Agile Development Processes Project 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.