Yo Yo Test App
Technical Specifications

VERSION: 1.0

27 OCTOBER 2020

Prepared by: Georgekutty George

Table of Contents

1.PROJECT OVERVIEW

In a YoYo test, athletes run between two set of cones placed 20 m apart. Athletes start running when a timer starts at a specific time according to a schema. Normally they will hear a beep which indicates the start of the timer. Athletes need to get behind the cone they are running towards before a certain time. If they don't make it, they will get a warning and if they miss a second time they are out of the test. Test results can be calculated automatically because all distances and times are included in a schema. Result will be shown as [Speed level – shuttle number], ex. 14-3. The schema will have different levels and respective shuttles; and each level and shuttle will get more difficult than the one before, as there will be less time to reach the cones. So over a period of time, the test will be harder to do and therefore a good way for a coach to measure an athlete's fitness level.

2.Technical Contribution

- The Application is developed in **.Net Core 3.1** Framework with MVC and RazorPages, following a Multitier architecture.
- The application is compatible with both Web and mobile devices.
- The application is divided into 8 layers strictly following dependency injection and SOLID principles.
- Exception handling is done globally.
- Logging is implemented using **Serilog** and all the exceptions and logs are written to log.text file.
- **AutoMapper** is used for mapping the Entity to Model.
- Unit Test is written for BAL layer using **xUnit** framework.
- Dapper Repository is added as an ORM in DataAccess layer.
- RegisterServices is separated from Startup.cs and created as a separate class library to avoid tightly coupled project.
- Proper commenting, regions are added for better readability of code.
- UI design is done using **bootstrap** and custom **CSS**. Flux design is used.
- Client side scripting is done in **JavaScript/jQuery** libraries.
- Partial Views are used for Displaying the Athletes list and Fitness Test Data.
- Toaster is added if user warn/stop an athlete.

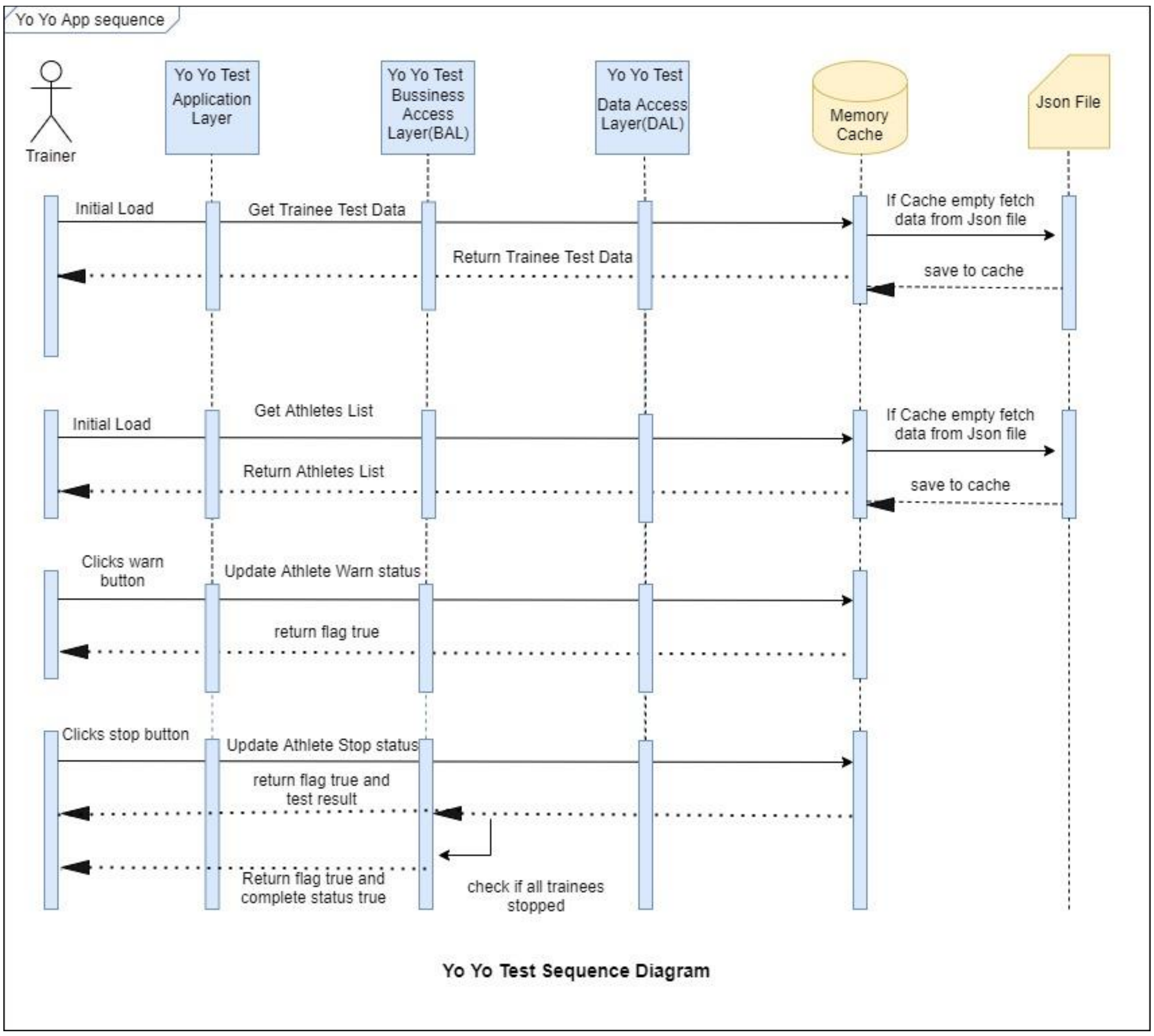
3.MULTI LAYER ARCHITECTURE

Below are the different layers of the Project:

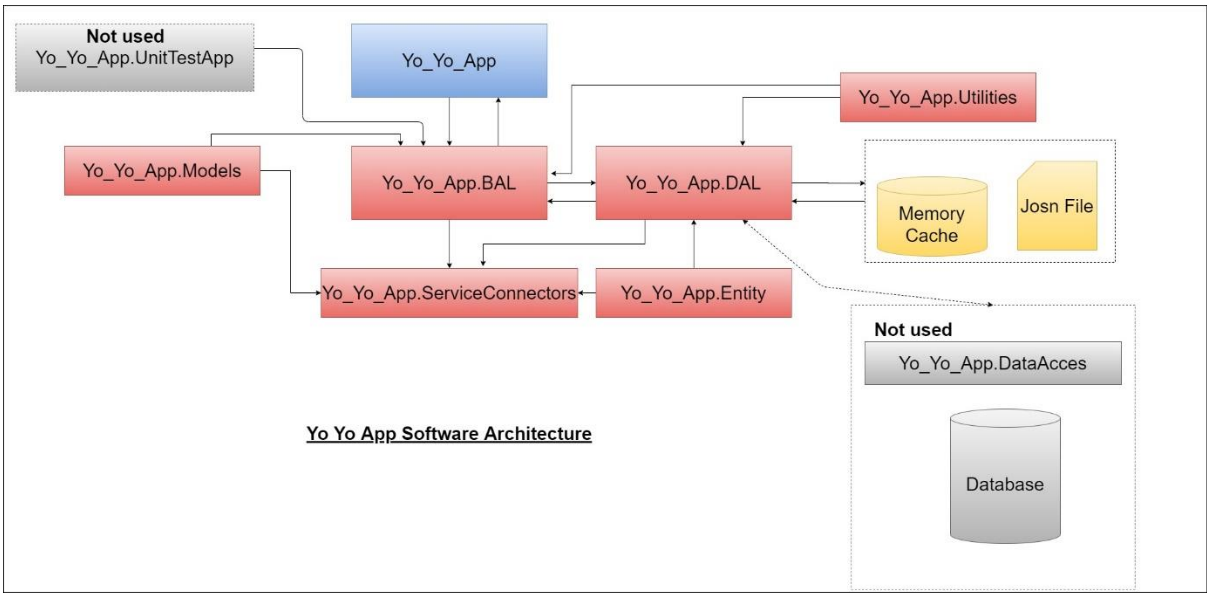
- **Yo_Yo_App** - Contains the Controller, HTML pages. It is the start of the application.
- **Yo_Yo_App.BAL** - All the Business logics are done in this layer.
- **Yo_Yo_App.DAL**- All the Database related operations are handled in this layer.
- **Yo_Yo_App.DataAcces** - This layer is not used as a part of this project but implemented Dapper Repository for future reference.
- **Yo_Yo_App.Entity** - All the database result sets are mapped to entity classes.
- **Yo_Yo_App.Models** - Model contains data to be showed in View.
- **Yo_Yo_App.ServiceConnectors** - Mapping the interface with concrete class is done in this layer. The class library is then referred in ConfigureServces method in Startup.cs.

- **Yo_Yo_App.Utilities** - All the common methods irrespective of project is added here as static methods. Example - ConvertTimeToSecond().
- **Yo_Yo_App.UnitTestApp** - This Project is currently not used as it is not coming under the current scope. Unit Test is being written for methods in BAL layer. The project is unloaded now.

4.SEQUECNCE DIAGRAM


Yo Yo App sequence
Trainer
Yo Yo Test Application Layer
Yo Yo Test Bussiness Access Layer(BAL)
Yo Yo Test Data Access Layer(DAL)
Memory Cache
Json File
Initial Load
Get Trainee Test Data
If Cache empty fetch data from Json file
Return Trainee Test Data
save to cache
Initial Load
Get Athletes List
If Cache empty fetch data from Json file
Return Athletes List
save to cache
Clicks warn button
Update Athlete Warn status
return flag true
Clicks stop button
Update Athlete Stop status
return flag true and test result
Return flag true and complete status true
check if all trainees stopped
Yo Yo Test Sequence Diagram

5.SOFTWARE ARCHITECTURE DIAGRAM



6.METHODS USED

These are the methods called form controller.

1. **GetFitnessRatingTestData**
   Method to retrieve next Fitness Rating Test Data based on Level and Shuttle. Method called when current shuttle ends.
2. **GetAllAthletes**
   Method to retrieve athletes list. Method called in initial load.
3. **GetAllAthletesCompleteResult**
4. Method to retrieve all athletes complete result list. Method called when test completes.
5. **UpdateAthleteStopStatus**
   Method to update individual athlete stop status and returns the test result for the particular athlete. Method called when user clicks on “Stop” button
6. **UpdateAthleteWarnStatus**
   Method to Update individual athlete warn status. Method called when user clicks on “Warn” button

7.ALGORITHMS

Method Name – **GetCurrentTrackDetailsByLevelAndShuttle**

```
START
    STEP 1 : Get all FitnessRating test data from Jsonfile/Cache memory and save to DATALIST.
    STEP 2 : Find the maximum level from the list and save to MAXLEVEL.
    STEP 3 : IF SHUTTLE==8
    STEP 4 : BEGIN
                SHUTTLE=0,
                LEVEL=LEVEL+1
    STEP 5 : END
    STEP 6 : ELSE
    STEP 7 : BEGIN
                SHUTTLE= SHUTTLE+ 1,
    STEP 8 : END
    STEP 9 : LOOP DATALIST
    STEP 10 : BEGIN
                Get the fitness test data from the DATALIST by LEVEL and SHUTTLE.
                IF Result is NULL
                BEGIN
                    IF SHUTTLE==8
                    BEGIN
                        SHUTTLE=0,
                        LEVEL=LEVEL+1
                    END
                    ELSE
                    BEGIN
                        SHUTTLE= SHUTTLE+ 1,
                    END
                    GO TO STEP 10
                END
    STEP 11 : END
    STEP 12 : RETURN Result.
STOP
```