**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**: tryn2hard

# Pocket Tally

## Description

Pocket Tally is an easy intuitive way to keep track of your cricket game score. Best part is it's free!

## Intended User

Avid darts players, bar-goers, and everyday people who want to have a good time playing darts.

# Features

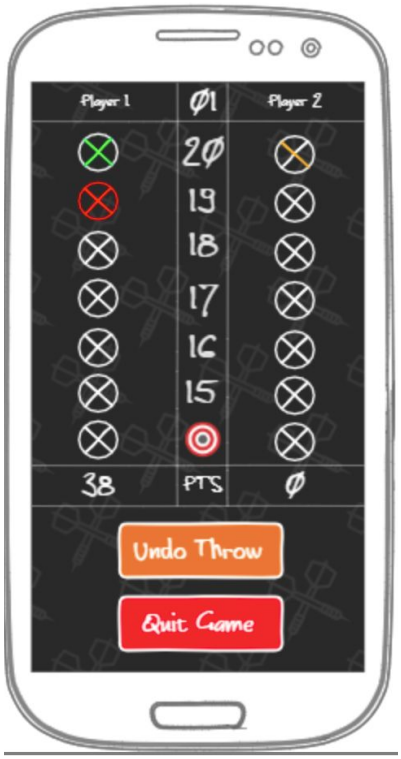
- 2 to 4 players in a game
- Standard or Cut-Throat game type
- Control the number of rounds in a game
- Create an online profile to keep track of game history
- Regularly update game history to online database

# User Interface Mocks

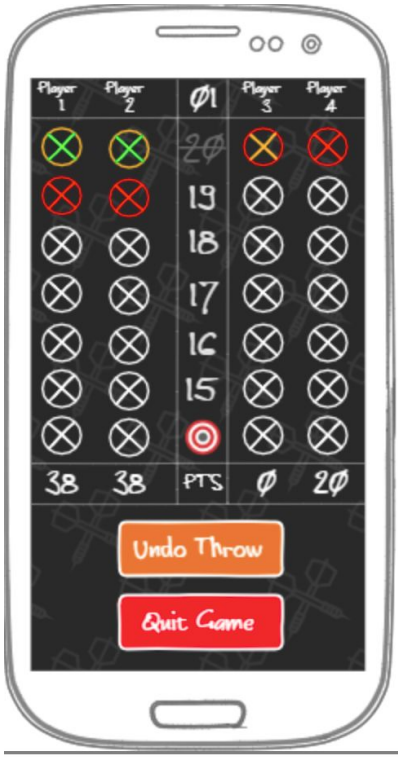
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

### Screen 1

## Screen 2




## Screen 3



## Screen 4


## Screen 5

### Screen 6


### Widget


## Key Considerations

#### How will your app handle data persistence?

The app will use preferences to store the user's settings. For each game of darts, the app will use Room to store the game data until the game is finished or ended.

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

In the case of the user leaving a current game, View Model will be used to make sure that the user can return to the game without the loss of any progress.

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

Android Studio version 3.1

Gradle version 4.7

Room -- to store the relevant game data

```
implementation "android.arch.persistence.room:runtime:$room_version"
```

ButterKnife -- to bindviews

```
implementation 'com.jakewharton:butterknife:8.8.1'
```

ViewModel -- to help with data persistence

```
implementation "android.arch.lifecycle:viewmodel:$lifecycle_version"
```

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

Google Mobile Ads -- to have ads

```
Com.google.android.gms:play-services-ads:15.0.1
```

Google Firebase Realtime Database -- to store game history

```
implementation 'com.google.firebase:firebase-database:16.0.1'
```

Google Firebase Job Dispatcher -- to schedule regular updates to the Firebase Realtime Database

```
implementation 'com.firebase:firebase-jobdispatcher:0.8.5'
```

# 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

- Create a github repo for the project and perform an init commit from AS
- Add all the necessary dependencies that will be used
- Configure the database
- Develop all logic for app in Java

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

- Build UI for the Starting Activity
    - Starting Activity will host three different fragments(Start Screen, Settings Screen, and How-to-Play)
- Build UI for the Dart Game Activity
    - Use fragments to change the number of players based on the user's settings

## Task 3: Connect the database to the Activity

- Use ViewModel and loaders to read/write to the database

## Task 4: Implement Google Mobile Ads

- Add an ad display to the Dart Game

## Task 5: Create a build variant for free and paid

- Create layout to remove the adds from the Dart Game

## Task 6: Support Accessibility

- Use vibrant colors to visually inform the user of any changes when marking dart tallies
- Leave enough space for textviews to be adjusted with larger text

## Task 7: Support RTL

- Add RTL to the manifest
- Define padding in dimens for RTL using paddingStart and paddingEnd
- Adjust player names and setting selection spinners for RTL

## Task 8: Use Firebase Realtime Database

- Store game history in Firebase Realtime Database (FRD)
- Schedule updates to the FRD of the user's game stats by using Firebase Jobdispatcher
- Request game history from FRD to generate user game stats to view in the widget
- Make users authenticate to access their personal game history data

## Task 9: Add widget to display user game data

- Program the widget to display the user's game stats from the Firebase Realtime Database
- When a user clicks on the widget; the app should open to a current game

---

**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**"