Universiteit van Amsterdam

Honours project

Score App

Students:

Max Wong 10751041

Alex Khawalid 10634207

Project supervisor:

Chris Schaffner

Contents

Abstract

1 Introduction

Have you ever been to a big tournament where multiple games are simultaneously being played on different fields? Did you find it difficult to keep track of which teams are playing and what the current scores are? Then this app might just be the right app for you.

We have programmed a website running on JavaScript for mobile phones, so users can easily look up which teams are playing and what the current scores are. The site helps the user to keep track of what matches are being played nearby and allows the user to add extra information about matches, such as the current score, location of the match and more.

To program such a website we have used the open-source JavaScript web application framework Meteor. Meteor allows us to easily program and test the app immediately, because whenever the code is saved, the site will automatically update. Furthermore, Meteor produces cross-platform code, so users with both IOS and Android handhelds are able to use the website. Meteor integrates with MongoDB, a cross-platform document-oriënted database, and uses publish-subscribe methods to automatically synchronise data between clients and database. This is the reason why Meteor is called reactive.

This report will provide insights in our work by covering various problems which we have encountered and how we solved them, which design choices we made and the reason why, and what functionality has been implemented and what developments are still possible.

2 Process

2.1 In general

The first meeting with our project supervisor, Chris Schaffner, was on September 3rd 2015 and was about agreements like pro-activity, honesty, coming to meetings on time and keeping track of progress in a log. We decided to meet weekly in his office, so we could actively work on the project, exchange ideas, talk about bugs and solutions and keeping track of what we have done and what still needs to be done.

A git repository was made, where all three of us had access to the project and where the code was saved. We decided that the project should be open source so after we finish working on the project, it can still continue to be developed by other people who wish to do so. We decided that using Github

was our best option as Github is a popular platform where many people will have access to our code.

In the later stages of the project the Meteor webhosting decided our website was using too many resources when syncing, this slowed down all of the functions of the application. These issues made hosting the application on the official meteor webhosting impossible. We switched to Modulus.io, as Modulus.io does not employ a cap on resources. However this is not the only argument for switching, Modulus.io also enables users to keep track of cpu and ram use of our application as well as providing an option to disable the application so no unnecessary sync operations would be performed.

2.2 Lessons Learned

Once the project started, our first task was to get familiar with Meteor. The official Meteor tutorial on meteor.com does not cover all of the 'basics'. Most importantly, it does not come with a built-in system to send users to different webpages. The most common solution to this problem is a routing plug-in which allows developers to use anchor tags for links by defining routes. Using routing plug-ins can be very straight forward, but for purposes where the site and the navigation have to be dynamic, a lot of data has to be passed from the frontend to the backend and vice versa. During the first iteration of the project we tried using iron-router, a package which could be used to define routes, but it was not part of the basic packages included in each app. Once it was added the entire site broke, links stopped working, entire pages did not load and the navigation would not be generated properly. Since a routing plug-in was absolutely required to make the site work and there were numerous problems with it, we started over from scratch, but this time we made use of iron-router from the beginning.

Our project supervisor, Chris Schaffner, gave us a link to a tutorial for meteor which also covered the iron-router basics. In the future it might be beneficial to see what functionality is not included in the basic package and to check if there is a plug-in to get that functionality, because this would have saved us a lot of time.

One of the challenging things about programming with Meteor is that debugging options are limited. There is a debugging mode for Meteor, but it was not very useful to the project as it did not reveal any errors in the code. Setting JavaScript to strict was of no help either. Meteor and JavaScript usually don't give any errors when the code is not written properly, things just break and then the code has to be inspected to see where the error is coming from.

In most cases the easiest way to find the error is to either inspect the code and look for syntax errors or by using the console.log() function to see where the error is. The console.log() function prints whatever is inside the

parentheses to the terminal or console in the web browser.

Missing brackets or semi-colons were the most common errors. The most important thing in the future is to write clean code, so the code does not have to be searched for syntax errors and in case an error does occur always start by using console.log() to see if everything is working as expected.

Another challenge the project faced was cross browser and cross platform compatibility. The site did not look the same on all devices and some functions like pinch zoom had to be disabled for the site to work properly. This proved to be problematic as these differences surfaced later in the project once testing with different devices started. Disabling pinch zoom for android and iOS works differently. Testing it was a little harder since there wasn't always an iOS device available. It would have been easier to implement if testing was done in the design phase with different devices.

The process could have gone smoother, but the essential functions were implemented and at the beginning of the project it is hard to plan ahead for problems you do not even know you might have. The project started with a plan and although the project deviated a few times, having a plan and setting a goal really contributed to the progress of the project.

2.3 Design choices

The current version of the app uses a hamburger menu, which slides in from the left. It is controlled using a button in the top left corner, but it is also possible to use a swipe gesture from left to right to open the menu. We chose this type of menu, because the project is focused on mobile users. Having the navigation take up a large chunk of the top of the screen would be problematic as the buttons would still have to be small and the most important content, like the score and the buttons to adjust the score, would not be available immediately.

The navigation is generated automatically since the content of the navigation is dynamic, for example: tournaments, field names and matches which can change at any time. There are two ways to find a specific game in our system. The first one is to use the navigation on the left and pick the tournament, field and match. The second one is to go to the fieldview and select the field. The first method was made so users can browse games, this way they can pick a game from a specific tournament. This view is also great for people who want to easily see who is playing at what field. The second view is for users who just want to see who is playing at the field they are at, this view will show a list of fields sorted on their proximity to the user if a location is available. These users may already be watching the game, but they might not know which team is which and what the score is. In these cases the second view is really convenient as it gives a list of the games that have been and will be played on the field.

The view for games was designed to display the most important information instantly. The names of the teams, the scores and the buttons to adjust the scores are all available without scrolling. The previous game and next game button are placed left and right respectively for an intuitive interface. The visual cues can be changed by clicking the team names, this was done so it does not get in the way when changing the score. The cues are only adjusted once in a game, so they do not have to be immediately available, while the other buttons are used several times per game. Information about the game is always displayed at the top of the page, this makes browsing the games on a field easier.

Some buttons on the view for games like the deactivate button or the plus and minus score buttons are chosen to be hidden, when a game has been deactivated. This way games are not allowed to change anymore from our website when two weeks have passed. But when two weeks have not yet passed and the game is deactivated, the buttons can come back by activating the game again. This way, when one accidentally deactivates the game or when modifications still have to be made, it is possible to do so without going to the Leaguevine site itself and changing it there.

For visual reference of the interface design see the screenshots section in the appendix.

3 Future Work

In the future a custom function could be added to the admin interface to easily add admin privileges to users. The only way to become admin now, is to launch the website, go to the /admin page and claim to be admin.

At the moment the source of the tournaments, games and fields is LeagueVine, it would be good if it was possible to make this variable so the app could work with different sources as long as a rest API is available. This way, sources which are not used anymore can be removed as source and new sources, such as Ultimate Central can be used.

It would also be beneficial to the app if there were more ways to deal with missing data, so when for example the field is not specified this would not break the entire navigation. At the moment the navigation relies on the presence of all information which includes tournaments, fields and games. If for example the fields are not available the navigation cannot show the games even if they have been added.

4 Conclusion

All in all our project was a success. We have completed the goals which we had set and programmed a website based on JavaScript, which is responsive

and can handle touch events from mobile phones or tablets. Users are able to find locations of matches by turning their GPS on and are allowed to contribute by providing information regarding scores, team colours and whether games are final or not in real time. Admins are allowed to add more tournaments and choose which tournaments to synchronise with Leaguevine. The code is freely available on github for further development and is documented properly for future developers.

A Code

The files score-app.js, score-app.html and score-app.css are included in the project

B Documentation

The documentation is written in a markdown file called readme.md, it is also available on the github project page.

C Screenshots

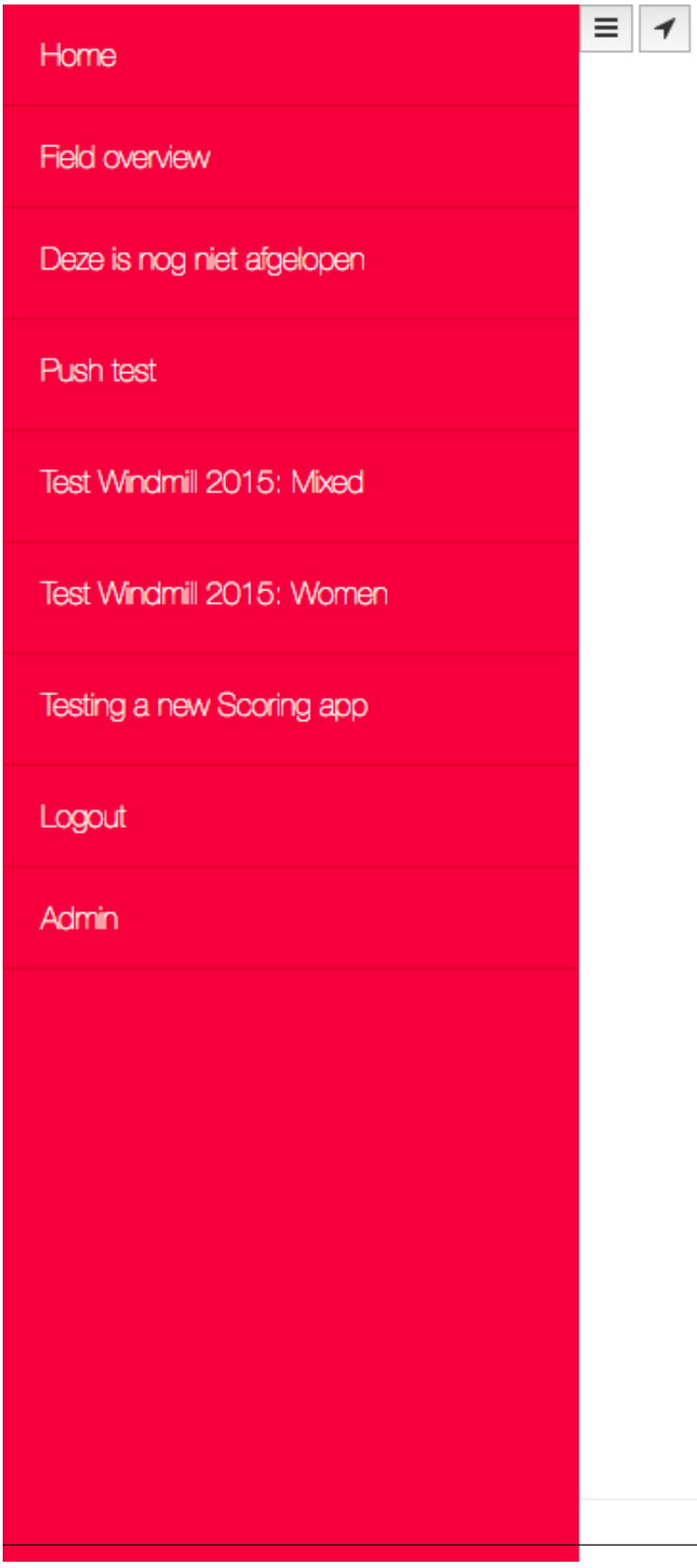

Figure 1: Navigation hamburger menu

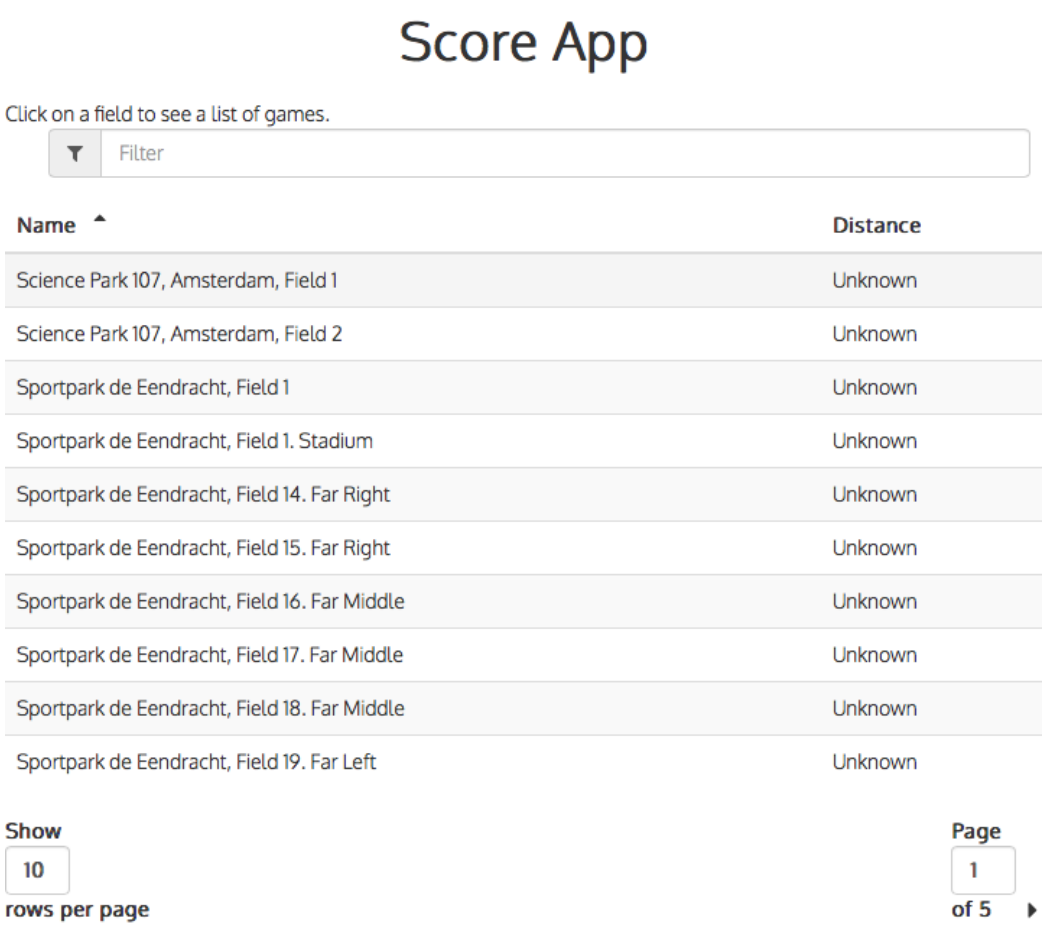

Figure 2: List of available fields

Figure 3: List of games on a specific field

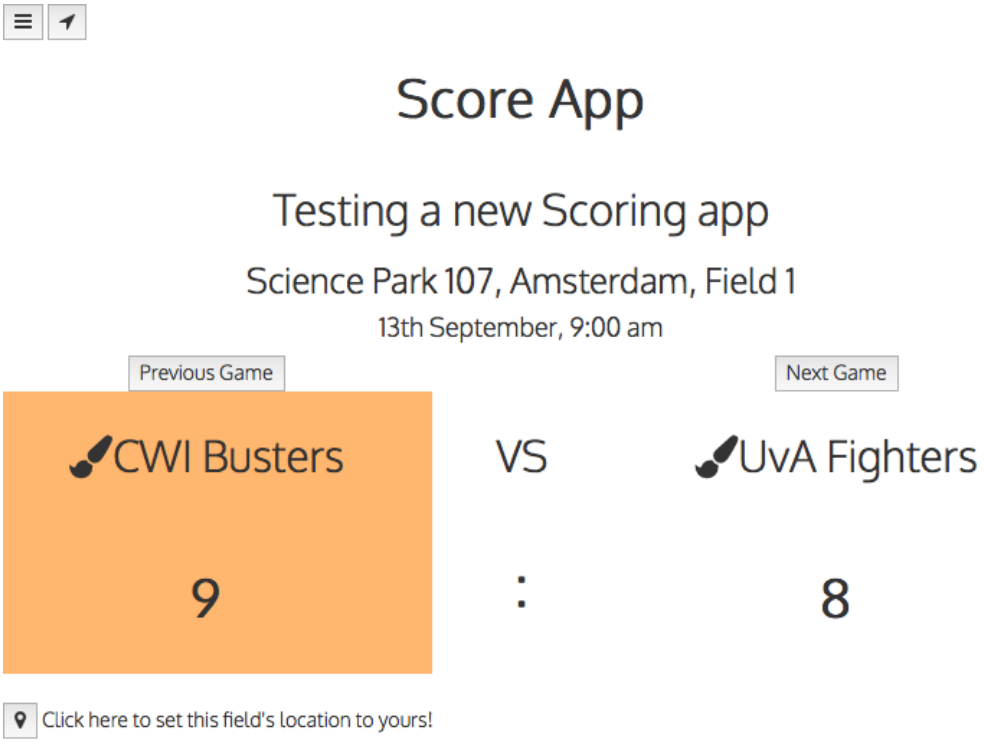

Figure 4: Game view

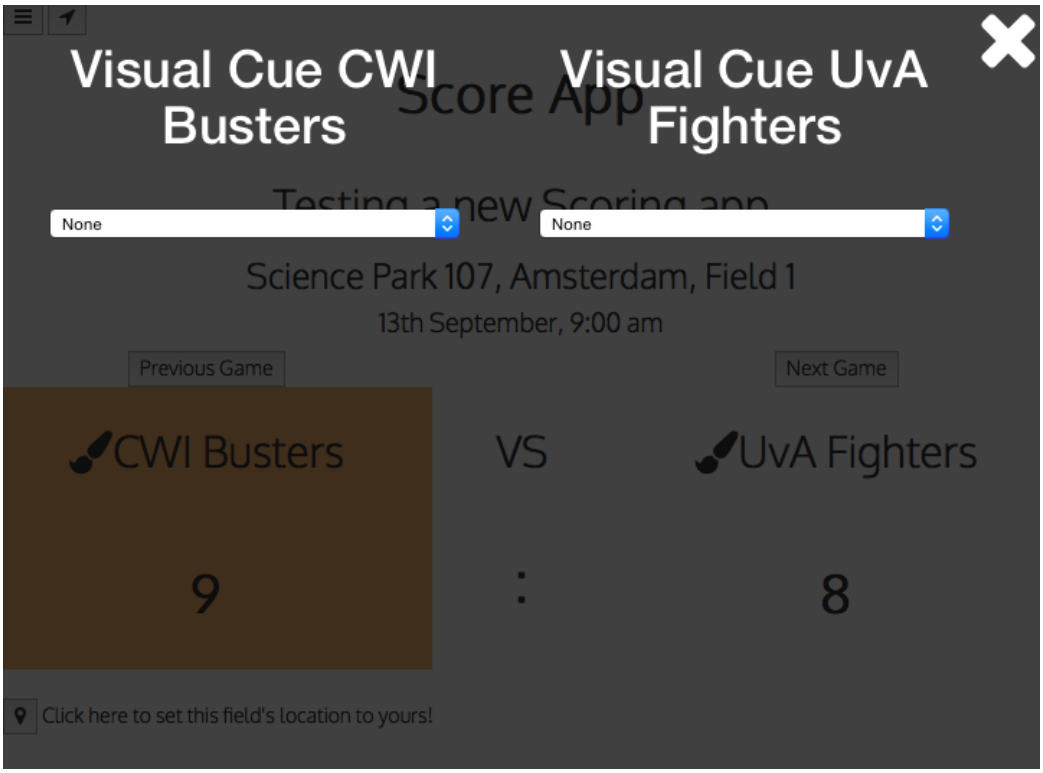

Figure 5: Interface for visual cues