[Description](#_Toc520923665)

[Intended User](#_Toc520923666)

[Features](#_Toc520923667)

[User Interface Mocks](#_Toc520923668)

[Screen 1 - News](#_Toc520923669)

[Screen 2 - Fixtures](#_Toc520923670)

[Screen 3 - Results](#_Toc520923671)

[Screen 4 – Log Table](#_Toc520923672)

[Key Considerations](#_Toc520923673)

[How will your app handle data persistence?](#_Toc520923674)

[Describe any edge or corner cases in the UX.](#_Toc520923675)

[Describe any libraries you’ll be using and share your reasoning for including them.](#_Toc520923676)

[Describe how you will implement Google Play Services or other external services.](#_Toc520923677)

[Next Steps: Required Tasks](#_Toc520923678)

[Task 1: Project Setup](#_Toc520923679)

[Task 2: Implement UI for Each Activity and Fragment](#_Toc520923680)

[Task 3: Firebase Setup](#_Toc520923681)

[Task 4: Implement the Business Logic](#_Toc520923682)

**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.

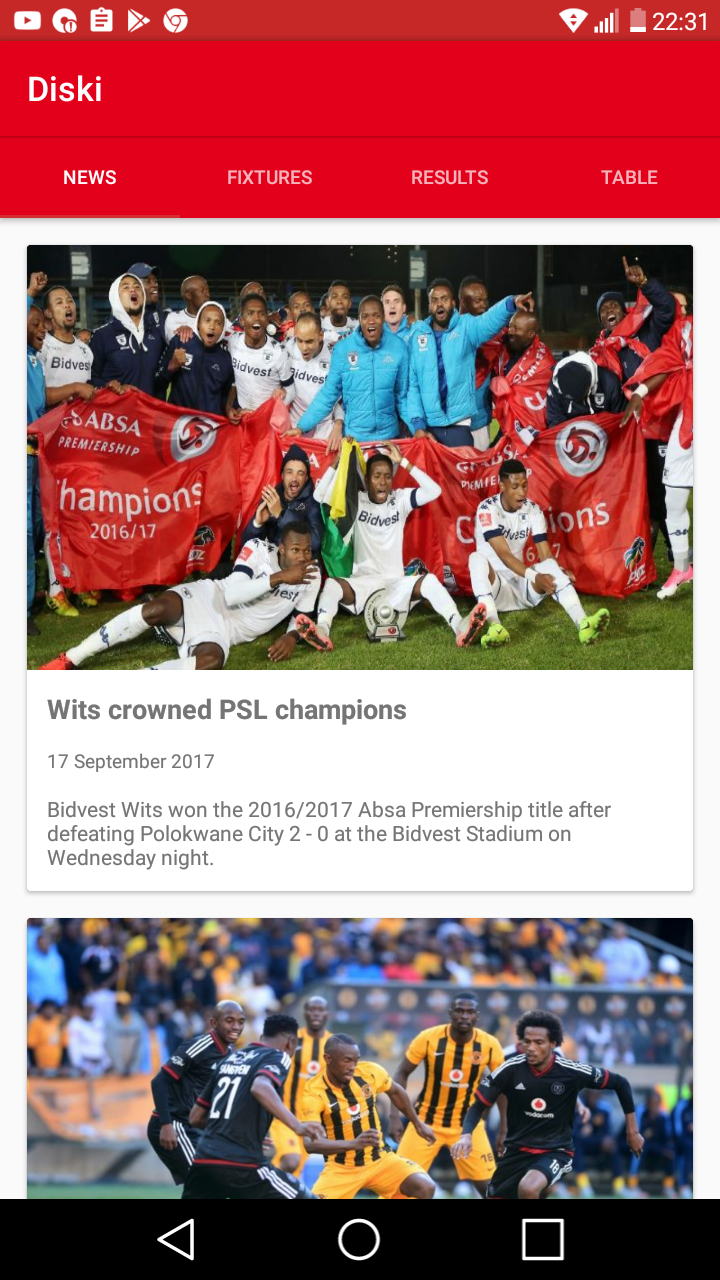
.

# Features

* News Feeds
* Match Fixtures
* Match Results
* League Table Standing

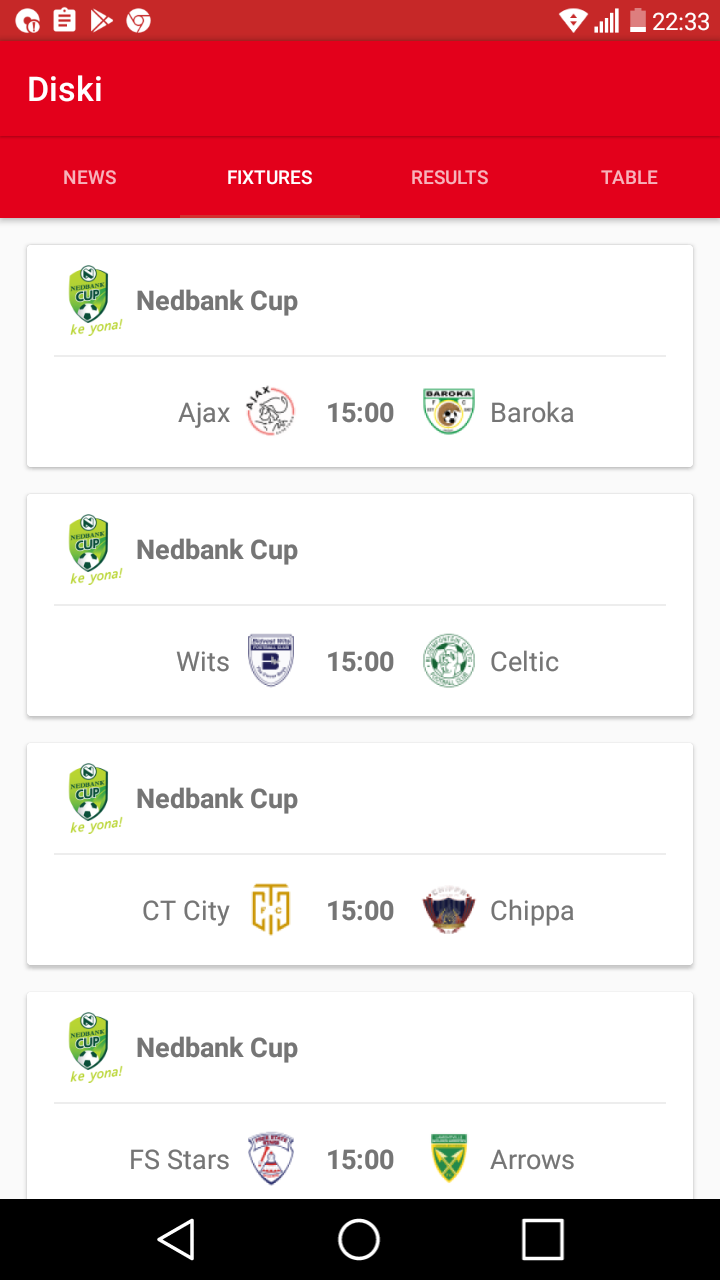
# User Interface Mocks

## Screen 1 - News



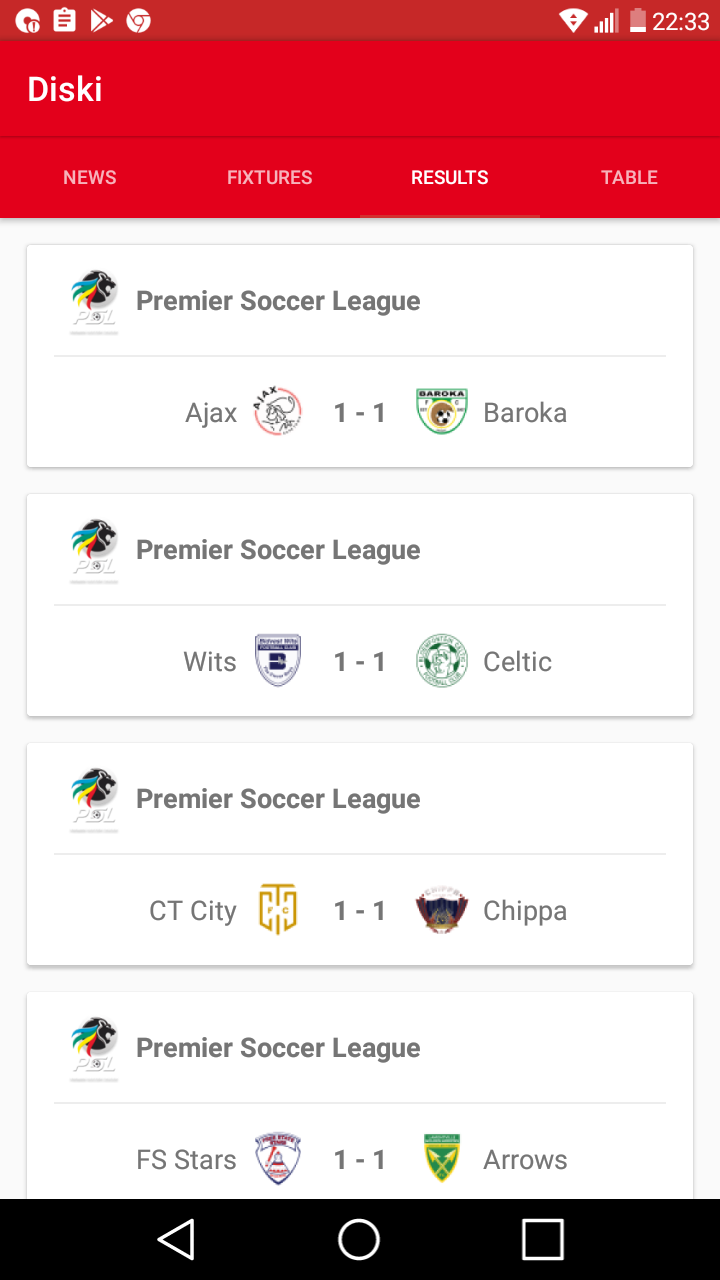
The news list screen is the land screen. When the user launches the app, they will be presented with the screen. The user can then click on the news cards to read the entire story.

## Screen 2 - Fixtures



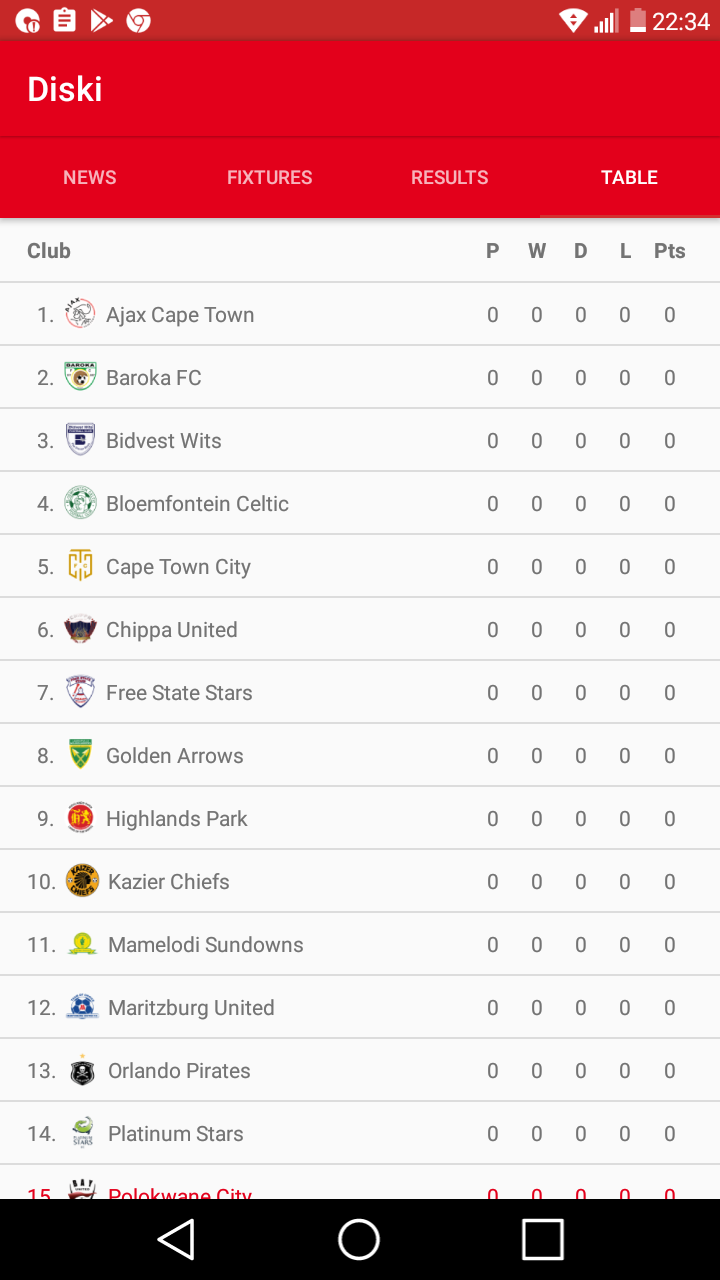
The fixture screen displays a list of upcoming fixtures for different tournament. This screen provides the user with the name of the teams playing, the time as well as the competition of fixture.

## Screen 3 - Results



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.

## Screen 4 – Log Table



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.

# Key Considerations

### 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

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

Glide – to handle the loading and caching of images

Retrofit – to consume web services

Dagger – for dependency injection

### 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.

# Next Steps: 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**”