4FitTime - Gym Area Scheduling System

Project Design Document

Team Number: 23

Team Members: Aiden Vigue, Connor Federoff, JongIn Lee, Noah Durand, Saimaurya Kanagala

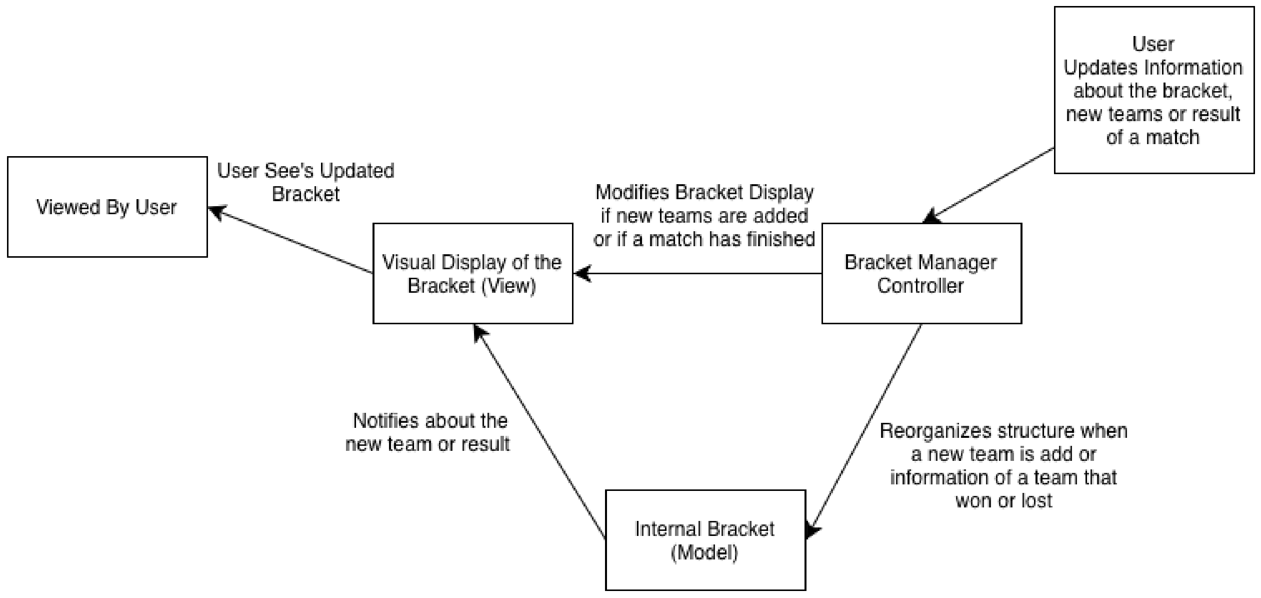
Purpose

College campuses with shared gymnasium and recreational facilities lack centralized, user-friendly systems that allow students and organizations to reserve spaces for practices, scrimmages, and tournaments. Current solutions often rely on fragmented processes such as email chains, physical sign-up sheets, or outdated booking systems that do not support real-time availability, team coordination, or competitive event organization. As a result, students often experience scheduling conflicts, miscommunication, and difficulty organizing athletic activities.

The purpose of this project is to design and implement a unified scheduling and sport management platform. The system enables users to to view gym avalibility, reserve time slots, form and manage teams, communicate through integrated messaging, and participate in tournaments with automated bracket generation and result tracking. By combining scheduling and competitive features in a single platform, the system aims to improve the efficiency, transparency, and overall experience of organizing and participating in sports at our campus.

Design Outline

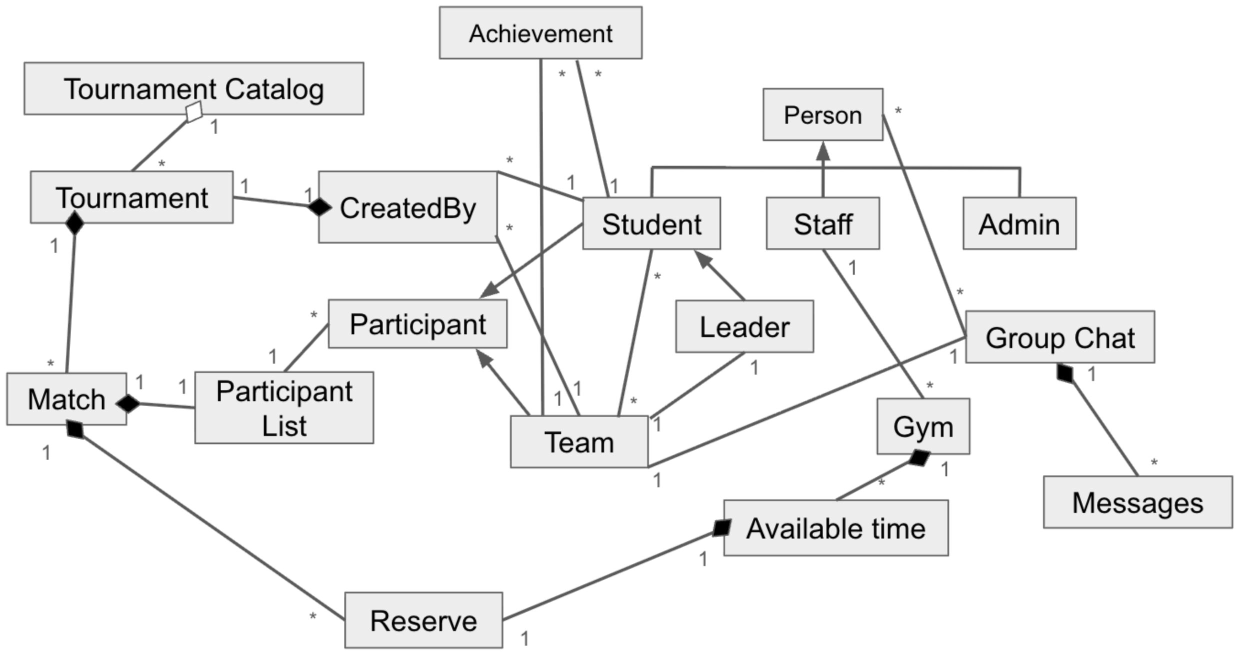
This project will be a web application that allows users to join teams containing other users and compete in tournaments created by other users or the staff. Results will be handled either by users or tournament organizers depending on the permissions given. This application will use the MVC model in which a controller will update a view and model so that the display of the bracket will be ready when the model has updated it's internal structure with a new team or results of a match.


User
Updates Information about the bracket, new teams or result of a match
Viewed By User
User See's Updated Bracket
Modifies Bracket Display if new teams are added or if a match has finished
Visual Display of the Bracket (View)
Bracket Manager Controller
Notifies about the new team or result
Reorganizes structure when a new team is add or information of a team that won or lost
Internal Bracket (Model)

- Bracket Manager (Controller)
    - Bracket Manager handles how to update brackets when a new team is added or when a result comes in
    - When a new team is added or removed, it will decide if the current bracket needs to be changed to a different size and the rankings of the teams will change
    - When a result of a match is added, it will tell which match needs to be updated to the View and give the result information to the model

- Internal Bracket
    - Stores all of the teams in the tournament
    - Stores which teams advanced to the next round of a tournament
    - Stores all the match data for each completematch
- Viewer Display of the Bracket
    - Shows teams off in a traditional bracket where teams that win play other teams that win until only 1 remain
    - Shows scores of matches that have happened and shows where teams are currently at in the bracket

High-level Structure of the Software



The software will have a person sign up for the website and then become a student, staff or admin. Students will play in tournaments and staff, and admin will give available times for gyms or locations. Students can then create teams to become the team leader and can then add other students to the teams. Students can then create tournaments so that other teams can play against their team. They will then be able to reverse gyms for a specific match. After being in a team, they can join tournaments and compete to see if they are the best team in the tournament. User/Team can start messaging with each other and communicate with their peers. Student/Team will be able to look at their achievements.

Design Issue

Functional Issues:

- What information will we need from the user in order to create an account?

    - Option 1: username, password, account access
    - Option 2: username, Password, Full name, Purdue email, phone number, account access
    - Option 3: Purdue Email, Purdue Password, phone number, account access, Full name

  Choice: Option 2

  Justification: Although option 3 would be a very strong option to choose since our users would be either Purdue students or Purdue faculty to both use and create new events within the application, it takes time and effort in order to gain access to the Api's that allow us to utilize Purdue's own verification system. If we were able to access Purdue's verification API then option three would be the better long term switch to choose, but for now option 2 allows us to ensure only Purdue users can join by making them use their Purdue email, and lets us set up a basic account with from information and ways of contacting them. It also creates a basic username and password verification method, and the email and phone number can be used to create a two-factor authentication system for better security.

- What details would I need to create tournament and tournament brackets within the application
    - Option 1: Purdue email, Interests, Team members
    - Option 2: Full Name, Tournament records, team members
    - Option 3: Purdue Email, Past tournament records, skill levels, Team members, Sports played

  Choice: Option 3

  Justification: With option 3, we will be able to make multiple different tournament brackets, each backets having players of the same skill gap, so that we wouldn't create random tournaments where the winners would dominate with a land slide, and the tournaments would all be equal in skill. Using the Purdue email will allow us to assign the players to their brackets and email them details about the tournaments so we can make sure this information would get to the participants.

- How should we let members within the tournaments communicate with each other?
    - Option 1: Giant group chats with everyone in the tournament

    - Option 2: Provide emails with all the members within the team

    - Option 3: Create small team group chats and one giant tournament discussion board

Choice: Option 3

Justification: If we just create giant group chats, they could get very messy and complicated with teams trying to communicate with each other all at once, and the chance for members to message entire groups of people in the tournament without them wanting to, making messy communication. For option 2, it could be the easy way, but then emails could either be flooded with messages from tournament officials. Option 3 allows for teams to communicate with one another in their own private group chats, and by having a giant discussion board, tournament officials can have an easy place to voice their opinions without everyone else within the tournament talking over them or hiding the notifications.

- How will we make sure that certain members can have more privileges compared to other members?

    - Option 1: Have a separate website sign up for admin users that will require users to be enrolled by other admins before they can log in

    - Option 2: Have a database that holds admin logins that can be accessed by admin officials, allowing for admins to log in regularly

Choice: Option 2

Justification: We chose option 2 because it allows us to use the same website, but by allowing these original admin members to log into and add the information for the new admin members, we don't require a unique sign up method for them that could have the chance to be abused or exploited by other students. Admins would have to be enlisted by previous admins, which would let them log in and access admin commands like creating tournaments or managing user accounts or other information that only higher ups should access.

- How will we manage group chat messages and adding/deleting members in group chats?

    - Option 1: Allow team leaders to manage adding/deleting users and monitor messages

    - Option 2: Allow team leaders to manage adding/deleting users, and admins and team leaders to monitor messages

- Option 3: Allow team leaders to manage adding/deleting users, and admins, team leaders, AI, and word banks to monitor messages

- Option 4: Allow team leaders to manage adding/deleting users, and admins, team leaders, and word banks to monitor messages

Choice: Option 4

Justification: We need to make sure that since these are school messaging groups chats meant for communication for these Purdue events, they shouldn't be exploited or used for other nefarious purposes or can send harmful messages to other members within the group. Although option 3 would allow us to have the most elements monitoring, feeding the messages of our students to an AI to analyze every time could be time consuming and on the unethical side, allowing the AI to read the information sent for every Purdue member. By allowing either the team members, admins monitoring reports or just flagged messages, or having a word bank analyse messages for any harmful content while being sent, we can ensure the integrity of these group chats, make sure that members with the right clearance can access and monitor the contents of the group chats, and just have certain words already filtered and flagged so they won't be send in the first place, creating a safe and secure messaging environment.

Non Functional Issues:

1. How are we going to host our backend services

   Option 1: AWS Option 2: Vercel **Option 3: ArgoCD** Option 4: OCI

   Decision: We chose to use ArgoCD and separate our databases from our backend system. ArgoCD gives us wide flexibility with how we want to structure connections from our databases to the backend, and from the backend to the ingress load balancer. ArgoCD also takes care of updating the deployment whenever we decide to make code changes.

2. What programming languages and frameworks will we use for implementing the backend

   Option 1: Typescript/NestJS Option 2: Go Option 3: Spring Boot

Decision: We chose to use Typescript and NestJS as the team is most comfortable with the Typescript / JavaScript runtime, and it allows us to iterate faster than using a full Spring Boot platform, and Go is more suitable for CLI tools and services that require the smallest possible latency at the expense of developer experience.

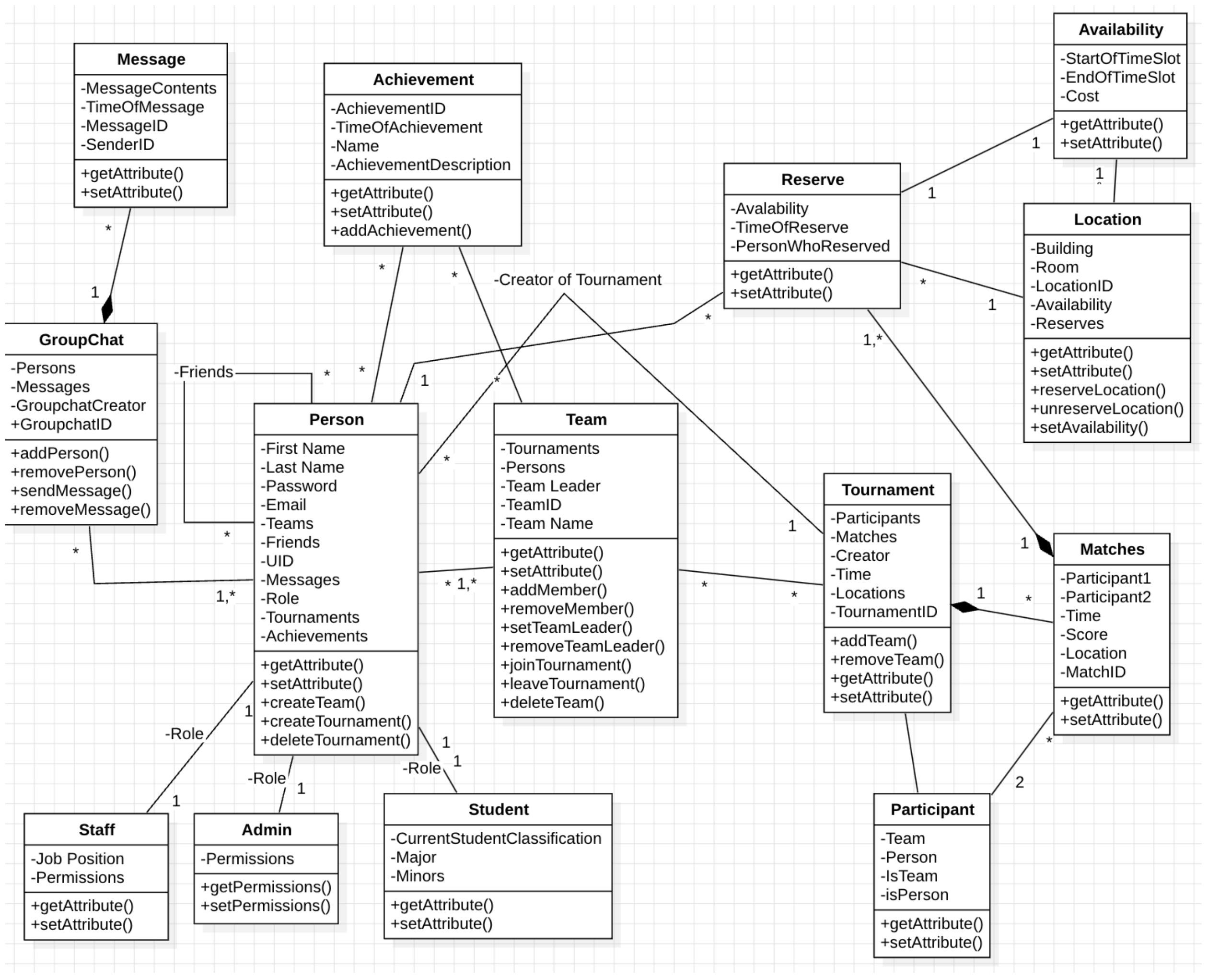
3. How will we ensure that tournament data is appropriately updated in the database / reflected on user's devices as results come in.

   Option 1: WebSockets Option 2: SSE **Option 3: Polling the server**

   Decision: We chose to simply poll the server for now. We may eventually implement a socket based interface to allow the app to get real time state updates, but if we design our polling interface well, we can utilize caching mechanisms like ETag and Last-Modified to reduce the amount of data sent to the client, while still having the simplicity of polling.

Design Details

Class Diagram



Class Purpose

- **Person**
  - Created every time someone signs up

    - First and Last Name, Password and Email data is collected when a user signs up
    - Each user has a Unique assigned ID
    - Every User can added other users as friends
    - Every User can join teams
    - Every User can create Tournaments
    - Every User will have achievements they have earned
- Student/ Staff/ Admin
    - They are a role of person, and they have different permission. Student can add student specific information. Staff can add new gyms. Admin has admin permission
- Team
    - It represents a group of students with one leader, who is a student who manages the team. It stores team ID/name, member list (Persons), team leader, and tournaments the team is involved in.
    - Every team will have a team leader who decides who gets added to a team, who gets removed from a team and which tournaments a team enters
    - Leader is a role assigned to a Student and not a physical part of the Team. Also, if the Leader is deleted, a new Leader could be assigned to the Team.
- Tournament
    - It represents a tournament. It stores tournament ID, creator, participants, matches, time, and locations.
- Matches

    - It represents individuals match within tournament. It stores participants, time, location, score, match ID.

- Participant
    - It represents an entity that can participate in a match/tournament. It can represent either a Person or a Team
- Group Chat
    - Each group chat is created by a user
    - Each group chat will have at least 1 person but can have many more
    - Each group chat will contain messages
    - Each group chat will have it’s own unique ID
- Messages
    - Each Message will have a sender id for the person who sent it
    - Each message will contain what the user sent
    - Each message will have a time of when it was sent and be given a unique ID for the message.
- Achievement
    - Each Achievement will be given a unique ID
    - There will be a name and time for each achievement along with a description of which tournament it was reward for and why it was rewarded
- Location
    - It represents a physical building that could be used for matches.

    - It stores the address, building, locationId

- Availability
    - It represents a time slot at a location
    - It stores start/end time slot, cost

- Reserve
    - A reservation record that ties a Person to a specific Availability at a specific time.

Relationship and Interaction

- Person – Person
    - Person has a self-association labeled Friends. This models that each Person can have multiple friends and each friendship connects two Persons.

- Student, Staff, and Admin - Person
    - Student, Staff, and Admin are all types of Person. Therefore, Student, Staff, and Admin have generalization (inheritance) relationships with Person. All common attributes such as name, email, and authentication of information are defined in Person and inherited by its subclasses.

- Team – Person
    - Teams store a list of Persons, and each Person may belong to multiple teams. Therefore, Teams and Person have a many-to-many association.

- Team – Team Leader
    - Teams include a designated Team Leader (a specific Person). This is modeled as a association/role relationship.

- Person/Team – Tournament
    - Each Tournament has a creator, which could be a Person/Team. A Person/Team may create many tournaments, but each tournament is created by one person/Team. Therefore, Person/Team and Tournaments have a one-to-many association relationship.
- Tournament - Matches
    - Each tournament could have multiple matches. Matches could be created once a tournament is opened, so Match depends on the Tournament. Therefore, there are one-to-many Composition relationships.
- Tournament - Paticipant
    - Tournaments include Participants. A tournament can have many participants, and a participant may join multiple tournaments. Therefore, Tournaments and Participant have many-to-many relationship.
- Participant – Student/Team
    - Participant will contain either a Team or Person since there are tournament where one person plays another person for a match like tennis
    - Participant is a role for Student/Team and it represent people or groups who are participating in match. Therefore, Participant has a generalization relationship with Student/Team.
- Matches – Participant
    - Each Match involves Participants. Matches includes exactly two participants. Therefore, Matches and Participant have an association where a match connects to two participants.
- Staff – Location

   - Staff members manage Gyms. A Staff member may manage multiple Gyms, and each Gym is managed by one Staff member. Therefore, there is a one-to-many association between Staff and Gym.

- Location – Availability

   - Each Location contains multiple Available Time slots. An Available Time slot only exists for a specific Gym. Therefore, Location has a one-to-many composition relationship with Availability.

- Availability – Reserve

   - Each Reservation refers to exactly one Available Time slot, and each Available Time slot may be reserved by at most one Reservation. Therefore, there is a one-to-one association between Available Time and Reserve.

- Match - Reserve

   - A Match may require a reserving a gym time. Matches may need different reservations. Therefore, there is a one-to-many association between Match and Reserve.

- Group Chat – Message

   - A Group Chat contains Messages. If a Group Chat is deleted, its messages should be deleted with it. Therefore, Group Chat and Message have a one-to-many composition relationship.

- Group Chat – Person

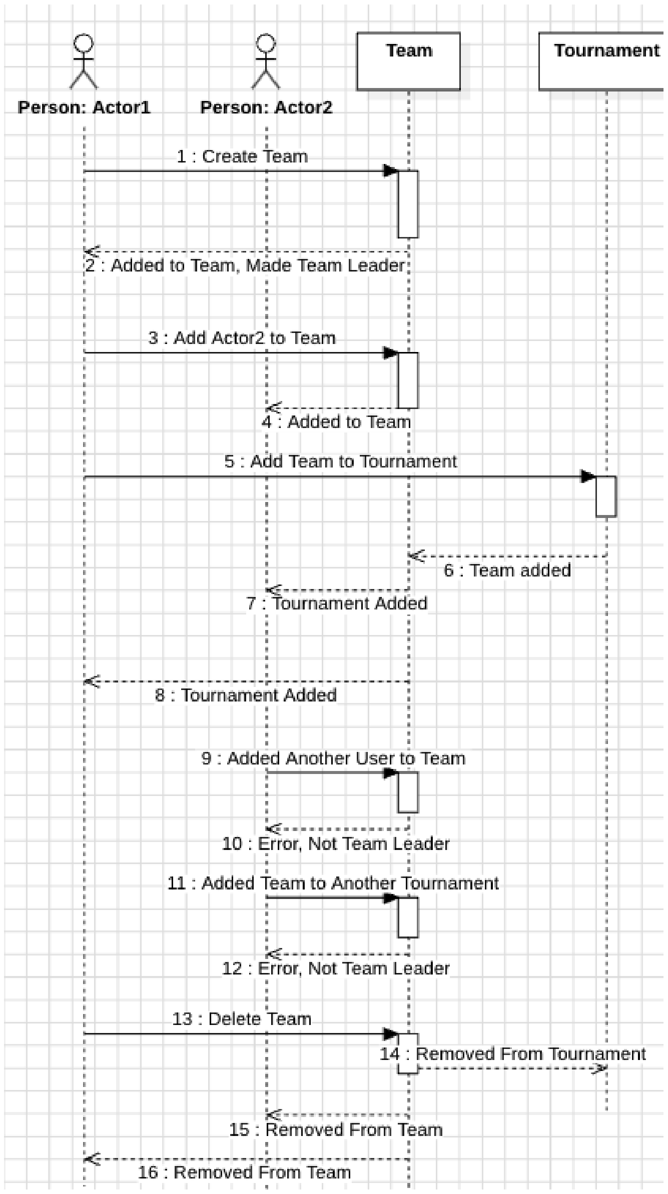
   - Group Chat contains Persons, and a Person can be in multiple group chats. Therefore, Person and Group Chat have many-to-many relationship.

- Person/Team – Achievement

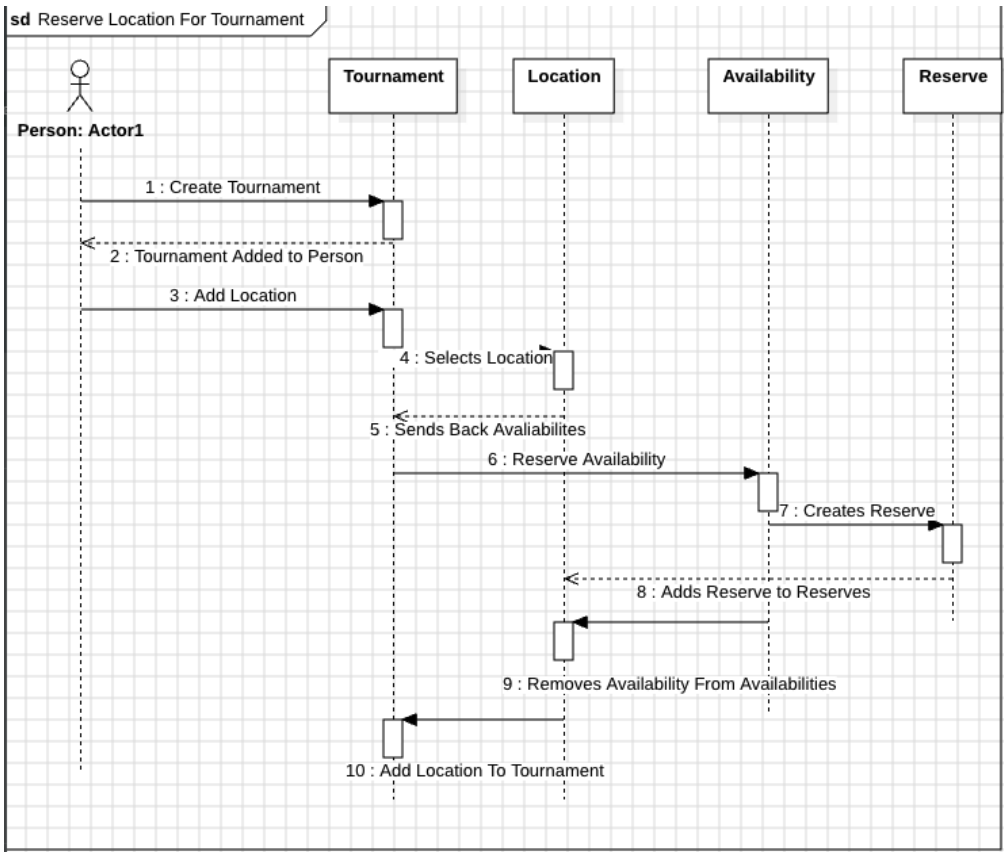
   - Person/Team can earn Achievements. A Person/Team can earn multiple achievements, and an Achievement can apply to many Persons / Team. Therefore, Person and Achievement have a many-to-many relationship.

Sequence Diagrams

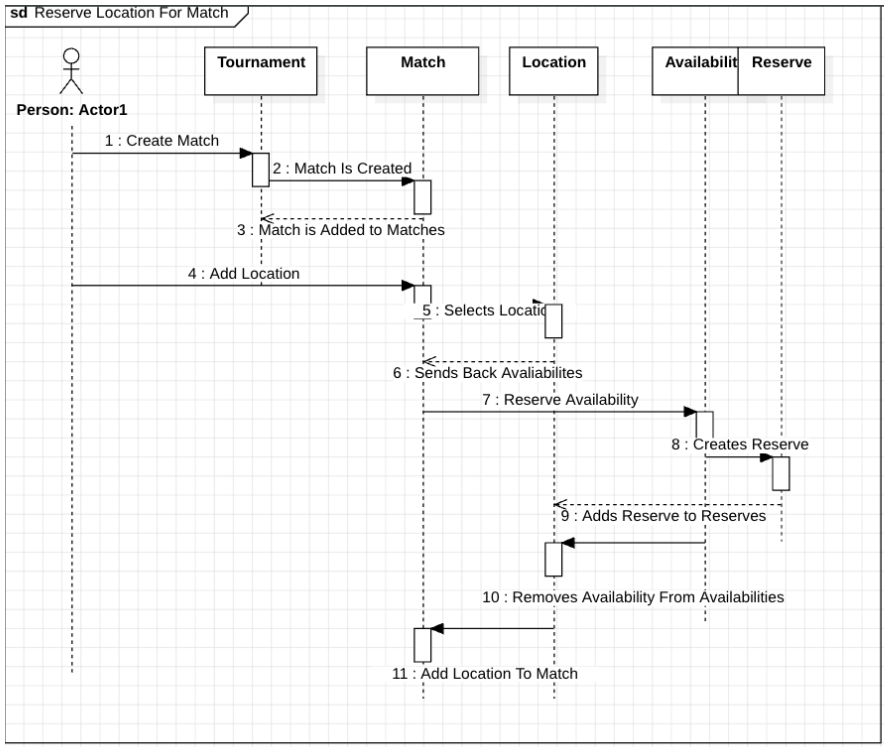
Team Actions


Person: Actor1
Person: Actor2
Team
Tournament
1 : Create Team
2 : Added to Team, Made Team Leader
3 : Add Actor2 to Team
4 : Added to Team
5 : Add Team to Tournament
6 : Team added
7 : Tournament Added
8 : Tournament Added
9 : Added Another User to Team
10 : Error, Not Team Leader
11 : Added Team to Another Tournament
12 : Error, Not Team Leader
13 : Delete Team
14 : Removed From Tournament
15 : Removed From Team
16 : Removed From Team

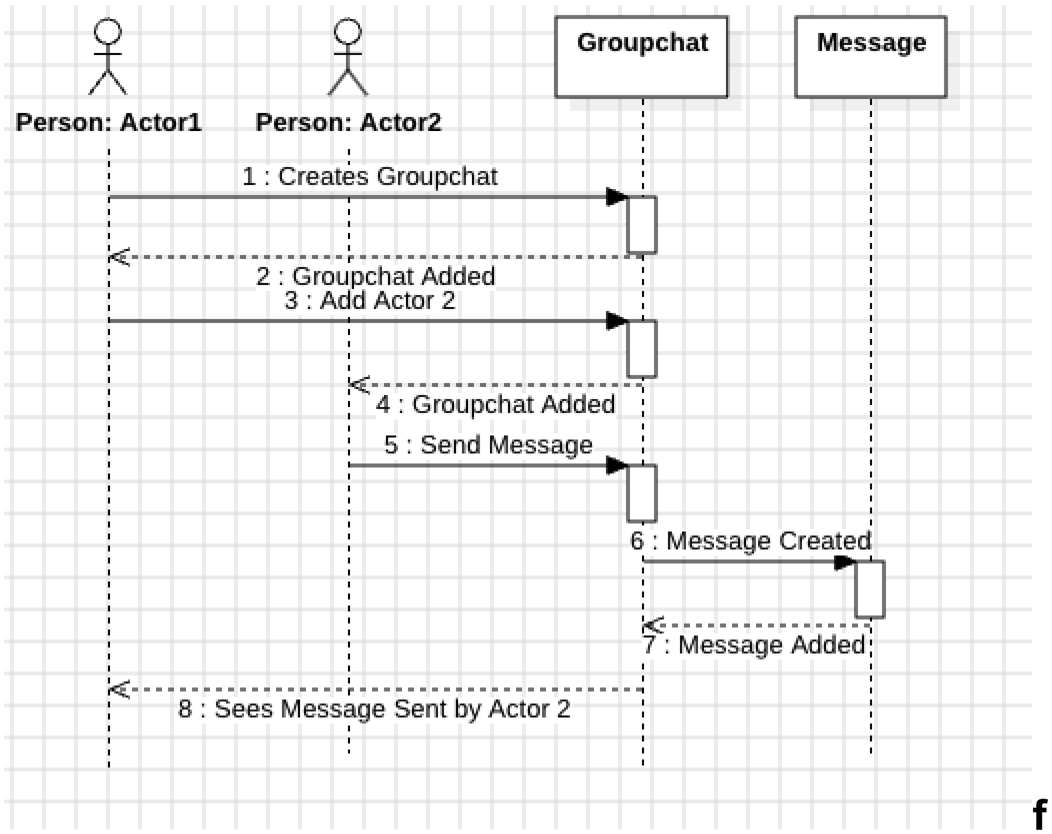
Adding a Location to a Tournament


sd Reserve Location For Tournament
Person: Actor1
Tournament
Location
Availability
Reserve
1 : Create Tournament
2 : Tournament Added to Person
3 : Add Location
4 : Selects Location
5 : Sends Back Avaliabilites
6 : Reserve Availability
7 : Creates Reserve
8 : Adds Reserve to Reserves
9 : Removes Availability From Availabilities
10 : Add Location To Tournament

Adding Location to Match


sd Reserve Location For Match
Person: Actor1
Tournament
Match
Location
Availabilit
Reserve
1 : Create Match
2 : Match Is Created
3 : Match is Added to Matches
4 : Add Location
5 : Selects Locatio
6 : Sends Back Avaliabilites
7 : Reserve Availability
8 : Creates Reserve
9 : Adds Reserve to Reserves
10 : Removes Availability From Availabilities
11 : Add Location To Match

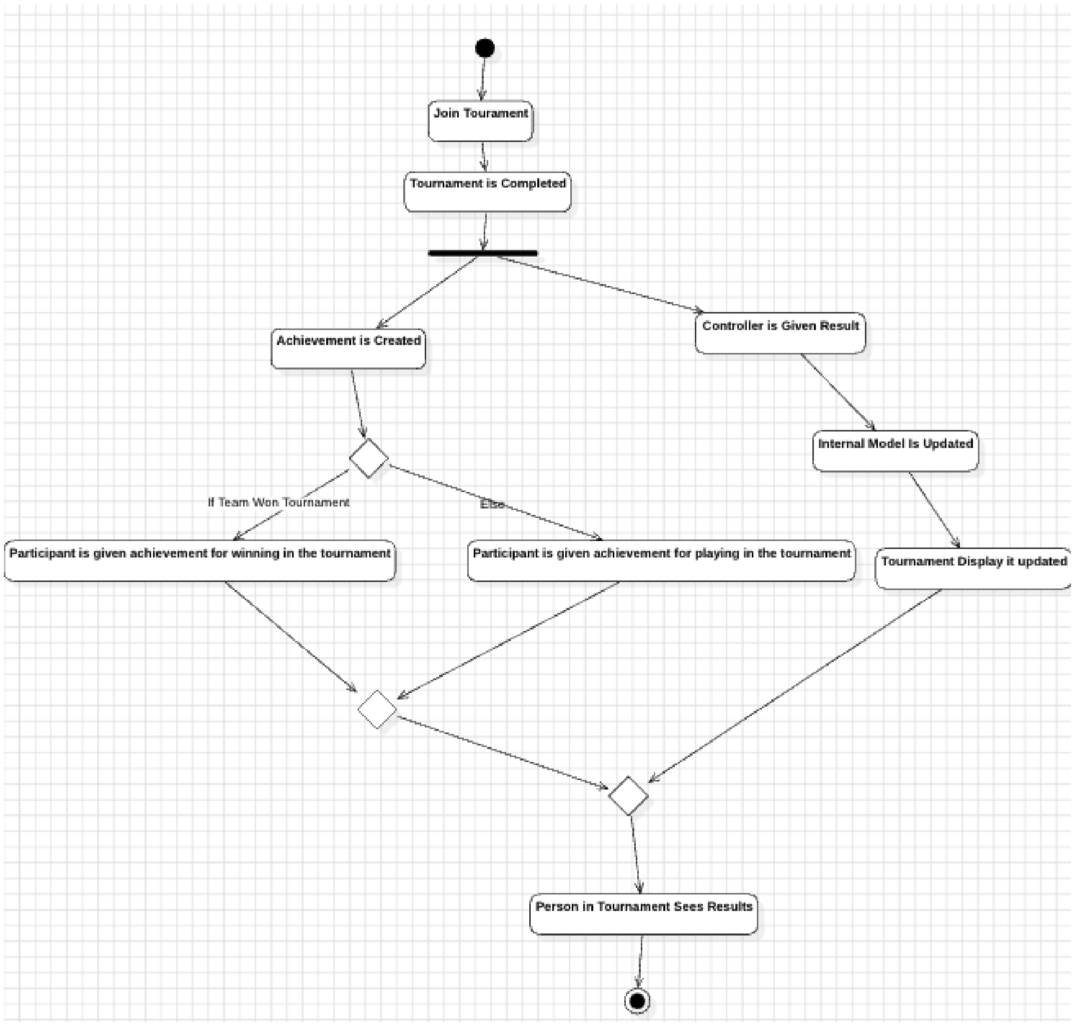
Messages


Person: Actor1
Person: Actor2
Groupchat
Message
1 : Creates Groupchat
2 : Groupchat Added
3 : Add Actor 2
4 : Groupchat Added
5 : Send Message
6 : Message Created
7 : Message Added
8 : Sees Message Sent by Actor 2

f

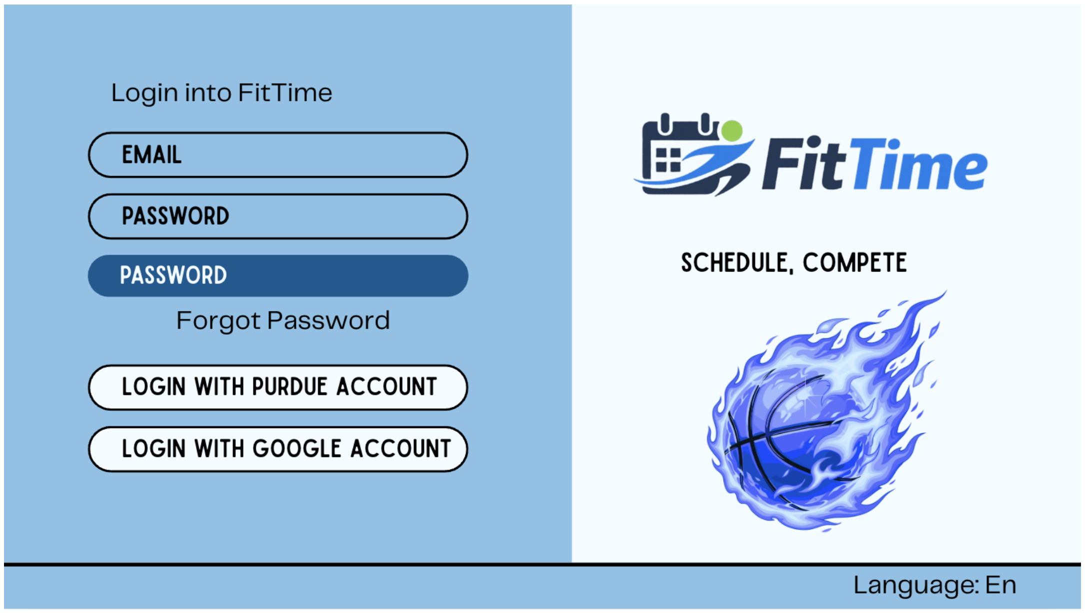
Activity Diagram for Tournament's Ending

After the tournament ends, the software must both give achievements to participants that played in the tournament and the participant that won and well as update the display of the tournament and the internal model of the tournament. The achievement is done so that users of the system are more enticed to enter more tournaments and win more tournaments.

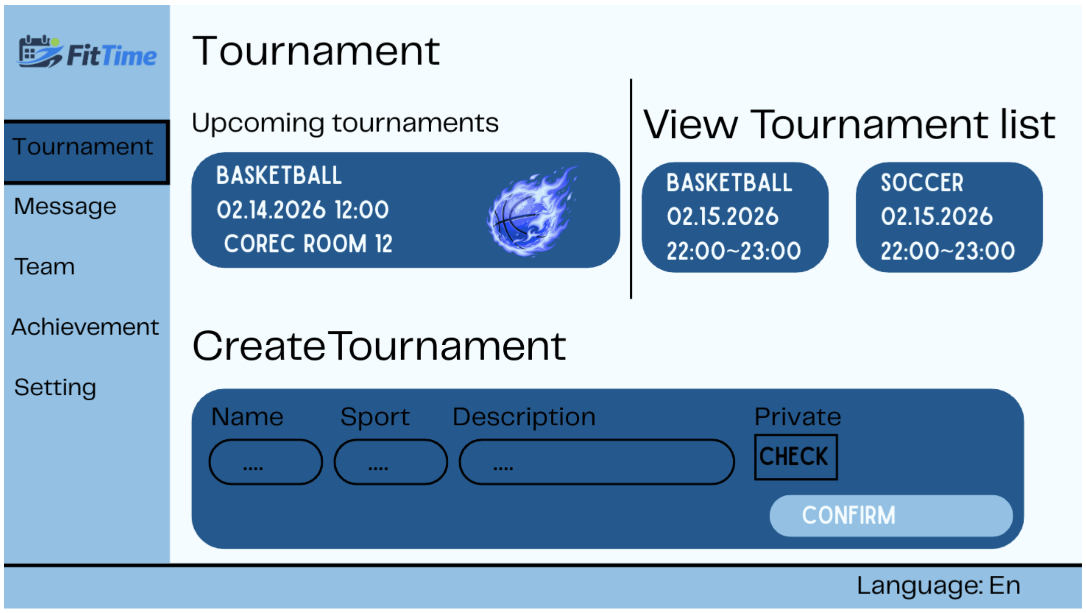

Join Tourament
Tournament is Completed
Achievement is Created
Controller is Given Result
Internal Model Is Updated
If Team Won Tournament
Else
Participant is given achievement for winning in the tournament
Participant is given achievement for playing in the tournament
Tournament Display it updated
Person in Tournament Sees Results

Mock UI

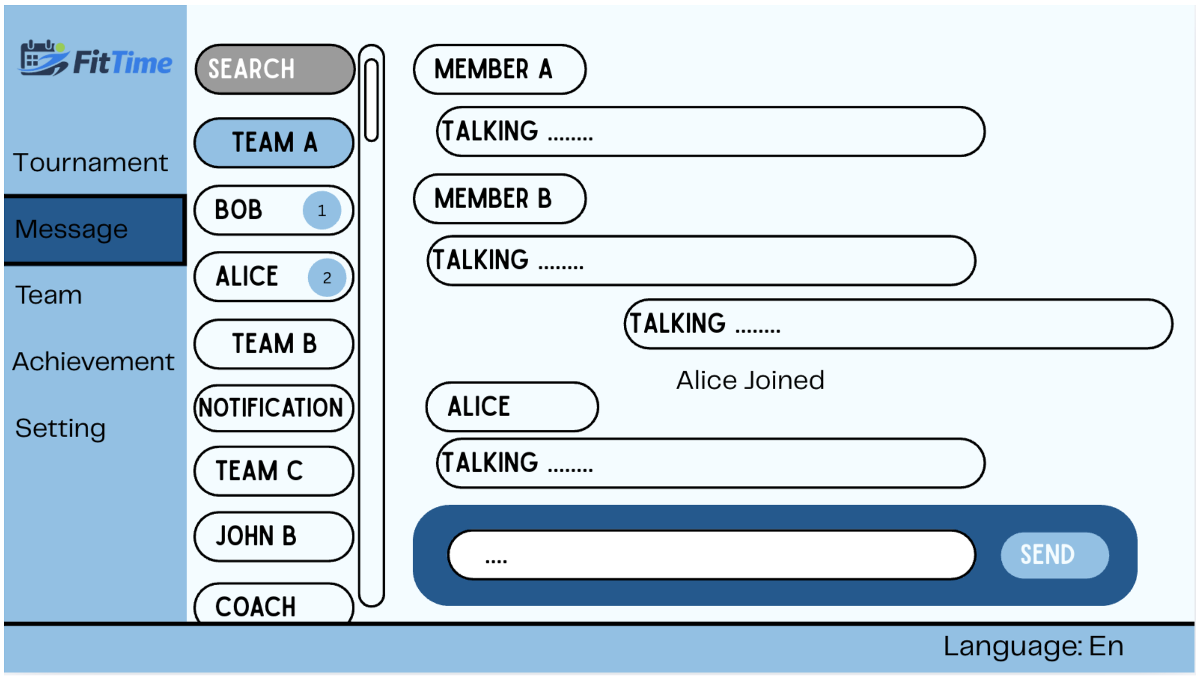
Login Page



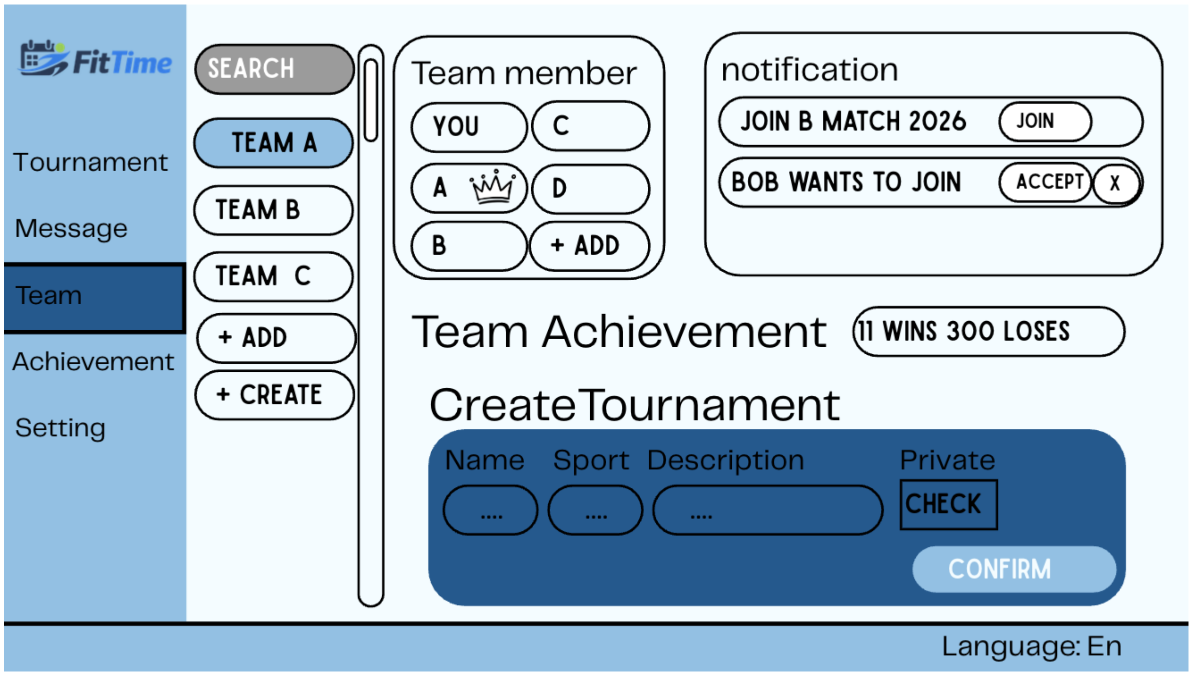
Tournament Page



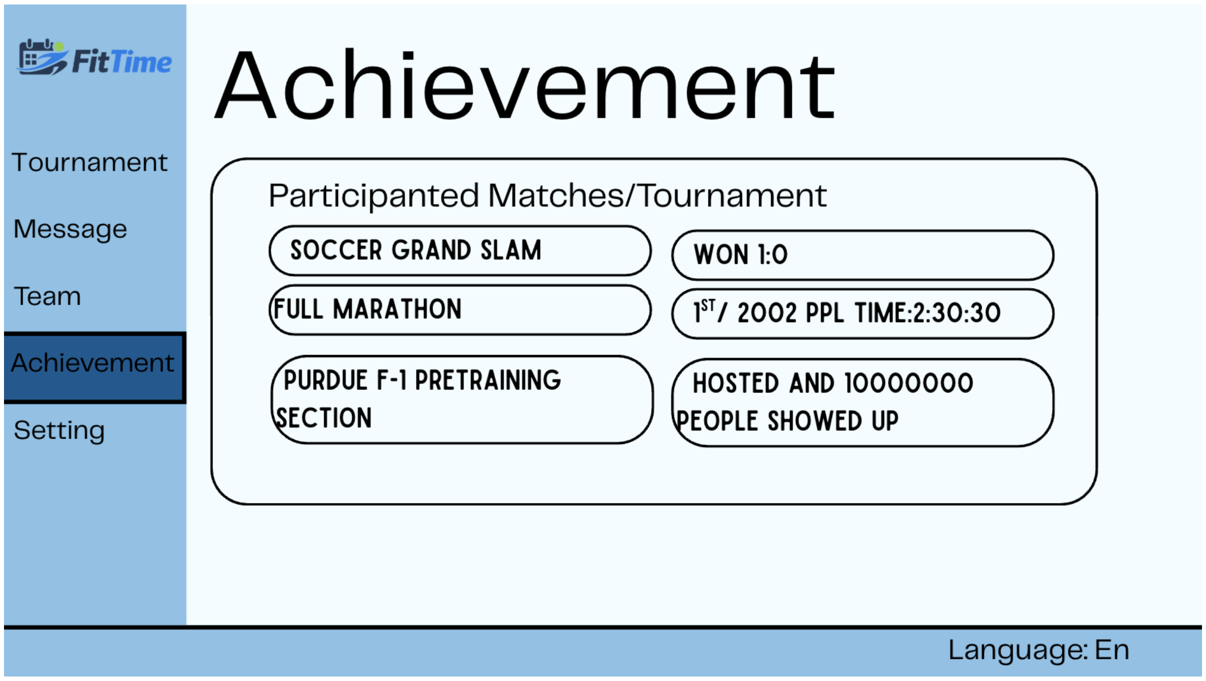
Message Page



Team Page



Achievement Page



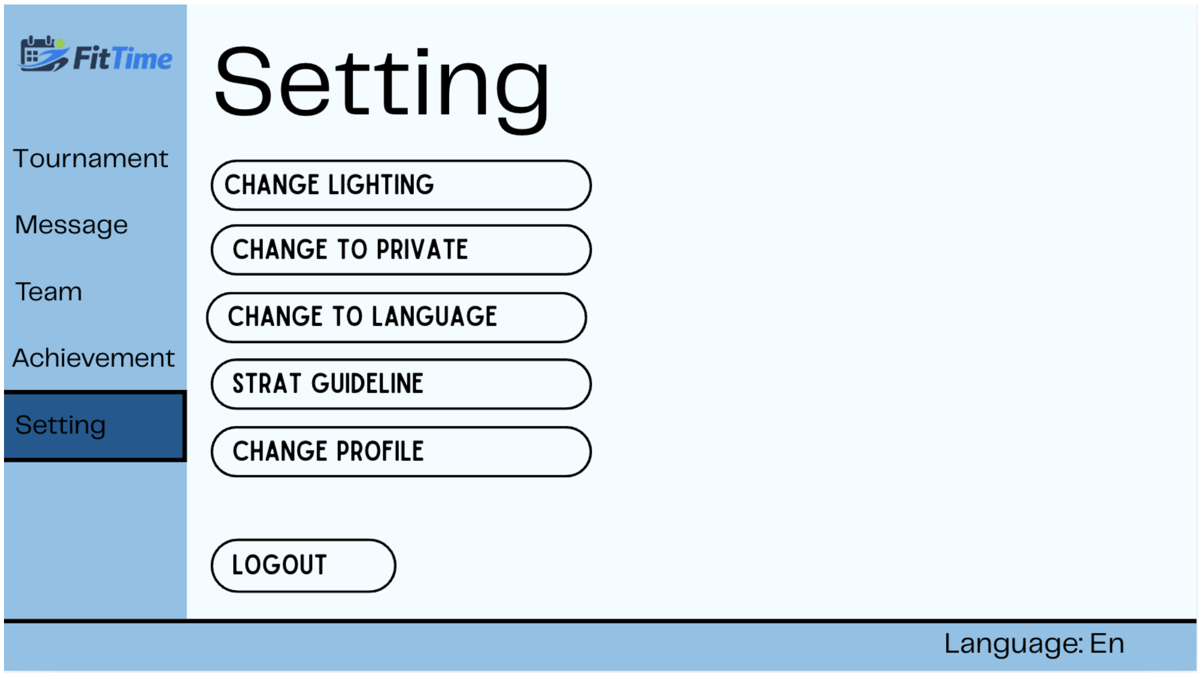
Achievement

Participanted Matches/Tournament

SOCCER GRAND SLAM	WON 1:0
FULL MARATHON	1ST / 2002 PPL TIME:2:30:30
PURDUE F-1 PRETRAINING SECTION	HOSTED AND 10000000 PEOPLE SHOWED UP

Language: En

Setting Page



Setting

Language: En