CA400 4th Year Project - Testing Document

Students:
Tom Callaghan (16449672)
Philip Donnelly (17518149)
Supervisor:
Dr Donal Fitzpatrick
Date: 03/05/2021

Abstract: *The overview of our project Last Man Standing is to create a fully automated version of the well-known competition "Last Man Standing". Last Man Standing is a league based game where players pick a Premier League team each week to represent them for that week. If the player's team wins their game against the opposition, the player proceeds to next week's stage. If the player's team of choice loses or draws against their opposition, the player is knocked out and does not proceed onto the next stage. The last player standing wins the competition/prize. This web application automates this whole process to aid charities or organisations alike to perform this competition on their terms. This document is a testing document for the web application, to show how the system was tested and validated.*

Table of Contents

1 Longitudinal Study / User Testing (Post Ethics Approval)

With over 100+ users using *Last Man Standing*, it was critical that a lot of testing was performed. As the nature of the application is a competition there needed not to be large functional errors throughout.

Before doing any user testing we had to create an ethics form and get it approved by DCU. Once approved, we pushed the application to family and friends to perform some soft testing of features. This helped us to get rid of all the small issues missed during the development before going public with the application. Due to Covid-19, it was a slightly longer process to get user testing but the results and feedback we got were very beneficial.

Once this user testing was finished we entered into our longitudinal study where we launched the application to the public. When a user signs up they are sent the feedback form. The feedback form is not mandatory as this is a longitudinal study. If they have any suggestions or feedback on the application they can fill out the form. When a form is filled out, we are notified. If there are any issues or feedback on the forms, we take the issue/feedback and place it on our JIRA board to ensure that it will be taken care of in the next sprint. This way we could make sure that any problems users were having we were taking care of as soon as possible. The application will continue to perform this longitudinal study to ensure all user requirements are satisfied.

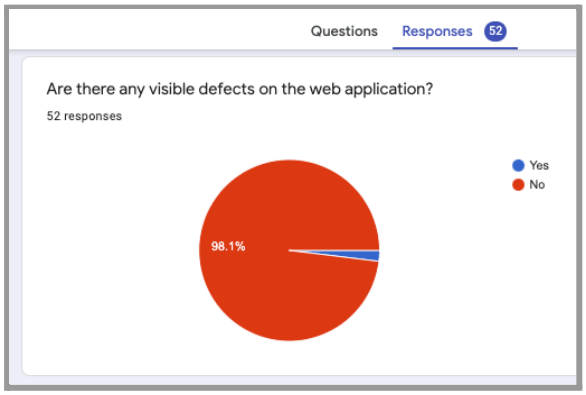
To date there have been over 50 responses received from users, this has helped to make our product what it is today. The majority of users were very happy with the visuals as well as the functionality of the application as seen in the results below.

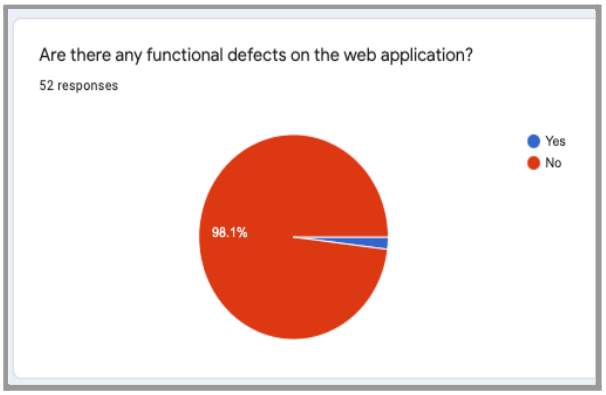
The link to the feedback form can be seen below along with some of the responses/feedback we have received so far.

Link to form: https://forms.gle/UMp2AtbcKbPxwebg8

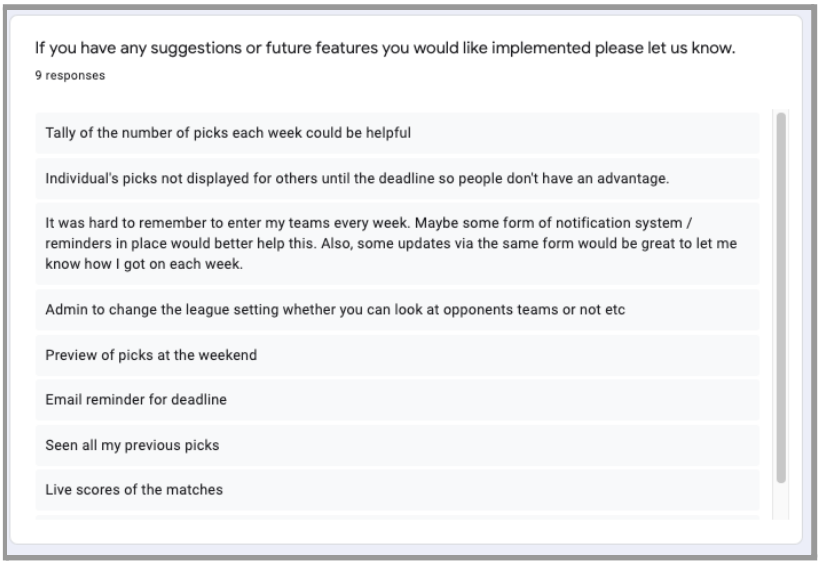
Feedback & Results:

The form asks the users if there were any visual defects or functional defects, with optional boxes to comment on the visuals and functionality. The results can be seen below.





Getting suggestions for additional features from the users was a great way to ensure that the product we were supplying was satisfying all user requirements. The suggestions/feedback can be seen below.



2 Unit Testing

Unit testing is a form of software testing which tests each component of a system to ensure the component works as expected. As *Last Man Standing* is a web application with numerous users, we needed to ensure that all aspects of the application work as intended. Below is a screenshot of the front end unit tests which test the code of all front end components of the app. These front end unit tests used the Jest Testing Library and consist of both assertion tests and snapshot tests.

Snapshot tests ensure the UI does not change unexpectedly when making changes to the code. This is done by creating a tree-like structure of the component when the snapshot is first created and is stored under a __snapshots__ folder. Then if a change is made to a code and the test suite is executed it will compare a new snapshot to the saved snapshot. If they are not the same the snapshot test will fail to indicate a change to the UI which can then be updated if correct.

Assertion tests ensure that the right information is being returned. This includes ensuring that button clicks trigger the correct functions and update the states correctly, that form inputs update the correct states and API requests result in the correct states being updated. These assertion tests ensure that our application updates the correct states to ensure the correct components and information are displayed to the user.

```
PASS  src/libs/crestBar/crests.spec.jsx
PASS  src/libs/profilePage/resultsRatio.spec.jsx
PASS  src/libs/signUp/signUp.spec.jsx (8.749s)
PASS  src/libs/leagues/selectTeam.spec.jsx
PASS  src/libs/navbar/navbar.spec.jsx (6.317s)
PASS  src/libs/forgotPassword/forgotPassword.spec.jsx
PASS  src/libs/signIn/signIn.spec.jsx
PASS  src/libs/profilePage/profilePage.spec.jsx
PASS  src/libs/splashScreen/splashScreen.spec.jsx
PASS  src/libs/crestBar/crestBar.spec.jsx
PASS  src/libs/whatIsLms/whatIsLmsCard.spec.jsx
PASS  src/libs/homePage/homePage.spec.jsx

Test Suites: 27 passed, 27 total
Tests:       57 passed, 57 total
Snapshots:   33 passed, 33 total
Time:        35.888s
Ran all test suites.
```

There were also unit tests performed on the backend of the application. As the backend lambda functions were written in Python, Pytest was used for unit testing. Unit tests were performed on all code in the lambda functions to ensure the output of the function was correct. For example, a unit test was performed on the function to calculate the over round that is taken from the bettings odds for the probability of a team winning.

5 Integration Testing

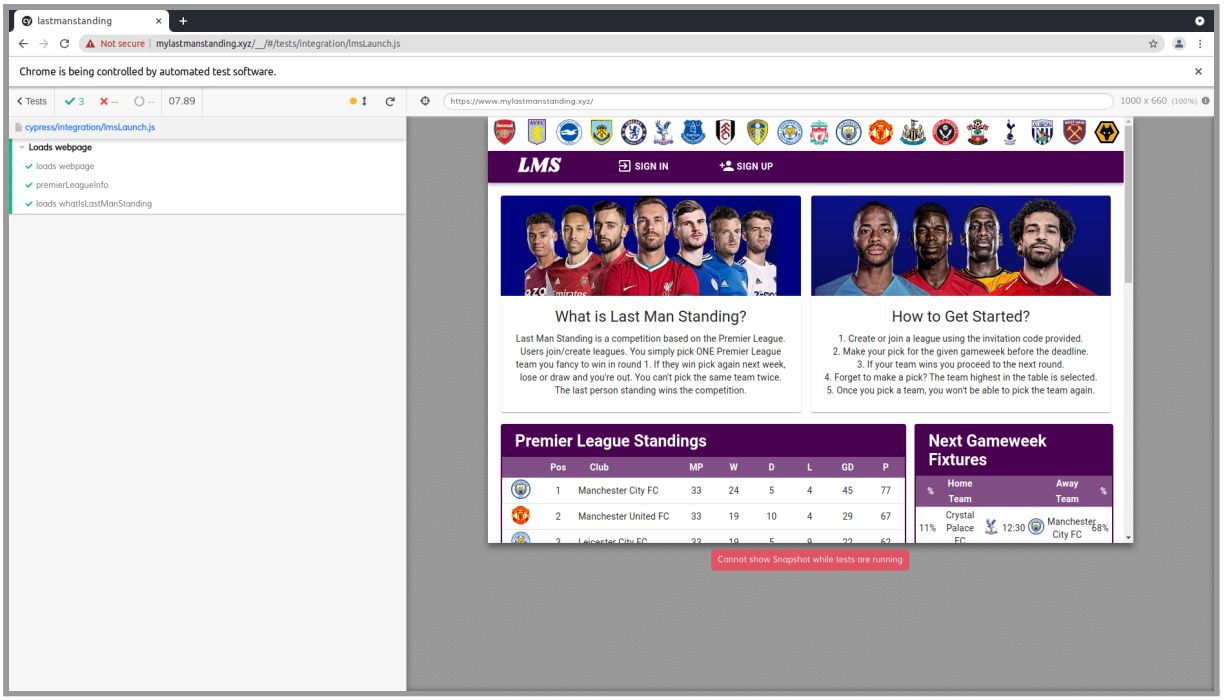
Integration testing is the phase in software testing in which individual components are combined and tested together. This was performed throughout our application. Pytest and moto were to mock the interaction between components and the database. By using moto we were able to mock a DynamoDB table to test the integration between the different components. The results from the integration tests on the backend can be seen below.

```
(venv3.8) ➜  lastmanstanding git:(master) ✗ PYTHONPATH=:$(echo $PWD) python -m pytest -s
============================================= test session starts ==============================================
platform darwin -- Python 3.8.2, pytest-6.2.2, py-1.10.0, pluggy-0.13.1
rootdir: /Users/tom/Desktop/2021-ca400-callagt4-donnep28/src/lastmanstanding
plugins: cov-2.11.1
collected 32 items

backendTests/test_LeagueInfoLambda.py .......
backendTests/test_adminActions.py ...
backendTests/test_createLeague.py ...
backendTests/test_deadlineReminder.py ..
backendTests/test_getLeagueInfo.py .
backendTests/test_getPremInfo.py .
backendTests/test_getProfile.py ..
backendTests/test_joinLeague.py ...
backendTests/test_lockLeagues.py ..
backendTests/test_myLeagues.py ..
backendTests/test_teamInfo.py .
backendTests/test_teamSelection.py .
backendTests/test_unlockLeagues.py ....

============================================== 32 passed in 4.53s ==============================================
```

Integration tests were also performed on frontend components using Cypress. For example, the authentication flow component was tested with the front end sign up and sign in components. The home page component was tested with the standings, fixtures and results components. The results from the homepage component integration with the standings, results and fixtures components can be seen below.



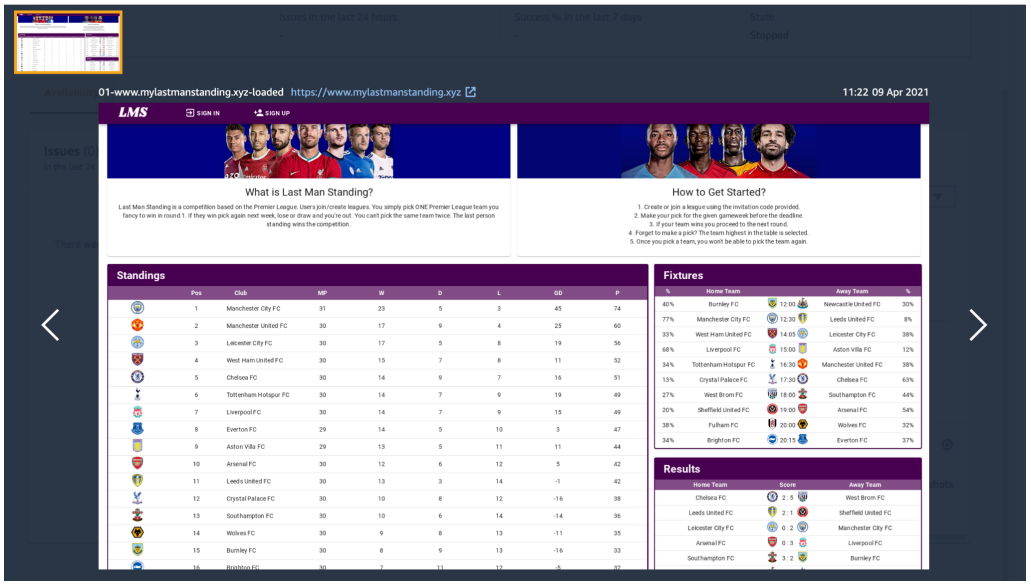
6 Regression Testing

Due to the extent of user testing performed different bugs and changes to components have been made throughout. As a result of this, a lot of regression testing had to be done to ensure that the change of one component did not affect another component/feature of the system. The best way to regression test our application was to run all unit tests, integration tests and systems tests every time a new change is pushed. This was done by running all tests in our CI/CD pipeline every time a commit was made to the repo. The results from the pipeline can be seen in the CI/CD pipeline section in the technical manual in the repo.

7 System Testing

7.1 Alarms & Canaries

To test the system on an ongoing basis due to a large number of users on the application, AWS CloudWatch Canaries were set up to run every 2 hours. This canary would take snapshots of each page of the application to ensure everything had loaded correctly. If there is an issue we would be emailed with the issue. The canary screenshot can be seen below.



There were also alarms and canaries set up on our API and main lambda functions to make sure that if something is being throttled or if there is a fault we are notified. These alarms are used to notify us of any eros along with any attacks we have on our API.

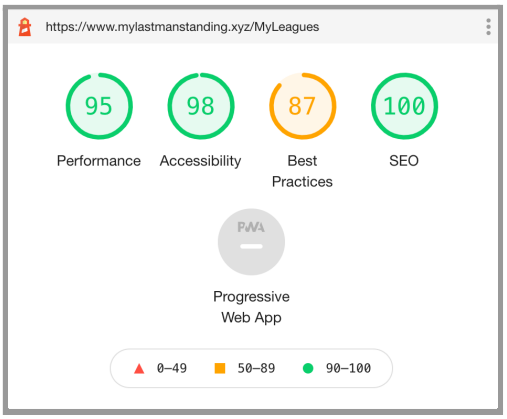
7.2 Performance / Device Testing (Mobile, Desktop)

The lambda functions in the backend were monitored while a large number of users were using the site. One of the lambda functions was operating relatively slowly in receiving the team's news. To fix this, they broke up the lambda functions to increase the speed of the operation

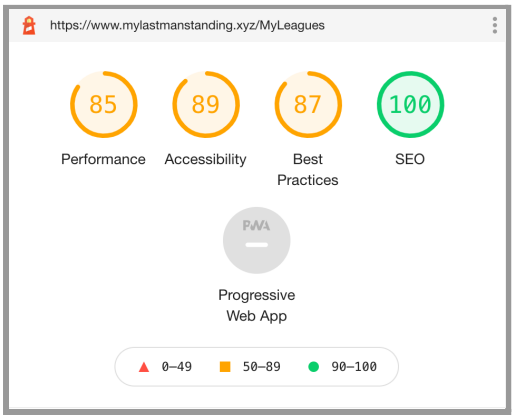
Due to a large number of users using mobile devices when accessing the web application, our web application needed to be compatible with mobile devices along with desktop.

To ensure mobile compatibility, all features implemented or large changes to the UI were checked on mobile along with the website. Several lighthouse tests were run on the mobile version to ensure not just the accessibility was correct but the performance on mobile was up to the same standard. The results for both the website on desktop and mobile can be seen below.

Desktop



Mobile



The UI components were developed to resize based on screen size. This allows it to be compatible on multiple platforms. As seen above there is a small drop in performance and accessibility on mobile this is common on web applications. The scores are still very high which suggest the application is compatible on both platforms.

8 Acceptance Testing

Acceptance testing as seen below was used to ensure the web application was meeting the necessary criteria for any investors or users of the app. The web application covered nearly all the criteria except live scores, which we found too expensive for us as students. These actions were set up early in the development to ensure all requirements were satisfied.

No.	Action	Accept	Reject	Comments
1	User must be verified before using the main features of the application	X		
2	The application runs with no standout errors	X		
3	User can successfully join or create as many leagues as they wish	X		
4	User can pick the given team once in each iteration of the competition	X		
5	User can change their pick up until the deadline	X		
6	User is eliminated if their team loses or draws their fixture	X		
7	User can view all other competitors pick during the game week	X		
8	Admin can reset the league	X		
9	Admin can stop users from joining	X		

	the league after the first game week of competition			
10	Admin can remove players from the league	X		
11	Admin can remove players from the league	X		
12	The user receives an email reminder of the deadline	X		
13	User can sign out at any time	X		
14	User receives news from the favourite team	X		
15	Live scores		X	Third-party API's available too expensive for students, not enough time to implement our own API

9 Heuristic (Schneiderman's Golden Rules)

9.1 Strive for consistency

Throughout the development of the application, we strived to stay consistent with both our design and use experience. All buttons are designed relatively the same. The same colour theme is used throughout the app. Refactoring was used in the code to try to keep the user experience the same. The use of a UI framework also helped with the consistency of components ensuring similar functionality to similar components.

9.2 Enable frequent users to use shortcuts

Several shortcuts were added to our application to ensure usability was optimised. The user can sign out no matter what page they are currently on.

9.3 Offer informative feedback

If a user is interacting with a feature that interacts with the backend of the application, we provide alerts and confirmations. This is to ensure that the user does wish to execute the task connected to the component clicked. If in the case that a user wants to do a task that results in an item being written to a database we will provide information in regards to the status of the job. Providing a successful message or an error message respective.

9.4 Offer simple error handling

Error handling was used throughout the application to prompt the user if something went wrong. Numerous components require text fields to be filled out. *Last Man Standing* has several handling methods to ensure the user is guided to entering the correct information into these text fields. If a user misses a text field or leaves a text field blank they will be prompted to enter something into the text field. The subsequent action will not be triggered if a text field is blank. Similar to this if a user enters an incorrect email when signing up they will be notified of this so they can amend the error before signing up.

9.5 Permit easy reversal of actions

Users make mistakes! This had to be in our minds throughout the whole development of the application. Throughout the whole application, there is at no point a section from which the user cannot back out from. The application has a minimalistic navigation bar on the top of each page, allowing users to return to a different page at any point during their use of *Last Man Standing.*

9.6 Design dialogues to yield closure

As a user interacts with different components on *Last Man Standing* we tried to ensure that every action had a beginning, middle and end. This helps the user feel relaxed and satisfied with their work when they complete an action. An example of this would be simply joining a new league. The beginning would be the user clicking the “Join League” button which opens a form for the user to fill out. The middle part is when the user is filling out the form as it takes the longest duration. Finally, the end would be the user clicking the “Join League” button on the end of the form. The user will feel accomplished and will see an alert notifying the user that they have successfully joined a league providing informative feedback to the user.

9.7 Support internal locus of control

The user is in complete control of the application. Throughout the whole app, the user does not have to partake in any process, unless they wish to do so. The app is there for the convenience of the user.

9.8 Reduce short-term memory load

As all the data from leagues are displayed on the page the memory required is very short term.

10 Accessibility Testing

10.1 Touchable buttons

To ensure users were aware of what buttons were clickable, we used hovering and changing the cursor to an “action” cursor. By using hovering the user can spot buttons as they change colour when hovered over.

10.2 Large fonts

Another piece of styling/design we tried to abide by when developing the web application was to make all text large as possible (without disrupting the clean/clear look) so it was easy for users to read and see valuable information on the screen.

10.3 High colour contrast

Colour Blindness had to be taken into account when testing and developing the application. To make the app accessible to people with this disability, a high colour contrast was essential.

10.4 Screen Reader

One feature we would have liked to have featured in our application was that of a screen reader for visually impaired users. Unfortunately between time constraints and lack of screen reader implementation knowledge we were unable to successfully include this in the project's final submission. Although this is a feature we would consider looking into if we were to develop this application any further.

11 Probability Accuracy

Within the application, we offer a probability of each team winning their match in that game week. This aids people who may not have a wide knowledge of the Premier League but are participating in the competition for another reason (supporting charity).

To check the accuracy of this probability, we have been actively calculating an accuracy score in spreadsheets for the last 4 game weeks. This way we can make the users aware of how accurate this probability is. It was very clear from our monitoring that the accuracy scores were not very accurate. This is due to the high standard of teams within the Premier League, although a team may have a high probability of winning a match the results tend to be very inconsistent. This results in the accuracy score being relatively low.

Game Week	HomeTeam Prob	Home Team	AwayTeam Prob	AwayTeam	Winner
31	40%	Burnley FC	30%	Newcastle United FC	Newcastle United FC
31	77%	Manchester City FC	8%	Leeds United FC	Leeds United FC
31	33%	West Ham United FC	38%	Leicester City FC	West Ham United FC
31	68%	Liverpool FC	12%	Aston Villa FC	Liverpool FC
31	34%	Tottenham Hotspur FC	38%	Manchester United FC	Manchester United FC
31	13%	Crystal Palace FC	63%	Chelsea FC	Chelsea FC
31	27%	West Brom FC	44%	Southampton FC	West Brom FC
31	20%	Sheffield United FC	54%	Arsenal FC	Arsenal FC
31	38%	Fulham FC	32%	Wolves FC	Wolves FC
31	34%	Brighton FC	37%	Everton FC	Draw
Game Week	HomeTeam Prob	Home Team	AwayTeam Prob	AwayTeam	Winner
32	23%	Newcastle United FC	50%	West Ham United FC	Newcastle United FC
32	54%	Arsenal FC	20%	Fulham FC	Draw
32	71%	Manchester United FC	10%	Burnley FC	Manchester United FC
32	51%	Southampton FC	23%	Crystal Palace FC	
32	62%	Chelsea FC	15%	Brighton FC	Draw
32	21%	Leeds United FC	56%	Liverpool FC	Draw
32	65%	Leicester City FC	14%	West Brom FC	Leicester
32	31%	Everton FC	41%	Tottenham Hotspur FC	Draw
32	59%	Wolves FC	16%	Sheffield United FC	Wolves FC
32	14%	Aston Villa FC	64%	Manchester City FC	Manchester City FC
Game Week	HomeTeam Prob	Home Team	AwayTeam Prob	AwayTeam	Winner
33	44%	Wolves FC	27%	Burnley FC	Burnley FC
33	70%	Liverpool FC	11%	Newcastle United FC	Draw
33	26%	Leeds United FC	48%	Manchester United FC	Draw
33	27%	West Ham United FC	46%	Chelsea FC	Chelsea FC
33	53%	Aston Villa FC	21%	West Brom FC	Draw
33	61%	Leicester City FC	15%	Crystal Palace FC	Leicester City FC
33	50%	Arsenal FC	23%	Everton FC	Everton FC
33	27%	Sheffield United FC	45%	Brighton FC	Sheffield Utd

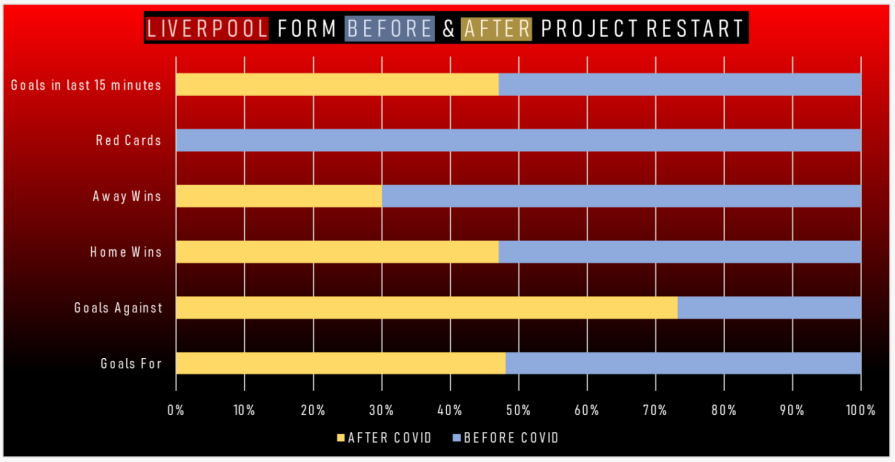
Accuracy Score (past 4 weeks): 44%

Incorrect	Correct	Total
20	16	36

Link to spreadsheet:
https://docs.google.com/spreadsheets/d/1xt7R72nfwjcZKqRpHqU6Q_uG2OckJQDopPiFIlcRis4/edit?ts=60716c19

For *Last Man Standing* the only way to progress to the next week is for your team to win. Due to this if a team draws we deem that as an incorrect production for the probabilities. In the table of fixtures above we only show the probability of each team winning but if you add the probability of the teams drawing onto the opposing team that you have selected the probabilities to skew away from the favourites or closer to a 50:50 prediction.

From our research, we found articles explaining how inaccurate predictions have been this year due to Covid-19. This unpredictability of Premier League results seems to be a result of several factors such as no preseason which has caused more injuries. As seen below, the Premier League champions of last year Liverpool saw a huge drop in wins since the return of the season.



Liverpool's form visualised in statistics, showing late goals, red cards, away wins, home wins, goals against and goals for, in pre-COVID form and post-June.

https://www.elartedf.com/statistical-impact-covid-19-premier-league-results/

12 Conclusion

To conclude, as there are 100+ users on the application at the moment, it was inevitable that the level of testing of the application had to be to a high volume and standard. Although we are aware that it is very difficult to have a fault-proof system, with 85+ unit and integration tests written as well as endless user testing, we feel that the coverage of testing the application is to a sufficient volume to be confident to have a large number of users.