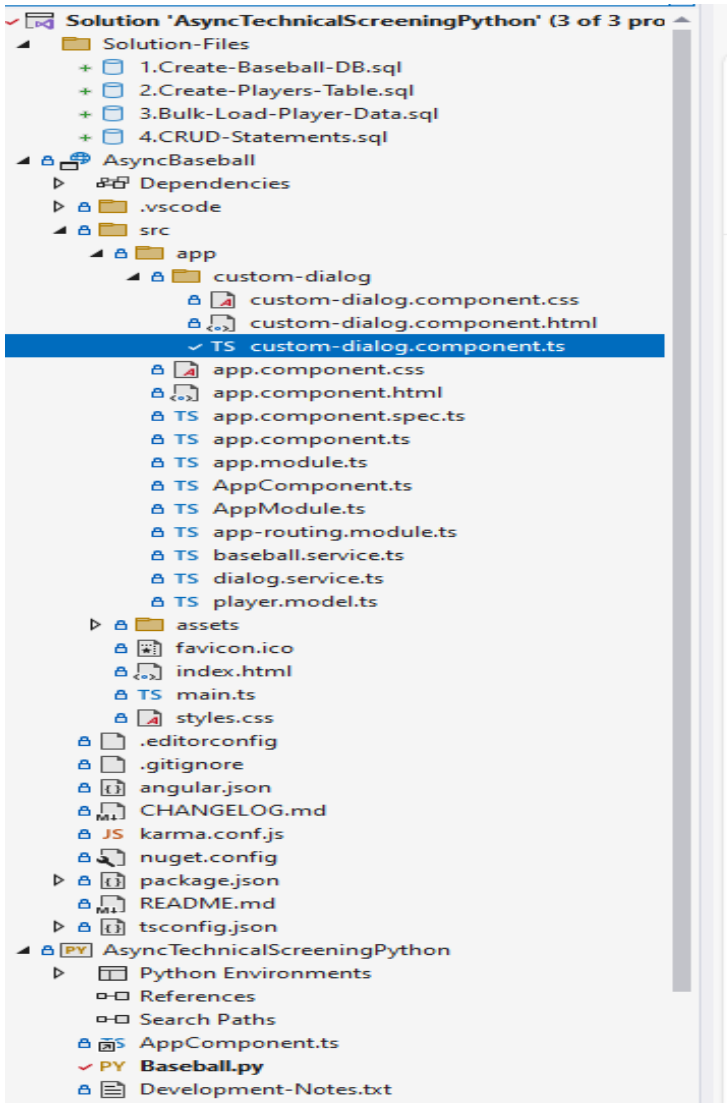
3. **Database (SQL Server)**
- Set up SQL Server connection.
- Create a schema for player data.
- - Implement data storage and retrieval.

High Level Overview



Architectural Design

- Front End
  - Angular MVVM Design
- Server
  - Python using flask for a lightweight web service
- Database
  - SQL Server
- Utility
  - C# Command line Utility using .net core 8 for Bulk loading of baseball data



Tools Used

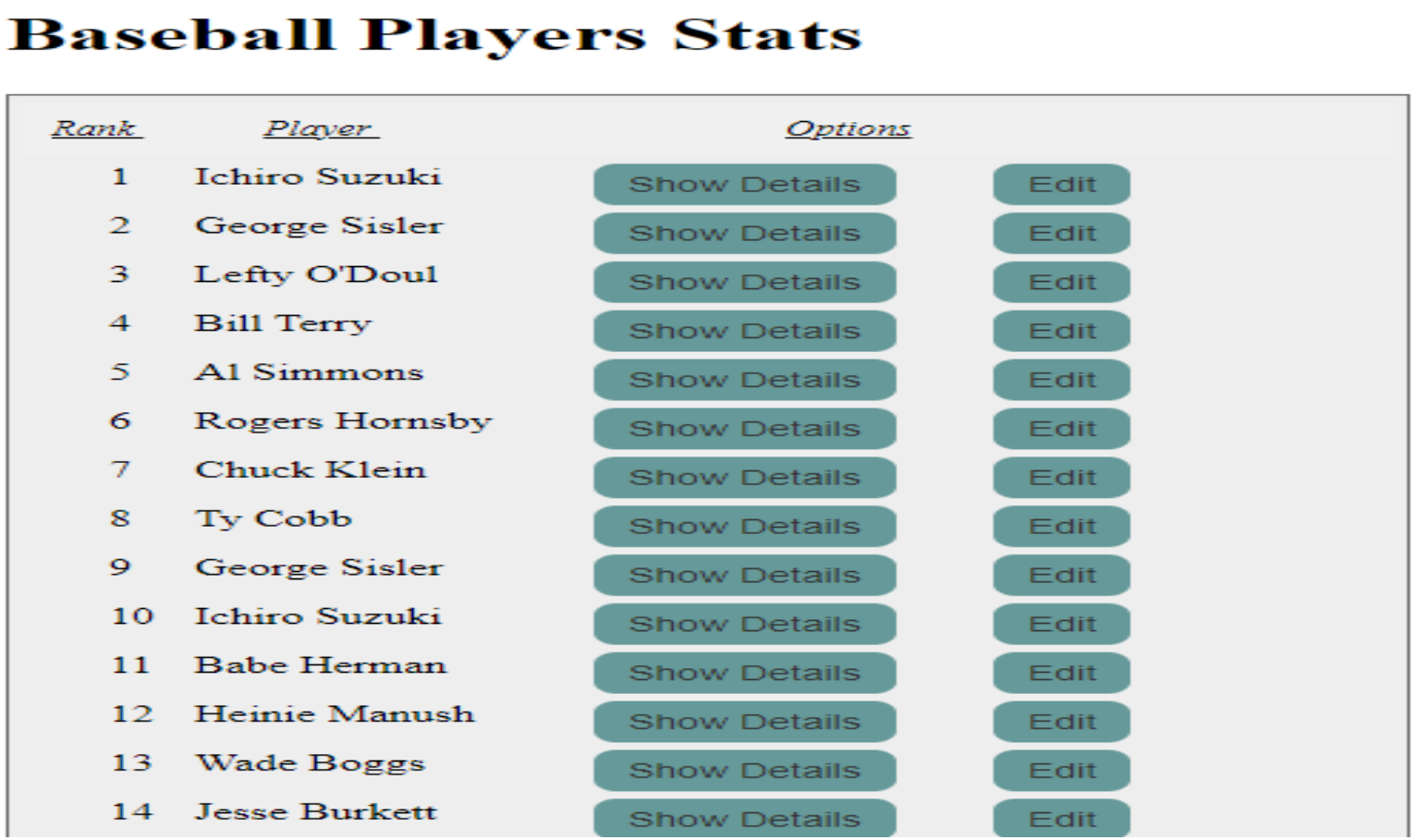
- Visual Studio Community 2022
- SQL Server Management Studio
- Github
  - GitHub - DrewIorio1/AsyncTechnicalScreeningPython
- Loom

Product Layouts

- AsyncTechnicalScreening
  - Top level Solution file
- AsyncBaseball
  - Angular – Web Client
- AsynchTechnicalScreeningPython
  - Used for call and loading data from the database api or sql server
- AsynUtlity
  - C# Command line utility used for bulk loading

Demo

Baseball Demo



Code Review Angular

- All code is under AsyncBaseball\src
  - Code initialed in the main.ts file
  - All code is designed to have a single use
  - 2 Main Components used
    - App.Component.html
    - \custom-dialog\custom-dialog.component.html
      - Handles the modal pop

```
/**
 * custom-dialog.compant - popup class to avoid binding errors during angular/material/dialog
 */


import { Component, EventEmitter, Input, Output, OnInit } from '@angular/core';
import { Player } from '../player.model';
import { Observable } from 'rxjs';
import { FormsModule } from '@angular/forms';
//import { FormGroup } from '@angular/forms';
@Component({
  selector: 'app-custom-dialog',
  templateUrl: './custom-dialog.component.html',
  styleUrls: ['./custom-dialog.component.css']
})
18 references
export class CustomDialogComponent implements OnInit {
  @Input() data: any;
  @Input() isEditMode: boolean = false;
  @Output() close = new EventEmitter<void>();
  @Output() save = new EventEmitter<Player>();

  specialValue: any

  ngOnInit() {
    //TODO : Implement 2 way binding
  }

  /**
   *  Close the current user
   */
  onClose() {
    this.close.emit();
  }

  /**
   *
   * @param Year - Year the player achived the rank
   * @param Age  - Age of the player
   * @param Hits - Total number of hits for the player in a given year
   * @param Bats - If the player batted left or right handed
   */
  onSave(Year: string, Age: string, Hits: string, Bats : string): void {
    // Implement save functionality here
    //let year =  Number.parseInt(Year) ;
    const updatedPlayer: Player = {
        Player: this.data.Player,
        Rank: this.data.Rank,
      AgeThatYear: Number.parseInt(Age),
      Hits: Number.parseInt( Hits),
      Year: Number.parseInt(Year),
      Bats: Bats,
        id: this.data.id
    };
    this.save.emit(updatedPlayer);
    this.onClose();
  }
}
```

AsyncTechnicalScreeningPython

- Python Webserver
  - Uses Python to call https://api.sampleapis.com/baseball/hitsSingleSeason
- Demonstration of Calling the Baseball API and Correcting Rankings for Display

```python
from flask import Flask, jsonify
from flask_cors import CORS
import requests
import pyodbc
# Create an instance of the Flask class that is the WSGI application.

app = Flask(__name__)


# Cors function to allow for client calls
CORS(app, resources={r"/api/*": {"origins": "http://localhost:4200"}})
# Flask route decorators map / and /hello to the hello function.
# To add other resources, create functions that generate the page contents
# and add decorators to define the appropriate resource locators for them.



API_URL = 'https://api.sampleapis.com/baseball/hitsSingleSeason'

# Fetch all players
@app.route('/api/players', methods=['GET'])
def get_players():
    response = requests.get(API_URL)
    data = response.json()
    # Check if the rank is blank in the list
    for index, player in enumerate(data):
        if not player.get('Rank'):
            player['Rank'] = str(index + 1)

    return jsonify(data)
```

Code Review

- AsynUtility
  - Uses C# .net core 8 to call to the ḩţţƥș ǻřî șǻŋřľêǻřîș çôŋ čǻșêčǻľľ ḩîţșŞîŋĝľêŞêǻșôŋ
    - With the assumption no data is loaded

```
/*
 * Class - Main control function of the Asynch Utility to grab data and assign to the database.
 *
 */
using AsynUtility;
using System.Text.Json;

0 references
class Program
{
    0 references
    static async Task Main(string[] args)
    {
        var players = await FetchPlayersFromApi();

        Console.Write($"Number of players {players?.Count}");
        //AddPlayersToDatabase(players);
        for (int i = 0; i < players.Count; i++)
        {
            if (players[i].Rank == 0)
                players[i].Rank = (i + 1);
        }

        AddPlayersToDatabase(players);

    }

    //
    // Helper function to add all the players from the API to the database
    //
    1 reference
    static void AddPlayersToDatabase(List<Player> players)
    {
        using var context = new BaseballContext();
        context.Players.AddRange(players);
        context.SaveChanges();
        Console.WriteLine("Players added to the database.");
    }

    //
    // Helper function to fetch all players from the sampleapi data api
    //
    1 reference
    static async Task<List<Player>> FetchPlayersFromApi()
    {
        var url = "https://api.sampleapis.com/baseball/hitsSingleSeason";
        using var client = new HttpClient();
        var response = await client.GetStringAsync(url);
        var options = new JsonSerializerOptions
        {
            PropertyNameCaseInsensitive = true,
            Converters = { new NullableIntConverter() }
        };
        return JsonSerializer.Deserialize<List<Player>>(response, options);
    }

}
```

0 2