For front end, all my latest update is in the getTournament branch

FRONTEND Interface

1. addNewField

‘**addNewField**’ component is a blueprint, based on that, created ‘**newTournament**’ component, ‘**newWager**’ component and ‘**registerTeam**’ component.

1.1. addNewField

Parameters that need to be passed in	Explanation
inputs (array)	An array of string including names of the input text fields
hasDescription (boolean)	
hasUploadImage (boolean)	
buttonText(string)	
titleText(string)	
route (string)	Inside the handleSubmit function, based on the route name, fetch the corresponding function in zustand store

1.2. newTournament

Function	Need backend api?
Create new tournament	Yes, and I have created it

PS. For the uploading image part, at the current stage, I choosed to use convertToBase64 function to store image inside the mongoDB. In the furture, you may change to other uploading files approachs such as AWS S3 and Google Cloud Storage.

1.3. newWager

Function	Need backend api?
Create new wager	Yes

1.4. registerTeam

Function	Need backend api?
Register new team	Yes

2. disputeDetail+ disputeCard

This is a webpage that lets referees handle disputes.

Function	Need backend api?
Get all disputes	Yes
Swap the outcome (the winner	Yes

become the loser; the loser becomes the winner)	
Ban team	Yes
Dismiss the dispute	Yes

3. leaderBoard

This is a webpage that ranks the users by their points. It can achieve pagination.

Function	**Need backend api?**
Get all users in leaderBoard	Yes
See the next page	Yes
See the previous page	Yes

4. support

This is a customer support page. In the future, you only need to replace the dummy context with the actual context, I don't think we need api if we choose to maintain the customer support rules for a long time.

Function	**Need backend api?**
Search the keyword	No
Dropdown menu	No

5. auth

This is a auth page to let user sign in or sign up by using their emails and passwords, or do it directly with Google.

Function	**Need backend api?**
Sign in with email and password	Yes
Sign in with Google	Yes
Sign up with email, password, confirmed password, username	Yes
Sign up with Google	Yes

6. getTournaments

In the tournaments page, I created delete, update and close/open function to allow admin to manipulate these tournamens.

Function	Need backend api?
Fetch all tournaments from database	Yes, and I have created it
Delete one certain tournament	Yes, and I have created it
Update one certain tournament	Yes, and I have created it
Close/open one certain tournament	Yes, and I have created it

PS. For the fetching image part, at the current stage, I choosed to directly retrieve image from mongoDB and show it as base64 format. In the furture, you may change to other way like fetch them from AWS S3 or Google Cloud Storage.

For back end, all my latest update is in the tournament branch

BACKEND APIs

1. Get All Tournaments

Endpoint	Method	Request	Response
/api/v1/tournament	GET	null	{} an object with all tournaments inside

2. Create New Tournament

Because the 'referee' in 'Tournament' Schema is referred to 'User' Schema, so when create a tournament and randomly assign it to a referee, the assigned referee's tournamentHistory is also correspondingly added the tournament.

Endpoint	Method	Request	Response
/api/v1/tournament /create	POST	name, game, gameMode, startDate, endDate, prize, platform, entryFee, limit, image, matchType,	If success: 201; If fail: 400

3. Update a Tournament By its Id

Endpoint	Method	Request	Response

/api/v1/tournament /update/:tid	PATCH	Name,
		matchType,
		limit,
		startDate,
		endDate,
		Prize,
		EntryFee,
		platform
			If success: 200;
			If invalid id: 400;
			If can't find that tournament: 404;
			If others:500

4. Delete a Tournament By its Id

Because the ‘referee’ in ‘Tournament’ Schema is referred to ‘User’ Schema, so when delete a tournament, this tournament will also be removed from the assigned referee’s tournamentHistory.

Endpoint	Method	Request	Response
/api/v1/tournament /delete/:tid	**DELETE**	**null**	**If success: 201; If fail: 400**

5. Close a Tournament By its Id

Endpoint	Method	Request	Response
/api/v1/tournament /close/:tid	**PATCH**	**null**	**If success: 201; If fail: 400**

6. Open a Tournament By its Id

Endpoint	Method	Request	Response
/api/v1/tournament /open/:tid	**PATCH**	**null**	**If success: 201; If fail: 400**