# Technical Architecture

## Use Cases

|  |  |
| --- | --- |
| Title | Menu |
| Primary Actor | User |
| Scope | Category Navigation |
| Level | High Level |
| (Story) | User will see a display of different sections of the Application they can navigate to. These will include all the main sections including the Ranking, Tutorial, Qualifying round, team selection and about section. |

|  |  |
| --- | --- |
| Title | Team Selection |
| Primary Actor | User |
| Scope | Choose team for personal Ranking View |
| Level | Sub-Function |
| (Story) | User can select a team of preference and that will present them with the latest information for that team specifically (Ranking, League position). |

|  |  |
| --- | --- |
| Title | Ranking Update |
| Primary Actor | User |
| Scope | Update Current Rankings |
| Level | Sub-Function |
| (Story) | User can update the current rankings which will display all teams’ positions in the current UEFA Ranking table. |

|  |  |
| --- | --- |
| Title | Tutorial |
| Primary Actor | User |
| Scope | Application Instructions |
| Level | Sub-Function |
| (Story) | User will be given a simple instruction of how they can use the application. |

|  |  |
| --- | --- |
| Title | Qualifying Rounds |
| Primary Actor | User |
| Scope | Qualifying Round Standings |
| Level | Sub-Function |
| (Story) | User can see the current standings of teams for the next Champions League qualifying rounds. |

# Technical Architecture

## Software Components

The software will contain both a backend to perform the mobile services and a front end containing the android app development.

### Back End

The backend is an Azure Mobile Service that uses .net framework. The Azure Mobile Service will be the cloud backend for the Application. It will be hosted through a .Net framework which will include 24x7 monitoring and management. The service can store data in to the Azure SQL and also allow Cloud-based sync. Through the .net framework it will have capabilities implemented to take data off both the UEFA Site and an appropriate site to pull off league tables. This data will then be used for a series of algorithms implemented through the Web API. The final data will then be provided to the front end (Java) for implantation of the actual android application.

The database to be used is Azure SQL. The will take a weekly storage of the current team ranking positions, qualifying rounds and information on positions for each league. This information can then be queried upon but this time to be used in a predictor algorithm for final ranking standings by taking standings from a series of weeks using the database.

Azure SQL includes excellent scalability, performance and security which are vital to the fundamental focus of the Application. It is self-managed so does not require much maintenance.

All functionality for the app will be developed here:

Examples –

1. Team Ranking – Algorithm for calculating the standings of **all** teams who have participated in UEFA competitions the previous five years. UEFA ranking is calculated over 5 years so an option to view the ranking for 2015 will be available. This will involve using the previous 4 years and the ranking points gained by each team this year (2014) to show rankings for 2015.
2. Country Ranking – Similar to the team ranking structure. A display of all countries in Europe by ranking position. Also calculated over 5 years, a similar algorithm will be implemented to calculate future years (2015).
3. Update current Rankings – An algorithm will be used to display the current ranking each week. A predictor algorithm will be introduced that will take a series of weeks of ranking and calculate (predict) what the final ranking tables could look like when all leagues finish.
4. Qualifying Round standings – Algorithms will be created to calculate the current standings in the qualifying rounds. The permutations for this are incredibly complex therefore an in depth level of coding will be required to do this correctly.

### Front End

The front end will be a Java application developed in Android Studio. The Java coded will focus on taking in the final data from calculated algorithms in the API and then produce the different sections of the app with the data in a presentable view for the user.

A personal view will be developed to allow a user to login and choose a team of preference. Java will be coded to allow the users to have easy navigation to sections that will give them information on their team easily without have to manually find the information.

An instruction section will also be created that will present tutorials on how to use the application. Other sections include displays for Country Ranking, Team Ranking and the Qualifying rounds. Each is unique in their own display depending on the complexity of data to be presented to the user.

A section for displaying predicted final qualifying round standings will include some neat features such as colour schemes for new teams that enter each week, teams that move to the seeded bracket for a qualifying round and teams that move to the unseeded bracket.

# Platform Libraries

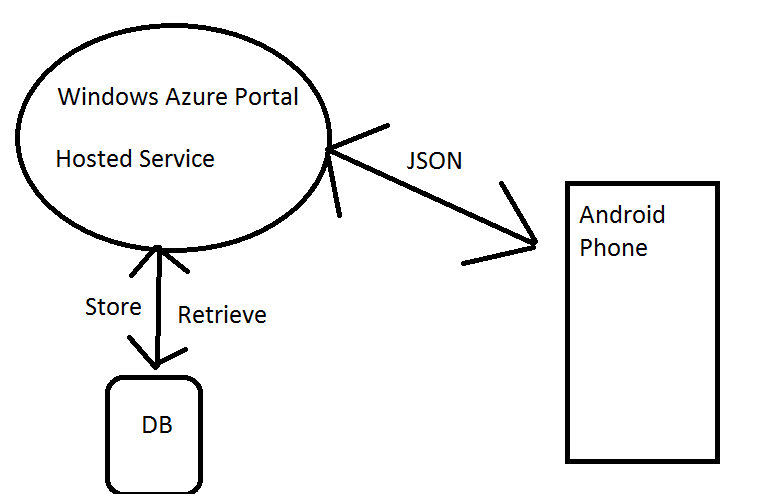
API –

* Web API 2.2
* ASP.Net

Languages –

* Java
* C#

## Distribution and Deployment



The Application will function through the Azure Mobile Services Components.

The Azure will provide User Authentication, Push Notification capabilities, Scripts and Data. It will also connect to the Azure SQL Database for storage and retrieval.

The hosted service will retrieve information for the UEFA Ranking and League stats from the appropriate sites and provide final data for the Front End of Java development through JSON that is calculated through algorithms in the .Net framework.

The Backend is a .Net framework which is coded thoroughly with C#.

## Security

Security will be vital for the Application and a high level of caution will be taken throughout development to prevent future breaches on release.

### Application Layer – Android

1. Storage of sensitive data on the user’s device will be very limited or non-existent. Whether the data is encrypted or not it can still be compromised and therefore essential storage will only be implemented with regard to sensitive data. The majority of data at run time will be stored in the RAM so any sensitive information will be cleared on application close.
2. Caching of the application data specifically the extent of it will be monitored throughout development. Should a registration section be implemented the caching of its URL history and page data for the actual web process could potentially be avoided. The Azure Mobile service has the capability of external login through Facebook/twitter, with a security element integrated for this case it could be beneficial for the user to login this way instead.
3. All query calls in the application must be protected. Should a query call be intercepted an attack could be executed that could breach the whole system. The .Net backend will use POST requests and temporary session keys which are secure for query calls and therefore will provide protection for any possible attacks.

1. Avoidance of crash logs occurring is crucial to the security aspect of the application. Logs provide data to the user and this could prove valuable in figuring out potential vulnerabilities in the coding. To prevent this ever occurring an extensive period of testing will be done throughout development to ensure the application does not crash. The application will be built without any warnings with assertion fails throughout should a potential error occur.
2. A Local Session Timeout will be employed not only for the protection of application data for the developer but also for the user. An easy breach of data includes the actual handset being used by unauthorised personnel and without a timeout this user can see all personalised data of the victim and perform all the permitted features they have when logged in. When the timeout occurs the user will be logged out and their personal team of preference for example will be cleared.
3. Debug logs like crash logs can store data which can be valuable to a user looking for breaches within the system. These debug logs could potentially be disabled for early releases to dilute the amount of information the user will obtain from a debug log.
4. The complexity of the code will be vital for security in the application. If the code is implemented in a simple way with regard to the functions and methods it can be easily understood and therefore altered for a hacker in breach. Reverse Engineering is easy for Android Apps and therefore it is vital the code is obfuscated to prevent malicious users making sense of it should they manage to obtain it.

1. The keyboard cache feature on an Android Operating System could become a security breach with regard to cached information. This is a user dictionary that is available to all applications and for potential sensitive key inputs the auto correct feature could provide an unauthorised user with the full keyword of what a previous user had entered on the App.
2. The simple copy and paste feature is available universally throughout the Android Operating System. For pages that require sensitive data to be entered the copy/paste feature will be disabled to avoid any exposure of potential data.
3. The selection of Third-party libraries will require thorough testing to ensure it can be implemented correctly in the code. Any third party can be used but some can contain vulnerabilities and weaknesses therefore the library selection is vital on the security side of things.
4. A snapshot preventer will be introduced for some pages with sensitive data to ensure they cannot be screenshotted/captured for viewing outside of the App itself. This will stop any easy means of taking the information for use other than on the Application.
5. For any files that need to be created (cache storage) it is essential there are no permissions on those files to be readable or writeable. Despite being stored in the private directory for that App it can still be accessed.
6. The implementation of intents is important and should be done in a careful manner. Starting services for example need to be done implicitly to make sure the right service is responding. Components accessed through intents should preferably be private for security.
7. Checking Activities is essential when implementing an app for security reasons. Despite being private a hacker can find ways around accessing authentication screens through an Activity in an Unlock state. With these potential scenarios in mind input validation is very important in case bad data gets inserted for these situations.
8. A series of Tamper checking will prevent an app being accessed through a back door or resigned by a hacker. A public key check that is read by the app’s certificate can verify the app has been signed with the developer’s private key.
9. Despite deleting files from the Application they can still be retrieved before they have been overwritten. This is important to note and another reason why sensitive data should be limited within local cached files.
10. Logic checks must be evident on the Azure Cloud to validate all potential input from the User. Attacks such as crash logs, buffer overflows can be done so it is vital checks are done to prevent any breaches.
11. Session Management will be done through the .Net framework to enhance security. Hosting a session outside of these session managements is risky and significant expertise would be required to have it done correctly.

(ViaForensics, 2014)

# Prototype

## Prototype Deliverable for week 8

The first prototype will be a basic functioning Android Application. The application will run in conjunction with an Azure Mobile Service and interact with the Azure SQL Database.

There will be a simple menu with hard coded examples of Ranking information. The actual GUI design (screens etc.) will set a very good gauge to how the proposed view will look for Release.

The testing strategy will involve complex data being inserted to the application for calculation and this will test the algorithms to ensure it runs smoothly. JUNIT testing could also be done through Android Studio to test different sets of data being inserted in to the algorithms to ensure efficiency, smooth scalability and correct output.

## Prototype Deliverable for week 11

The second prototype will again provide basic functionality however it will work with JSON data that will be retrieved from appropriate web sources through the Azure Mobile Service. This data will contain ranking and team information that can be passed in to the algorithms for calculation of Rankings.

The testing strategy will involve different JSON data being inserted in to the Java application and in to Algorithms. Various levels of data will be inserted to fully test the capabilities of the implemented algorithms.

# Risks

There are some potential risks that may affect the delivery of the project. When doing the calculation of ranking all of the permutations **must** be implemented correctly and to complete accuracy for the majority of the applications features to function. Should any issues arise with any of the permutations it could take up more time than proposed to solve and become a risk in affecting the delivery of the project.

The integration of security throughout the application is essential for the development. The Application should not be released without the proposed security features being included and running correctly. This could become a risk and affect the delivery of the project.

The GUI design for the Application is important and must be designed to suit a wide range of phones and facilitate their hardware specifications including screen sizes. Should the application be developed with a bad code design it can cause delays due to fixes being required to work with most phones and therefore affect the delivery of the project.

The GET operation must be configured correctly to pull the correct data from the chosen API. With the application being ‘live’ it is essential the correct data is received constantly as the algorithms cannot be calculated without the correct data. Should the data not be received correctly during development it could delay development and becoming a risk in affecting the delivery of the project.

Bibliography

viaForensics, (2014). viaForensics. [online] Available at: https://viaforensics.com/resources/reports/best-practices-ios-android-secure-mobile-development/ [Accessed 16 Oct. 2014].