How to Use this Template

1. Create a new document, and copy and paste the text from this template into your new document [Select All → Copy → Paste into new document]
2. Name your document file: "**Capstone_Stage1**"
3. Replace the text in green

GitHub Username: geekanamika

Sportify

Common requirements:

1. App keeps all strings in a strings.xml file and enables RTL layout switching on all layouts.
2. App uses accessibility by: Content descriptions, navigation using a D-pad.
3. Java language will be used for development.

Description

Write a brief summary of what your app does. What problem does your app solve?

Sportify is an Android app that allows users to read their favorite sports news headlines everyday and share it with their friends easily. This app is intended for anyone who is a sports lover and wants to keep himself abreast with the latest sports updates happenings in the world. Sportify allows you to check details, last updated, favourites, share & add widgets on the screen for last updated news headlines.

Intended User

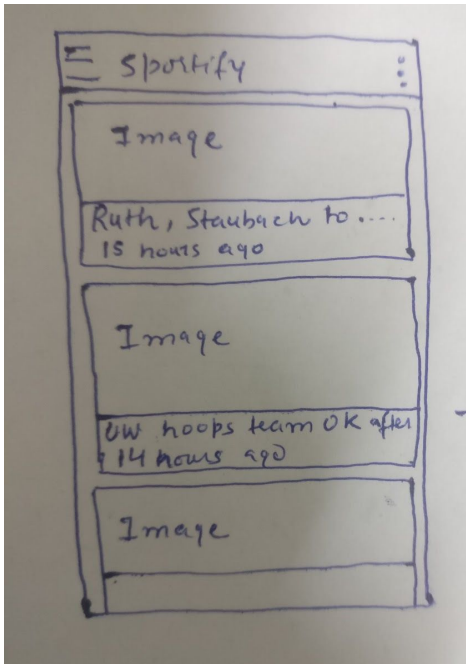
Who is your intended user? (For example, is this an app for dog owners? Families? Students? Travelers?)
Sportify is for all the people who are crazy about the Sports and are very consistent Checking the news updates. It will help them to check news headlines from different news channels at one place.

Features

- Read latest updated news.
- Ability to choose different sources for sports news.
- Favorite/Bookmark news headlines that you like so that you can read them later even if user is offline
- User friendly Material Design UX
- Giving periodic notifications(time selected by user) about the news headlines

User Interface Mocks

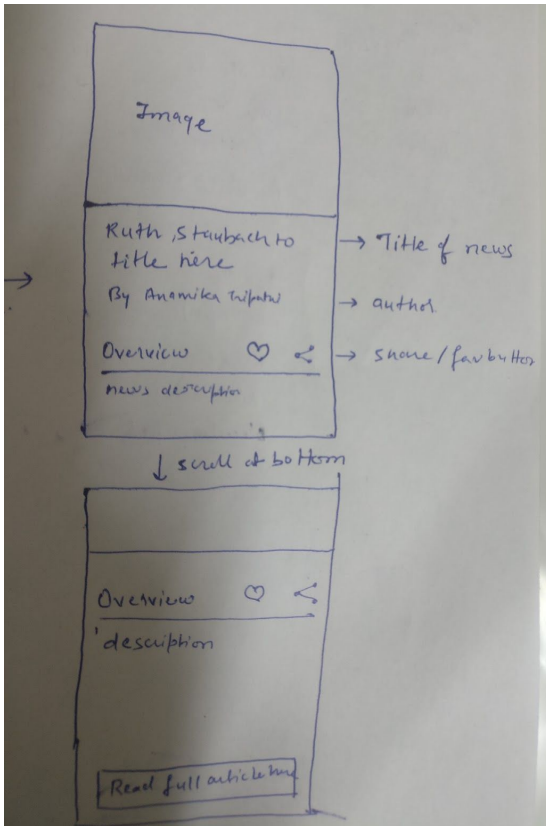
Screen 1



The screen shows the populated Sports news from APIs in recyclerview.
Card details:

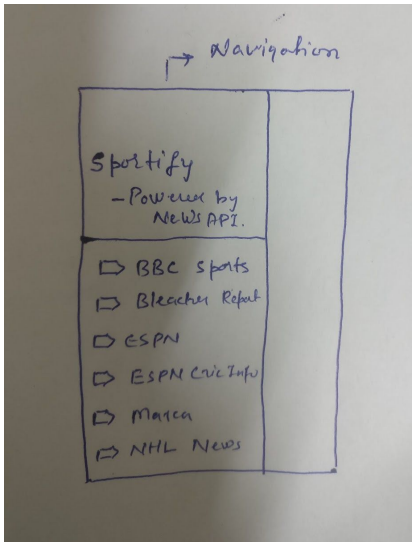
- Image from api
- News headline
- Updated time

Screen 2



Detail screen:

- It shows image related to news from API.
- News title, author, & description.
- Heart icon lets you add news as favourite so that user can see it offline too later.
- Share button can help in sharing the news.
- At the bottom, user can click on “Read full article to read article on web”.

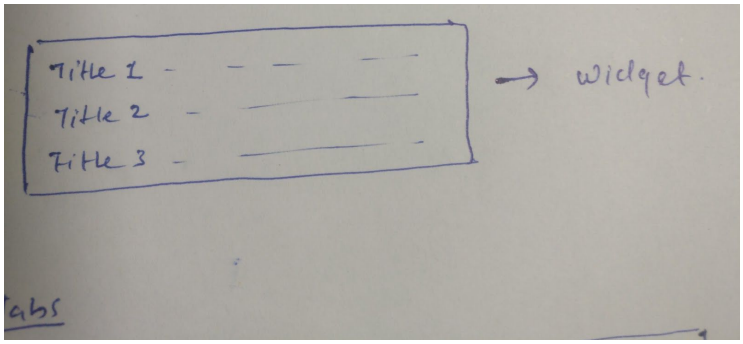


Navigation View/Bar

It allows the user to check headlines from selected news channels.
If ESPN is selected, it'll display ESPN headlines in home screen.
User can also select default home screen from a favourite news channel in Settings.

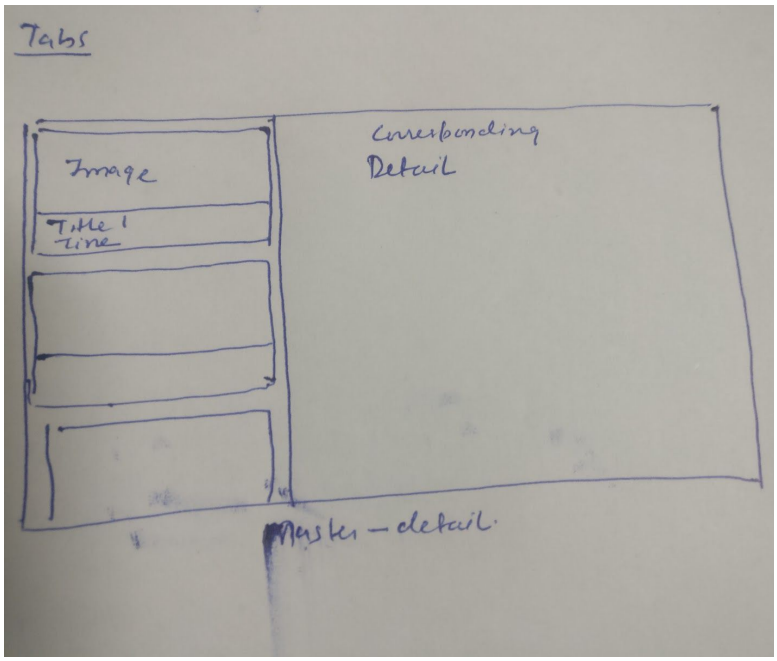
Favourite

This screen displays all the favourite news marked by User.
User can click on any card & can see detail screen.
It is also available offline.
If no favourite is available, It will display a snackbar with message that No favourite news found.



Widgets

It displays latest three headlines fetch over the network.
If user has marked any particular news channel favourite in settings, it'll display only favourite channel's latest three headlines.



Tab View

Master Detail view for tablet screen
Left pane: It contains all the cards with news headlines, image & updated time.
Right pane: It displays the detail screen.

Key Considerations

How will your app handle data persistence?

The data will be stored in SQLite Database using Room Library to store favourite news marked by User. App will use executors & Livedata for easy access to database.

Describe any edge or corner cases in the UX.

For example, how does the user return to a Now Playing screen in a media player if they hit the back button?

- If no favourite news is available, it shows a snackbar displaying No favourite news found.
- If no internet connection is available, It shows a snackbar displayed, No internet connection.
- If a channel is marked as favourite in Settings, It'll be set as default in home screen.

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

App uses Android studio 3.2.1
Gradle details:

- Gradle build tool: classpath 'com.android.tools.build:gradle:3.2.1'
- distributionUrl=https\://services.gradle.org/distributions/gradle-4.6-all.zip
  // Sdk and tools
- minSdkVersion = 16
- targetSdkVersion = 28
- compileSdkVersion = 28
- buildToolsVersion = '28.0.3'
  // App dependencies
- supportLibraryVersion = '28.0.0'
- viewPagerIndicatorVersion = '2.4.1@aar'
- constraintLayoutVersion= '1.1.3'
- lifecycleVersion = '1.1.1'
- roomVersion = '1.1.1'
- picassoVersion = '2.5.2' (Over displaying images over network & caching it)
- timberVersion = '4.7.1' (For logging)
- gsonVersion = '2.8.4' (Serialization and Deserialization POJOs.)
- butterKnifeVersion = '8.8.1' (As a view injector for avoiding findViewById.)
- retrofitGsonVersion = '2.1.0' (For API calls)
- FirebaseJobDispatcher: For setting job periodically to sync data over network
- Google play services

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

- Google Admobs: To display ads in banner form at the bottom of the application
- FirebaseJob Dispatcher to update application for news

- Firebase Dynamic link for sharing news data effieciently

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

- Set up build.gradle file for libraries needed.
- Create Util files like Constants, Executors for Room, etc
- SetUp app's theme by saving all colors, creating styles, etc
- Create Toolbar to use in application following app's theme

Task 2: Set Up data directory

- Create pref directory for saving preferences
    - Create pref's interface which contains all methods
    - Pref's implementation
- Create entities for db
- SetUp Retrofit for fetching data & parse using GSon
- SetUp Local db using Room, use livedata
- SetUp repository class which is single source of data for rest of the application

Task 3: Create MainActivity, mainactivity fragment, navigation bar

- Create main activity, fragment layout, item layout for recyclerView
- Set up navigation bar
- Fetch data from remote using data repo & display in list

Task 4: Create Detail Activity, detail fragment & it's related implementation

- Create layout for fragment and activity
- Display data passed using parcelable
- Set Up favourite button implementation
- Set up share button button implementation

Task 5: Create App supported for larger screen

Use above fragments to support large screen using master detail flow.

Task 5: Set Up Admobs

- Create a banner admob in the bottom for home screen
- Create interstitial admob when user click favourite button & then displayed it as marked favourite

Task 6: Schedule job to fetch data every 6 hours

- Fetch data from API every six hours using firebase job dispatcher & cache it.

Task 7: Create setting screen

- Save user's favourite channels for their preferences updates for home screen
- User will get widget update according to saved preferences
- Save notification preferences

Task 8: Create widget & display latest 3 headlines

- Widget will be updated every-time new data is sync from remote.
- Widget displays can be configured to user's favourite channel using settings

Task 9: Give user notification update (optional)

- For preferred channels once in a day (default)
- User can set notification timing in setting screen

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**"