**Playoffs Technical Document**

Application Flow

[1. Splash 4](#_Toc124164002)

[1.1 Checking if the user has already logged or not 4](#_Toc124164003)

[2. Sign Up 5](#_Toc124164004)

[2.1 Register 5](#_Toc124164005)

[2.2 OTP Verification 6](#_Toc124164006)

[2.3 Role Select 7](#_Toc124164007)

[2.4 Create Account 8](#_Toc124164008)

[2.5 Create Team 9](#_Toc124164009)

[2.6 Add Members 10](#_Toc124164010)

[2.6.1 Add Members through Phone Contact 10](#_Toc124164011)

[2.6.2 Add Members Manually 11](#_Toc124164012)

[2.7 Add schedule 12](#_Toc124164013)

[2.7.1 Add Game 12](#_Toc124164014)

[2.7.2 Add Event 14](#_Toc124164015)

[3. Login 15](#_Toc124164016)

[3.1 Login with Mobile 15](#_Toc124164017)

[3.2 OTP Verification 16](#_Toc124164018)

[3.3 Team List 17](#_Toc124164019)

[4. Invite Member 18](#_Toc124164020)

[4.1 Join Team 18](#_Toc124164021)

[5.Team Landing 19](#_Toc124164022)

[5.1 Home 19](#_Toc124164023)

[5.2 Schedule 20](#_Toc124164024)

[5.2.1 Edit Schedule 22](#_Toc124164025)

[5.2.1.1 Edit Game 22](#_Toc124164026)

[5.2.1.2 Edit Event 23](#_Toc124164027)

[5.2.2 Schedule Details 24](#_Toc124164028)

[6.2.2.1 Details 24](#_Toc124164029)

[5.2.2.2 Availability 25](#_Toc124164030)

[5.2.2.3 Assignments 26](#_Toc124164031)

[5.3 Teams 27](#_Toc124164032)

[5.3.1 Edit Team 28](#_Toc124164033)

[5.3.2 Team Details 29](#_Toc124164034)

[5.4 Members 30](#_Toc124164035)

[5.4.1 Add New Member 31](#_Toc124164036)

[5.4.2 Edit Member 32](#_Toc124164037)

[5.4.3 Member Details 33](#_Toc124164038)

[5.5 Settings 34](#_Toc124164039)

[6.Hamburger Menu 34](#_Toc124164040)

[6.1 My Profile 35](#_Toc124164041)

[Common Files: 36](#_Toc124164042)

# 1. Splash

****

**Component Name:**

* SplashScreenComponent

**Functionality:**

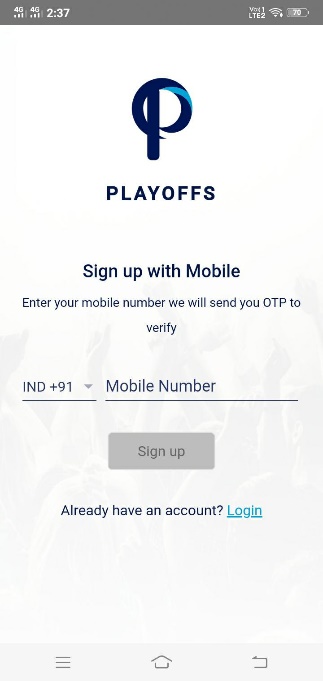
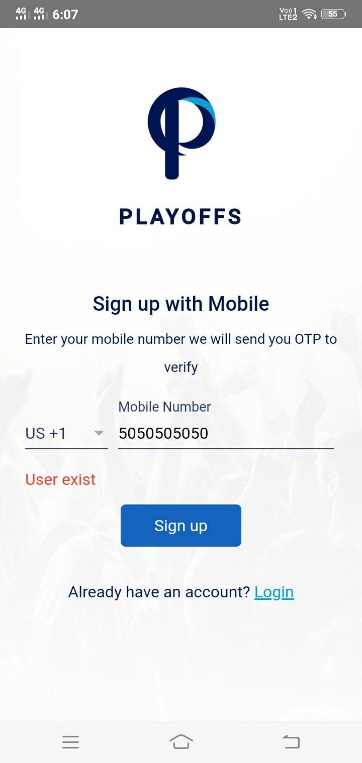
* Whenever open the application it will shows first.
* Inside the **ngOninit**() method setting the timeout for the splash screen as **2 sec.**

# 1.1 Checking if the user has already logged or not

* If the user login or signup the token will generate and will stored in localstorage through **localStorage.setItem("authToken", apiResponse.token)** this method.
* The expiry time for the token is 1 day. So, if the user already logged-in and open the application before the expiration time the screen will take it to the user’s last action screen.
* Otherwise, the API’s response is 401(unauthorized) the **localStorageService’s** method **clearAllLocalStorage()** automatically execute and it will take to the login screen.

# 2. Sign Up

# 2.1 Register

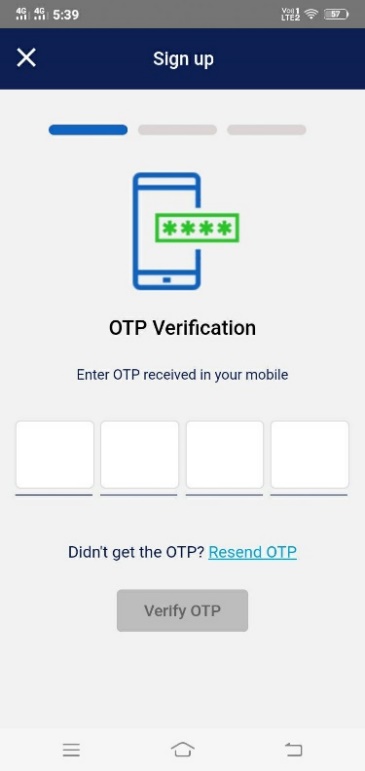
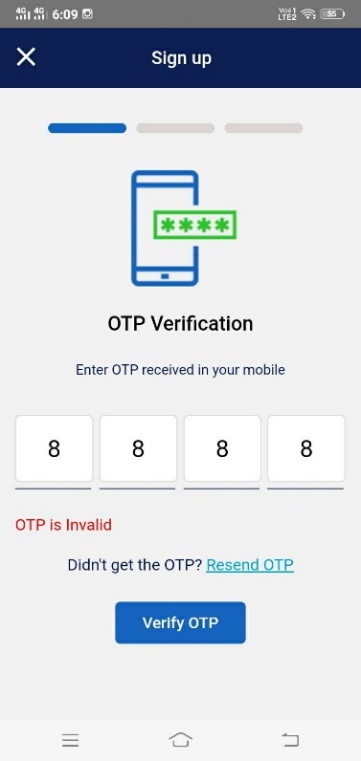
**Component Name:**

* RegisterComponent

**Functionality:**

* When the user enters the mobile number and clicks the signUp button the **OnSubmit()** method will be called.
* Inside the **OnSubmit()** method use the **AuthService’s getDataFromService()** methodwhich is to make an HTTP get call. This will be getting the OTP for the mobile number. Then passing two parameters to the method **getDataFromService(SignupUrl, query params).** The first param is an **environment** variable which is having an API endpoint for getSignUpOtp and second param is query parameter for get method, which is passing the username as a query param.
* If the API statusCode is 200, then storing user data in the localStorage with the help of localStorageServices. Then it will redirect to otpVerification screen. Otherwise, will show the back-end API response status message in the front end.
* **pressEnterToSubmit**() method is used for while clicking the enter key the onSubmit() method will be called.
* **Key**() is used for restrict the alphabets and special characters while entering.

# 2.2 OTP Verification

**Component Name:**

* OtpVerificationComponent

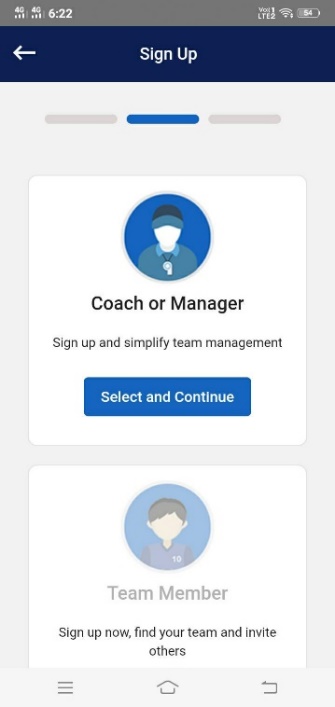
**Model Name:**

* SignUpModel

**Functionality:**

* In the constructor setting the top bar title and buttons with the help of NavigationModel.
* Inside the **ngAfterViewInit()** setting the OTP box auto focus.
* When user enters the OTP which is received from the registered mobile number, then click Verify OTP button the **onSubmit()** method will be called.
* Inside the **onSubmit()** method using the **AuthService’s** **postDatatoService**() method which is to make an HTTP post call. This will be validating the OTP. Then passing two parameters to the method **postDataToService** (**apiSignupOTPUrl, SignUpModel**). The first param is an **environment** variable which is having an API endpoint for validateSignupOtp and the second one is SignUpModel object.
* If the API statusCode is 200 and the playerInvite parameter is true, then store user data in the localStorage with the help of localStorageServices and it will redirect to **team invitation** screen. if it is false it will redirect to **Role** screen.
* Otherwise, the api response status message will show OTP invalid or Expired in the front end. we have the option Resend OTP. If the Resend OTP button is clicked then resendOTP() method will be called and inside the method, **AuthService’s** **getDataFromService(SignupUrl, query params)**  is called and send the new OTP to the mobile number.
* **pressEnterToSubmit()** method is used for while clicking the enter key the onSubmit() method will be called.
* **Move()** method is used while entering the OTP, once the form field reaches the maxlength the focus will automatically move to the next field.
* **Key**() is used for restrict the alphabets and special characters while entering.

# 2.3 Role Select



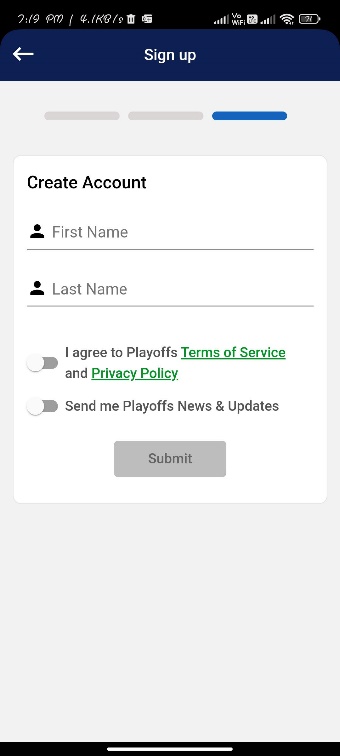
**Component Name:**

* RoleComponent

**Functionality:**

* In the constructor setting the top bar title and buttons with the help of NavigationModel.
* On this screen we have enabled only one option which is **Coach or Manager.**
* When the user selects the **Coach or Manager** and clicks the **select and continue** button **onSubmit(role:any)** method will call.
* While calling this method there is no backend service is called, will store only the parameter passing data to localstorage using **localStorageService’s setUserActionData()** method. Then the page will redirect to create account screen.

# 2.4 Create Account



**Component Name:**

* CreateAccountComponent

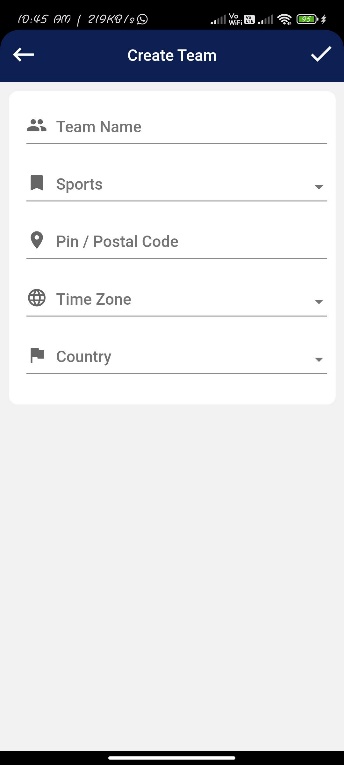
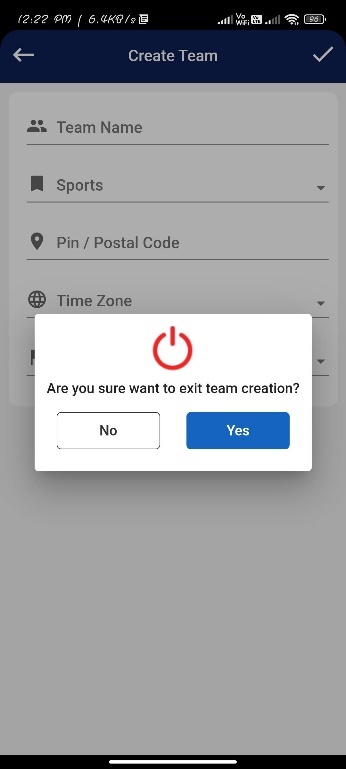
**Model Name:**

* RegisterUserModel

**Functionality:**

* In the constructor setting the top bar title and buttons with the help of NavigationModel.
* **ngOnDestroy()** is used for preserving the entered data, even if the component is destroyed.
* The form is filled and click submit button, the **onSubmit()** method will call.
* Inside the **onSubmit()** method we use the **AuthService’s postDataToService()** method which is to make an HTTP post call. This will be creating a user. Passing two parameters to the method **postDataToService (apiSaveUser, RegisterUserModel).** The first param is an environment variable which is having an API endpoint for createUser and the second one is RegisterUserModel object.
* If the API response statusCode is 200, then get the authorization token from the response and store in localstorage through **localStorage.setItem("authToken", apiResponse.token)** method. And also, **userRole, username** and **actionmessages** are store to localStorage with the help of **localStorageService’s** methods.
* **Omit\_special\_char()** methodis used for restrict the numbers and special characters while entering.

# 2.5 Create Team

**Component Name:**

* CreateTeamComponent

**Model Name:**

* TeamModel

**Functionality:**

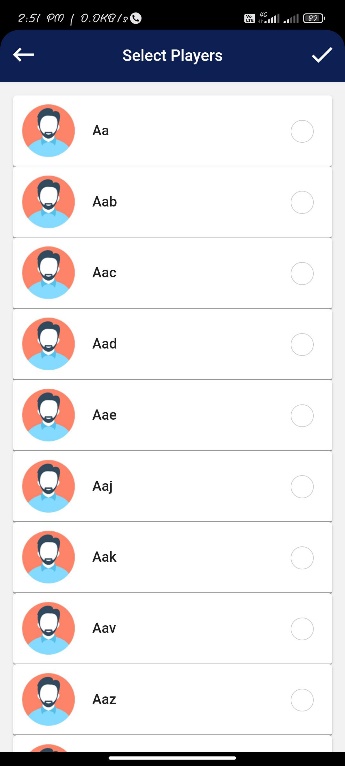
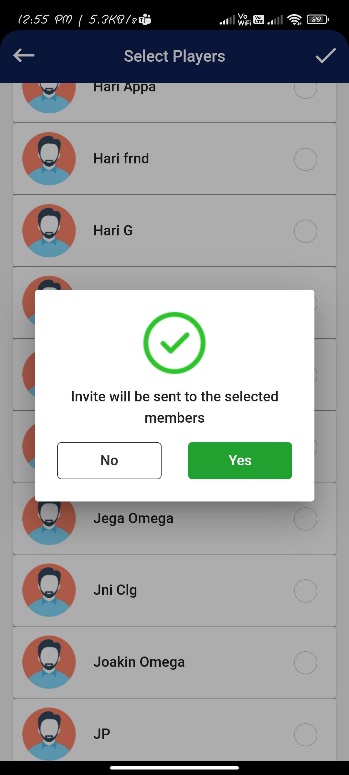
* In the constructor setting the top bar buttons and titles with the help of **NavigationModel.**
* Sports, timezone and country fields are the dropdowns and the values dynamically come from the backend.
* Inside the **ngOnInit**() method call the timeZone, Sports, and countryLists api’s to get the data. Below AuthService’s http methods are used to get the data.

1. Sports - **getListOfData(apiSportsList, Header)**
2. Timezone - **getListOfData(apiTimezone, Header)**
3. CountryList- **getListOfData(apiCountryList, Header)**

* The first parameter for the above http methods is an environment variable which is having api endpoints. The second one is a token.
* Inside the ngOnInit() method first checking the localstorage action messages. If it is createAction the component will be used for creating a new team and the top bar also be dynamically changed.
* The form is filled and click the tick button, the **submitForm**() method will be called. Inside this manually initialize the values to TeamModel and then check the action type is create or edit.
* If it is create action the **createTeam()** method will call. Inside this method use the **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token. This will be creating a team. Passing three parameters to the method **postDataToServiceWithToken (apiCreateTeam, TeamModel, Header).** The first param is an environment variable which is having an API endpoint for createTeam and the second one is teamModel object and the third one is a token.
* If the API response statusCode is 200, then store the teamId and teamName in localstorage with the help of localStorageService. Then it will redirect to the respective screen based on the action messages.
* **Key**() is used for restrict the alphabets and special characters while entering.
* If click the backbutton the exit popup will open with the help of navigation model.

# 2.6 Add Members

# 2.6.1 Add Members through Phone Contact

**Component Name:**

* SelectPlayerComponent

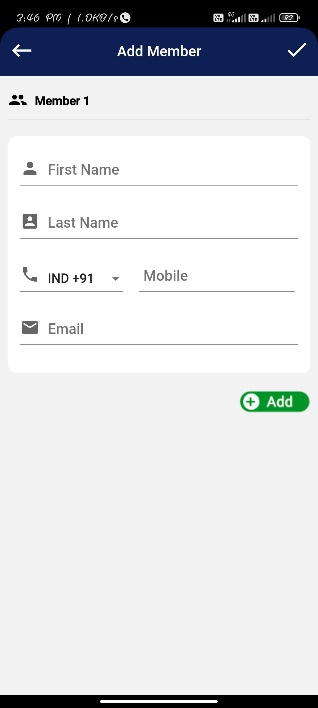
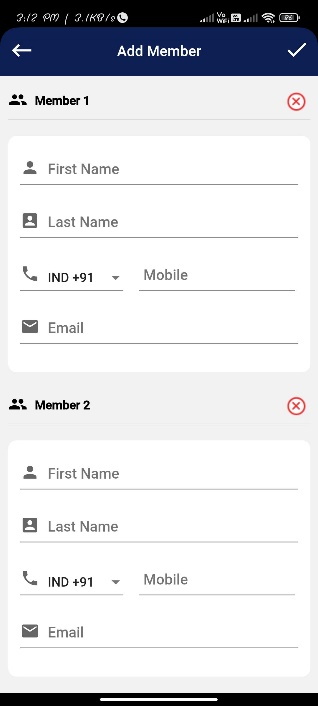
**Model Name:**

* AddPlayerModel

**Functionality:**

* In the constructor setting the top bar buttons and titles with the help of **NavigationModel.**
* **onCheckboxChange()** method is used for selecting the contacts from the list.
* Created **openContacts()** method and call it inside the ngOnInit().
* Inside the **openContacts()** first get the phone contact permission with the help of **Contacts plugin.** Using **Contacts.getPermissons()** method to get the permission for accessing the phone contact.
* If permission is granted list the phone contacts to this component.
* Then check whether the contact list mobile numbers have country codes or not. If it is not then adding the owner’s country code to the respective mobile number.
* After that formatting all the mobile numbers. **(Format: 91-1234567890).**
* Members are selected then click the tick button **onSubmit()** method will be called and open the confirmation popup with the help of matDialog.
* In the popup if click the send Invites button **onSubmitConfirmation()** method will be called.
* Next, manually initialize the values to AddPlayerModel. Then using **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token. This will be adding the players. Passing three parameters to the method **postDataToServiceWithToken (apiAddPlayer, addPlayerModel, Header).** The first param is an environment variable which is having an API endpoint for addingPlayer and the second one is playerModel object and the third one is a token.
* If API’s status code is 200 then it will redirect to the AddSchedule screen.

# 2.6.2 Add Members Manually

** **

**Component Name:**

* AddMemberManuallyComponent

**Model Name**:

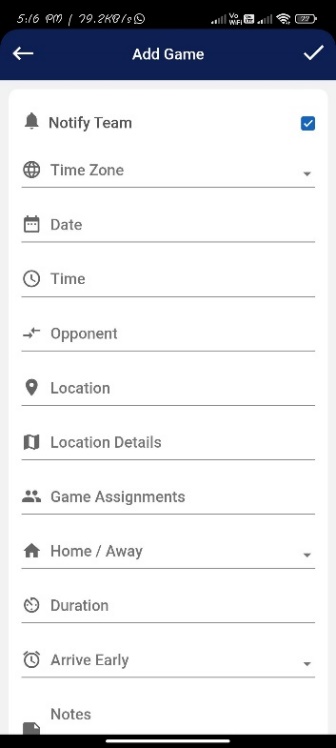
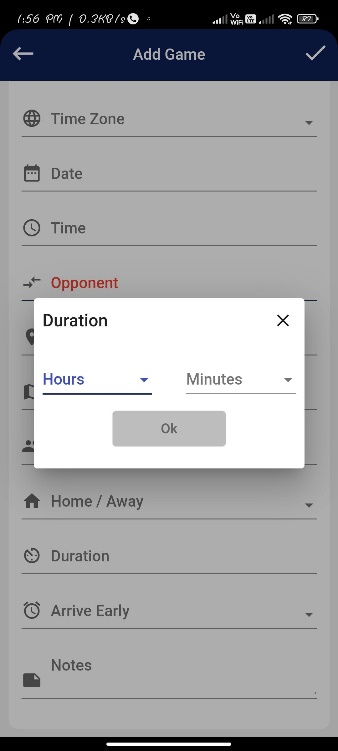
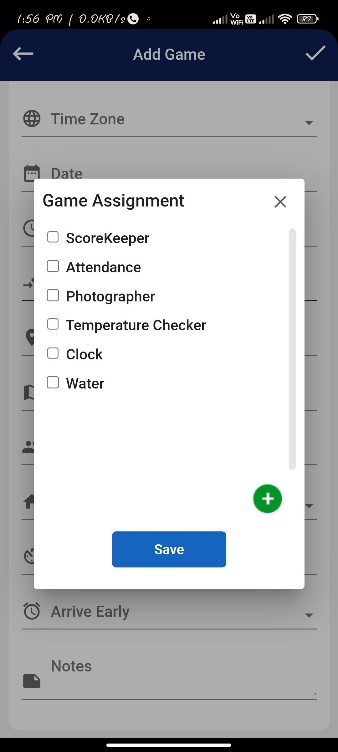
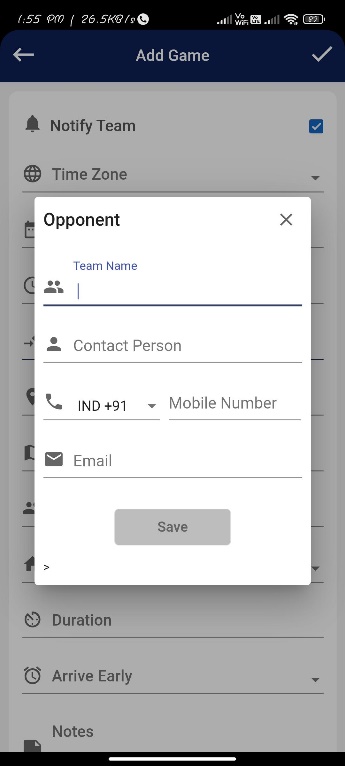
* AddPlayerModel

**Functionality:**

* In the constructor setting the top bar title and buttons with the help of NavigationModel.
* In this component uses a form array for adding multiple members.
* Inside the formGroup created one form array name as **members.**
* When clicking the Add button **addAnotherMember()** method will be called and dynamically create another player form.
* While removing the player **removePlayer(index:any)** method will call and remove the player based on the index passed in the parameter.
* When the players are added and click the tick button **addMemberSubmit**() method will call.
* Inside the **addMemberSubmit**() manually initialize the values to AddPlayerModel.
* Then using the **AuthService’s postDataToServiceWithToken**() method which is to make an http post call with the token. This will be adding the players. Passing three parameters to the method **postDataToServiceWithToken (apiAddPlayer, AddPlayerModel, Header)**. The first param is an environment variable which is having an API endpoint for addPlayer and the second is playerModel object and the third is a token.
* If the API’s status code is 200, it will redirect to the respective screen.
* **Omit\_special\_char()** methodis used for restrict the numbers and special characters while entering.
* **Key**() method is used for restrict the alphabets and special characters while entering.

# 2.7 Add schedule

# 2.7.1 Add Game

**Component Name**:

* AddgamesComponent
* OppnentComponent (For opponent popup)
* DurationComponent (For Duration popup)
* GameAssignmentsComponent (For Assignments popup)

**Model Name**:

* GameModel

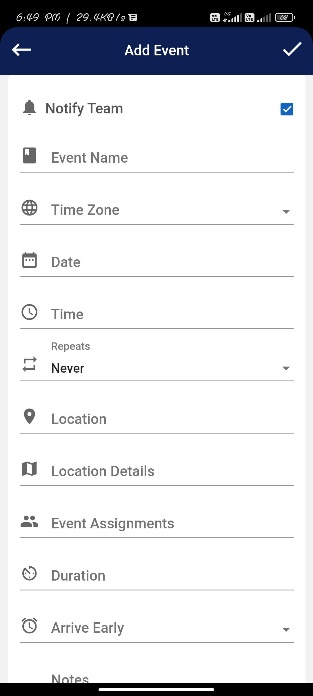
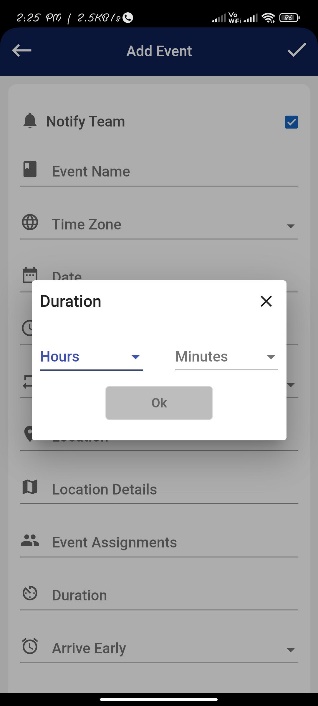
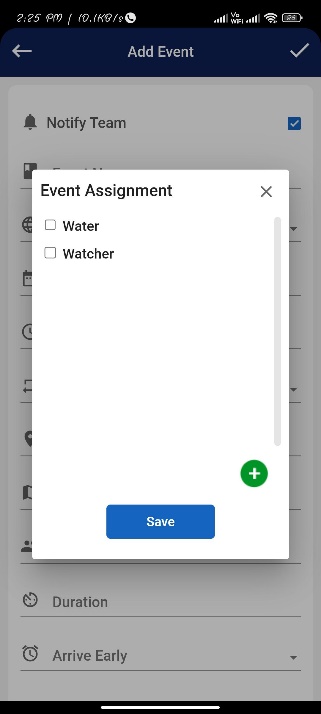
**Functionality**:

* Inside the constructor first checking the localstorage action messages. If it is a editAction, the component will be used for editng a game and the top bar also be dynamically changed. Otherwise the component is used for creating a new game.
* Timezone, assignments, duration, and arriveEarly are the dropdowns and the values dynamically come from the backend.
* Created **initGamePage**() method and calling it inside the ngOninit().
* Inside the **initGamePage**() method call the timeZone, duration, and arriveEarly api’s to get the data. Below AuthService’s http methods are used to get the data.

1. timeZone - **getListOfData(apiTimeZone, Headers)**
2. assignments - **getListOfData(apiGameAssignments, Headers)**
3. duration and arrive early- **getListOfData(apiArriveDuration, Headers)**

* The first parameter for the above http methods is an environment variable which is having api endpoints. The second one is a token.
* **onLocationSelected**() and **onAutoCompleteSelected**() methods are used for get google map searched location.
* **notifyTeamSelected**() method is used for passing data yes or no while select and unselect the notifyTeam checkbox.
* when clicking the opponent field the **openOpponentDialog()** methodwill be called and it will open the OpponentComponent inside the popup with the help of matDialog. Whenever open the popup selected data will pass addgame component to opponent component and whenever close the dialog opponent component data will pass to addgame component.
* **openAssignmentDialog()** methodis used while clicking the assignments field, the GameAssignmentsComponent will open inside the popup with the help of matDialog. This also shares the data same as the opponent popup.
* **openDurationDialog**() methodis used while clicking the duration field, DurationComponent will open inside the popup with the help of matDialog. This also shares the data same as the opponent popup.
* The form is filled and click the tick button the **addGame**() method will be called. Inside this method checking the localStorage Action messages. If it is createAction **createGame**() method will be called.
* Inside the **createGame**() method using **AuthServices’s postDataToServiceWithToken**() method, to make a HTTP post call with token. Passing three parameters to the method **postDataToServiceWithToken (apiAddGame, gameModel, Header).** The first param is an environment variable which is having an API endpoint for createGame and the second one is gameModel object and the third one is a token.
* If API statusCode is 200 it will redirect to a respective screen.

# 2.7.2 Add Event

**Component Name**:

* AddEventComponent
* DurationComponent (For Duration popup)
* EventAssignmentsComponent (For Assignments popup)

**Model Name**:

* EventModel

**Functionality**:

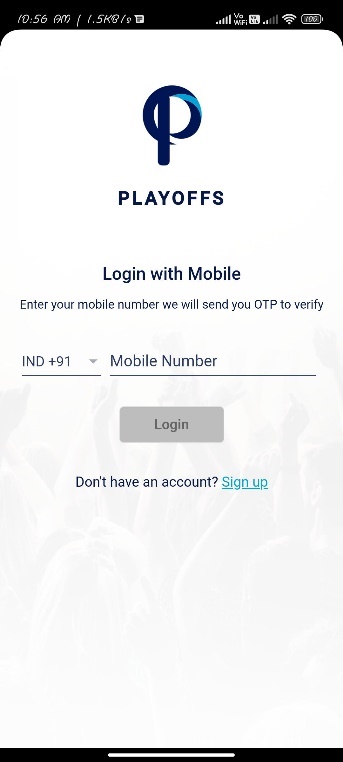
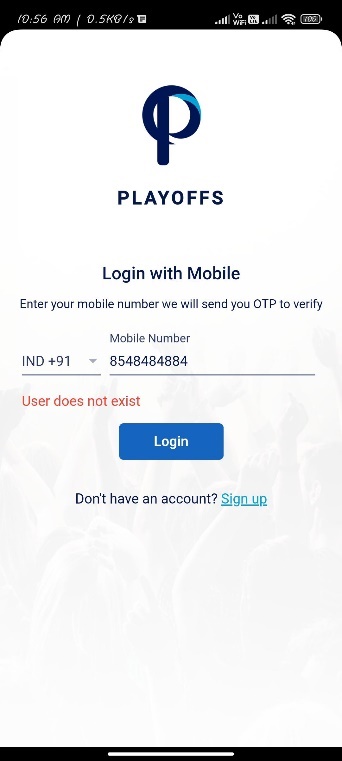
* Inside the constructor first checking the localstorage action name. If it is an editAction the component will be used for editing a event and the top bar also be dynamically changed. Otherwise the component is used for creating a new event.
* Timezone, assignments, duration, and arriveEarly are the dropdowns and the values dynamically come from the backend.
* Created **initEventPage**() method and calling it inside the ngOninit().
* Inside this **initEventPage** () method call the timeZone, duration, and arriveEarly api’s to get the data. Below AuthService’s http methods are used to get the data.

1. timeZone - getListOfData(apiTimeZone, Headers)
2. assignments - getListOfData(apiEventAssignments, Headers)
3. duration and arrive early- getListOfData(apiEventDuration, Headers)

* The first parameter for the above http methods is an environment variable which is having api endpoints. The second one is a token.
* **onLocationSelected**() and **onAutoCompleteSelected**() methods are used for get google map searched location.
* **notifyEventSelected**() method is used for passing data yes or no while select and unselect the notifyTeam checkbox.
* **openAssignmentDialog**() method is used while clicking the assignments field, the EventAssignmentsComponent will open inside the popup with the help of matDialog. Whenever open the popup selected data will pass addevent component to EventAssignmentComponent component and whenever close the dialog EventAssignmentComponent data will pass to addEvent component.
* **openDurationDialog**() method is used while clicking the duration field, DurationComponent will open inside the popup with the help of matDialog. This also shares the data same as the assignment’s popup.
* The form filled and click the tick button the addEvent() method will call. Inside this method checking the localStorage action name. If it is create action **createEvent()** method will be called.
* Inside the createEvent() method using **AuthServices’s postDataToServiceWithToken()** method, to make a HTTP post call with token. Passing three parameters to the method **postDataToServiceWithToken (apiAddEvent, eventModel, Header).** The first param is an environment variable which is having an API endpoint for createEvent and the second one is eventModel object and the third one is a token.
* If API statusCode is 200, it will redirect to the respective screen.

# 3. Login

# 3.1 Login with Mobile

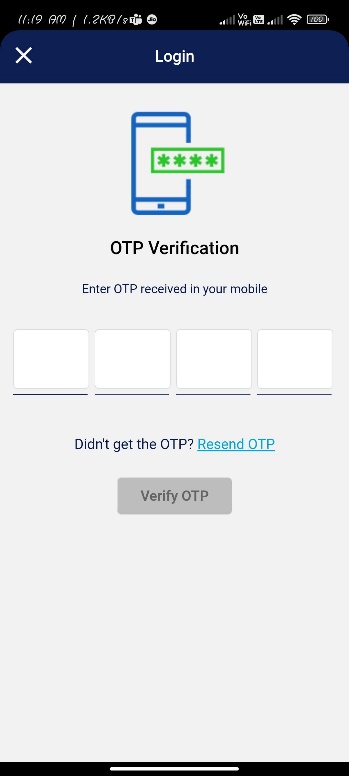
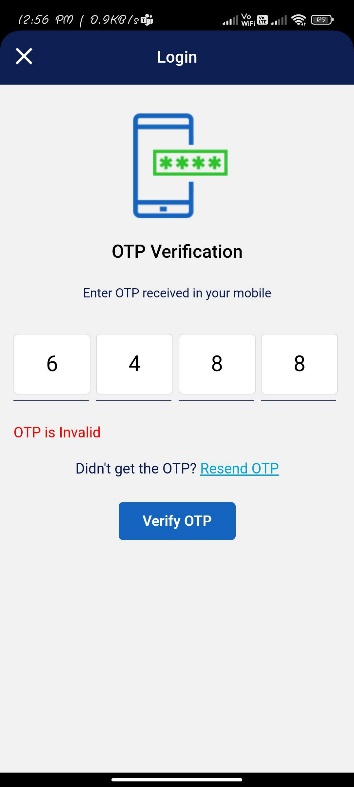
**Component Name:**

* LoginComponent

**Functionality:**

* When the user enters the mobile number and click the Login button the **OnSubmit()** method will be called.
* Inside the **OnSubmit()** method uses the **AuthService’s getDataFromService()** methodwhich is to make an HTTP get call. This will be sending an OTP for the mobile number. Then passing two parameters to the method **getDataFromService(apiLoginUrl, query params).** The first param is an **environment** variable which is having an API endpoint for getLoginOtp and second param is query parameter for get method, which is passing the username as a query param**.**
* If the API statusCode is 200, then the user data will store in the localStorage with the help of localStorageServices. Then it will redirect to loginOtpVerification screen. Otherwise, will show the back-end API response status message in the front end.

# 3.2 OTP Verification

** **

**Component Name:**

* LoginOtpVerificationComponent

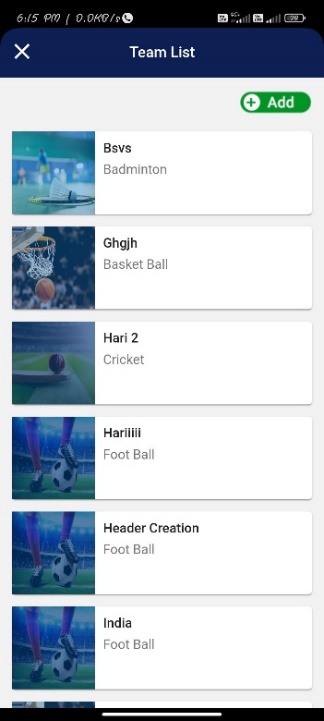
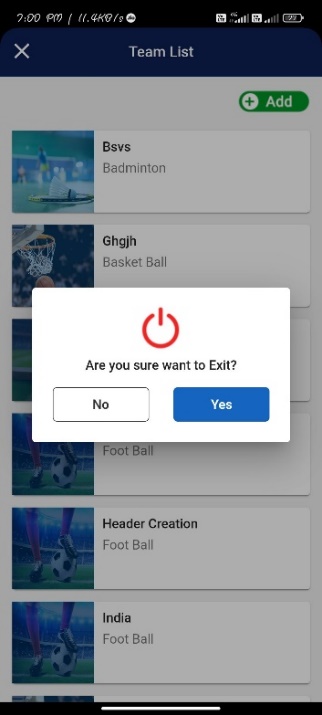
**Model Name:**

* LoginModel

**Functionality:**

* In the constructor setting the top bar title and buttons with the help of NavigationModel.
* When user enter the OTP which is received from the registered mobile number, then click Verify OTP button the **onSubmit()** method will be called.
* Inside the **onSubmit()** method manually initialize the values to loginModel. Using the **AuthService’s** **postDataFromService**() method which is to make an HTTP post call. This will be validating the OTP. Then passing two parameters to the method **postDataToService** (**apiLoginOTPUrl, SignUpModel**). The first param is an **environment** variable which is having an API endpoint for validateLoginOtp and the second one is LoginModel object.
* If the API response statusCode is 200, then get the authorization token from the response and store in localstorage through **localStorage.setItem("authToken", apiResponse.token)** method. And also, user data are stored in localStorage with the help of **localStorageService’s** setmethods. Then it will redirect to teamList screen.
* Otherwise, the API’s response status message will show OTP invalid or Expired in the front end. we have the option Resend OTP. If the Resend OTP is clicked then resendOTP() method will call and inside the method, **AuthService’s** **getDataFromService(apiLoginUrl, query params)**  is called and send the new OTP to the mobile number.
* **pressEnterToSubmit()** method is used for while clicking the enter key the onSubmit() method will be called.
* **Move()** method is used while entering the OTP, once the form field reaches the maxlength the focus will automatically move to the next field.
* **Key**() is used for restrict the alphabets and special characters while entering.

# 3.3 Team List

**Component Name:**

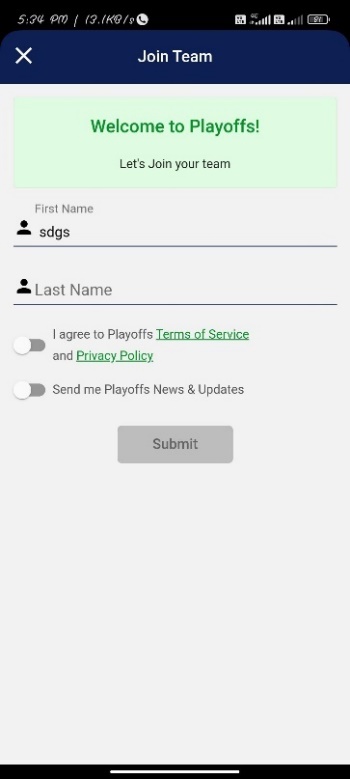
* TeamListComponent

**Functionality**:

* In the constructor set the top bar title and buttons with the help of NavigationModel.
* Inside the **ngOnInit**() call the **AuthService’s getListOfData()** method to get the team lists for their username. Passing two parameters for the method **getListOfData(apiTeamList + queryParam, Header).** The first param is an environment variable which is having an API endpoint for getTeamList with the concatenation of queryparam which is a username and the second one is a token.
* While selecting any team **selectTeam()** method will be called and store the team and member data in localStorage with the help of localStorageService, then it will redirect to the team landing screen.
* When clicking add button it will redirect to create team screen.
* If click the close button the close confirmation popup will be opened with the help of NavigationModel.

# 4. Invite Member

# 4.1 Join Team



**Component Name:**

* JoinTeamComponent

**Model Name:**

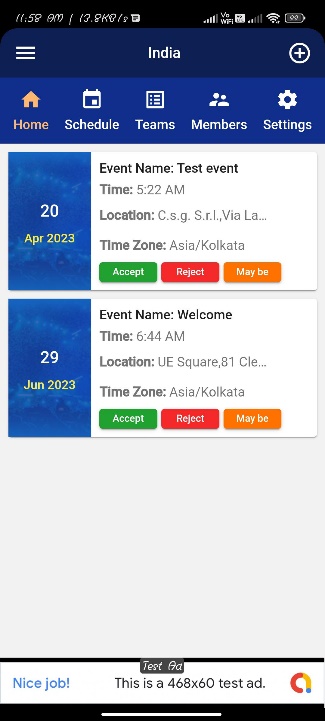
* RegisterUserModel

**Functionality:**

* In the constructor set the top bar title and buttons with the help of NavigationModel. Then manually initialize the localStorage data to RegisterUserModel.
* When clicking submit button **onSubmit()** method will be called.
* Inside the **onSubmit()** methodusing **AuthServices’s postDataToService()** method, to make an HTTP post call. Passing two parameters to the method **postDataToService(apiSaveUser, RegisterUserModel).** The first param is an environment variable which is having an API endpoint for createUser and the second one is RegisterUserModel object.
* If the API response status code is 200, then get the authorization token from the response and store in localstorage through **localStorage.setItem("authToken", apiResponse.token)** method.

# 5.Team Landing

# 5.1 Home

****

**Component Name:**

* TeamDashboardComponent
* TeamsLandingComponent

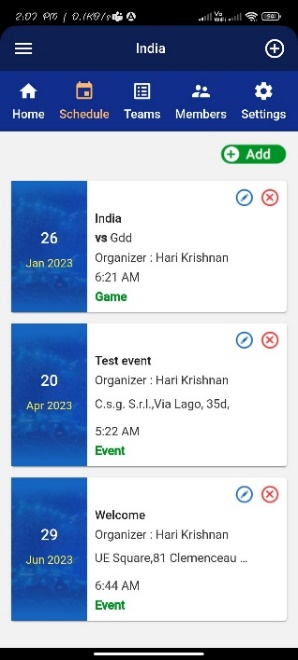
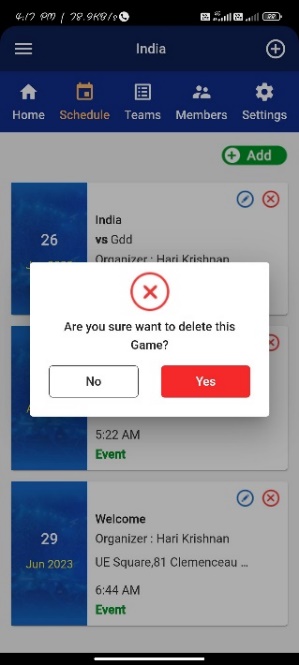
**Model Name:**

* Acceptance Model

**Functionality:**

* This screen **TeamDashboardComponent** is the main component. **TeamLandingComponent** is injected into **TeamDashboardComponent**.
* In **TeamDashboardComponent** inside the ngOninit() get user data and store them into localStorage. Using the **AuthService’s** **getListOfData**() method which is to make an HTTP get call to get the user details. Then passing two parameters to the method **getListOfData** (**apiUserDetail + queyParam, Headers**). The first param is an **environment** variable which is having an API endpoint for getUserDetails with the concatenation of queryparam which is a username and the second one is a token.
* Using **LocalStorageService** to store theuser data to localStorage.
* **TeamLandingComponent** is used to show the event lists and event acceptance status.
* In this component created **initLandingPage()** method and call it inside the ngOnInit().
* Inside the **initLandingPage()** method using **AuthService’s** **getListOfData**() method which is to make an HTTP get call to get the event lists. Then passing two parameters to the method **getListOfData** (**apiUserDetail + queryParam, Headers**). The first param is an **environment** variable which is having an API endpoint for getUserDetails with the concatenation of queryparam which is a username and teamId, and the second param is a token.
* When clicking the Accept, Reject, Maybe buttons **updateEventAcceptanceStatus()** will be called, passing event, team, member data and acceptance string as a parameter for this method.
* Inside the **updateEventAcceptanceStatus()**  method manually initialize the values to AcceptanceModel. Then using **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token. This will be updating the event acceptance status for the respective user. Passing three parameters to the method **postDataToServiceWithToken (apiEventAcceptance, AcceptanceModel, Header).** The first param is an environment variable which is having an API endpoint for userEventAcceptance and the second one is AcceptanceModel object and the third one is a token.

# 5.2 Schedule

** **

**Component Name:**

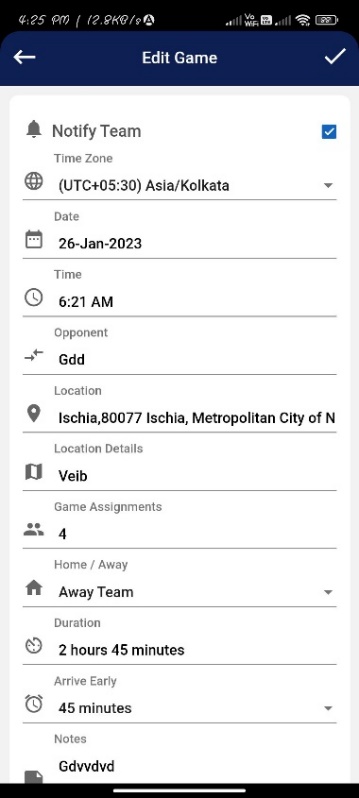
* TeamsScheduleComponent

**Functionality:**

* Create one method **listSchedules**() and call it inside the **ngOnInit**().
* Inside this method make a service call and get the schedule list for the respective username and teamId.
* Using **AuthService’s getListOfData()** method which is to make an HTTP get call to get the schedule list. Passing two parameters to the method **getListOfData** (**apiScheduleList + queryparams, Headers**). The first param is an **environment** variable which is having an API endpoint for getScheduleList with the concatenation of queryparam which is a username and teamId, and the second param is a token.
* Clicking add button **openSchedule()** method will call and it will redirect to the addSchedule screen to create a game or event.
* Clicking the particular schedule, the **scheduleDetails()** method will be called and passing **scheduleId** and **schedule** **type** as a parameter. Then the parameter values are stored in localstorage with the help of localStorageService. After that, it will redirect to **eventDetails** screen.
* When clicking the edit icon from the particular schedule, the **editScheduleById()** method will be called and passing **scheduleId** and **schedule** **type** as a parameter. Then the parameter values are stored in localstorage with the help of localStorageService. If the type is a game it will redirect to the **edit game screen(adddGame),** if it is an event it will redirect to the edit event screen (addEvent).
* When clicking the delete icon from the particular schedule the **openDialogWithTemplateRef**() method will be called and it will open the delete popup with the help of matDialog. If click the popup’s delete button **deletecheduleById()** method will be called and passing **scheduleId** and **type** as a parameter.
* Inside the **deletecheduleById()** method using **AuthServices’s postDataToServiceWithToken()** method, to make a HTTP post call with token. If it is a game **postDataToServiceWithToken (apiDeleteGame + Queryparams, Header)** method will be called, if it is an event **postDataToServiceWithToken (apiDeleteschedule + Queryparam, Header)** will be called. The first param is an **environment** variable which is having an API endpoint for deleteGame/deleteEvent with the concatenation of queryparam which is a username and gameId/eventId, and the second param is a token.

# 5.2.1 Edit Schedule

# 5.2.1.1 Edit Game

****

**Component Name:**

* AddgamesComponent
* OppnentComponent (For opponent popup)
* DurationComponent (For Duration popup)
* GameAssignmentsComponent (For Assignments popup)

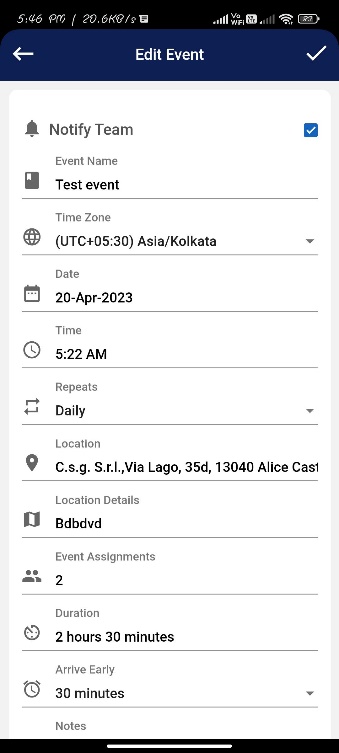
**Model Name:**

* GameModel

**Functionality:**

* Inside the constructor first checking the localstorage action messages. If it is an editAction the component will be used for edit a game and the top bar also be dynamically changed.
* Inside the **initGamePage**() method check if it is an editAction make an API call for the respective gameId and get the gamedetails. Using **AuthService’s getListOfData()** method to get the game details for their respective game. Passing two parameters for the method **getListOfData(apiGetGameDetailsById + queryParam, Header).** The first param is an environment variable which is having an API endpoint for getGameDetails with the concatenation of queryparam which is a gameId and the second one is a token.
* Then manually initialize the game data to gameModel and send the opponents, durations, assignments data to the respective components through local variables, and bind the gameModel values to the respective html fields.
* The game is edited and click the tick button the **addGame**() method will be called. Inside this method checking the localStorage Action messages. If it is an edit action **updateGame**() method will be called.
* Inside the **updateGame**() method using **AuthServices’s postDataToServiceWithToken**() method, to make a HTTP post call with token. Passing three parameters to the method **postDataToServiceWithToken (apiUpdateGame, gameModel, Header).** The first param is an environment variable which is having an API endpoint for updateGame and the second one is gameModel object and the third one is a token.
* If the API statusCode is 200 it will redirect to a respective screen based on the action messages.

# 5.2.1.2 Edit Event

****

**Component Name:**

* AddEventComponent

**Model Name:**

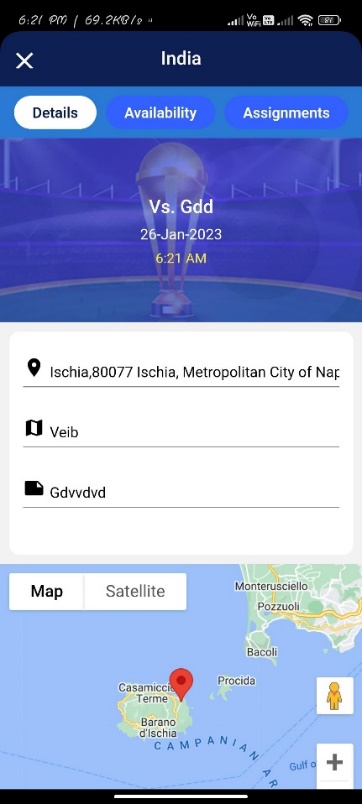
* EventModel

**Functinality:**

* Inside the constructor first checking the localstorage action name. If it is an editAction the component will be used for editing an event and the top bar also be dynamically changed.
* Then check if it is an editAction make an API call for the respective eventId and get the event details. Using **AuthService’s getListOfData()** method to get the event details for their respective event. Passing two parameters for the method **getListOfData(apiGetEventDetailsById + queryParam, Header).** The first param is an environment variable which is having an API endpoint for getEventDetails with the concatenation of queryparam which is an eventId and the second one is a token.
* Then manually initialize the event data to eventModel and send the durations, assignments data to the respective components through local variables, and bind the eventModel values to the respective html fields.
* The event is updated and click the tick button the addEvent() method will call. Inside this method checking the localStorage Action name. If it is an editAction **updateEvent()** method will be called.
* Inside the **updateEvent** () method using **AuthServices’s postDataToServiceWithToken()** method, to make a HTTP post call with token. Passing three parameters to the method **postDataToServiceWithToken (apiUpdateEvent, eventModel, Header).** The first param is an environment variable which is having an API endpoint for upateEvent and the second one is eventModel object and the third one is a token.
* If API statusCode is 200 it will redirect to the respective screen.

# Schedule Details

# 6.2.2.1 Details

****

**Component Name:**

* TeamsEventDetailsComponent

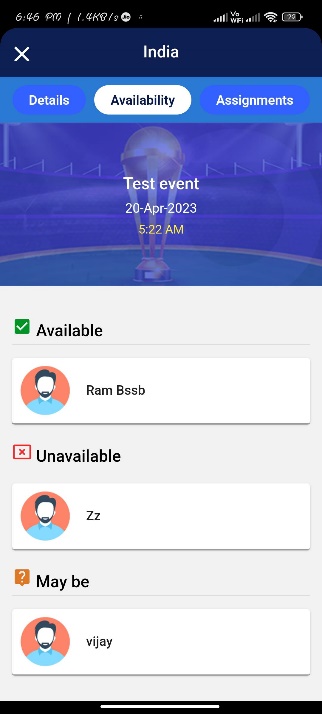
**Model Name:**

* GameModel

**Functionality:**

* Inside the **ngOnInt**() make an API call to get the game/event details. Using **AuthService’s getListOfData()** method to get the game/event details. Passing two parameters for the method **getListOfData(apiGetEventDetailsById / apiGetEventDetailsById + queryParam, Header).** The first param is an environment variable which is having an API endpoint for getEventDetails/getGameDetails with the concatenation of queryparam which is an eventId/gameId and the second one is a token.
* Then initialize the game/event details data to gameModel/eventModel and bind the model values to the respective html fields.

# 5.2.2.2 Availability

****

**Component Name:**

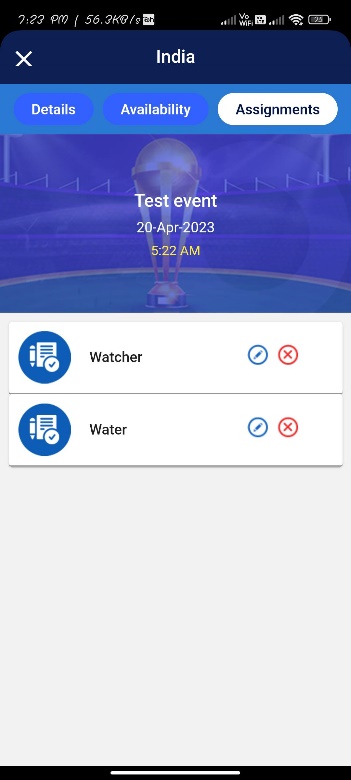
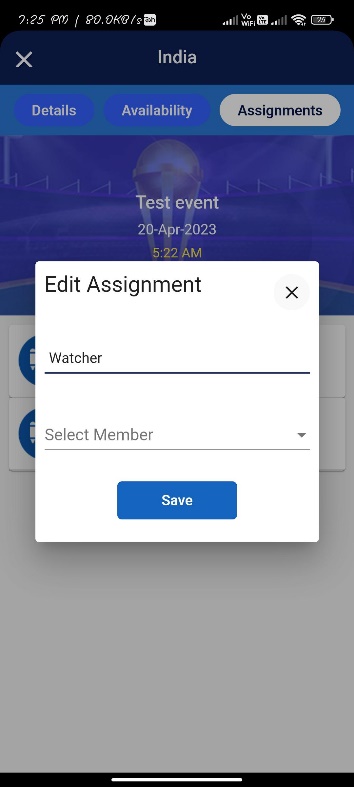
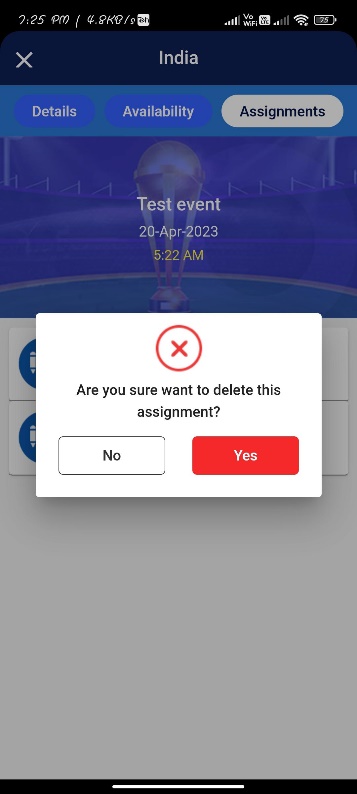
* TeamsEventAvailabilityComponent

**Note: Currently implemented only the events.**

**Functionality:**

* Created **getAvailabilityList()** methodand call it inside the ngOnInit().
* Inside the **getAvailabilityList()** method using **AuthService’s getListOfData()** method to get the event member availability list. Passing two parameters for the method **getListOfData(apiEventAvailability+ queryParam, Header).** The first param is an environment variable which is having an API endpoint for getEventMemberAvailability with the concatenation of queryparam which is an eventId and the second one is a token.

# 5.2.2.3 Assignments

**  **

**Component Name:**

* TeamsEventAssignmentsComponent

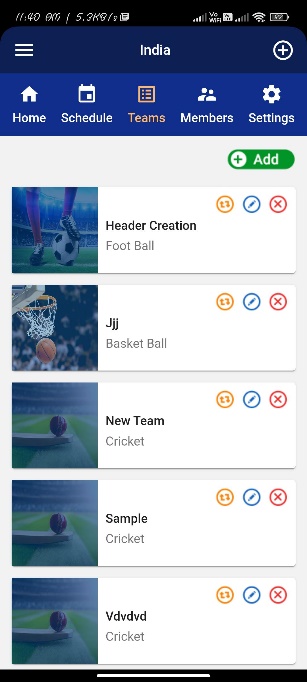
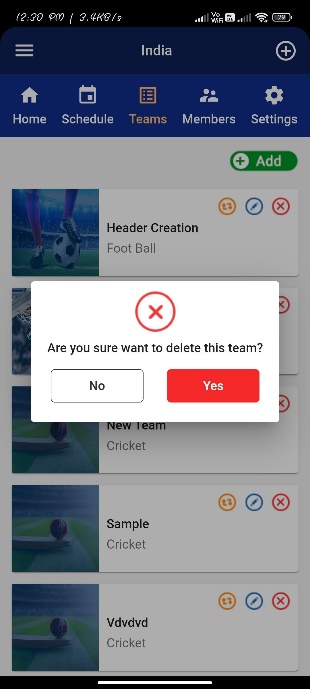
**Model Name**:

* SaveMemberForAssignments

Functionality:

* Created **getAssigendDetails()** method and call it inside the ngOnInit() method.
* Inside the **getAssigendDetails()** using **AuthService’s getListOfData()** method to get the event/game assignment lists with members assigned and member lists for the respective team. Passing two parameters for the method **getListOfData(apiGetAssignmentsForEvent / apiGetAssignmentsForGame + queryParam, Header).** The first param is an environment variable which is having an API endpoint for getAssignments for event/game with the concatenation of queryparam which is an eventId / gameId and the second one is a token.
* When editing the assignments click the edit icon **openDialogWithTemplateRef()** method will be called and it will open the edit assignments popup with the help of matDialog.
* After editing the assignement and assign to the particular team member then clicking the save button **saveAssignment()** method will be called.
* Inside the **saveAssignment**() method manually initialize the values to SaveMemberForAssignments model. Using the **AuthService’s** **postDatatoServiceWithToken**() method which is to make an HTTP post call to save the member for the respective assignment. Then passing three parameters to the method **postDatatoServiceWithToken** (**apiSaveMemberForEventAssignments / apiSaveMemberForGameAssignments, SaveMemberForAssignmentsMOdel, Header).** The first param is an **environment** variable which is having an API endpoint for saving the member for respective game / event assignment and the second one is **SaveMemberForAssignmentsModel** object and third one is a token.
* While deleting the assignment again **openDialogWithTemplateRef()** method will be called and it will open the delete popup with the help of matDialog. If clicking the yes button **deleteAssignment()** method will be called, inside this using the **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token to delete the assignment. Passing two parameters to the method **postDataToServiceWithToken (apiDeleteAssignmentForEvent / apiDeleteAssignmentForGame + queryParam, Header).** The first param is an environment variable which is having an API endpoint for deleting the game / event assignment with the concatenation of queryparam which is an eventId/ gameId , teamId, assignemntId, userName and the second one is a token.

# 5.3 Teams

** **

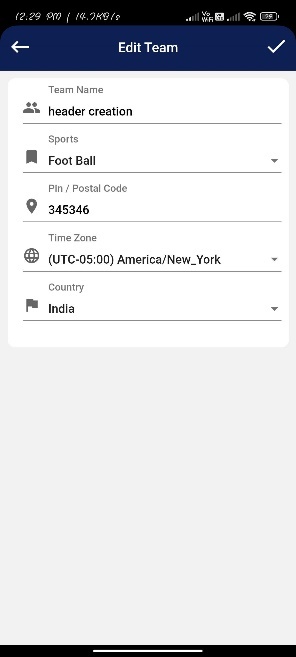
**Component Name:**

* MyteamsComponent

**Functionality:**

* Inside the constructor set the toastr messages based on the action message with the help of **ToasrService**.
* Created **getTeamsList()** method and call it inside the ngOnInit(). Inside this use the **AuthService’s getListOfData()** method to get the team lists for their username. Passing two parameters for the method **getListOfData(apiTeamList + queryParam, Header).** The first param is an environment variable which is having an API endpoint for getTeamList with the concatenation of queryparam which is a username and the second one is a token.
* When clicking the add button the **createTeam()** method will be called and it will redirect to the createTeam screen to create a new team.
* If clicking the particular team **getTeamById()** method will be called and it will redirect to **teamDetails** screen. Here team details are shown.
* If clicking the shift(first) icon the **openDialogTemplateRef()** method will be called and opened the team shifting confirmation popup with the help of matDialog. If clicking the yes button the **changeTeamById()** method will be called and shifted to the selected team.
* If clicking the edit icon **editTeamById()** method will be called and it will redirect to the **editTeam** screen.
* If clicking delete icon **openDialogWithTemplateRef()** method will be called and opened the delete confirmation popup with the help of matDialog. If clicking the yes button the **deleteTeamById()** method will be called. Inside this method using **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token to delete a team. Passing two parameters to the method **postDataToServiceWithToken (apiDeleteTeam + queryParam, Header).** The first param is an environment variable which is having an API endpoint for createTeam with the concatenation of query param which is the username and teamId and the second one is a token.

# 5.3.1 Edit Team

****

**Component Name:**

* CreateTeamComponent

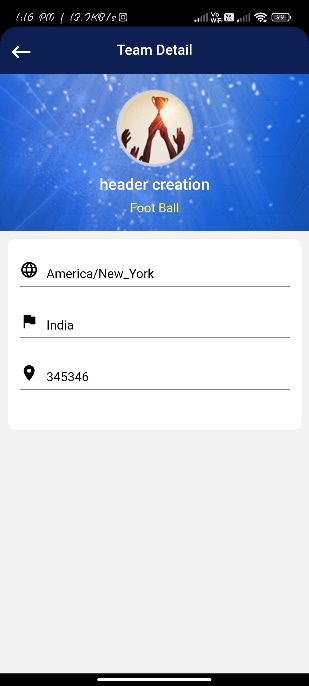
**Model Name:**

* TeamModel

**Functionality:**

* Inside the **ngOnInit()** method first check the localstorage action messages. If it is an edit action the top bar dynamically changed. Then make an API call to get the respective team details. Using **AuthService’s** **getListOfData**() method which is to make an HTTP get call to get the team details. Then passing two parameters to the method **getListOfData** (**apiViewTeam + queyParam, Headers**). The first param is an **environment** variable which is having an API endpoint for getTeamDetails with the concatenation of queryparam which is a teamId and the second one is a token.
* Then manually initialize the team details to TeamModel and bind the team model values to the respective html fields.
* After updating a team and clicking the tick button, the **submitForm**() method will be called. Inside this method check whether the action type is create or edit. If it is an edit the **editTeam()** method will be called.
* Inside the **editTeam()** method use **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token to update the team. Passing three parameters to the method **postDataToServiceWithToken (apiEditTeam, teamModel, Header).** The first param is an environment variable which is having an API endpoint for updating the team and the second is a TeamModel object and the third is a token.
* And the API response statuscode is 200, it will redirect to the respective screen.

# 5.3.2 Team Details



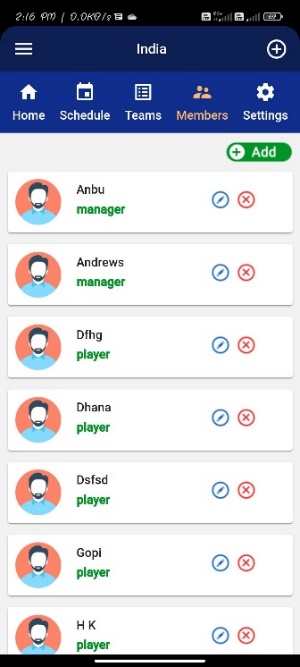
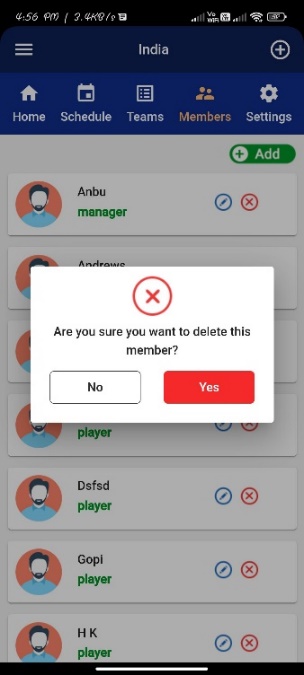
**Component Name:**

* TeamsTeamdetailsComponent

**Functionality:**

* Inside the **ngOnInit()** Using **AuthService’s** **getListOfData**() method which is to make an HTTP get call to get the team details. Then passing two parameters to the method **getListOfData** (**apiSingleTeamData + queyParam, Headers**). The first param is an **environment** variable which is having an API endpoint for getTeamDetails with the concatenation of queryparam which is a teamId, and the second one is a token.

# 5.4 Members

** **

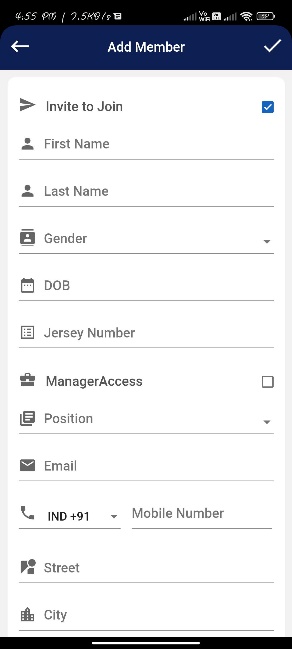
**Component Name:**

* TeamsMemberComponent

**Functionality:**

* Inside the constructor set the toastr messages based on the action message with the help of **ToasrService**.
* Created **initTeamMembers()**methodand inside this calling API to get the member list. Using **AuthService’s** **getListOfData**() method which is to make an HTTP get call to get the member list. Then passing two parameters to the method **getListOfData** (**apiListMember + queyParam, Headers**). The first param is an **environment** variable which is having an API endpoint for getMemberList with the concatenation of queryparam which is a teamId and username, and the second one is a token.
* When clicking the add button the **teamSetup()** will be called and it will redirect to the addmember screen, let to add a new member.
* When clicking the particular member **getTeamById()** method will be called and it will redirect to the **memberdetails** screen.
* When clicking edit icon **editTeamMemberById()** method will be called and it will redirect to the add member screen to edit a member.
* When clicking the delete icon **deleteMember()** method will be called. Inside this method using **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token. This will be deleting the selected member. Passing three parameters to the method **postDataToServiceWithToken (apiDeleteMember, TeamMemberData, Header).** The first param is an environment variable which is having an API endpoint for deleteMember and the second is member data variable and the third is a token.

# 5.4.1 Add New Member

****

**Component Name:**

* AddmemberComponent

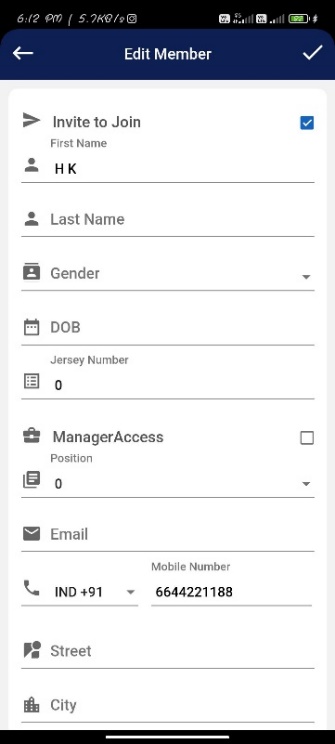
**Model Name:**

* PlayerModel

**Functionality:**

* Inside the constructor checking the localstorage action name. If it is an edit action the component will work for an edit member and the top bar also dynamically change. Otherwise it will work for add member.
* **selectNotifyTeam()** method is used for passing a string value “manager” or “player” while checking and unchecking the ManagerAccess checkbox.
* When filled the member details and click the tick button the **addPlayer()** method will be called. Inside this method check the action messages. If it is an edit action editPlayer() method will be called, otherwise **createPlayer()** method will be called.
* Inside the **createPlayer()** methodUsing the **AuthService’s** **postDatatoServiceWithToken**() method which is to make an HTTP post call to add the members. Then passing three parameters to the method **postDatatoServiceWithToken** (**apiAddPlayer, PlayerModel, Header).** The first param is an **environment** variable which is having an API endpoint for addPlayers and the second one is **playerModel** object and third one is a token.
* If the API status code is 200, it will redirect to the respective screen.
* **Omit\_special\_char()** methodis used for restrict the numbers and special characters while entering.
* **Key**() method is used for restrict the alphabets and special characters while entering.

# 5.4.2 Edit Member

****

**Component Name:**

* AddmemberComponent

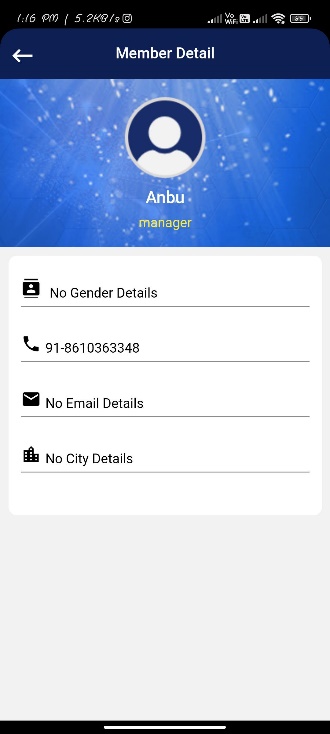
**Model Name:**

* PlayerModel

**Functionality:**

* Inside the constructor check the localstorage **action** **name**. If it is an edit action the component will work for an edit member and the top bar also dynamically change.
* And also make an API call to get the member details. Using **AuthService’s getListOfData()** method to get the member details for their respective member. Passing two parameters for the method **getListOfData(apiEditTeamPlayer + queryParam, Header).** The first param is an environment variable which is having an API endpoint for getPlayerDetails with the concatenation of queryparam which is a playerId and the second one is a token. Then the player details and manually initialized to playermodel and bind the playerModel values to the respective html fields.
* After updating the player and then clicking the tick button the **addPlayer()** method will be called. Inside this method check the action messages. If it is an edit action **editPlayer**() method will be called.
* Inside the **editPlayer**() method Using **AuthService’s postDataToServiceWithToken()** method to update the member. Passing three parameters for the method **postDataToServiceWithToken (apiEditTeamNewPlayer, PlayerModel, Header).** The first param is an environment variable which is having an API endpoint for updatePlayer second one is a PlayerModel object and the third one is a token.
* If the player is updated successfully, it will redirect to the respective screen.

# 5.4.3 Member Details



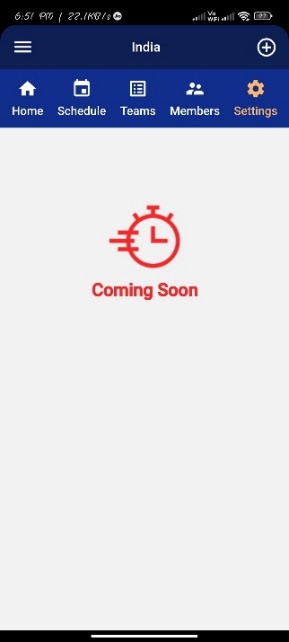
**Component Name:**

* TeamsMemberdetailsComponent

**Functionality:**

* Inside the **ngOnInit()** using **AuthService’s** **getListOfData**() method which is to make an HTTP get call to get the member details. Then passing two parameters to the method **getListOfData** (**apiMemberDetails + queyParam, Headers**). The first param is an **environment** variable which is having an API endpoint for getPlayerDetails with the concatenation of queryparam which is a playerId, and the second one is a token.

# 5.5 Settings

****

**Component Name:**

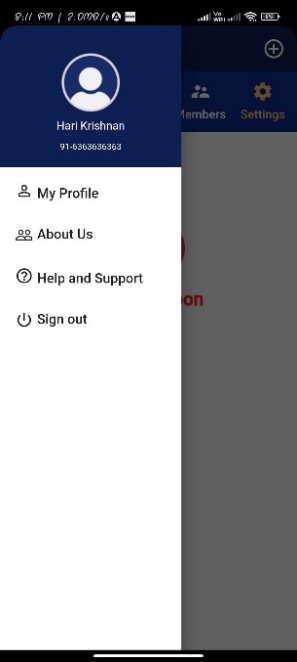
* TeamsSettingsComponent

**Functionalities:**

* The below functionalities are there in this component. Currently in a hidden stage. Will implement it later.

1. Change Language
2. Notification Preferences
3. Delete Account

# 6.Hamburger Menu

****

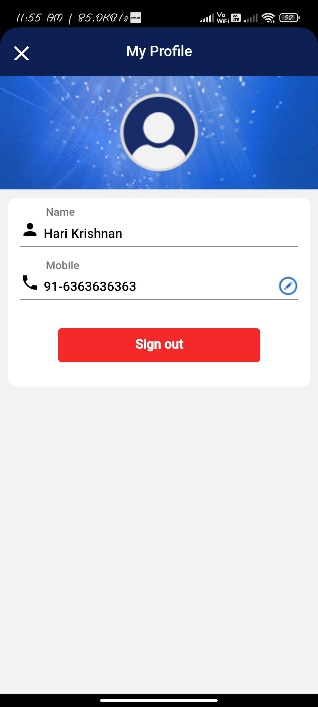
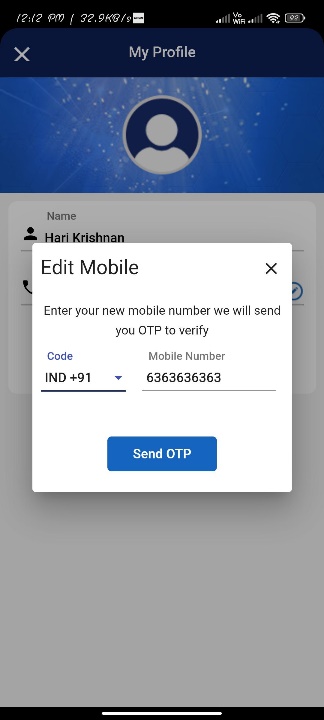
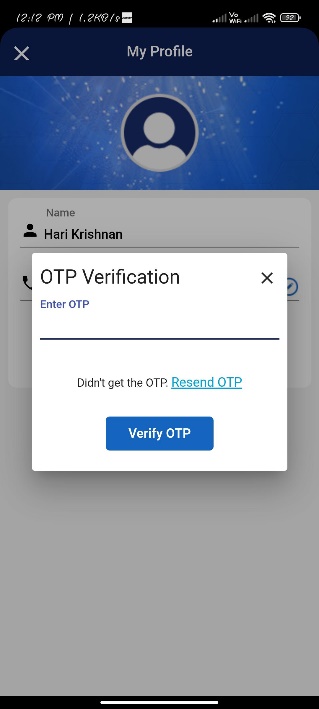
**Component Name:**

* SideBarMenuComponent

**Functionality:**

* Clicking My Profile, it will redirect to the MyprofileComponent.
* Clicking About Us, it will redirect to the AboutusComponent.
* Clicking Help and Support, it will redirect to the HelpandsupportComponent.
* Clicking sign out the **openDialogTemplateRef()** method will be called and it will open the signout popup with the help of matDialog. When click the yes button **logout()** mwthod will be called and inside this use **localStorageService’s clearAllLocalStorage()** method to clear the local storage and then will redirect to the login.

# 6.1 My Profile

**  **

**Component Name:**

* MyprofileComponent

**Model Name:**

* ProfileDetails

**Functionality:**

* Inside the constructor set the nav bar title and buttons with the help of NavigationModel.
* When clicking the Sign out button the **logOut()** method will be called and inside this use **localStorageService’s clearAllLocalStorage()** method to clear the local storage and then will redirect to the login.
* When clicking the edit icon the **openDialogWithTemplateRef**() method will be called and will open the edit mobile number popup with the help of matDialog.
* After editing the mobile number and click the send OTP button the **editProfile()** method will be called. Inside this method use **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token. This will be sending the OTP for the updated mobile number. Passing two parameters to the method **postDataToServiceWithToken (apiMobileNumberUpdate + queryParam, TeamModel, Header).** The first param is an environment variable which is having an API endpoint for sendOTPforMobileNumberUpdate with concatenation of query param which is an userName and the second one is a token.
* If the api status code is 200, then it will open the **OTPverification** popup. After enter the OTP and click the verify OTP button **otpVerify()** method will be called. Inside this method using **AuthService’s postDataToServiceWithToken()** method which is to make an HTTP post call with the token. This will be verifying the OTP and update the mobile number. Passing three parameters to the method **postDataToServiceWithToken (apiProfileUpdate, ProfileDetailsModel, Header).** The first param is an environment variable which is having an API endpoint for update the user profile and the second one is **ProfileDetailsModel** object and the third one is a token.

# Common Files:

|  |  |  |
| --- | --- | --- |
| **File Name** | **Usage** | **File Type** |
| AuthService | All the http methods, spinners methods and getting authorization token methods are written in this file. | TS file |
| ErrorHandlerService | Handling front-end errors and calling api service to send the errors to service. | TS file |
| LocalStorageService | Store the necessary information / data to the local storage | TS file |
| Shared | Written common functions and storing temporary data. | TS file |
| AppConstants | Application constant strings are declared in this file. | TS file |
| environment | All the api endpoints are declared. | TS file |
| TopBarComponent | Common top bar for the login and signup flow. It will dynamically change the screen buttons and screen title. | Component |
| Model | All the model files are here. | Folder |
| SideBarMenuComponent | This component will be displaying hamburger menu which is common for all the screens. | Component |

**Note:**

* Whenever the API is called **AuthService’s** **showSpinner()** method will execute and start the spinner. After getting the API’s response **AuthService’s** **hideSpinner()** will execute and stop the spinner.