**This is a live project-based coding question. Read the task description below and solve using the VS Code IDE provided alongside:**

**Task description:**

This task is to create an Angular application that allows users to create a cricket team by selecting players from a list.

The application allows users to create a team by choosing from a list of players. The users can also perform the following operations:

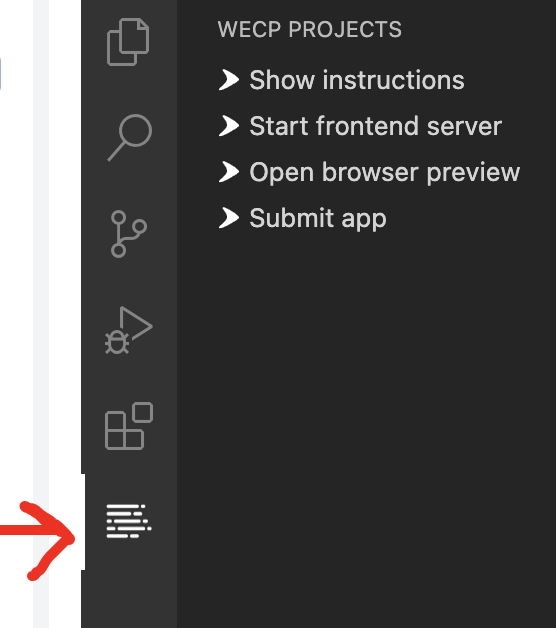
* If the user cannot find a required player in the given list, they can add a new player to the list and use that player to form the team
* Users can change the role of an already existing player (Batsman, Bowler, Wicket Keeper)
* Users can delete a player from the players' list

**Important notes:**

1. Click on the Explorer button on the VS Code Side toolbar to access the project directory.
2. Any changes made to a player in the players' list, must reflect in the team's list.
3. A team must have **exactly 2 Batsmen, 2 Bowlers, and 1 Wicket Keeper.**
4. When the number of batsmen in the team is not exactly 2, the application must throw the message *"You do not have the required number of batsmen in your team"*
5. When the number of bowlers in the team is not exactly 2, the application must throw the message *"You do not have the required number of bowler in your team"*
6. When the number of wicket-keeper in the team is not exactly 1, the application must prompt the message "You do not have the required number of wicket-keeper in your team"
7. You can look into **src/app/players/player.model.ts** to understand the player's model used across the app.
8. Most of the code for the application is already written. You need to complete only the following two files: **src/app/players/player.service.ts** and **src/app/team/team.service.ts** to successfully build this application
9. There is no need to write code to style this application. This is already written.
10. Do not make any changes in other files except other than the **src/app/players/player.service.ts** and **src/app/team/team.service.ts**
11. You must pass all the test cases in order to get marks on this question

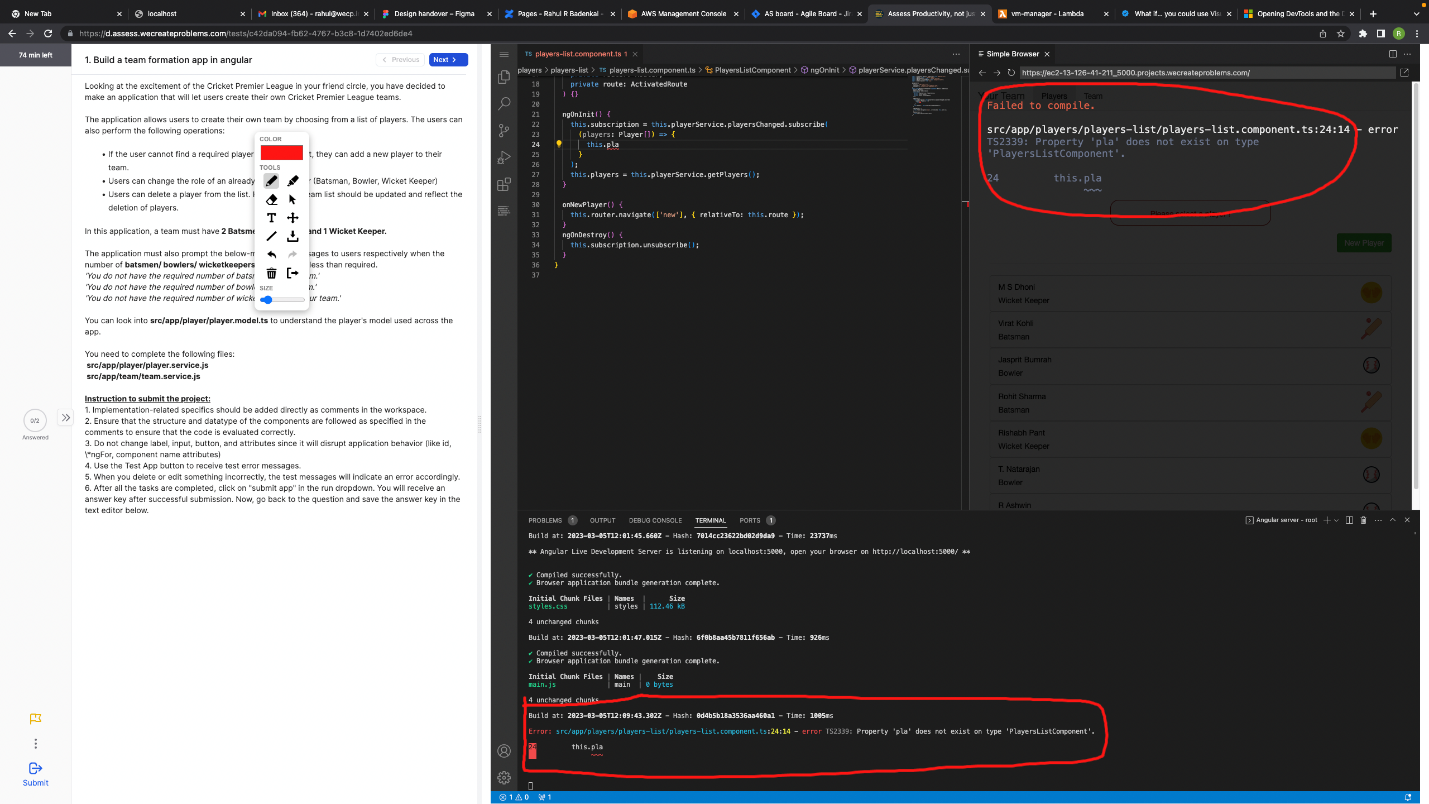
**Testing & Submitting your code:**

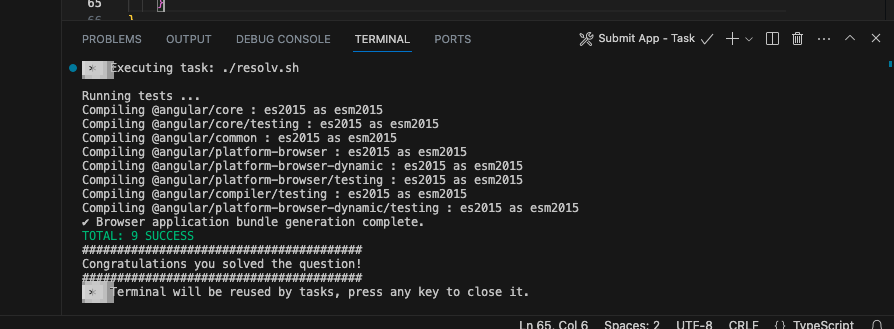
**Step 1:** Click on the **WeCP Projects Button** as shown below and click to **Start frontend server** to start the angular server.



**Step 2:** Click on the **Open browser preview** to test and preview your angular application inside the VS Code Browser.

**Step 3**: Write your code to complete the task(s) as asked in the question. Click on the **Test and Submit App** to execute your code and confirm if the application is working as expected. If your code has errors, it will appear on the browser preview as well as in the terminal as shown in the below display.



**Step 4:** You can check your result in the terminal after you click on **Test and Submit App.﻿** 

**Team.service.ts**

import { Player } from '../players/player.model';

import { EventEmitter } from '@angular/core';

export class TeamService {

teamChanged = new EventEmitter<Player[]>();

playerId: number;

private team: Player[] = [];

getTeam() {

return this.team.slice();

}

/\*

Should edit a player of given id to nwPlayer

\*/

editPlayer(id: number, nwPlayer: Player) {

const pi=this.team.findIndex(player=>player.id===id);

this.team[pi]=nwPlayer;

this.teamChanged.next(this.getTeam());

}

/\*

Should add a player to team, if the player

already exist alert "This Player already exist in your team !!"

\*/

addPlayer(player: Player) {

const pi=this.team.find(player1=>player1.id===player.id);

if(pi){

alert('This Player already exist in your team !!');

}

else{

this.team.push(player);

this.teamChanged.next(this.getTeam());

}

}

/\*

Should delete a player from team

\*/

deletePlayer(id: number) {

const pi=this.team.findIndex(player=>player.id===id);

this.team.splice(pi,1);

this.teamChanged.next(this.getTeam());

}

/\*

Returns error messages if a team does not have exactly 2 batsmen

2 bowlers and 1 wicket-keeper

\*/

getStatus() {

var status: string[] = [];

let c1=0,c2=0,c3=0;

this.team.forEach(player=>{

if(player.description==='Batsman')

c1++;

if(player.description==='Bowler')

c2++;

if(player.description==='Wicket Keeper')

c3++;

});

if(c1<2)

{

status.push('You do not have required number of batsmen in your team');

}

if(c2<2){

status.push('You do not have required number of bowler in your team');

}

if(c3<1){

status.push('You do not have required number of wicket-keeper in your team');

}

return status;

}

}

**Player.service.ts**

import { Player } from './player.model';

import { Injectable } from '@angular/core';

import { Subject } from 'rxjs';

@Injectable()

export class PlayerService {

playersChanged = new Subject<Player[]>();

count: number = 7;

private players: Player[] = [

new Player(1, 'M S Dhoni', 'Wicket Keeper'),

new Player(2, 'Virat Kohli', 'Batsman'),

new Player(3, 'Jasprit Bumrah', 'Bowler'),

new Player(4, 'Rohit Sharma', 'Batsman'),

new Player(5, 'Rishabh Pant', 'Wicket Keeper'),

new Player(6, 'T. Natarajan', 'Bowler'),

new Player(7, 'R Ashwin', 'Bowler'),

];

getPlayers() {

return this.players.slice();

}

generateID() {

this.count = this.count + 1;

return this.count;

}

/\*

Returns the player for a given player id

\*/

getPlayer(id: number) {

const pi=this.players.find(player=>player.id===id);

return pi;

}

/\*

Adds the player to player list

\*/

addPlayer(player: Player) {

let x=this.generateId();

const pi=this.players.find(playr=>playr.id===player.id);

this.players.push(player);

this.playersChanged.next(this.getPlayers());

}

/\*

Updates the player given the index and new changed values

\*/

updatePlayer(index: number, newPlayer: Player) {

const pi=this.players.findIndex(player=>player.id===index);

this.players[pi]=newPlayer;

this.playersChanged.next(this.getPlayers());

}

/\*

Deletes a player from player list

\*/

deletePlayer(index: number) {

const pi=this.players.findIndex(player=>player.id===index);

this.players.splice(pi,1);

this.playersChanged.next(this.getPlayers());

}

}

**Build a Sales Dashboard application using Angular**

You are asked to build a Sales Dashboard application using Angular.

In this application a sale data entity consists of three fields; **saleTotal**, **buyerName** and **creditCard**. You will find the data in **sales.json** file in the assets folder.

The application requires four dashboard reports. Implement relevant calculations in **reports.service.ts** and use them in **app.component.ts** file.

1. Total Sale Report: Implement **calculateTotalSales** in **reports.service.ts**. You must return the sum of all sales.

2. Total Cash Sale Report: Implement **calculateTotalCashSale** function in **reports.service.ts**. You must return the sum of all sales where creditCard attribute is false.

3. Total Credit Card Sale Report: Implement **calculateTotalCreditSale** function in **reports.service.ts**. You must return the sum of all sales where creditCard attribute is true.

4. Buyer With Most Sale: Implement **calculateBuyerWithMostSale** function in **reports.service.ts**. You must return the buyer with the most saleTotal. Thus, if Loron Chaun is the person with the most purchased items, return Loron Chaun and sum of the purchases he made. Return format must be **{"buyerName": "Loron Chaun", "saleTotal": 34000}**. The **sales.json** may or may not contain above names and figures. The above example is just to demonstrate the expected data format.

**Your task is to complete the function defined in the file below:**

1. **src/app/app.component.ts**

2. **src/app/reports.service.ts**

3. **src/app/sales.interface.ts**

**Notes:**

1. It is given that each report has a different implementation, but each function will require the sale data. Use **getSalesData** function in **reports.service.ts** to fetch sales data. The **getSalesData** function is already implemented for you.

2. Do not change file names, class names , method declarations.

3. Use **Test App & Submit** option often so you will be guided by test error messages.

**Testing & Submitting your code:**

**Step 1:** Click on the WeCP Projects Button.

A screenshot of a computer

Description automatically generated

**Step 2:** Click on **Test & Submit** app button shown above to test your code.

**Step 3:** Click on **Start server** button shown above to start the server.

**Step 4:** Click on **Open browser preview** button shown above to view the GUI preview as shown below.

A screenshot of a computer

Description automatically generated

**Step 5:** You will receive a congratulations message as shown below upon successful completion of the task.

A screen shot of a computer

Description automatically generated

Report.service.ts

*import* { HttpClient } *from* '@angular/common/http';

*import* { Injectable } *from* '@angular/core';

*import* { filter, map, Observable, of } *from* 'rxjs';

*import* { Sales } *from* './sales.interface';

@Injectable({

  providedIn: 'root',

})

*export* class ReportsService {

  constructor(private *http*: HttpClient) {}

  getSalesData(): Observable<Sales[]> {

    const jsonFile='assets/sales.json';

*return* this.http.get<Sales[]>(jsonFile);

  }

  calculateTotalSales(*salesData*: Observable<Sales[]>): Observable<number> {

   let totalSale=0;

*salesData*.subscribe(*data*=>{

*data*.forEach(*element*=>{

      totalSale+=*element*.salesTotal;

    });

*return* of(totalSale);

   })

*return* of(totalSale);

  }

  calculateTotalCashSale(*salesData*: Observable<Sales[]>): Observable<number> {

  let cashTotal=0;

*salesData*.subscribe(*data*=>{

*data*.forEach(*element*=>{

*if*(!*element*.creditCard){

        cashTotal+=*element*.salesTotal;

      }

    });

*return* of(cashTotal);

  })

*return* of(cashTotal);

  }

  calculateTotalCreditSale(*salesData*: Observable<Sales[]>): Observable<number> {

   let cashTotal=0;

*salesData*.subscribe(*data*=>{

*data*.forEach(*element*=>{

*if*(*element*.creditCard){

        cashTotal+=*element*.salesTotal;

      }

    });

*return* of(cashTotal);

   })

*return* of(cashTotal);

  }

  calculateBuyerWithMostSale(

*salesData*: Observable<Sales[]>

  ): Observable<{ buyerName: string; saleTotal: number }> {

  let mostSale=0;

  let buyer:{buyerName:string;saleTotal:number}={buyerName:"",saleTotal:0};

*salesData*.subscribe(*data*=>{

*data*.forEach(*element*=>{

*if*(mostSale<*element*.salesTotal){

        mostSale=*element*.salesTotal;

        buyer={buyerName:*element*.buyerName,saleTotal:*element*.salesTotal};

      }

    });

*return* of(buyer);

  })

*return* of(buyer);

  }

}