# Agile Development Processes Postmortem Report

## Team 7


- Goitom Abrehaley
- Tobias Alldén
- Rahwa Araya
- Pooriya Balavi
- Alexander Graham
- Erik Karlkvist
- Johan Ljungberg


---

# Project Description

The project is to create an application for a football cup held in a city. The application will contain information about upcoming matches, scores for played matches, rankings etc and also having the possibility for users to find nearby restaurants and events, and admin functionality to post events, change scores etc. The definition of the project is as follows

> This customer is the marketing manager of a huge football cup that will take place during the summer
> of 2018. In the event, there will most likely be thousands of football interested tourist coming to town. These tourists will need a software tool to help them with, e.g., finding results, coming games, restaurants, shops, maps, other services, etc. The software tool should also help the tourists to schedule their planned activities. The marketing manager would like the tool to be simple to use by the tourist, but the organization also need to be able to publish new events (like results and new upcoming events) into the same
> software application. The marketing manager really likes advertising in general, but his knowledge about football is quite
> weak, so he will most likely need some help from the developer team to come up with brilliant ideas of football-related things that would be of interest for this specific group of tourists.

# Technical Info

Target Platform: React Native, focus on Android,
Programming Language: JavaScript,
FootballTown.

# Links

- Git repository: https://github.com/ErikKarlkvist/FootballTown
- Issue tracker: https://trello.com/invite/b/eOCzPjKE/00fdf0bbdaa001eb9c3829cbf87c03df/footballtown
- Continuous integration builds: https://travis-ci.org/ErikKarlkvist/FootballTown

---

# Sprint 1 Log

## 2018-03-22: Initial meeting

- Populating the backlog
- Selecting platform and programming language
- Initial team meeting with customer, coach and the full team

## Commitment

This (short) sprint we have committed to setting up the project and learning a bit about React Native and git for those who had not spent any time with this before.

## Work Done

| Feature | Time estimated | Time spent per team member |
|---|---|---|
| Initial setup | *3.5 hours* | All members spent the same amount of time |

## Reflections

Initial meeting worked well. We populated the background and set up the environment.

---

# Sprint 2 Log

## Commitment

During this sprint we committed to doing the following:

- Creating the navigation bar at the bottom of the app
- Add views for each tab (page) in the app
- Creating a games component allowing you to view the games
- Creating a news component allowing you to view summaries of the latest news items
- Creating a structure for colouring and styling application elements
- Creating database structure for games and news items

## Work Done

| Feature | Time estimated | Time spent per team member |
|---|---|---|
| *1: Nav bar* | *4 hours* | *Goitom 2.5 hours, Rahwa 2.5 hours* |
| *2: Start Views* | *1 hour* | *Goitom 0.5 hours, Rahwa 0.5 hours* |
| *3: Game Component* | *3 hours* | *Tobias 0.5 hours, Pooriya 2.5 hours* |
| *4: Game Component* | *3.5 hours* | *Tobias 3 hours, Pooriya 0.5 hours* |
| *5: News Component* | *2 hours* | *Tobias 2 hours, Pooriya 0 hours* |
| *6: Game Database Structure* | *3 hours* | *Erik 1 hours, Alex 1 hours, Johan 1 hours* |
| | | |

| | | |
|---|---|---|
| *7: News Database Structure* | *6 hours* | *Erik 2.5 hours, Alex 2.5 hours, Johan 2.5 hours* |
| *8: Color config file* | *3 hours* | *Erik 1 hours, Alex 1 hours, Johan 1 hours* |
| *9: Discussion: Task allocation & Prioritisation of backlog* | *7 hours* | *Full team 1 hour* |
| *10: Setup travis* | *4 hours* | *Erik 2 hours, Tobias 2 hours* |

# Reflections

As a team we worked well together this sprint. Everyone attended all group sessions and thoroughly contributed to group meetings. We worked primarily in the second week of the sprint due to Easter. We adopted most of the agile principles introduced from the literature although our primary focus over this sprint was pair programming and was divided as can be seen in the Work Done table.

## Pair Programming

One of the observations we made from pair programming is it often fostered conversation on how to complete tasks and resulted in more efficient code and less errors to debug as the non-programmer in the pair would often spot mistakes. In theory the practice could be inefficient as it relies on two people doing the same task so double the time, although we noticed it meant faster iteration as less time was spent stuck on problems. The course text states "few companies have applied pair programming to “all” their developments for very long. But many programmers have found some dose of pair programming beneficial" (page 105). Our experience aligns with finding this useful, especially considering The React Native framework is new to many of us, however we can also see how this may be challenging to use all the time. From our own experience we often would not be together or have clashing timetables so this would not be practical. One of the areas of pair programming we found the most valuable was the opportunity to give each other continuous feedback, whether positive or negative. This factor helps each individual member to improve themselves. Resource utilization is also one positive aspect of pair programming as it only requires one computer per two-person team, thus making advances in the project possible if any member should not have their computer available.
At the beginning, it was little hard to apply the practice of agile/pair programming, since we are seven members working on a single product. But we managed it by dividing the project into smaller modules and assigning two to three members to work on each module. Even though this action was very great to practice the principles of agile, it was little hard to manage

the team as well as to synchronise the modules.

## Scrum Meetings

Another area of Agile we used a lot was the scrum meetings. Each dev session we worked we would all discuss where we were up to and assess our progress. This was particularly useful as it allowed us to see how the project was evolving, what new modules were being added and provide feedback to the rest of the group. This helped keep the group familiar on the structure of the project and added code at a rate we could have an understanding of what things do even when we didn't write that code ourselves. It also allowed the team to be up to date with everyone's development and whether the project is reaching a bottleneck or the continuous integration is working well. Further, it also allowed us to work together and help one another to fix any errors or unclarities with the project, for example if one of the teams did not know how to solve a given problem or implement something.
In general, in the scrum meeting we tried to answer 3 questions

```
1) What did we do yesterday?
2) What will we do today?
3) What obstacles do you anticipate?
```

Once these questions were answered we could proceed with the development. These questions allowed us to make improvements in our project and to increase the motivation of involvement in the project.

## Product Owner/Customer

The project has a customer which knows what different functionalities are the most important for their business case. During the first sprint presentation we noticed some difficulties of managing our customer expectations as the customer would want more features that we believed we could deliver in the given time-frame. To combat this we had to work with the customer and make tradeoffs to move some functionality to the backlog for upcoming sprints. The customer will allow for continuous feedbacks on the product backlog and priorities for the upcoming iterations.

## Code Ownership and Refactoring

Some of the group members were new to GitHub version control and the React Native framework, thus working with Pair programming helped the team to tackles issues by sharing knowledge among each other. Using Git and version control provided the opportunity for members to work on a task simultaneously and then later show each other their work.

Additionally, version control was beneficial as it allowed us to revert to previous versions of the code for debugging and quality control purposes (when required). Also it facilitates concurrent development as each team could work on their own branch and merge these with the other work once the feature was completed, which is one of the main advantages of version control.

## Continous Integration

To manage continuous integration, travis is used. This allows the running of the application whenever a new change is made to the project and if this change makes the application not run, we are notified via an email. There were some issues setting up travis as the project are dependant on several modules such as cocoapods (dependency manager) and the testing framework. The tests refused to run given that the project was missing these dependencies, forcing us to spend some time configuration the travis file to make it run the application. This was completed and the integration platform is now up and running.

## Communication Channels

Other than the face-to face meetings we conducted during the sprint (for example during pair programming sessions), another communications platform was needed. This platform needed to be both efficient and support richness of communication to allow for project progress outside of the 'physical' meetings. For this task, slack was selected. Slack is useful as it allows to split the communication into several different channels which could all have a different purpose (one for development, one for non-work related banter etc) and also supported integration for various external applications, such as travis. Thus allowing us to get notified in the slack once a build had failed.

To summarise, the agile techniques we focused on this sprint complemented each other well. None of them produced conflicts in methodology and they helped the team to work together efficiently. For the following sprint we will continue to work with agile methodologies and also add new items to the application. As the members are now more familiar with the framework and development environment, the pace will hopefully be higher the next sprint.

Pair programming was good when we learned about React Native but it also limited how much we were able to do. Two or three team members were doing the same thing and each task took longer time than if we would have done it individually. Therefore we should be able to increase the workload of the upcoming sprints when we move away from pair programming to more individual work.

# Sprint 3 Log

# Commitment

During this sprint we commited to:

- Create app concept plan & designs
- Overhaul the bottom navigation to match concept
- Creating the admin view
- Create Article views for Events and News
- Create feed views for Events, News, Matches and Ranks
- Testing the created components
- Create the home page
- Overhaul the look of the application
- Create Mockup

# Work Done

Log what was accomplished, and how.
Please report on all activities; for example, in addition to coding, planning and design discussion.

| Feature | Time estimated | Time spent per team member |
|---|---|---|
| *Sprint Discussion* | *7 hours* | *Each team member: 1 hour* |
| *Planning Poker* | *3.5 hours* | *Each team member: 0.5 hours* |
| *Design Mockup* | *17 hours* | *Alex: 10 hours, Pooriya: 7 hours* |
| *Overhaul bottom navigation* | *3 hours* | *Alex: 3 hours* |
| Create Admin View | *5 hours* | *Erik: 5 hours* |
| Create Event View | *2 hours* | *Pooriya: 2 hours* |
| Backend database functionality | *2 hours* | *Erik: 2 hours* |
| General Bug fixing | *3 hours* | *Erik: 3 hours* |
| News Article View | *4 hours* | *Johan: 4 hours* |
| | | |

| | | |
|---|---|---|
| Game Details View | *8 hours* | *Johan: 4 hours, Rahwa: 4 hours* |
| News Feed View | *6 hours* | *Tobias: 6 hours* |
| Homepage view | *4 hours* | *Tobias: 4 hours* |
| Games Navigation | *4 hours* | *Goitom: 2 hours, Rahwa:2 hours* |
| Games Tab | *4 hours* | *Goitom: 4 hours* |
| Points Tab | *3 hours* | *Goitom: 1.5 hours, Rahwa:1.5 hours* |

# Reflections

The group worked as a team to carry out the tasks defined in the backlog and tackle the problems that occurred during development. All team members showed up to all meetings with a positive attitude. Each member contributed to the progress of the project relative to their level of programming competence. Once again team-work and pair-programming played a major role in accomplishment of tasks. This sprint the work was more divided than previously, making concurrent progress more easily than before. Further, all the different agile practices used complemented each other and made the structuring and administration of this sprint easier than the ones before. The different practices used in the sprint are described below.

## Backlog Refinement

We spent some time refining the backlog this sprint. Initially we had Epics and some user stories, but to break these down into more manageable chunks we spent some time refining these by discussing the epics trying to break these down as far as possible. Once the epics were broken down planning poker was conducted. The tasks that were to be undertaken during this sprint were the following:

### 1.0 Admin View & Functionalities [EPIC]

The admin views and functionality will contain the necessary views and tools for an admin to add items to the backend, for example games and events.

- **1.1 Create/Edit Event:**
  As an admin I want the functionality to create and edit events.

- **1.2 Create/Edit News Article:**
  As an admin I want the functionality to create and edit news articles.
- **1.3 Create/Edit Game:**
  As an admin I want the functionality to create and edit Games.

## 2.0 Homescreen View [EPIC]

- **2.1 Display Recent relevant information:**
  As a user I want a simple representation of the relevant recent information for the cup so I can keep up to track quickly.

## 3.0 Feed View [Epic]

The feed view will contain news and events.

- **3.1 Top Bar Navigation:**
  As a user I want to be able to switch between two feed tabs, one for news and one for events. (Implemented through a set of tabs in the top bar)

- **3.2 Event Tab:**
  As a user I want a tab to view a list of upcoming events for the football cup.

- **3.3 News Tab:**
  As a user I want a tab to view a list of recent news related to the football cup.

- **3.4 Event Page:**
  As a user I want an event page to display the information about a specific event when selected from the events feed.

- **3.5 News Page:**
  As a user I want a news page to display the information about a news article when selected from the news feed.

## 4.0 Games View [Epic]

Information about games and the current points for the different teams

- **4.1 Top Bar Navigation:**
  As a user I want to be able to switch between two games tabs, one for news and one for events. (Implemented through a set of tabs in the top bar)

- **4.2 Games Tab:**
  As a user I want a tab to view a list of upcoming events for the football cup.

- **4.3 Points Tab:**
  As a user I want a tab to view the current scores in the cup for the different teams.

- **4.4 Game Page:**
  As a user I want a game page to display the information about a specific game when selected from the games feed.

## 5.0 Mockup

We also decided on creating mockups for the different screens that would be created during the course of the sprint, as to help us with the design and to get a consistant look and feel in the application.

# Planning Poker

Planning poker is one of the Key agile practices we used while we develop our project in this sprint, this was used to approximate how exhaustive the different tasks were. As per standard practices the planning poker consisted of every team member assigning a score to each task based on their understanding of the difficulty of the task. Once every team member had given the task a score, the two team members with the smallest and highest score discussed with the group why they gave the task the score they did to try to reach a consensus.

The total points of the tasks are to be used in the upcoming sprint planning as an estimation of our velocity, thus giving us an approximation of how much we can manage to complete in the next sprint. This was advantageous in planning however took quite a while to conduct. Also, as we have not been working together as a agile team for a long time period we occasionally found it difficult to guage timings accurately.

# Pair Programming

Even though the notion of pair programming was adopted during this sprint, members did not sit with each other to code as much as the previous sprints. This was because the team strived to work more at home in order save time and keep up with the larger work-load. We found Slack particularly useful for keeping in touch. We often used the meetings to go through each other’s issues and problem. This allowed the members with better programming skills helping others with their issues.

## Scrum Board

To be able to keep track of who is doing what in the project, we are using a scrum board with all the tasks to be done, as well as the backlog for future sprints. This scrum board is hosted on Trello. Every time a developer starts a task, they put their name on the card and move it into the "In progress" section of the board. When they are eventually done, they move it into the testing column and later into the "done" column. This proved useful as it reduces the risk of people doing the same tasks when they are unsure of what to do.

## Scrum Meetings

Scrum meetings, one of the backbones of the agile methodology, played a crucial role in keeping members up to date with each other’s progress and the project’s current situation. During these meetings, members give a short summary of what they have been working on and how it went with their tasks. In regard to individual's progress, members gave each other feedback and comments. Furthermore, future ideas and plans were discussed whilst a backlog was prepared and updated for the upcoming sprint. We found the time these meetings took worthwhile.

## Product Owner/Customer

The customer’s level of satisfaction regarding progress and development of the application was higher compared to the previous sprint, when she requested for more features and asked for prioritization of tasks. Additionally, the customer appeared to be happy with the layout of the application and the available set of features.

## Code Ownership and Refactoring

During this sprint, there were less problems with GitHub and version control as almost all the members had gained decent knowledge about the GIT commands and how they can collaborate with GitHub. In the beginning the group used various branches to work on their tasks but that seemed to be tedious and inefficient as each individual branch was not updated with other’s latest work. Thus, a branch was created called “develop” which was dedicated to this sprint and everyone pulled and pushed to it, resulting in faster iteration and overall better productivity. The only trade-off was slightly more conflicts.

## Continuous Integration

Travis was originally the team's first choice for carrying out the testing of application. However implementing Travis in our project and getting it to work with Firebase, has not been easy. The team is still working on the issue to be able to carry out the testing. Using Travis would utimatly allow us to run the application whenever a new change is made to the project and if this change makes the application not run, we are notified via an email.

## Communication Channels

As in the last sprint, we were using Slack as our primary communication channel. This week we also created a shared folder on Google Drive where we put a document with user stories and epics.

## Incremental Design

During the first two sprint we have been working based on simple white board designs that includes the basic functionalities of our project. But in this sprint we produced a rich design of our project through continuous discussion with the team members and our customer. The designs are now finalized as mock ups and uploaded to our project directory in order to have common understanding among the team members and the customer to avoid major modifications while the project is in its around final stage.

## Work at a sustainable pace

In order to avoid degrading quality and errors we worked at a sustainable pace. This really helded utilize our efforts wisely and learn from every sprint.

# Sprint 4 Log

## Commitment

During this sprint we commited to:

- Create Team Selection such that information about a given team can be viewed
- Create a splash screen.
- Show players in a team
- Finalize Start screen
- Get overall similar look and feel in the application

- Testing the created components

# Work Done

Below follows the various tasks conducted in this sprint

| Feature | Time estimated | Time spent per team member |
|---|---|---|
| *Sprint Discussion* | *7 hours* | *Each team member: 1 hour* |
| *Map implementation* | *4.5 hours* | *Tobias: 4.5 hours* |
| *Splash Screen* | *1 hour* | *Tobias: 1 hour* |
| *Teams Rank page* | *5 hours* | *Goitom: 4 hours, Alex: 0.5 hours* |
| *Styling the Event page* | *0.5 hours* | *Pooriya: 0.5 hours* |
| *Team view* | *2 hours* | *Pooriya: 2 hour* |
| *Team view styling* | *2 hours* | *Alex: 2 hours* |
| *Pick team view* | *3 hours* | *Alex: 3 hours* |
| *Players view* | *3 hours* | *Alex: 3 hours* |
| *App logo & icon* | *2 hours* | *Alex: 2 hours* |
| *The My-Team connection to database* | *3 hour* | *Pooriya: 3 hour* |
| *Write Tests* | *4 hour* | *Rahwa: 4 hour* |
| *Create Team Admin View* | *2 hours* | *Erik: 2 hours* |
| *Admin update Events, Games and News* | *4 hours* | *Erik: 4 hours* |
| *Fixing bugs and styles* | *2 hours* | *Erik: 2 hours* |
| *Generally helping others* | *2 hours* | *Erik: 2 hours* |
| *Adding realistic data to database* | *3 hours* | *Johan: 3 hours* |

| | | |
|---|---|---|
| *Adding realistic data to database* | *3 hours* | *Johan: 3 hours* |
| *Adding games and table to the homepage* | *3.5 hours* | *Johan: 3.5 hours* |
| *Fixing page with game details* | *0.5 hours* | *Johan: 0.5 hours* |
| *Linking "Load more" buttons to corresponding screen from homepage | *1 hour* | *Tobias: 1hour* |

# Reflections

During this sprint we continued to work in the same fashion as in the earlier sprints, we also had some idea of our velocity given the completed tasks in the former sprint. However, as this is our final sprint we did not do any planning poker but instead took on finishing all the tasks as we felt that this was manageable. All the promised functionalities of the application seem to be implemented, but good be polished a bit more. Some minor styling may be done after the final sprint to make the application look as good as possible.

## Pair Programming

Like the previous sprint, pair programming was not largely utilised this sprint as each member have an understanding about how the framework works. However, we did spend time working together such that potential errors could be quickly solved which yields somewhat of a similar approach to the pair programming methodology. The main reason pair programming was not used was a lack of time, where we wanted to deliver as much as possible each sprint.

## Scrum Board

The scrum board was utilised in this sprint as well, similarly to the previous one. It gave an understanding of what was in progress and how much was left to do. However, sometimes the scrum board was forgotten and there were occasions were team members worked on the same thing without knowing about it. While the scrum board is good within itself, it requires that each team members actively uses it. Most team members have been good at this, but there have been moments of bad communication within the group. This miscomunication and lack of scrum board usage may be reduced in a real-life setting in which the programmers work in the same enviroment (office) and have access to a physical scrum board, such as a white board with post-it-notes. As this is not feasable in this kind of educational project, a digital scrum board such as trello yields a simular result. However, as previously stated, this requires that the team members actively uses it.

## Scrum Meetings

The scrum meetings consisted of catching up on what everyone was doing and working together as described in the *pair programming* section. The meetings seem very beneficial for the success of the project as members assisted each other with their problems or even gave each other new ideas for making the code better. However, the meetings during this sprint was less formal and did not really reflect a scrum meeting in its ordinary fashion. The meetings were more an oppertunity to discuss the work we were undertaking, help each other with potential problems that had arisen and try to get the look of our individual parts to become consistent.

## Product Owner/Customer

The customer satisfaction after the previous sprint was positive, the customer vented their opinions and wishes for this sprint and we as a development team tried to reach a consensus on what could be delivered within the final timeframe. As the number of sprints was very limited, the wishes of the customer may not have had developed/changed as much as in a real life project which made our approach to the project somewhat linear as the initial planned version of our app largely reflect what is in our final version. In a real life setting, scope creep may be more apparent thus making larger compromises needed.

## Code Ownership and Refactoring

As predicted, there were almost no problems with GitHub and version control. All members seem to be adopted to GitHub and how it works. We did some refactoring, mainly moving the navigators from inside views to their own classes, which made the structure of the files better.

Overall the file structure is rather lacklustre, but due to time limitations it has not been a priority. If this was an actual application that were to be released, then we would probably spend more time cleaning up and be more consistent with the architecture and naming. For example, some files are named using camelCase, and some using underscores. The internal definition of what a component is versus a view has not been clear to all group members, leading to some unnecessary confusing when looking for specific parts in the code. Finally, indentation is also heavily inconsistent, in some places two spaces were used, others 4 and also alignment is often incorrect. The spacing issue is most likely due to the team using different text editors. These problems could largely be combated by defining code- and naming conventions at the beginning of the project and by having code reviews during the development cycles.

## Testing

Testing using Travis has worked well this sprint and we've written test for the database classes, where we tested Login User and CRUD for Events, News, Teams and Games. We added some rendering tests for the main tab views, however found it hard to test more than this since we don't do any heavy computations, we just downloads data from the database and display it.

As we are using Firebase, most of the backend code is given by the library, and is not tested by us. We also couldn't test anything with actual HTTP calls, since firebase doesn't allow it. We therefore had to mock all functions that we use in Firebase, which took a lot of effort and time. All in all, we've learned how to setup a testing environment for react native, as well as creating mock-ups, while developing this app, but not much in terms of actual testing and how to do test driven development.

## Continuous Integration

After each team has worked separately on specific feature based on the catalogue on the scrum board, continuous integration has been done by the team seating together. More than the previous two sprints.

In this sprint continuous integration has been used throughout. As testing was utilized more in this sprint and the product is in its final stage continuous integration were more useful.

This gave an approximation whether the changes made the tests fail or pass.
Working alone, each team member produces many line of code which may have some integration bugs when it is integrated with his/her team members work. Sitting together during integration has helped us minimise the impact of these bugs as they are picked upon immediately and the offending coders should be able to diagnose the cause more easily.

## Communication Channels

Slack was largely utilised as the main platform for communication. This was consistent with communication in previous sprints and we have no new observations regarding its usefulness.

# Post-mortem

# Reflections

The group's overall impression of the project is positive. The team not only learned and expanded their knowledge regarding agile practices but also learned the fundamentals of creating a mobile app, using the React Native framework.
The team adopted the essential practices of agile to develop the application that helped the project go forward smoothly and allowed the team to cope with changes very well. Practices such as pair programming, prototyping, test-driven development, refactoring, continuous integration and scrum board were crucial in accomplishing the project.

## Pair Programming

Pair programming was largely utilized in the earlier phases of the project and served as a means for the team members to get familiar with the framework as only one of the team members had previous experience. The idea was to speed up the learning process by combining the knowledge team members had collected from reading up on the framework.

One of the main advantages of pair programming is that the code will supposedly be of a much higher quality than if a single programmer were to write the code. This is due to the other, monitoring programmer, who can spot potential errors and technical debt that may find its way into the code. This is more effective once both programmers are knowledgeable in the frameworks and language they are working in. As such, given that most of the team were unfamiliar with the framework we might as such not have noticed the full set of advantages related to pair programming, but it did help us get familiar with the framework faster. A side advantage of this is it also helped us to get to know and understand the team faster.

As the project went forward and members were assigned to their own tasks, pair programming started to evolve in format. Each person would write code for specific tasks until the next sprint and if they had any issues, they discussed it during the next meeting, getting help from team members. This allowed a rather quick and effective pair programming session for the group where it made members work together when it was necessary.

## Scrum Board

The scrum board proved to be a very valuable asset to our team during the project. The board allowed us to keep track on the current state of the project such as completed tasks, tasks in progress, etc… It allowed for a backlog where new ideas and items delayed to future sprint could reside. As such, the scrum board was one of the central practices used in the project.

The scrum board was realized using Trello as we did not have the necessary resources for a

physical scrum board. The online version gave the same functionality as a physical one would. However, there were some drawbacks with the board, such as people forgetting to update the cards. One example of this was when several people would work on the same task as the individual that had initially taken on finishing the task had forgot to update the task to "in progress" on the board. This caused unnecassary duplicate work and potential delays to the project. On the positive side, these duplications of work were often discovered early and did as such not cause too much of a distraction to the project. Having a physical scrum board could potentially fix this issue as a developer could physically "take" the task if it were written on, for example a post-it note. Overall the board was an extremely important feature to the team.

## Scrum Meeting

Throughout the entire project we found scrum meetings the most important and useful agile practice. In each meeting, every group member has participated and tried to share their experiences and challenges faced. Moreover, this practice was useful in determining challenges that developed during the project work and enabled us to figure out which tasks were requiring more work/time. It was vital to facilitate the development of the project iterations and track the project progress.

During these meetings, the members not only talked about the problems they were facing but also their achievement and what they have learnt. Members also reviewed each other’s progress and gave feedback, especially when it came to the user interface and designs.

## Product Owner/customer

Meeting the customer on set dates pushed the team members to be accountable for their work and take more responsibility with their tasks. This consequently made the team work together to accomplish the tasks on the backlog in order to satisfy the customer and get feedback in order to develop the application in regard to customer’s preferences. Having meetings regularly was particularly useful as it meant the customer could change their mind about decisions without massive setbacks in terms of workload.

## Code ownership and refactoring

During the early sprints the responsibility of setting up the GitHub version control and continuous integration builds tools, such as Travis was under the responsibility of specific group members. This due to most of the group members were new to GitHub version control and the React Native framework. But as we could proceed with the higher sprints this responsibility was shared among all group members. This has really helped us to minimize the

set-up time for the project environment.

Throughout all sprints, we worked through the features listed in the backlog. The backlog was populated with features/modules based on the interest of our customer. The team could prioritize and select specific modules for each sprint. The selected module was moved to the to-do catalogue. Every group member was given the freedom to choose a module to work with from the to-do catalogue. After choosing a feature/module the group member put his/her name in the module and moved it from the "to-do" scrum board section to "in progress". After he/she was done with the module he/she moved it from the "in progress" section to "testing". When every test and integration of the module with other modules was successful, the feature was finally marked as completed.

After the feature was completed the team was using collective code ownership. i.e. anyone within the team was given the right to modify the module as far as he/she could enhance its features/there was no strict “ok” requirement from the module owner to modify it. This had its own positive and negative sides. To start with positive side, every module was not under the sole control of a specific group member means, if any group member thought something was missed and wanted to add it, he /she was able to do it easily with no more restrictions. This might have helped us, if any one of our team member was not able to complete the course. From the negative side, when a group member used to modify a specific module, other modules dependent that module was messed up.

Our experience on the project really complements with the literature, for example where it says “during collective code ownership anyone on the team can improve any part of the system at any time. But one of the worst risks is a general degradation due to inconsiderate extensions (“creeping featurism”)”.

Finally, even though the team has tried its best to implement refactoring, such as pulling out common modules to a separate module to avoid duplications here and there, we haven’t followed strict coding standards. Some examples of thins include: some files are named using camelCase, and some using underscores. Indentation is also heavily inconsistent, in some places two spaces were used, but in others 4 space was used. The spacing issue is most likely due to the team was using different text editors. These problems could have been largely combated by defining code and naming conventions at the beginning of the project and by having code reviews during the development cycles. This counteracts with the literature where it comments, “agile methods include the idea that teams should adhere to strict coding standards to help quality”.

# Testing

Considering that there were no complicated computation in the application, testing was mainly usable when the data was being fetched from the database. For this purpose, test cases were

written using Travis where we tested user login and CRUD (Create, Read, Update and Delete functions) for Events, News, Teams and Games as well as rendering tests for the main views of the app.

Additionally, most of the backend code is carried out by the Firebase library where it was not testable. For this reason, functions used in Firebase were mocked to be able to be test. Even though this process took extra time and effort, the team learned how to setup a testing environment for react native, as well as creating mock-ups during the development of the application.

We realised during the project that test driven development is not really a feasible practice for all types of software development, and can in some cases actual harm the development process. In our case, we were trying to really do some test driven development but due to issues with travis and lack of testable code, we never had the chance. The code that we finally tested was already written by the time we were ready to do test driven development, and waiting for travis to be fixed would have stalled our progress a lot, since nobody could work before they had data to display. This realisation is also observed in the course book, where the author ranks test driven development as one of the bad and ugly practices.

## Continuous Integration

One of the advantages of using Git was it allowed us to integrate quickly and easily. We could merge branches, see where conflicts occurred and make amendments where required. We did this regularly as a group (between every few hours to every day we worked on the project). Towards the beginning of the project we started with many development branches, however we were finding the code was too widespread and taking longer to put the project together. By the end of the project we were down to about three branches, allowing work on major projects (such as the Maps view) without distrupting others workflow.

Ultimately, we found continuous integration useful as it allowed us to see, physically how the project was coming together.

## Communication channel

We found Slack was the most important communication channel. We had a smooth communication among our group members through Slack. We had a vocal group, sharing views, questions and notification to other group members via Slack.

Another important communication channel was verbally in person. We had group code sessions in the same room which made communication **much** faster. This eliminated hold ups as we weren't waiting for responses like you would with Slack. Finally, it allowed for more

innovative solutions to problems as there was a lot of input and discussion.

What would you do differently in a future but similar project?

# Project outcome

## Concept of project:

We were given the task to create a software tool for a football cup. This tool would then be used by tourists and fans so they could keep track of things like results, upcoming games, restaurants and shops in the area etc. An admin should be able to manage the content (news, games, results etc.) through the same application.

Our final product is a smartphone application which cover all those areas. It can show news, events, games, a table and a map with all happenings placed. An admin can log in and edit stuff that exist in the app and also add new stuff through an admin view. We have also added functionality for a user to follow teams to see information about them and the players in the squad. There is also a homepage which shows a few games, news and the top teams in the table.

## Frameworks used:

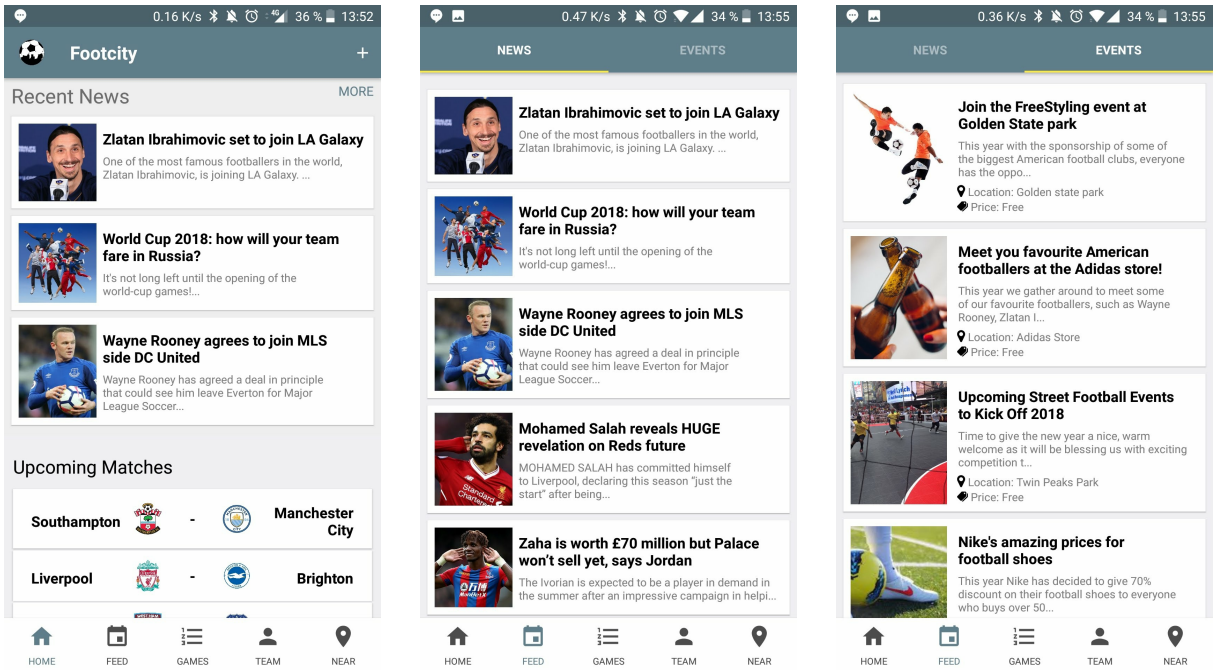
The application is built using React Native, which is a front-end framework for building applications. The framework is an extension from the React framework which was created to be able to develop fast and modern web pages. The framework is built using javascript and allows for writing cross platform code for iOS and Android with very little native modifications and as such may reduce the development time for developing cross platform applications as there is no need to write two separate versions.

The back end of the application is handled via Firebase which is a modern database solution that is easy to handle and scales easily. Further, as the setup is rather simple for Firebase the time nessesary for enviroment setup was reduced, increasing the time available for work on the actual application.

## Application walkthrough:

In this section we will describe the application at it currents stage, discuss the different screens and how the functionality works. We will start off with the main tab navigator and the

screens in it. All data shown in the screenshots are retrieved from a live database that we created.

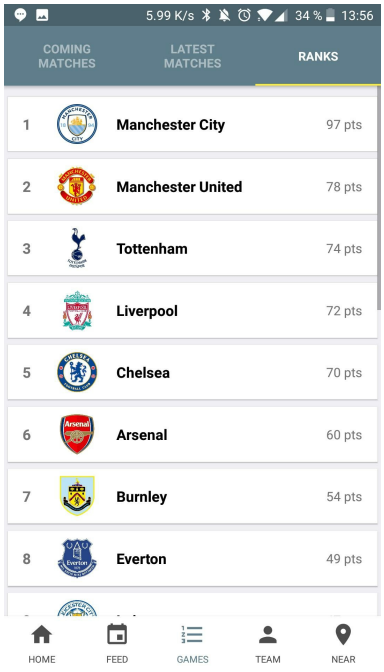


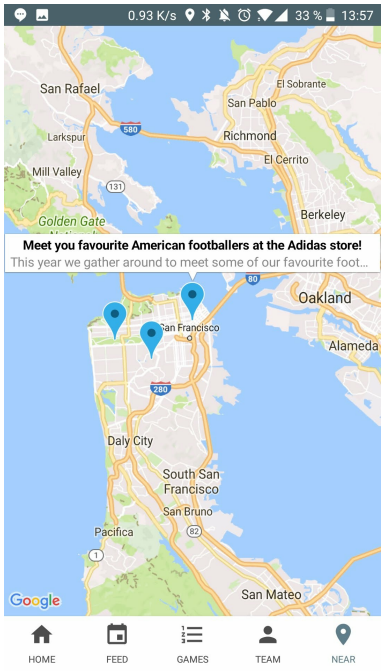

The left image above this text displays the home page. In the tab bar at the bottom the "Home"-tab with the house icon is highlighted with the app colour, which is a dark blue/grey colour. Each view in the app has a header, displaying where in the application you are. Since this is the home screen, the app logo and name is in the header. The plus sign is for admin functionality, where you can login and add things like news, events, etc. The content of the main screen is basically a summary of other tabs. At the top you can see the latest news, after that upcoming matches are shown, followed by the latest matches. At the bottom of the main screen the top 5 teams are shown.

The middle screen displays the latest news, and the screen to the right displays upcoming events. These are both tabs in the tab feed. We initially wanted the events to be dependent on the teams you followed, but they ended up being more general events for everyone. These are all managed by the admin who can create, update and delete events and news. The admin also manages teams, games and players.

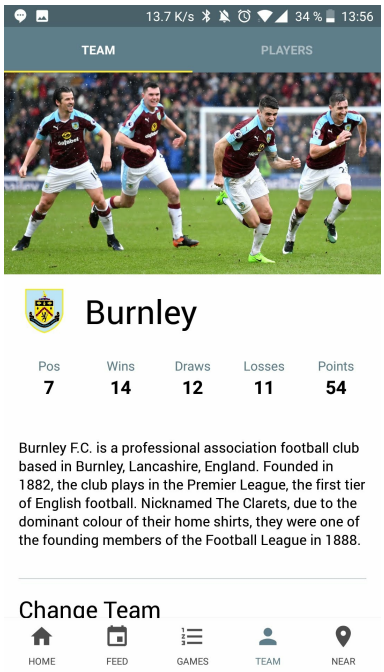
The left screen below displays the latest matches and their scores. This is part of the Games tab, where Upcoming Matches, Latest Matches and Ranks are sub-tabs. The Upcoming matches looks like the latest matches, but since they haven't been played yet they do not have scores. Clicking on any of these games will open a detailed view of the game where you can read a description of the game, the location and date. The middle screen above depicts the ranks that the teams currently has, which is based on the scores the team has accumulated this far.

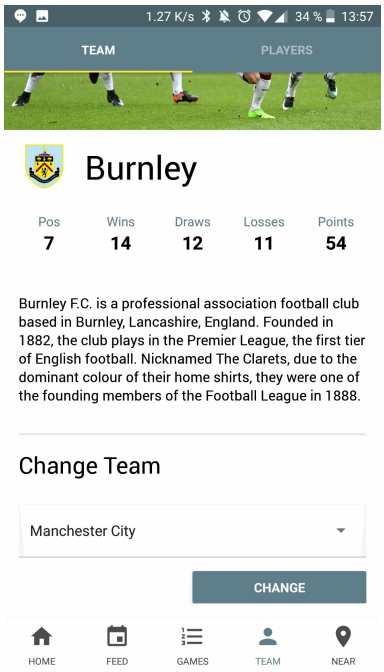
The screen to the right is the maps view, where you can see where your position is and where

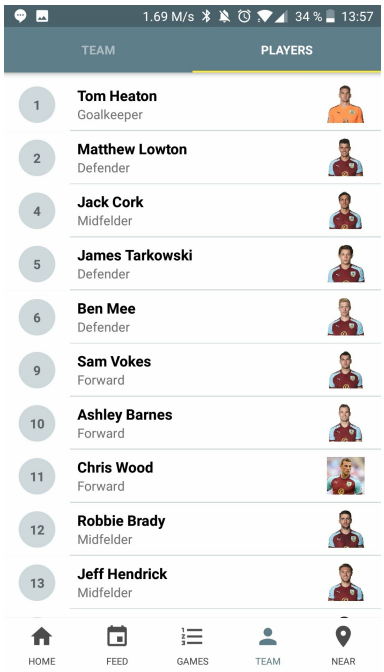



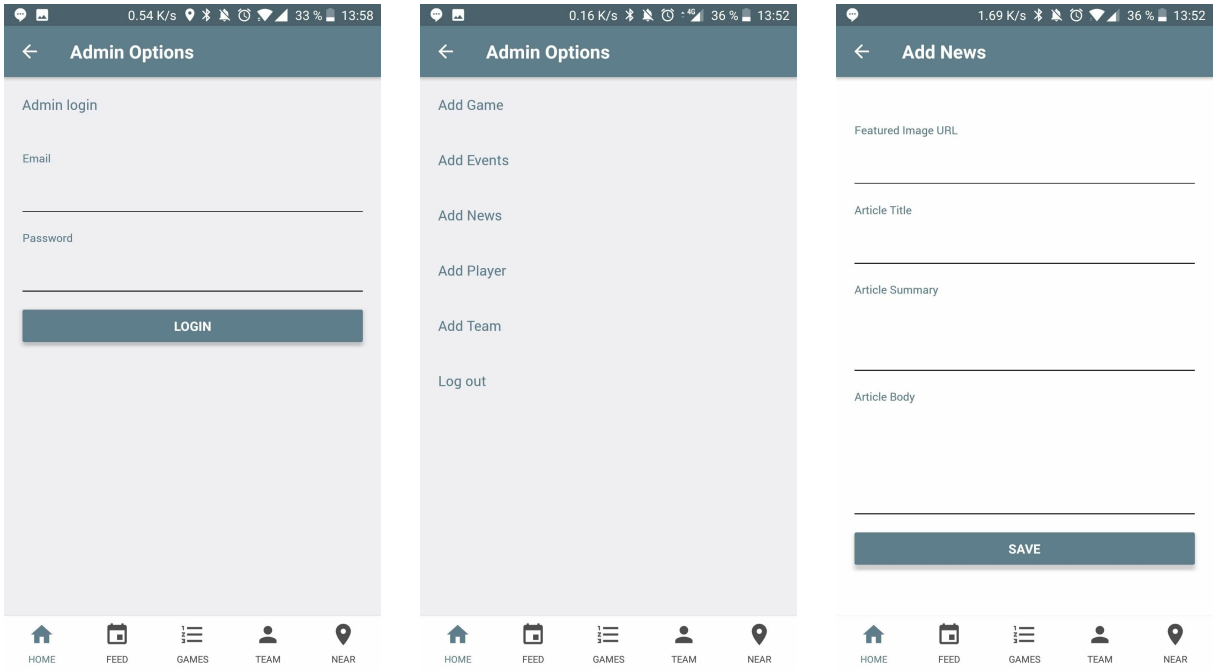

the events positions are. The blue markers are event locations, and they use latitude and longitude information stored in the database for location mapping. We used Google maps for this since it is free and highly documented.



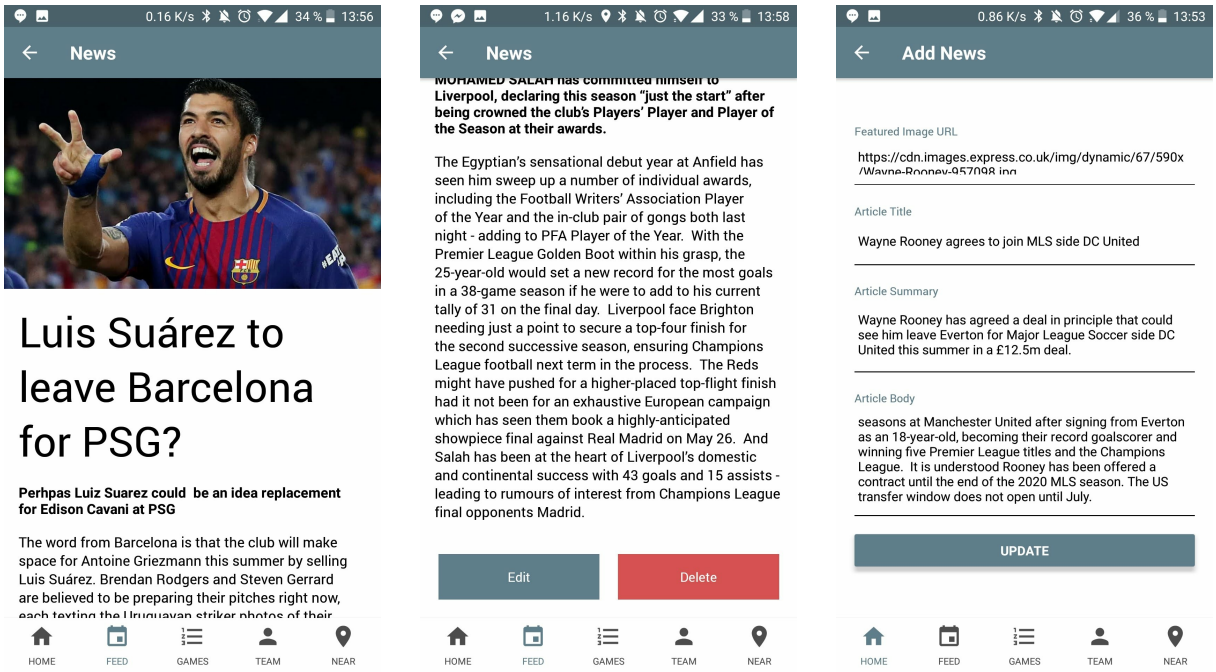




These screens above are all from the teams tab. The left and middle are both from the sub-tab Team, where you can see your team's stats like wins, losses and rank, together with a description of the team. The middle screen is the same tab but scrolled down, and shows the change team selector. If you change team the new team's data will be visible, and you will be marked as supporting that new team instead. The screen to the right display the players in each team, with their name, image, position and number. We have over 400 players added manually to the database.




The last part of the app are admin functions. The screens above show what the admin can do. The left screen is a simple login, which is the first thing you see when clicking on the plus sign on the Home-page. Once the admin successfully logins, they are redirected to the middle screen where they can choose to add something new. The screen to the right show how the add news screen looks like. The other add screens look very similar. The admin only needs to log in once, and then they are kept logged in even if they close the app. They contain all the input for adding a new news article. One issue with the current design is adding images, since you must find an image online and then copy that URL into the app for it to work. We would like to be able to upload images to the database, however this was too complex for this project.




The last unique screens in the app are the detailed views. The left and middle one both depict

a detailed view of a news article. The detail view of events and games look similar. The middle screen shows the bottom of the news detail, and once the admin has logged in they can edit and delete it there. If they press edit, the same screen as before where you add a new news article appears, however this time the data is filled with the current data. The admin can edit any input here and then hit update to update the news article on the database.