Description
Intended User
Features
User Interface Mocks
    Screen 1
    Screen 2
Key Considerations
        How will your app handle data persistence?
        Describe any corner cases in the UX.
        Describe any libraries you'll be using and share your reasoning for including them.
        Describe how you will implement Google Play Services.
Next Steps: Required Tasks
    Task 1: Project Setup
    Task 2: Implement UI for Each Activity and Fragment
    Task 3: Your Next Task
    Task 4: Your Next Task
    Task 5: Your Next Task

GitHub Username: kildares

FutMatcher

Description

Organizing and maintaining recreational football games is a very difficult task, like any social event. This is a sport which requires a significant number of people, generally at least 10 for an indoor match. If the minimum number of people do not show up, the match may not happen. On the other hand, if too many people show up, the game could be too boring. Also, previous to the start of the game, people generally take too much time picking up the teams.

This app aims to help organizers in managing matches. With this app, they will be able to register football matches, control the number of players which will show up and pick up teams quickly by randomly sorting them, based on preferred positions.

Intended User

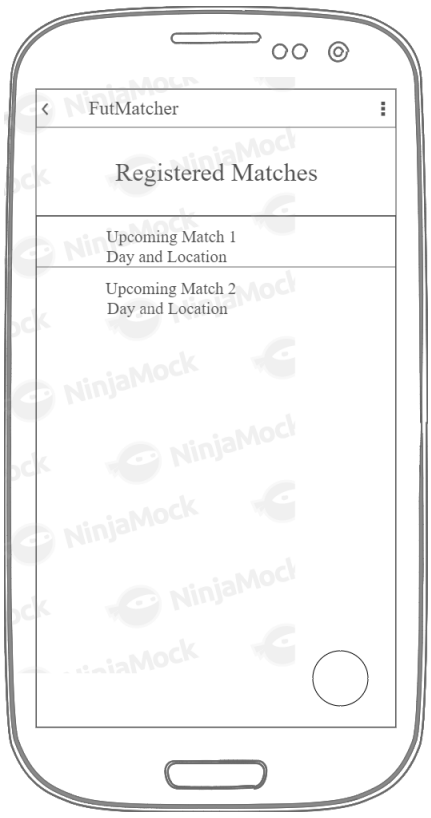
This app is intended for football players attempting to organize football matches.

Features

List the main features of your app. For example:

- Register a football match, saving date and location, minimum and maximum allowed people.
- Choosing a player's preferred position in the game
- Notify user of an upcoming football match
- Pick teams randomly or based in previously appointed preferred position

Screen 1



The main activity of the app will contain a RecyclerView with a list of registered football matches. When an item in the list is clicked, the app will open an activity with the match details (screen 2).

This activity will also contain a FAB which prompts to create a new match. When clicked, will go to screen 3.

Screen 2

This screen displays the match details and the allows players to be registered through a button “Add Player”. When pressed, it shows an Alert Dialog to prompt the user to select the player name and it’s preferred position (screen 4).
After registration, the players will be listed in a RecyclerView containing the inputted data. It is possible to edit a player data by selecting an item of the RecyclerView. The same AlertDialog will be prompted.
If the Pick Teams option is selected, the app will go to screen 5.

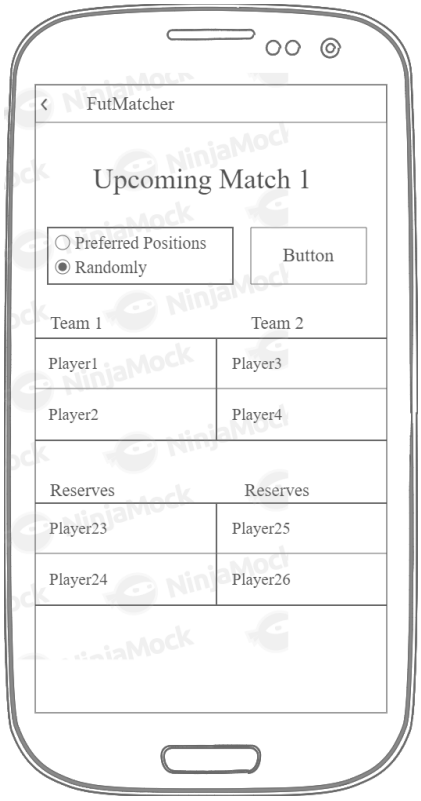
Screen 3

This screen allows the creation of a match. The user must provide a match title, a location and date. The user must provide the number of players per team. The allowed numbers are: 11, 7 and 5.
The app calculates the minimum number of players allowed based on the number of players provided, but the user may insert a bigger value.
The maximum number of players cannot be smaller then the minimum number of players.
After clicking the Create option, the match will be registered.

Screen 4

AlertDialog shown when registering a player. The player name is a textInput. The Preferred Position Spinner is used to select the player's preferred position in the game. The accepted values are: Goal, Defense, Left, Right, Midfield and Attack.

Screen 5



In the PickTeam screen the user can divide the teams randomly or based on the players preferred position. The remaining players will be listed as reserves.

Key Considerations

How will your app handle data persistence?

For this app I will create a content provider to access a SQLite Database. This database will contain three tables: one to maintain the match data, another to maintain the registered players associated to the match and a third to maintain the picked teams for each match.

Describe any edge or corner cases in the UX.

The app will make use of the master/detail flow. When using devices with resolution greater than 600dp, screen 5 will be combined with screen 1 and screen 2 will be combined with screen 1, to appear as only one screen.

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

ButterKnife - As there will be many views and events in this app, this library will ease the development process.
JUnit and Mockito - To make use of unit testing, JUnit and Mockito will be used.
Espresso - To test the screenflow and its functionalities, the app will make use of the Espresso lib.

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

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 Project
- Import Libraries
- Push to GitHub

Task 2: Implement UI for Each Activity and Fragment

- Build UI for Screen 1 to support Screens 2 and 4 in High-Res devices
- Build UI for Screen 2
- Build UI for Screen 3
- Build UI for Screen 5
- Build AlertDialog for Screen 4

Task 3: Implement Content Provider

This task will focus on developing the content provider functionality. The SQLite database, the associated SQL functionalities and the provider functions.

- Create ContentProvider class
- Create Contract classes and SQLite database definitions

- Create URI Matcher
- Create Provider SQL Query
- Create Provider SQL Insert
- Create Provider SQL Update
- Create Provider SQL Remove

Task 4: Loader for the Content Provider

This task will focus on implementing the loader callbacks for the content provider in order to load match and player datas from the content provider.

Task 5: Pick up Teams

This Task will focus on implementing the algorithm to pickup the teams:

- Pick teams randomly
- Pick teams by preferred position

Task 6: Notifications

This Task will focus on implementing the User Notification of an upcoming game.

Task 7: Error handling

This task focus on error treatment for the most common use cases

- Empty fields
- Error loading data from content provider
- Content Provider protection from SQL Injection

Task 8: Tests

In this task the unit and instrumented tests will be developed