[**Description**](#_Toc521016063)

[**Intended User**](#_Toc521016064)

[**Features**](#_Toc521016065)

[**User Interface Mocks**](#_Toc521016066)

[News List Screen](#_Toc521016067)

[News Details Screen](#_Toc521016068)

[Fixtures List Screen](#_Toc521016069)

[Screen 3 - Results](#_Toc521016070)

[Screen 4 – Log Table](#_Toc521016071)

[Key Considerations](#_Toc521016072)

[Tool Requirements](#_Toc521016073)

[How will your app handle data persistence?](#_Toc521016074)

[Describe any edge or corner cases in the UX.](#_Toc521016075)

[Describe any libraries you’ll be using and share your reasoning for including them.](#_Toc521016076)

[Describe how you will implement Google Play Services or other external services.](#_Toc521016077)

[Required Tasks](#_Toc521016078)

[Task 1: Project Setup](#_Toc521016079)

[Task 2: Implement UI for Each Activity and Fragment](#_Toc521016080)

[Task 3: Firebase Setup](#_Toc521016081)

[Task 4: Implement the Business Logic](#_Toc521016082)

**GitHub Username**: NalediMadlopha

**Diski**

# **Description**

Diski is a mobile app that provides users with real time updates about anything related to soccer in South Africa. It mainly focuses on the Premier Soccer League (PSL), providing real time news feeds, live score updates, fixtures and results as well as the latest league table standings.

We live in an information age, where people have access to almost any kind of information that is of interest to them. However most of the information is offered with a certain level of delay, e.g. Back in the days people who would have missed a soccer match would have to wait till Monday morning to read about all the weekend soccer action in a newspaper. Now are days, people have televisions (TV), radios and the Internet to keep them posted about the current happenings of the matches taking place. However, they would have to wait for a specific time for a TV or radio programme to find out what transpired in a specific match of their interest. Some may even go online to read about it which is arguably the best approach to find the latest news of one’s interest. The Diski app provides real time information so the user does not have to wait for a specific time to get information or even go online to search for soccer related information. The app provides the user with information to the user as it happens.

# **Intended User**

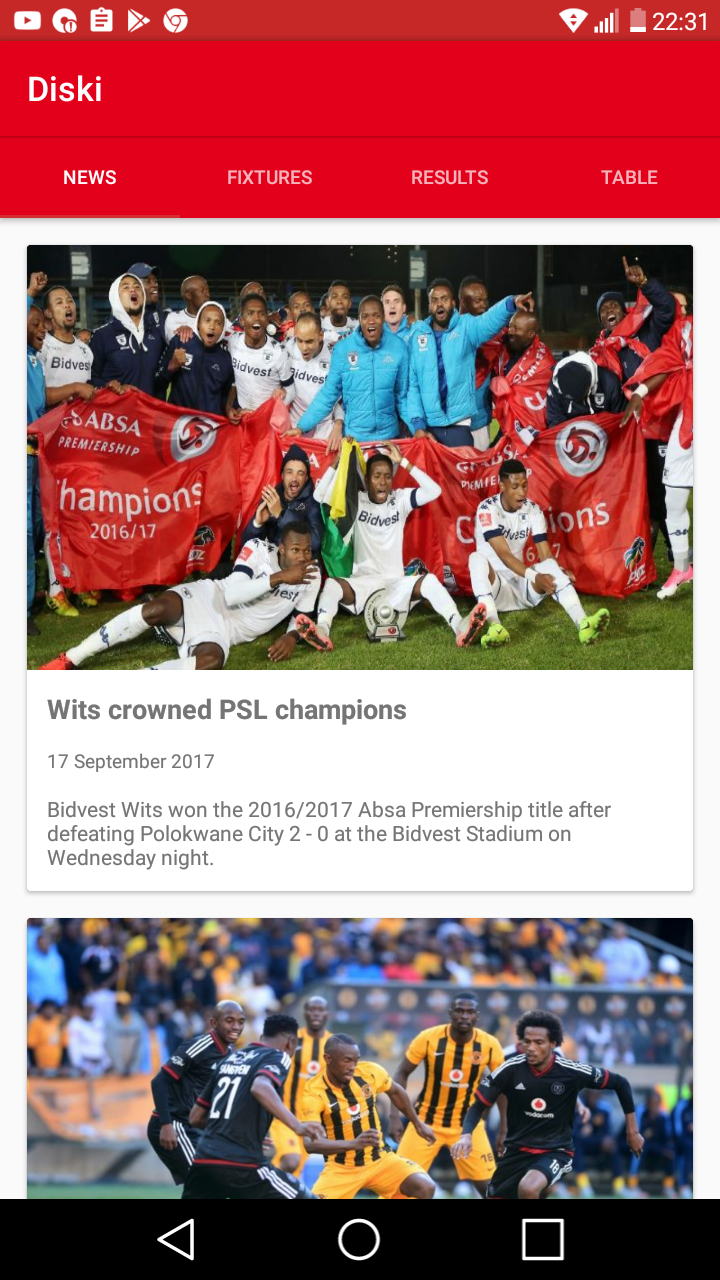
This app is intended for anyone who has interest in South African football and wants to keep themselves up to date with what’s happening in the world of football in this country.

# **Features**

* **News Feeds** **–** the news feeds feature provides soccer news feeds to the user. The user will be able to select an article from a list of articles to display its full content allowing the user to read the story.
* **Match Fixtures –** this feature provides the user will a list of fixtures for different competitions.
* **Match Results –** the match results feature provides the user with a list of match results, display the scores between two teams that we playing against each other.
* **League Table Standing –** displays the log table detailing the games played, wins, loses, draws as well as the total points each team in the league has.

# **User Interface Mocks**

## News List Screen

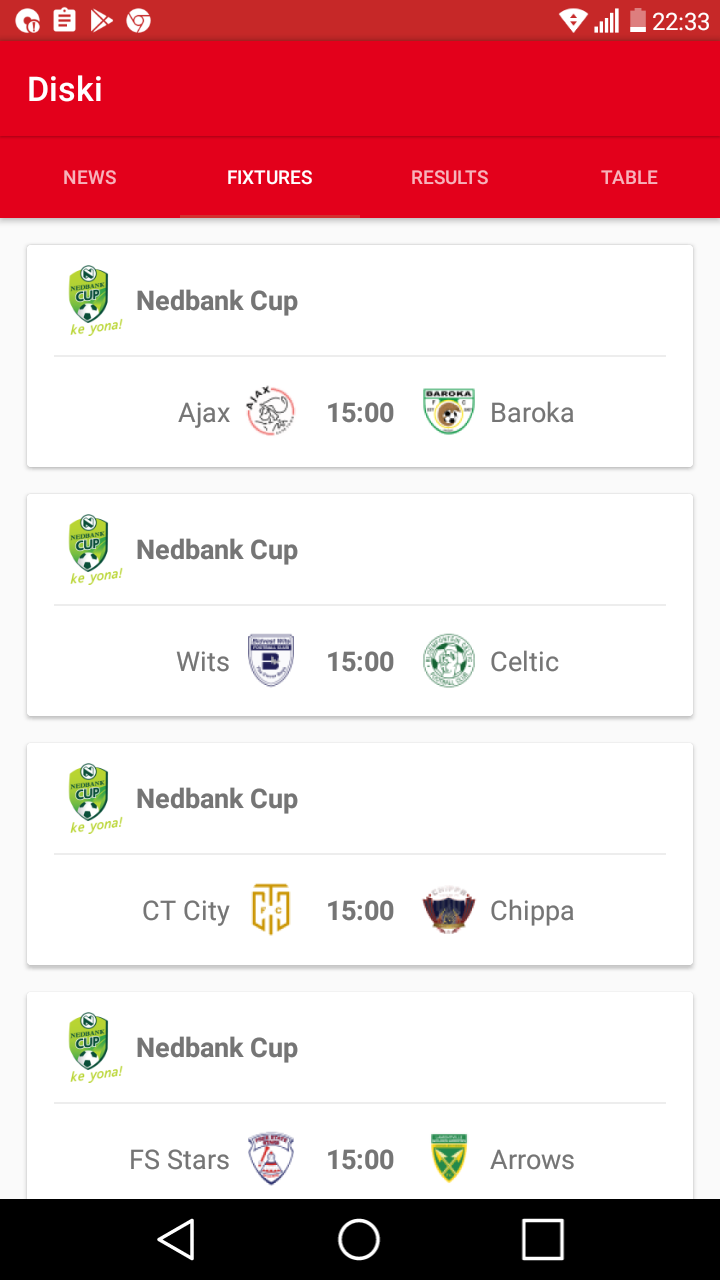


The news list screen is the land screen. When the user launches the app, they will be presented with the screen. The user can then click on the news cards to read the entire story.

## News Details Screen

The news details screen displays the entire story of the article that was selected from the news list screen.

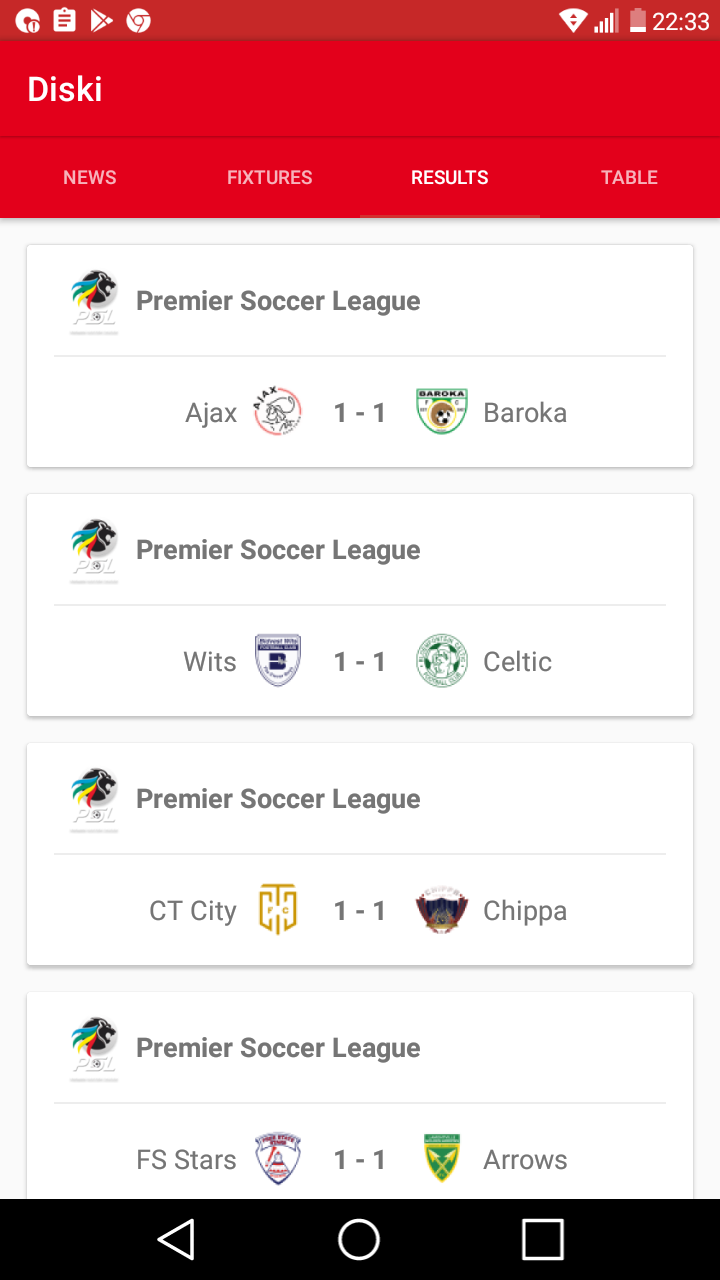
## Fixtures List Screen



The fixture screen displays a list of upcoming fixtures for different tournament. This screen provides the user with the name of the teams playing, the time as well as the competition of fixture.

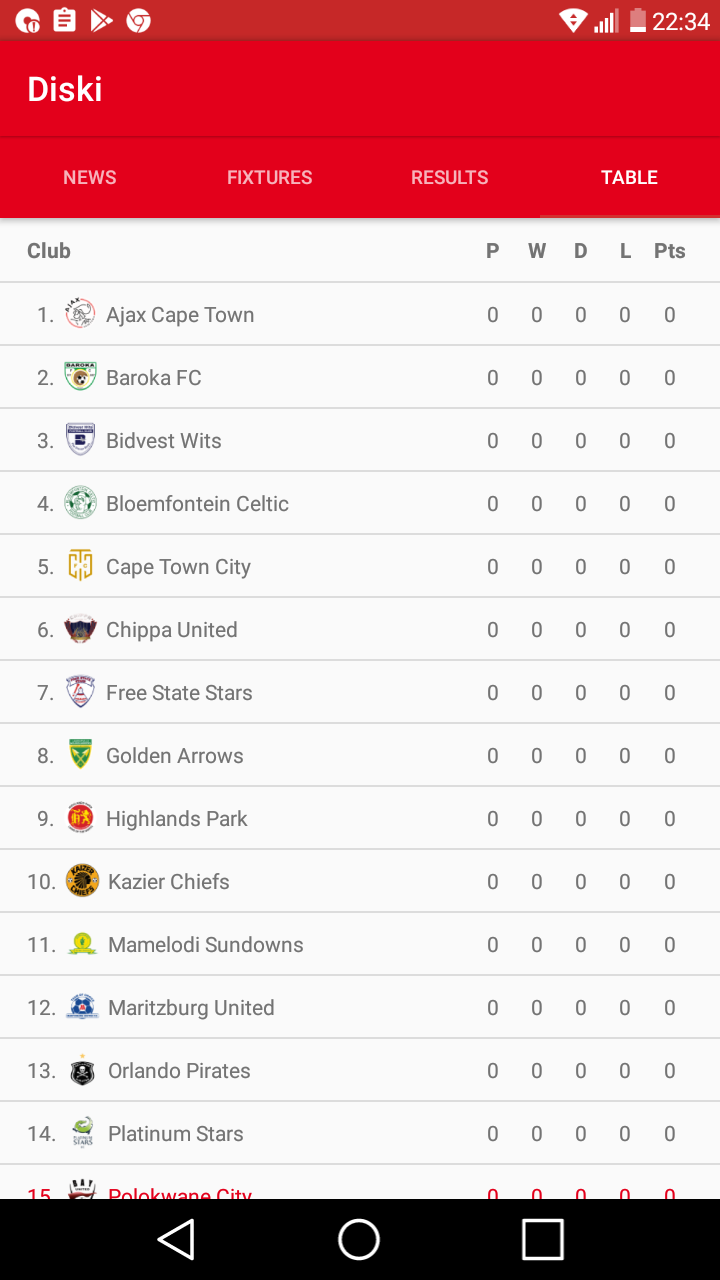
## 

## Results List Screen



The results list screen displays a result list of the most recently played matches. It displays the result score, competition of the match as well as the teams that were playing against each other.

## Log Table Screen



The log table screen displays the log standings of the league, detailing the matches played, wins, loses, draws as well as points for each team.

# 

# Key Considerations

### Tool Requirements

* Android Studio 3.1.3

### How will your app handle data persistence?

The app will be using Firebase Realtime Database to store data remotely. This will allow the app to sync data while online and persist it to disk, making it available offline, even when the user or operating system restarts the app. The app will work as it would online by using the local data stored in the cache.

### Describe any edge or corner cases in the UX.

The app will land the user on the news list screen when it is first launched. The user can will be able to click on certain tab then the app will slide to the screen linked to that tab. E.g. If the user launches the app for the first time then click on the fixtures tab, the app will slide to the fixtures fragment display a list of upcoming fixtures. While on the fixtures screen if the user clicks on the fixtures tab, the app will not change the screen.

The user will also be able to slide left and right to switch between the tabs. If there news screen is displayed, the user will not be able to slide to the left though because this is the first screen of the view pager. Similarly the user will not be able to slide to the right when the table log screen is displayed since this is the last screen of the view pager.

The user will be able to select a news article to read by clicking on a desired news article card on the news list screen. A screen displaying the entire article will be presented to the user the user. If the user hits the back button from this screen they will be taken back to the news list screen.

If the user hits back while any other screen (news list, fixtures list, results & table log) is displayed, the app will be exited.

### Describe any libraries you’ll be using and share your reasoning for including them.

Mockito – to mock out dependencies when unit testing

implementation 'org.mockito:mockito-core:2.18.3'

Butterknife – to annotate fields and cast corresponding view in the layouts

implementation 'com.jakewharton:butterknife:8.8.1'  
annotationProcessor 'com.jakewharton:butterknife-compiler:8.8.1'

Glide – to handle the loading and caching of images

implementation 'com.github.bumptech.glide:glide:4.6.1'

Retrofit – to consume web services

implementation 'com.squareup.retrofit2:converter-gson:2.3.0'  
implementation 'com.squareup.retrofit2:retrofit:2.3.0'

Dagger – for dependency injection

implementation 'com.google.dagger:dagger-android:2.16'  
implementation 'com.google.dagger:dagger-android-support:2.16'  
annotationProcessor 'com.google.dagger:dagger-android-processor:2.16'  
annotationProcessor 'com.google.dagger:dagger-compiler:2.16'

Card View – for displaying the news articles

implementation 'com.android.support:cardview-v7:27.1.1'

Junit – for unit testing

testImplementation 'junit:junit:4.12'

Expresso – for UI testing

androidTestImplementation('com.android.support.test.espresso:espresso-core:3.0.1', {  
 exclude group: 'com.android.support', module: 'support-annotations'  
})

### Describe how you will implement Google Play Services or other external services.

The app will use Firebase for all external services.

For data storage, Firebase Realtime Database will be used to store data such as the clubs, fixtures, results and the log table data.

Firebase Storage will be used to store images that will be used for the news articles as well as the images used for club logos.

Firebase – for real time data storage

com.google.firebase:firebase-database:16.0.1

Firebase – to store the images

com.google.firebase:firebase-storage:16.0.1

Firebase - Analytics

com.google.firebase:firebase-core:16.0.1

Firebase – to report on crashes

com.google.firebase:firebase-crash:16.0.1

Firebase – for push notifications

com.google.firebase:firebase-messaging:17.1.0

# Required Tasks

This is the section where you can take the main features of your app (declared above) and break them down into tangible technical tasks that you can complete one at a time until you have a finished app.

## Task 1: Project Setup

* Configure build variants
* Configure product flavors
* Configure libraries
* Implement the package structure using the MVVM architecture design pattern
* Configure Automation on Jenkins

## Task 2: Implement UI for Each Activity and Fragment

* Build UI for MainActivity
* Build UI for NewsListFragment
* Build UI for NewsDetailsActivity and NewsDetailsActivityFragment
* Build UI for FixtureListFragment
* Build UI for ResultListFragment
* Build UI for LogTableFragment
* Build UI for ErrorScreenActivity
* Build UI for AppWidget

## Task 3: Firebase Setup

* Create Firebase project
* Configure Firebase
* Create JSON file to upload to Firebase Realtime Database

## Task 4: Implement the Business Logic

* Implement logic to perform CRUD operation on the Firebase Realtime Database
* Handle Error Cases (No Internet connection, bad data etc.)

**Submission Instructions**

* After you’ve completed all the sections, download this document as a PDF [ File → Download as PDF ]
  + Make sure the PDF is named “**Capstone\_Stage1.pdf**”
* Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

* Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
* Add this document to your repo. Make sure it’s named “**Capstone\_Stage1.pdf**”