

CMPUT 401

Measure What Matters

Testing Documentation

<b>1.0.0 Testing Strategy</b>	<b>3</b>
1.1.0 Frontend Testing Strategy	3
1.1.1 How to run	3
1.1.2 Testing Methodology	3
1.2.0 Backend Testing Strategy	3
1.2.1 How to run	3
1.2.2 Testing Services	3
1.2.3 Testing Controllers	3
1.3.0 Selenium Testing Strategy	4
1.4.0 User Stories Testing Strategy	4
<b>2.0.0 Manual Acceptance Testing User Stories</b>	<b>5</b>
1.1.1 Create an Account	5
1.1.2 User Login	6
1.1.3 User Logout	8
1.1.4 View Account Profile and Info	9
1.1.5 Edit Account Info	10
1.2.1 Create A Team	11
1.2.2 Edit Team Roster	13
1.2.3 Create a Match Lineup	16
1.4.2 Start A Match Recording Session	19
2.1.1 Game Event - Substitutions	20
2.1.3 Game Event - Shots on Net	21
2.1.5 Game Event- Goal	22
2.1.6 Game Event- Assists	23
2.1.9 Cache Event Recordings if Network is Down	23
2.1.10 Game Event - Touches	26
2.1.11 Game Event - Ball Possession	27
2.3.1 On Field Timer	28
3.1.1 Stat Dashboard	30
3.2.1 Stat- Time on Field	31
3.2.2 Stat - Plus Minus	32
3.2.3 Lineup During Goals	33
3.2.6 Stat - Ball Possession Time/ Percentage	33
<b>4.0.0 Selenium Automated Testing</b>	<b>34</b>
<b>5.0.0 Incomplete User Stories</b>	<b>36</b>

# 1.0.0 Testing Strategy

## 1.1.0 Frontend Testing Strategy

### 1.1.1 How to run

Front end testing was done using the React Testing Library. The relative path of tests from the root folder is `/frontend/src/test`. After using the command `>npm install` from the frontend directory, tests can be run using `>npm test`. The tests are connected to github using a CI pipeline.

### 1.1.2 Testing Methodology

With our testing strategy, we avoided testing implementation details when testing our components - this is in line with the React Testing Library recommendations. We favoured testing as if our tests were the final user interacting with the app. To that end we tested what the user could see wherever possible and found elements by their text. We used dependency injection for decoupling api fetching and to enable easier testing.

## 1.2.0 Backend Testing Strategy

### 1.2.1 How to run

Backend testing was done using the Jest testing framework. The relative path of tests from the root folder is `/backend/test`. After using the command `>yarn install` from the backend directory, tests can be run using `>yarn test`. The tests are connected to github using a CI pipeline. Tests were done for both controllers and services to ensure correct function for both.

### 1.2.2 Testing Services

Service tests are tests in the `/backend/test` folder with the `*.service.spec.ts` file names. Services were tested by first creating a `TestingModule` with the service to be tested and relevant repositories as providers. Then a check was run to ensure that they were defined. All functions were then tested, ensuring that any helper functions were called an appropriate amount of times (depending on function), the database was called with the correct call and the right amount of times, and that the return value correctly corresponded to the expected value. For database calls inputs and returns were mocked using Jest.

### 1.2.3 Testing Controllers

Controller tests are tests in the `/backend/test` folder with the `*.controller.spec.ts` file names. Controllers were tested by first creating a `TestingModule` with the controller to be tested and

relevant services as providers. Then a check was run to ensure that they were defined. Each function in the controller was tested to ensure that they called the right function in the service, the call was made using the correct input, with the correct return value, and the call was made the correct number of times. Lastly, errors were mocked to ensure that they were returning the right error and message to the frontend, depending on the error encountered.

### 1.3.0 Selenium Testing Strategy

The automated testing can be found in the “/selenium\_test” folder (relative to the root folder). In this directory “>npm install” will be run to download dependencies, and then tests can be run using “>npm run test”. Selenium was used for automated testing, to ensure that the application runs correctly. The driver ran through all of the application's key functionality. Checks were also done to check for different error messages that should pop up when the user inputs an incorrect value. Mocha and Mochawesome were run for testing and report generation respectively. The output reports of Mochawesome can be found in “/selenium\_test/ReportDirectory” in both html and PDF forms. More information on Selenium Tests can be found in Section 4.0.

### 1.4.0 User Stories Testing Strategy

All user stories were tested using manual testing, which can be found in section 2.0. A majority of them were also tested using Selenium, with a report summary in the “/selenium\_testing” folder relative to the root folder. This can also be found in the testing docs folder under “/docs/docs” from the root folder.

# 2.0.0 Manual Acceptance Testing User Stories

## 1.1.1 Create an Account

**Acceptance:** User has own account to keep local information and rosters, unique username and password satisfied

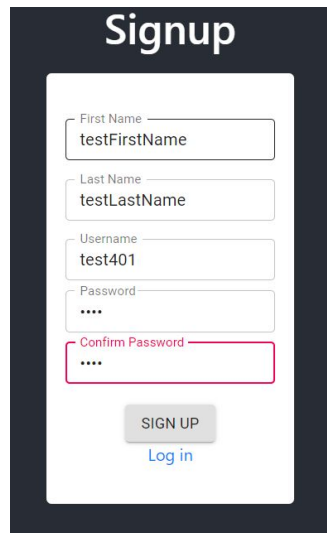
Clicking the sign up button from the dashboard is shown below.

The image shows two side-by-side screenshots of a web application interface. The left screenshot is titled "Login" and features a white background with a dark border. It contains two input fields: "Username" and "Password". Below these fields is a grey "LOG IN" button and a blue link that says "Sign up here". The right screenshot is titled "Signup" and also has a white background with a dark border. It contains five input fields: "First Name", "Last Name", "Username", "Password", and "Confirm Password". Below these fields is a grey "SIGN UP" button and a blue link that says "Log in".

Error checking is then done as follows to ensure the user has a complete account creation experience. The following two pictures show error handling on incorrect account creation. The first displays what occurs when passwords do not match, the second if a username is taken, and lastly the message type that occurs if a parameter is empty.

The image shows three side-by-side screenshots of the "Signup" form, each with a dark background and a white form area. The first screenshot shows an error message "Passwords must match!" in red text at the top. The form fields are: "First Name" (empty), "Last Name" (filled with "test"), "Username" (filled with "test"), "Password" (filled with "\*\*\*\*\*"), and "Confirm Password" (filled with "\*\*\*\*\*"). The "SIGN UP" button is greyed out, and the "Log in" link is blue. The second screenshot shows an error message "Username is taken." in red text at the top. The form fields are: "First Name" (empty), "Last Name" (filled with "test"), "Username" (filled with "bardtest"), "Password" (filled with "\*\*\*\*\*"), and "Confirm Password" (filled with "\*\*\*\*\*"). The "SIGN UP" button is greyed out, and the "Log in" link is blue. The third screenshot shows an error message "Please enter a valid username. It should not contain spaces." in red text at the top. The form fields are: "First Name" (empty), "Last Name" (filled with "test"), "Username" (filled with "test"), "Password" (filled with "\*\*\*\*"), and "Confirm Password" (filled with "\*\*\*\*"). The "SIGN UP" button is greyed out, and the "Log in" link is blue.

Correct information was then inputted:

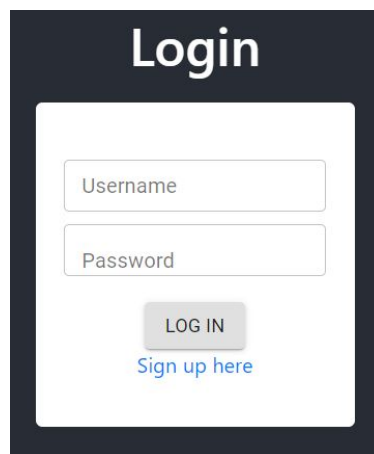


The image shows a 'Signup' form with the following fields and values:

- First Name: testFirstName
- Last Name: testLastName
- Username: test401
- Password: \*\*\*\*
- Confirm Password: \*\*\*\*

Below the fields are two buttons: 'SIGN UP' and 'Log in'.

The sign up button was then clicked: Upon complete account creation the user is taken back to the login page:



The image shows a 'Login' form with the following fields and values:

- Username: [empty]
- Password: [empty]

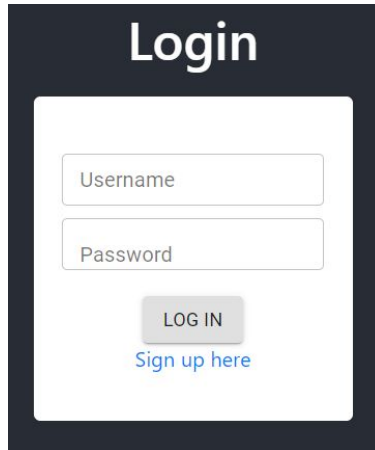
Below the fields are two buttons: 'LOG IN' and 'Sign up here'.

PGadmin was also checked to ensure that the new user was saved to the database.

## 1.1.2 User Login

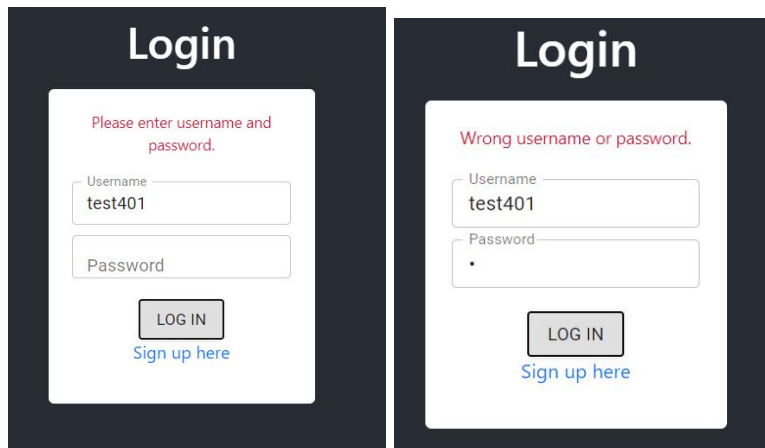
**Acceptance:** Log in process completes, logs into correct account, profile shows up.

The manual version of this test was done as follows:



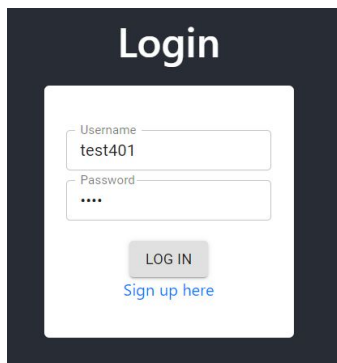
The image shows a dark blue rectangular frame containing a white login form. At the top, the word "Login" is written in a large, white, sans-serif font. Below the title, there are two white input fields with rounded corners. The first field is labeled "Username" and is empty. The second field is labeled "Password" and is also empty. Below these fields is a grey button with the text "LOG IN" in white. Underneath the button is a blue link that says "Sign up here".

The username used from the previous test was then inputted into the login. Errors for empty or incorrect credentials were then tested for, to provide the user with complete login capability.



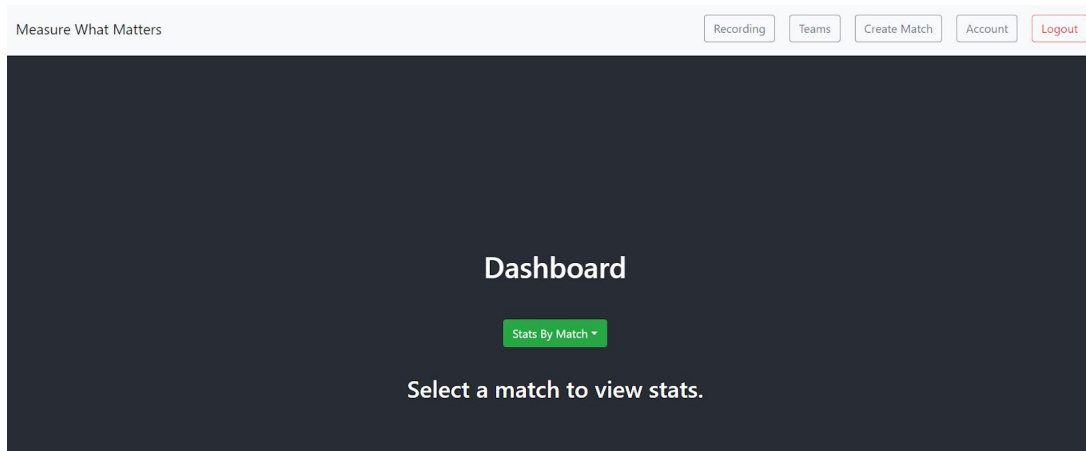
This block contains two side-by-side screenshots of the login form. The left screenshot shows the form with the message "Please enter username and password." in red text above the input fields. The "Username" field contains the text "test401", and the "Password" field is empty. The right screenshot shows the form with the message "Wrong username or password." in red text above the input fields. The "Username" field contains "test401" and the "Password" field contains a single black dot. Both screenshots feature the "LOG IN" button and the "Sign up here" link at the bottom.

Correct login was inputted



The image shows the login form with the "Username" field containing "test401" and the "Password" field containing four black dots. The "LOG IN" button is now highlighted in a light blue color, indicating it is active. The "Sign up here" link remains blue.


Validation was completed, as shown below.




To verify this is the correct user account and close the acceptance test, the account button was chosen from the top right and credentials were verified.

**Account Information**

Username:

First Name:  

Last Name:  

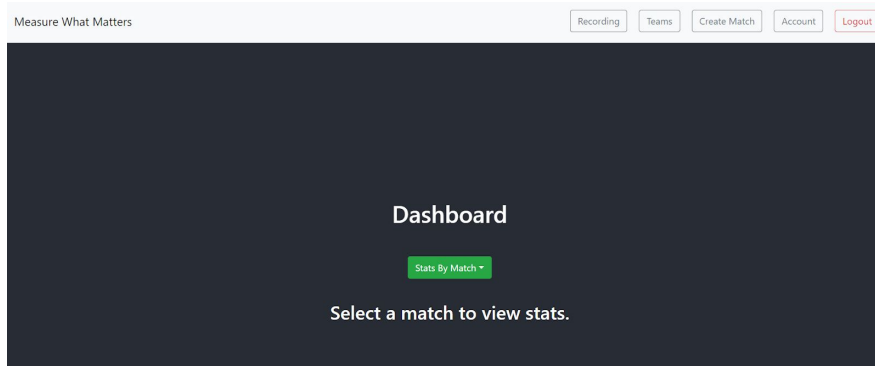
This was also done and verified using Selenium testing.

### 1.1.3 User Logout

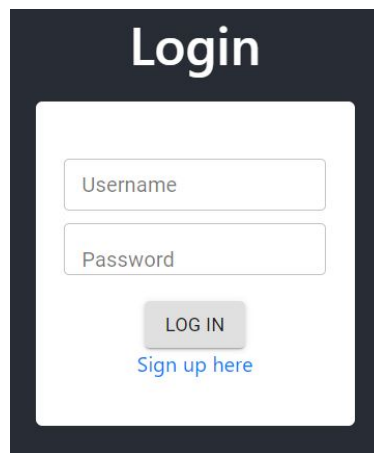
**Acceptance:** Log out completes and return to home screen achieved.

Starting from the dashboard in picture below (login shown in 1.1.2):





Clicking the logout button on the top right :



This completes the acceptance test, as we are brought back to the home screen and logged out. The JWT key was verified to be wiped from the front end.

This was also done and verified using Selenium testing.

#### 1.1.4 View Account Profile and Info

**Acceptance:** Interface with correct account information showing accessible

As shown in test 1.1.2 the account information is accessible from the dashboard using the Account button:

The screenshot shows a form titled "Account Information" with three input fields: "Username" containing "test401", "First Name" containing "testFirstName", and "Last Name" containing "testLastName". Each of the last two fields has a small pencil icon to its right. A "BACK" button is located at the bottom center of the form.

This verifies the acceptance test as the desired interface was accessible, and was also verified using Selenium .

### 1.1.5 Edit Account Info

**Acceptance:** Edits have a simple interface, view account information shows updates correctly.


The account edit page was pulled up by clicking the account button from the dashboard, as shown in test 1.1.2:


This screenshot is identical to the one above, showing the "Account Information" form with fields for Username, First Name, and Last Name, and a BACK button.

The pencil icon beside the first name was clicked, allowing access to the input box, where the name "newFirstName" can be seen inputted below.

**Account Information**

Username:

First Name:    
[Save](#) [Cancel](#)


Last Name:  


[BACK](#)

The save button was clicked, and the same thing was done with the last name using the new name of “newLastName”. The result can be seen below. This verifies the acceptance test of a simple interface with correct updates shown and saved to the database.

**Account Information**

Username:

First Name:  

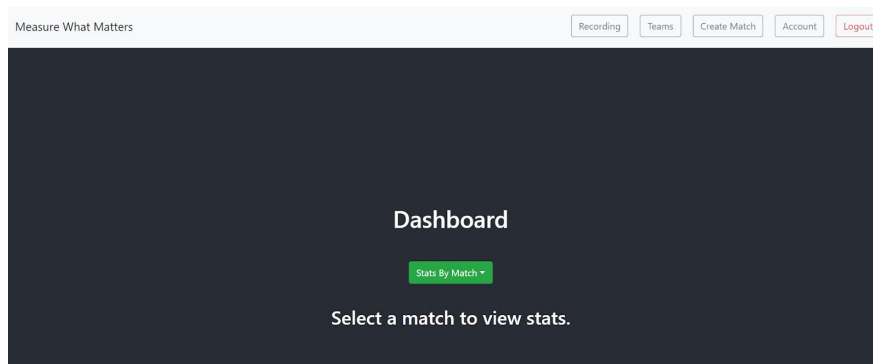
Last Name:  

[BACK](#)

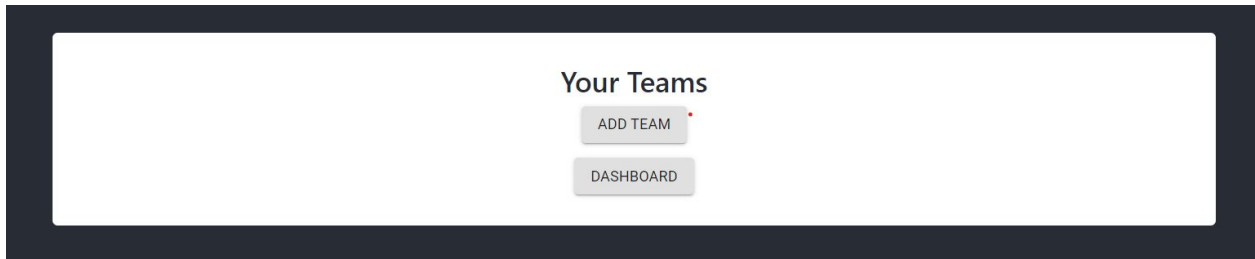
## 1.2.1 Create A Team

**Acceptance:** Interface for team creation has a team name and initial roster available. Once executed the team information is shown correctly.

Starting at the dashboard:



The Teams button was clicked opening the following interface:



Clicking add team:

### Add A New Team

Player Name	Player Number
-------------	---------------

Adding inputs, and the players for the team using Add Player:

### Add A New Team

 newTeamTest  
**newTeamTest**  
 Zoey  Zoe  2  
  

Player Name	Player Number	
Joey Joe	1	<input type="button" value="REMOVE"/>

Finally, after creating the team via the add team button:

## newTeamTest

First Name	Last Name	Jersey Number
Joey	Joe	1
Zoey	Zoe	2
slowey	slow	3

EDIT ROSTER

ADD TEAM

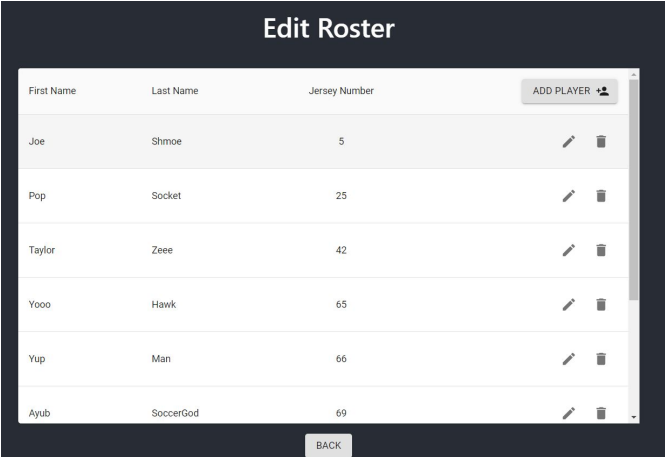
DASHBOARD

This verifies that team creation can be correctly executed, and that the team information is then shown correctly, passing the acceptance test.

### 1.2.2 Edit Team Roster

**Acceptance:** Roster edit interface easily accessible and simple, roster shows up correctly after edits

From the teams interface at the end of the previous test (1.2.1), if the edit roster button is clicked, the roster for that team will show up:



The screenshot shows a mobile application interface titled "Edit Roster". It features a table with three columns: "First Name", "Last Name", and "Jersey Number". There are seven rows of player data. To the right of each row are two icons: a pencil for editing and a trash can for deleting. In the top right corner of the table area, there is a button labeled "ADD PLAYER" with a person icon. At the bottom center of the screen, there is a "BACK" button.

First Name	Last Name	Jersey Number		
Joe	Shmoe	5		
Pop	Socket	25		
Taylor	Zeee	42		
Yooo	Hawk	65		
Yup	Man	66		
Ayub	SoccerGod	69		

From here the edit button can be clicked, clicking it on Joe Shmoe:

### Edit Player

First Name  
Joe

Last Name  
Shmoe

Jersey Number  
5

CANCEL SAVE

Adding edits to Joe Shmoe to make him Joey Shmoey with a new jersey number:

### Edit Player

First Name  
Joey













Last Name  
Shmoey


Jersey Number  
61

CANCEL SAVE

After clicking save it can be seen that the roster has been updated with a new Joey Shmoey:

### Edit Roster

First Name	Last Name	Jersey Number	
Pop	Socket	25	 
Taylor	Zeee	42	 
Joey	Shmoey	61	 
Yooo	Hawk	65	 
Yup	Man	66	 
Ayub	SoccerGod	69	 

ADD PLAYER 

BACK

Editing a player on the roster works! How about adding players? Clicking Add Player:

Add Player

First Name  
Enter a valid name with no spaces.

Last Name  
Enter a valid name with no spaces.

Jersey Number  
Jersey number must be a number greater than 0.

CANCEL SAVE

We want to add our TA to our lineup:

Add Player

First Name  
Kalvin













Last Name  
Eng


Jersey Number  
7

CANCEL SAVE

Clicking save takes us back to the roster where we can see Calvin added:

### Edit Roster

First Name	Last Name	Jersey Number	
Kalvin	Eng	7	 
Pop	Socket	25	 
Taylor	Zeee	42	 
Joey	Shmoey	61	 
Yooo	Hawk	65	 
Yup	Man	66	 

ADD PLAYER 

BACK

If we tried to add Calvin when the jersey number 7 was already taken:

### Add Player

First Name

Kalvin

Last Name

Eng

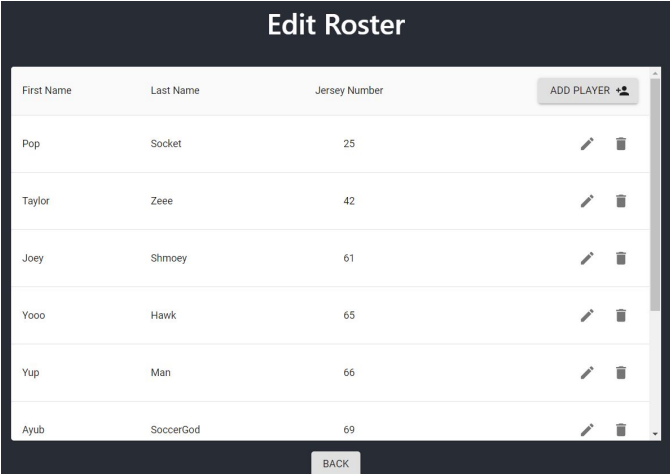
Jersey Number

7













Jersey number must be unique.

CANCEL SAVE

Now from the edit page we can click the delete button beside Calvin's name:



The screenshot shows the 'Edit Roster' interface. It features a table with columns for 'First Name', 'Last Name', and 'Jersey Number'. There is an 'ADD PLAYER' button with a plus icon in the top right corner. Each row in the table has an edit icon (pencil) and a delete icon (trash can) to its right. At the bottom of the interface is a 'BACK' button.

First Name	Last Name	Jersey Number	
Pop	Socket	25	 
Taylor	Zeee	42	 
Joey	Shmoey	61	 
Yooo	Hawk	65	 
Yup	Man	66	 
Ayub	SoccerGod	69	 

Kalvin is now deleted. This verifies that deletion, edit, and adding new players to the roster from the edit roster interface update correctly, passing the acceptance test.

## 1.2.3 Create a Match Lineup

**Acceptance:** Match lineup interface allows for simple roster creation, the lineup for the game is correct after added

From the dashboard clicking create a match:



## Create A Match

Team  
*None*

Opponent Name

Game Venue  
*Home*

When  
2020-11-01 10:30 AM

CREATE MATCH

BACK

Inputting a team, opponent name, and time:

## Create A Match

Team  
*newTeamTest*

Opponent Name  
*Liverpool*

Game Venue  
*Home*

When  
2020-12-25 10:30 AM

CREATE MATCH

BACK

We are returned to the dashboard, clicking the recording button on the top of the dashboard:

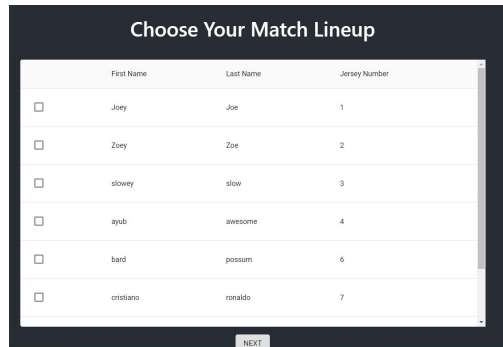
# Upcoming Matches

Select a match to begin

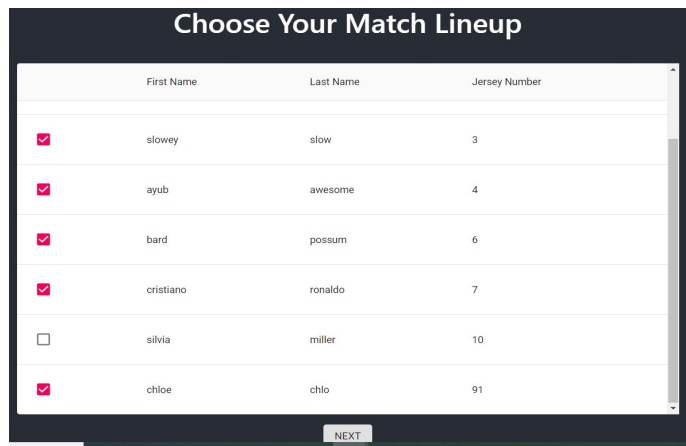
12/25/2020, 10:30:00 AM  
newTeamTest vs. Liverpool  
Home Game

BACK

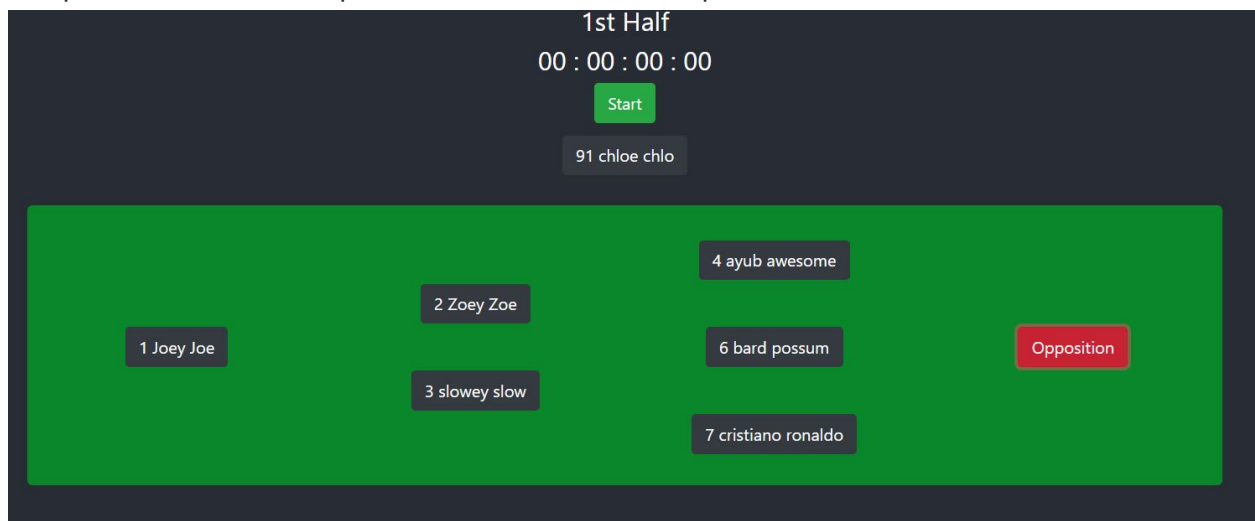
Our match is available. Clicking on this match will bring us to the pick match lineup screen:



We pick all our players, except for Silvia who is not available for this match tonight:



The match recording starts, and by clicking and dragging a player to a bench we can see our whole lineup, with Silvia out as expected, as shown in the next picture :



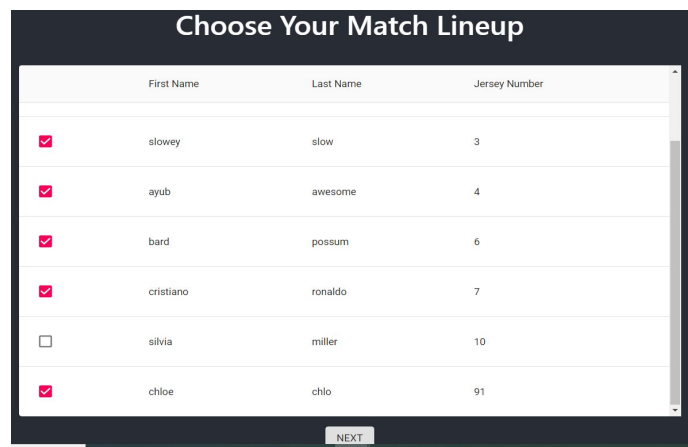
Therefore once a match is created, you simply have to click on a match and pick your lineup, the lineup will show correctly.

This verifies that roster creation is simple and the correct roster is shown, passing the acceptance test.

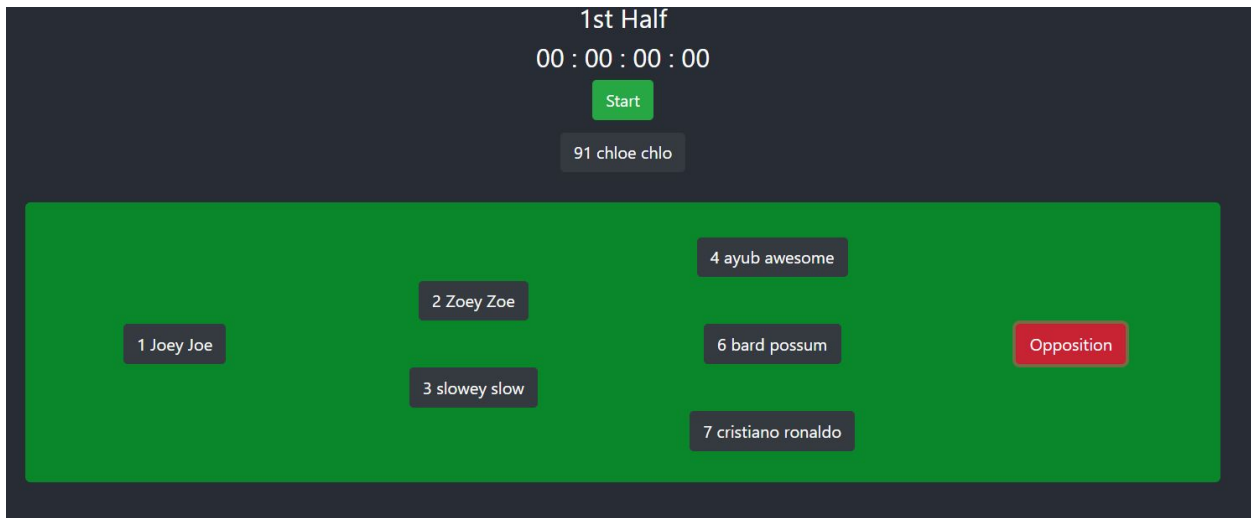
## 1.4.2 Start A Match Recording Session

**Acceptance:** Selection of icon properly puts manager and recorders into “during game” activity.

Starting from the lineup creation in test 1.2.3 above:



Clicking the next button starts the match, and then clicking and dragging a player to the bench will activate the bench, the result is shown below:

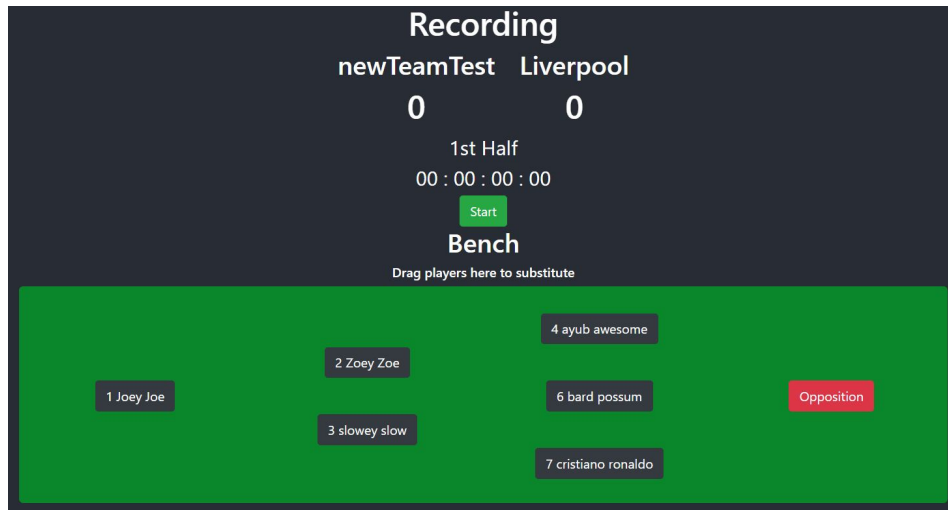


Therefore, the icon selected correctly put the recorder into the “during game” activity, passing the acceptance test.

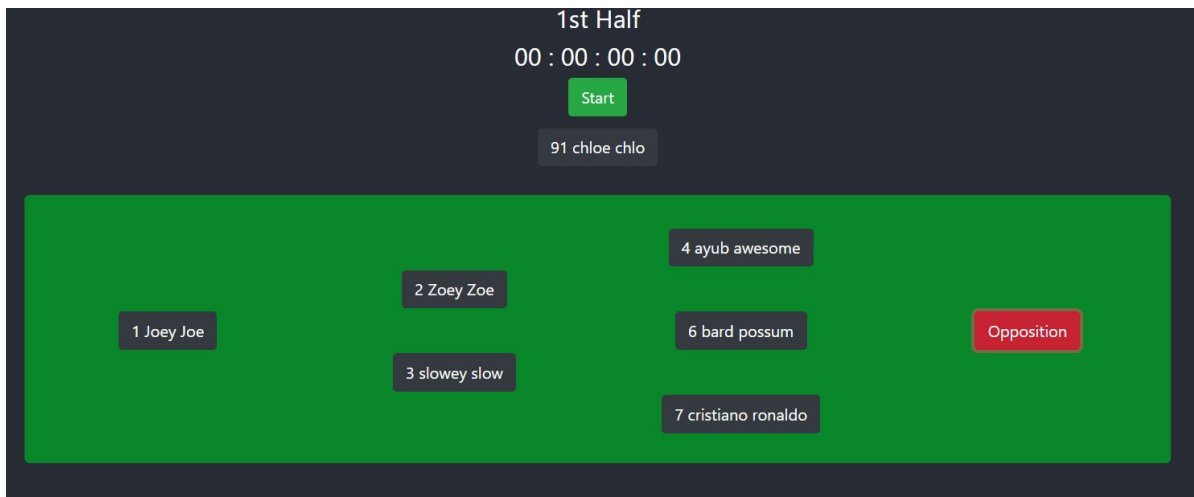
## 2.1.1 Game Event - Substitutions

**Acceptance:** Selection of players to substitute results in the lineup being updated.

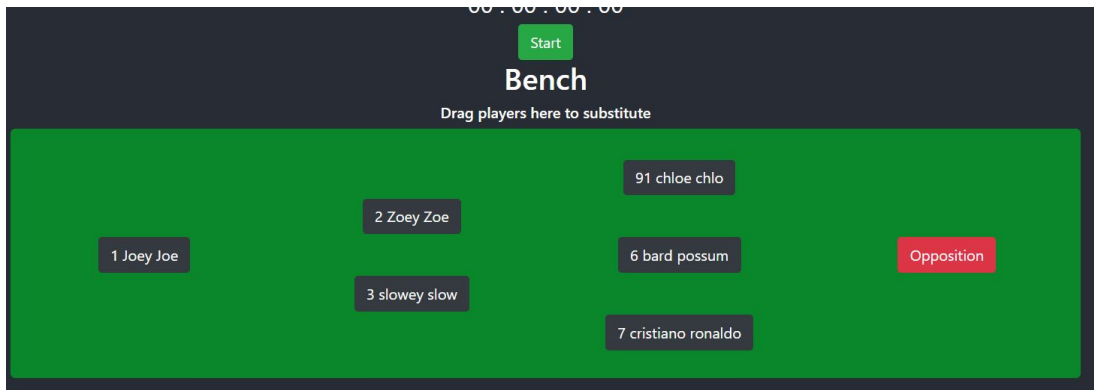
Starting from the recording interface (getting to this is shown in test 1.4.2):



We can click and drag a player to the bench to activate the bench:



Now we click on Ayub, and then on Chloe to substitute Ayub out for Chloe:

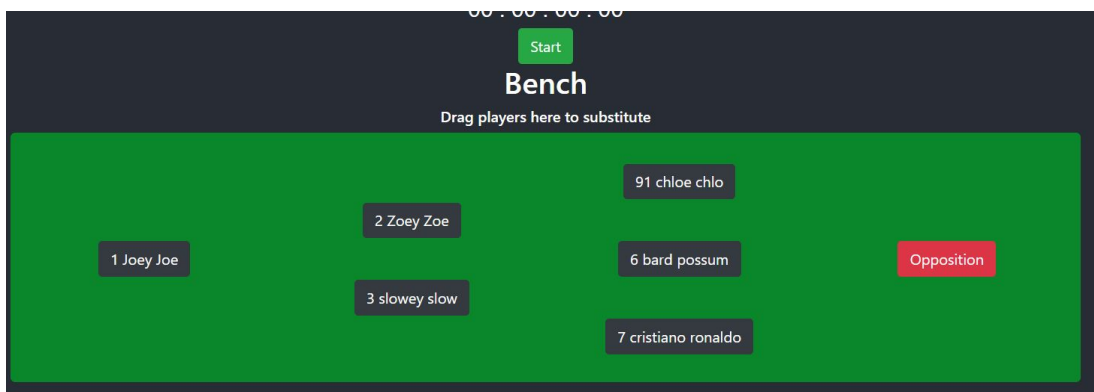


As can be seen, the substitution is executed properly, with proper lineup update, passing the acceptance test.

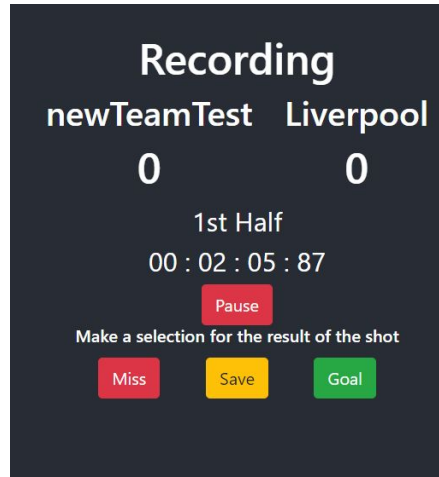
### 2.1.3 Game Event - Shots on Net

**Acceptance:** A shot taken event including the location and time is persisted and associated with the current game.

From the recording interface (see test 1.4.2 to get to) we click the start icon to start the timer:



Now we click on a player so they have possession. Next we drag them to the field to take a shot.

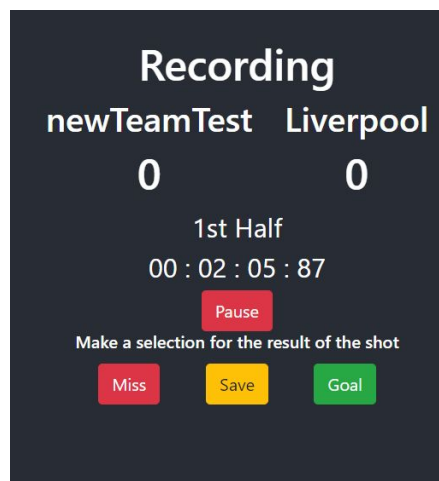


If we pick miss or save, it returns to the game with the same score. We verified in the database that the shot is recorded correctly and that it correctly states if it is on target (save) or a miss. If the event is a goal it will increment the score for the team that scored it.

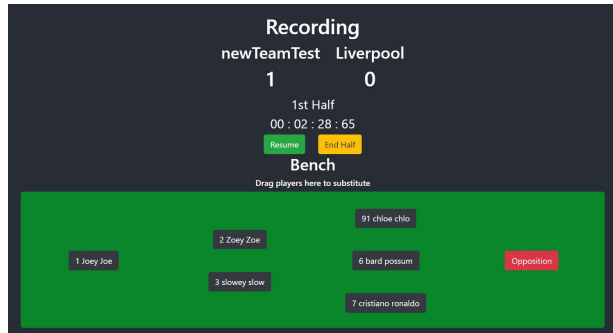
## 2.1.5 Game Event- Goal

**Acceptance:** A goal event is persisted with the time it occurred and associated with the current game and the player who scored it.

Starting from the shot interface at the end of test 2.1.3:



If we choose goal the recording screen updates as follows:



The goal was also checked to be correctly saved into the database with the correct matchId, playerId, and time were recorded.

## 2.1.6 Game Event- Assists

**Acceptance:** An assist event is persisted with the time it occurred and associated with the current game and the player who got the assist.

The assist event is synonymous with the goal event in 2.1.5. When a goal is recorded, if the player to touch the ball previous to the scorer was from the same team (not the opposition), the player to touch the ball previously is registered for the assist in the database. Checking the database, the player and time are saved correctly, passing the acceptance test.

## 2.1.9 Cache Event Recordings if Network is Down

**Acceptance:** Events recorded without network connection should be persisted when the network connection is reestablished.

From the recording interface (see test 1.4.2 to see how to get to):

Recording  
MWM PSG  
1 0  
1st Half  
00 : 00 : 12 : 27  
Resume End Half  
Bench  
Drag players here to substitute

1 Aaminah Hood  
2 Addison Connor  
3 Khaleesi Harrison  
4 Usman McIntyre  
5 Wayne Townsend  
6 Marco Lu  
Opposition

Name	Status	Type	Initiator	Size	Time	Waterfall
player	204	Other	0.0	2 ms		
player	204	Other	0.0	2 ms		
player	201	xhr	xhr.js:184	411.0	20 ms	
player	204	Other	0.0	2 ms		
neutral	204	Other	0.0	1 ms		
neutral	201	xhr	xhr.js:184	414.0	18 ms	
neutral	201	xhr	xhr.js:184	414.0	19 ms	
assists	201	xhr	xhr.js:184	378.0	22 ms	
assists	201	xhr	xhr.js:184	159.0	94 ms	
assists	204	Other	0.0	2 ms		
goals	201	xhr	xhr.js:184	159.0	97 ms	
shots	201	xhr	xhr.js:184	412.0	39 ms	
assists	204	Other	0.0	2 ms		
assists	204	Other	0.0	2 ms		
shots	201	xhr	xhr.js:184	412.0	34 ms	
goals	204	Other	0.0	2 ms		
shots	204	Other	0.0	5 ms		
shots	204	Other	0.0	10 ms		

20 requests | 3.5 kB transferred | 1.2 kB resources

When the network is connected, we can see in the network requests that all the events are recorded successfully.

Recording  
MWM PSG  
2 0  
1st Half  
00 : 00 : 20 : 81  
Resume End Half  
Bench  
Drag players here to substitute

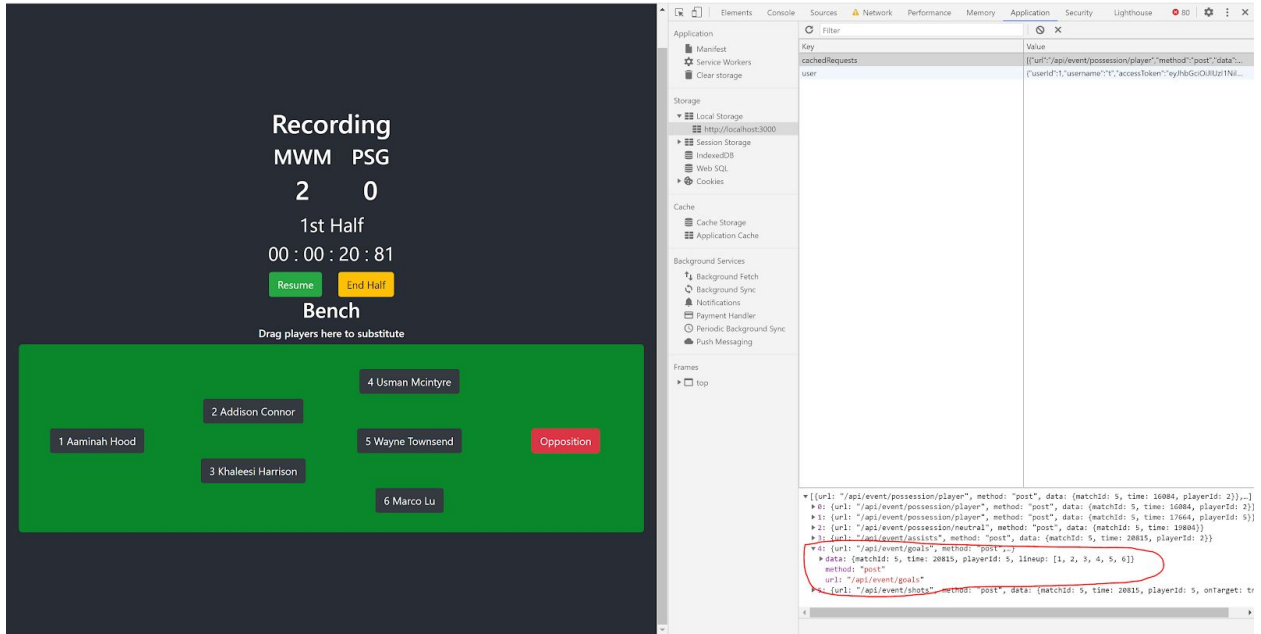
1 Aaminah Hood  
2 Addison Connor  
3 Khaleesi Harrison  
4 Usman McIntyre  
5 Wayne Townsend  
6 Marco Lu  
Opposition

Name	Status	Type	Initiator	Size	Time	Waterfall
shots	204	Other	0.0	10 ms		
goals	201	xhr	xhr.js:184	159.0	94 ms	
shots	201	xhr	xhr.js:184	412.0	39 ms	
shots	201	xhr	xhr.js:184	412.0	34 ms	
player	failed	xhr	xhr.js:184	0.0	6 ms	
player	failed	xhr	xhr.js:184	0.0	6 ms	
player	204	Other	0.0	2 ms		
Outchunks	failed	fetch	index.js:1	0.0	2 ms	
player	failed	xhr	xhr.js:184	0.0	2 ms	
player	failed	xhr	xhr.js:184	0.0	3 ms	
Outchunks	failed	fetch	index.js:1	0.0	2 ms	
neutral	failed	xhr	xhr.js:184	0.0	4 ms	
neutral	204	Other	0.0	2 ms		
neutral	204	Other	0.0	1 ms		
Outchunks	failed	fetch	index.js:1	0.0	3 ms	
assists	failed	xhr	xhr.js:184	0.0	6 ms	
assists	failed	xhr	xhr.js:184	0.0	6 ms	
goals	failed	xhr	xhr.js:184	0.0	6 ms	
goals	failed	xhr	xhr.js:184	0.0	7 ms	
assists	204	Other	0.0	2 ms		
shots	failed	xhr	xhr.js:184	0.0	7 ms	
assists	204	Other	0.0	2 ms		
goals	204	Other	0.0	3 ms		
shots	204	Other	0.0	2 ms		
shots	204	Other	0.0	2 ms		
Outchunks	failed	fetch	index.js:1	0.0	3 ms	
Outchunks	failed	fetch	index.js:1	0.0	2 ms	
Outchunks	failed	fetch	index.js:1	0.0	1 ms	
player	failed	xhr	xhr.js:184	0.0	4 ms	
player	204	Other	0.0	2 ms		

50 requests | 3.5 kB transferred | 1.2 kB resources

When we turn the network offline and try to record some events, all the requests fail to save the event to the backend. We see these failed requests in red in the image above.

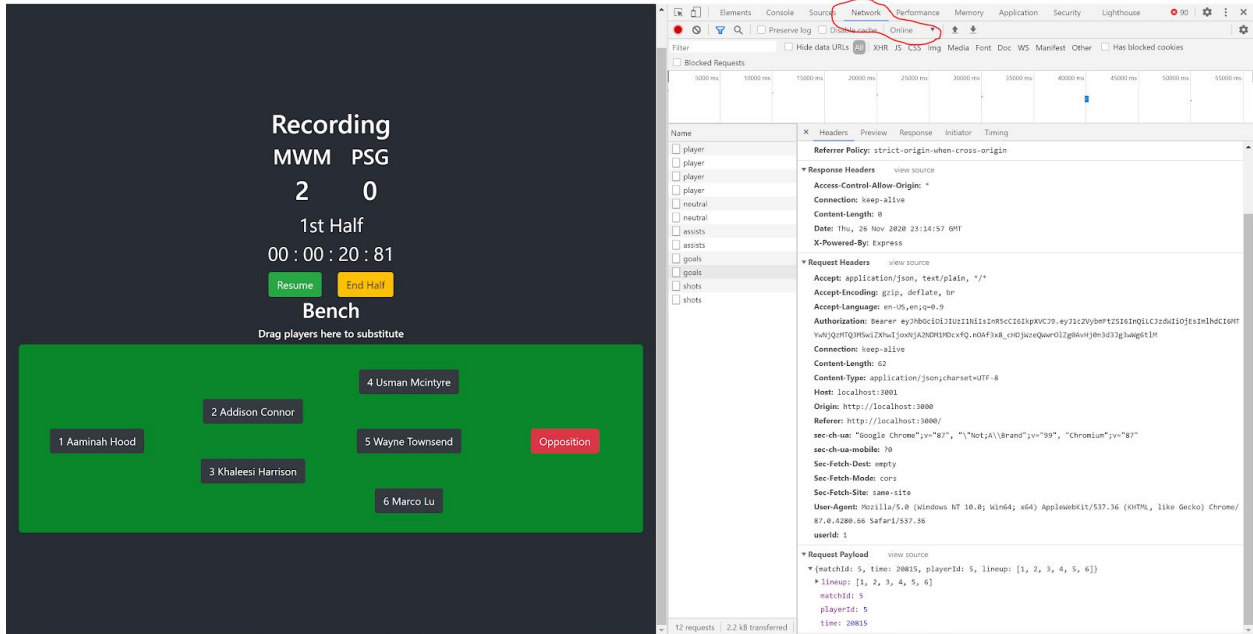




If the cause of failure is a network error then the failed requests are stored into local storage and are retried every 10 seconds until they resolve.

	id [PK] integer	time integer	playerId integer	matchId integer	lineup integer[]	createdDate timestamp without time zone	updatedDate timestamp without time zone
1	1	2611	4	1	{1,2,3,4,5,6}	2020-11-25 02:34:04.387937	2020-11-25 02:34:04.387937
2	2	2611	4	1	{1,2,3,4,5,6}	2020-11-25 02:34:04.38908	2020-11-25 02:34:04.38908
3	3	9212	[null]	1	{1,2,3,4,5,6}	2020-11-25 02:34:12.809263	2020-11-25 02:34:12.809263
4	4	9212	[null]	1	{1,2,3,4,5,6}	2020-11-25 02:34:12.810673	2020-11-25 02:34:12.810673
5	5	12212	2	1	{1,2,3,4,5,6}	2020-11-25 02:34:17.872031	2020-11-25 02:34:17.872031
6	6	12212	2	1	{1,2,3,4,5,6}	2020-11-25 02:34:17.873792	2020-11-25 02:34:17.873792
7	7	21393	5	1	{1,2,3,4,5,6}	2020-11-25 02:34:28.975451	2020-11-25 02:34:28.975451
8	8	21393	5	1	{1,2,3,4,5,6}	2020-11-25 02:34:28.976449	2020-11-25 02:34:28.976449
9	9	12273	2	5	{1,2,3,4,5,6}	2020-11-26 22:59:16.905557	2020-11-26 22:59:16.905557
10	10	12273	2	5	{1,2,3,4,5,6}	2020-11-26 22:59:16.906217	2020-11-26 22:59:16.906217

Currently, there are 10 goal events in the backend. When the network is reconnected the number should change from 10 to 11 recording a goal event with the player id 5, match id 5, time 20815, and lineup [1,2,3,4,5,6].



When the network is reenabled, we can see that the goal request that we highlighted in the previous pictures has succeeded along with the other requests that failed due to network error.

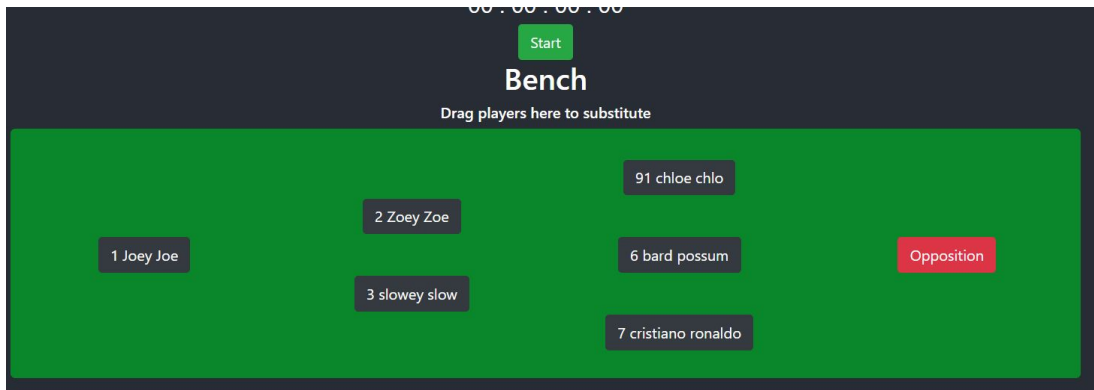
id [PK] integer	time integer	playerId integer	matchId integer	lineup integer[]	createdDate timestamp without time zone	updatedDate timestamp without time zone
1	1	2611	4	{1,2,3,4,5,6}	2020-11-25 02:34:04.387937	2020-11-25 02:34:04.387937
2	2	2611	4	{1,2,3,4,5,6}	2020-11-25 02:34:04.38908	2020-11-25 02:34:04.38908
3	3	9212	[null]	{1,2,3,4,5,6}	2020-11-25 02:34:12.809263	2020-11-25 02:34:12.809263
4	4	9212	[null]	{1,2,3,4,5,6}	2020-11-25 02:34:12.810673	2020-11-25 02:34:12.810673
5	5	12212	2	{1,2,3,4,5,6}	2020-11-25 02:34:17.872031	2020-11-25 02:34:17.872031
6	6	12212	2	{1,2,3,4,5,6}	2020-11-25 02:34:17.873792	2020-11-25 02:34:17.873792
7	7	21393	5	{1,2,3,4,5,6}	2020-11-25 02:34:28.975451	2020-11-25 02:34:28.975451
8	8	21393	5	{1,2,3,4,5,6}	2020-11-25 02:34:28.976449	2020-11-25 02:34:28.976449
9	9	12273	2	{1,2,3,4,5,6}	2020-11-26 22:59:16.905557	2020-11-26 22:59:16.905557
10	10	12273	2	{1,2,3,4,5,6}	2020-11-26 22:59:16.906217	2020-11-26 22:59:16.906217
11	11	20815	5	{1,2,3,4,5,6}	2020-11-26 23:14:57.90415	2020-11-26 23:14:57.90415

In our database there are now 11 goal events as expected with the correct parameters: player id 5, match id 5, time 20815, and lineup [1,2,3,4,5,6].

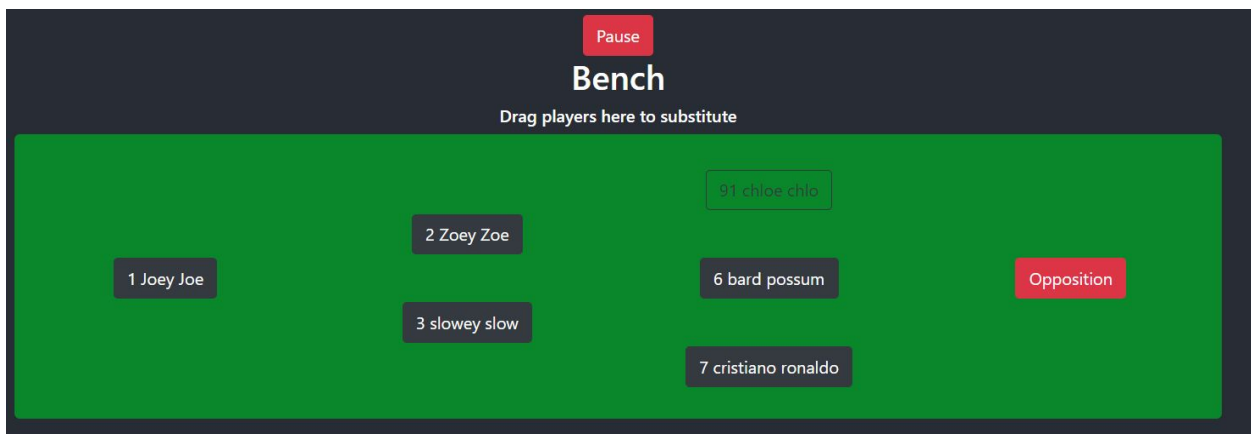
### 2.1.10 Game Event - Touches

**Acceptance:** A ball touch event is persisted with the time it occurred and associated with the current game and the player who touched the ball.

From the recording interface (see test 1.4.2 to see how to get to):



Starting the timer continues a match session. At this point we can click on a player to dictate them having a touch of the ball. The player will be highlighted to indicate that they have taken a touch and have possession of the ball:



We confirmed the touch is sent to the database and saved. The absolute amount of touches is then used to see if our best players are getting enough touches in later reports.

## 2.1.11 Game Event - Ball Possession

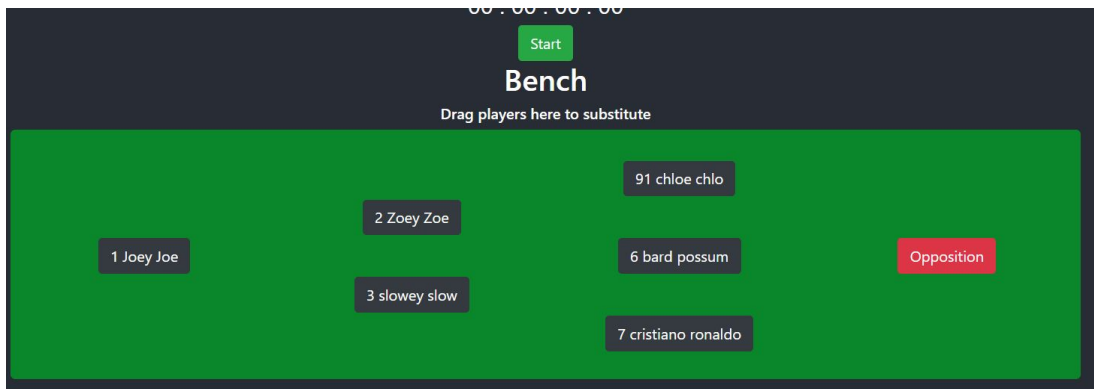
**Acceptance:** A possession change event is persisted with the time it occurred and associated with the current game.

Similar to 2.1.10, clicking on a player activates that player having possession. Clicking on another player or the opposition stops this possession. The absolute amount between our team and the opposing team is used to verify percentage splits of possession in later reports. This is verified in the database, passing the acceptance test.

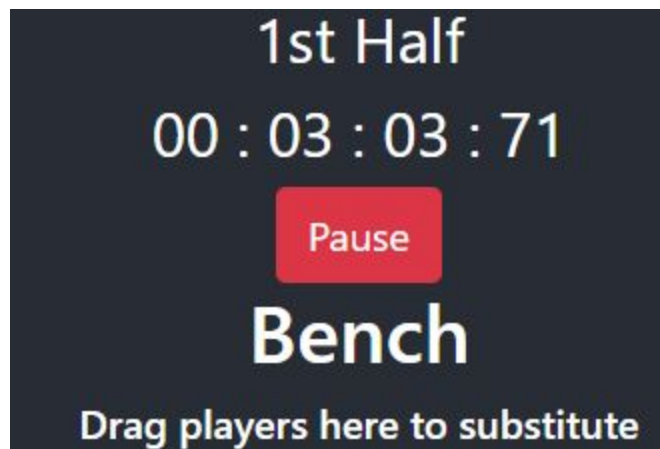
## 2.3.1 On Field Timer

**Acceptance:** At the request of the coach they should be able to view the amount of time each player on the current lineup has been playing.

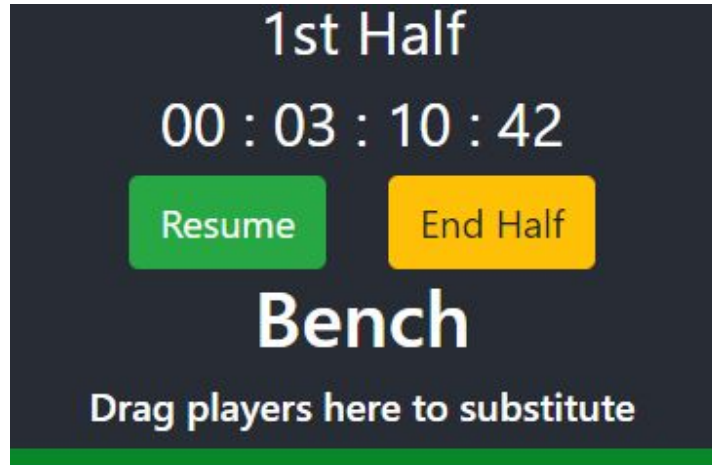
Starting from the recording page (see test 1.4.2 to see how to get to):



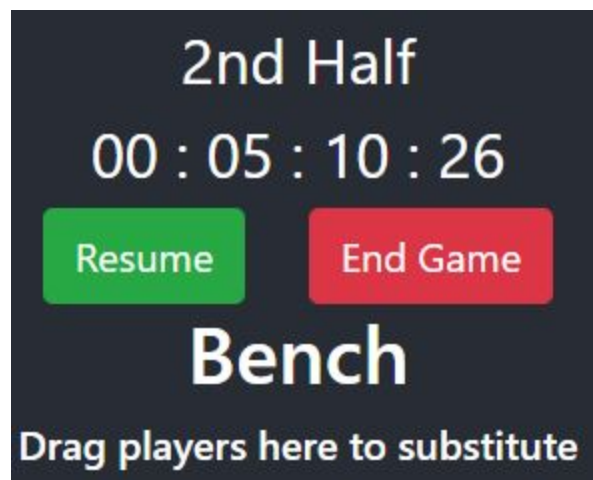
Clicking start on the timer activates the timer:



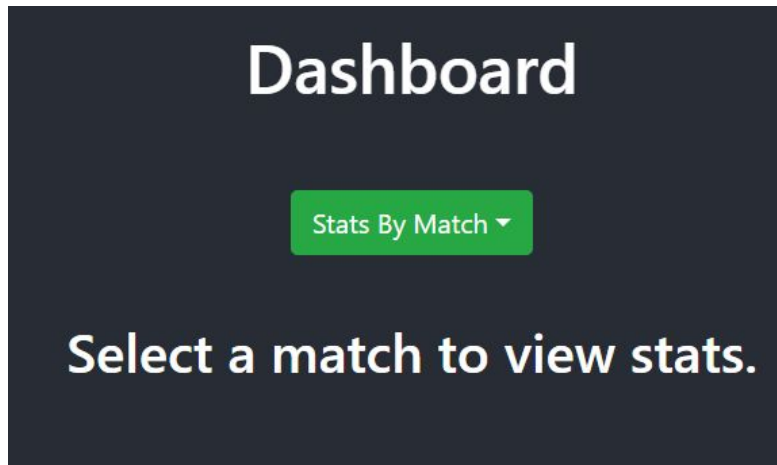
This timer can be paused by clicking pause:



Clicking resume was then selected to ensure the timer was still working. We then pause again and click “End Half” to end the half, we verified this sends the half-time of the match to the backend, which can be used for splitting statistics by half later on. Clicking the end half should restart the timer to zero and update the half.



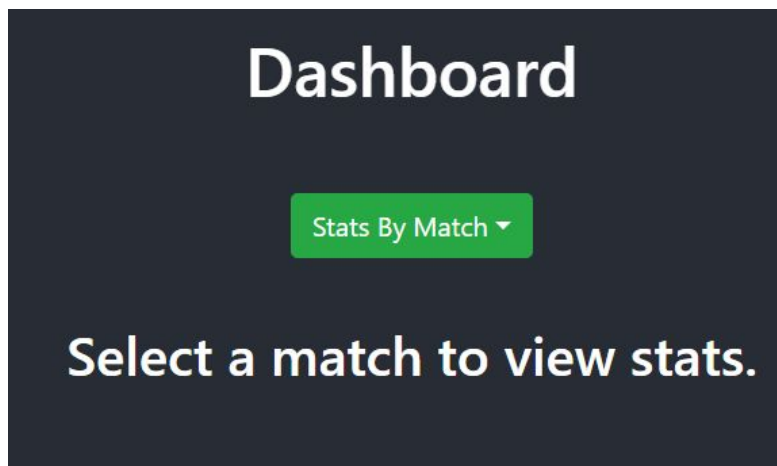
We verified the previous steps also work for the second half. Pausing during the second half will now end the match and bring the user to the select match dashboard:



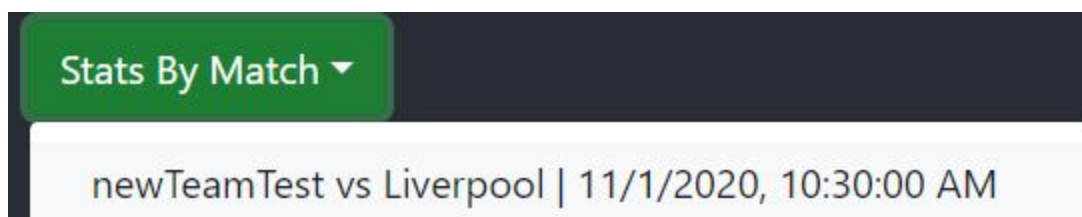
### 3.1.1 Stat Dashboard

**Acceptance:** User can navigate here and view the match summaries.

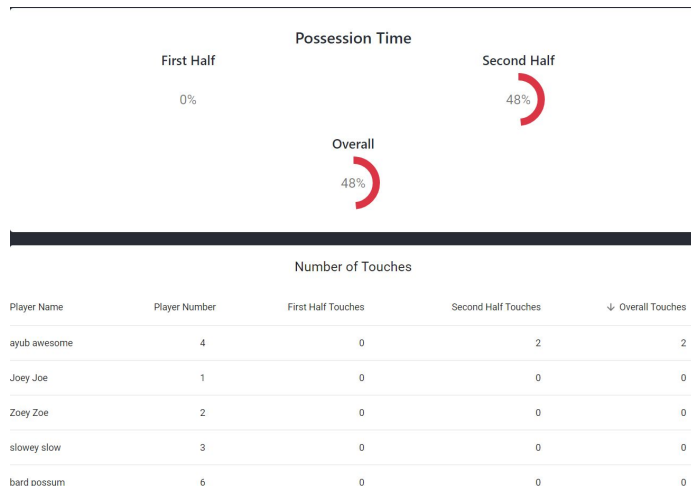
From the main dashboard:



The user can click the Stats by match button and choose a match:



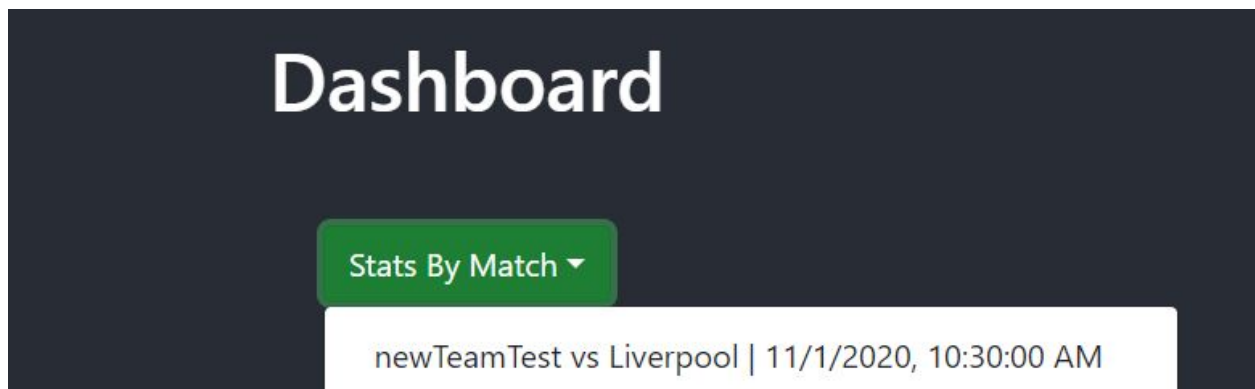
Choosing a match takes them to a page with statistics from that match:



There are multiple reports on this dashboard that can be found in the following tests. This passes the acceptance test as the stats dashboard is easily navigated too and contains tables summarizing the match statistics.

### 3.2.1 Stat- Time on Field

**Acceptance:** Can view time on the field for a given match on the stat dashboard. This table will have data for all individuals that played in that match.



From the stat dashboard (main page on application) we can select a completed match, selecting it will take us to a page with numerous reports, one of these reports is the Time On Field for all players in a given match:

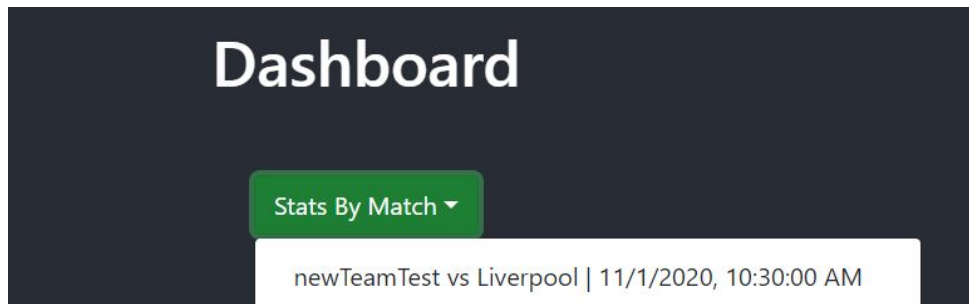
Time on Field		
Player Name	Player Number	↓ Minutes Played
Joey Joe	1	8.8
bard possum	6	8.8
cristiano ronaldo	7	8.8
Zoey Zoe	2	8
slowey slow	3	6.4

Rows per page: 5 1-5 of 7 < >

Therefore, this passes the acceptance test as the time on field for all the players is shown.

### 3.2.2 Stat - Plus Minus

**Acceptance:** Can view the net plus/minus for all players in a given match on the stat dashboard.



From the stat dashboard (main page on application) we can select a completed match, selecting it will take us to a page with numerous reports, one of these reports is the Plus Minus for all players in a given match:

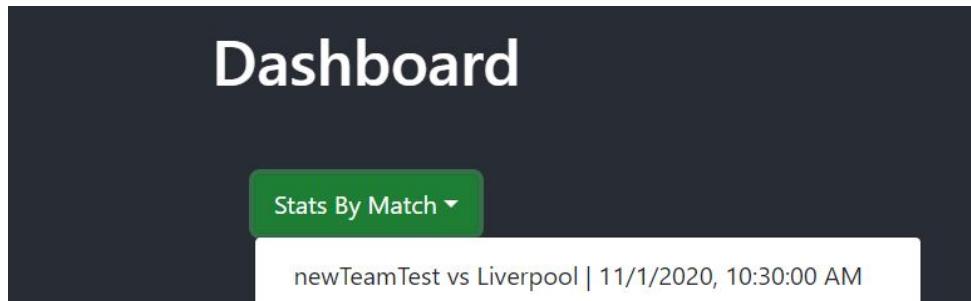
Plus Minus			
First Name	Last Name	Jersey Number	Plus Minus
Joey	Joe	1	4
Zoey	Zoe	2	4
slowey	slow	3	4
ayub	awesome	4	2
bard	possum	6	4




Therefore, this passes the acceptance test as the Plus Minus for all the players is shown.

### 3.2.3 Lineup During Goals

**Acceptance:** The user can see a list of players on the field for each goal scored.

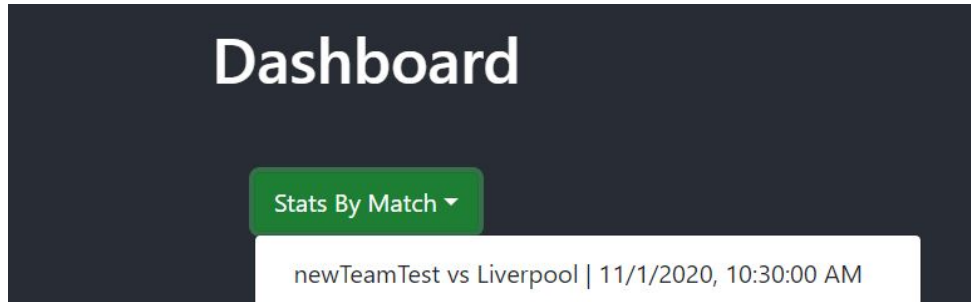


From the stat dashboard (main page on application) we can select a completed match, selecting it will take us to a page with numerous reports, one of these reports is the Lineup During Goal for all players for a given goal:

Lineup for Goals	
 opponents goal	
Goal	Line During Goal
1	Joey Joe, Zoey Zoe, slowey slow, ayub awesome, bard possum, cristiano ronaldo
2	Joey Joe, Zoey Zoe, slowey slow, ayub awesome, bard possum, cristiano ronaldo
3	Joey Joe, chloe chlo, slowey slow, ayub awesome, bard possum, cristiano ronaldo
4	Joey Joe, chloe chlo, slowey slow, ayub awesome, bard possum, cristiano ronaldo
5	Joey Joe, chloe chlo, slowey slow, ayub awesome, bard possum, cristiano ronaldo
6	Joey Joe, chloe chlo, slowey slow, ayub awesome, bard possum, cristiano ronaldo
7	Joey Joe, chloe chlo, slowey slow, ayub awesome, bard possum, cristiano ronaldo
8	Joey Joe, chloe chlo, slowey slow, ayub awesome, bard possum, cristiano ronaldo

### 3.2.6 Stat - Ball Possession Time/ Percentage

**Acceptance:** The percentage of possession for each team is reported on the stats dashboard for a match. These percentages will be available for the first half, second half, and the whole game.



From the stat dashboard (main page on application) we can select a match, selecting it will take us to a page with numerous reports, one of these reports is the Team Ball Possessions for all players for a given goal:



## 4.0.0 Selenium Automated Testing

Selenium testing results were outputted into a report generated using Mocha and Mochawesome. This report can be found in the '/selenium\_testing/ReportDirectory' relative to the root folder. PDF and html file versions are available. The reports were observed and it was found that the correct outputs were generated and all tests passed.

There were 5 tests run on Selenium covering a range of User Stories:

### Test#1:

The first test simply opens the web browser and ensures the url is correct.

### Test#2:

The second test goes through the signup process. It opens the application web page and clicks the sign up button. After this it will input the boxes for sign up, but put in not matching passwords, then clicks the sign up button. We check for the expected error message and then correct the passwords to match, and sign up.

This covers the user story 1.1.1 - Create an Account.

### Test#3:

The third test goes through the login process. It pulls up the website and attempts to sign in with an empty password, checking for the correct error message. Then it attempts to sign in with an incorrect password, again checking for the correct error message. Lastly, it will put in the correct password, click the login button and assert that it is on the dashboard page.

This covers the user story 1.1.2 - User Login.

#### Test#4:

The fourth test creates a team and a match. It starts by pulling up the website and logging in. Then it clicks the "Teams" button on the navigation bar. It then clicks add team, adds a team name, and then adds 8 players. It then clicks the add team button to create a team, and then navigates back to the dashboard. At this point it clicks the create match button, picks the team we created, and inputs an opponent name. Then it clicks create match to create the match.

This covers the user stories :

1.2.1 - Create a Team

1.2.3 - Create a Match Lineup

#### Test#5:

The final test picks a match lineup, runs through a recording session, checks to make sure statistics tables are present after the recording session, and logs out. It starts by pulling up the website and logging in. It then clicks the record button and chooses a match. This will bring it to the match lineup page where it will pick the 8 players we created to be in the match lineup. Then it clicks next to start a recording session, and clicks the start button to begin the timer. The time is checked to ensure it is changing. The pause, and end half buttons are chosen. The period is checked to make sure we are in the second half. The timer is then started again and the game is ended. This brings the user back to the dashboard. From here we click on stats by match and choose a match. Once the stats page is loaded, we check to make sure that we are on the correct url and that all the tables are properly loaded.

This covers user stories:

1.1.3 - User Logout

- 1.2.3 - Create a Match Lineup
- 1.4.2 - Start a Match Recording Session
- 2.3.1 - On Field Timer
- 3.1.1 - Stat Dashboard
- 3.2.1 - Stat - Time on Field
- 3.2.2 - Stat - Plus Minus
- 3.2.3 - Stat - Lineup During Goals
- 3.2.6 - Stat - Ball Possession Time/ Percentage

## **5.0.0 Incomplete User Stories**

All must haves were completed.

The incomplete user stories are shown below.

Should have:

- 2.1.12 - Flag An Event for Later Accuracy Check
- 3.2.7 Visualization and Event Timeline
- 3.2.9 - Export as CSV
- 3.2.10 - Stat Date Range Selection
- 3.2.11 - List and Edit Game Events Postgame
- 3.2.12 - Player Stat Summary

Could have:

- 2.1.2 - Goal Location Details
- 2.1.4 - Game Event - Throw Ins
- 2.1.7 - Game Event - Corner Kicks
- 2.1.8 - Gaven Event - Penalty Shots
- 3.2.4 - Visualization - Shots on Net Location on Field

These user stories were incomplete due to the 2 month time constraint of this project, as it was not feasible to complete these after higher priority user stories were accounted for.