How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: shubham1g5

Six Four Fantasy

Description

Six Four Fantasy is an extremely addictive free cricket fantasy game that allows you to build your own team and earn fantasy points based on how your team players actually performs in real world cricket matches. But there is a catch! Unlike all other cricket fantasy games here you only earn points for the boundaries hit by your batsmen while losing points for the boundaries committed by your bowlers. That's it! Are ready to score some Sixes and Fours ?

Intended User

Cricket lovers all round the world who wants to apply their predictive cricket skills and earn fantasy points.

Features

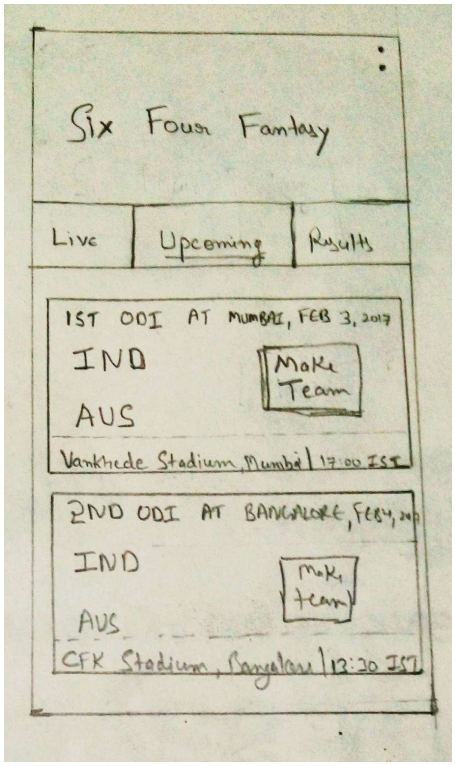
Six Four Fantasy has following features:

- Users can see ongoing and upcoming matches.
- Users can see scores and their fantasy points for the ongoing matches.
- Users can see results and fantasy points of matches concluded in past.
- Users can build their fantasy team for upcoming matches by selecting 11 players of their choice out of all available players.
- Users can see their rank and a leaderboard for all the matches ongoing or concluded in past.
- App sends notification to the user about the upcoming matches of their home team.

User Interface Mocks

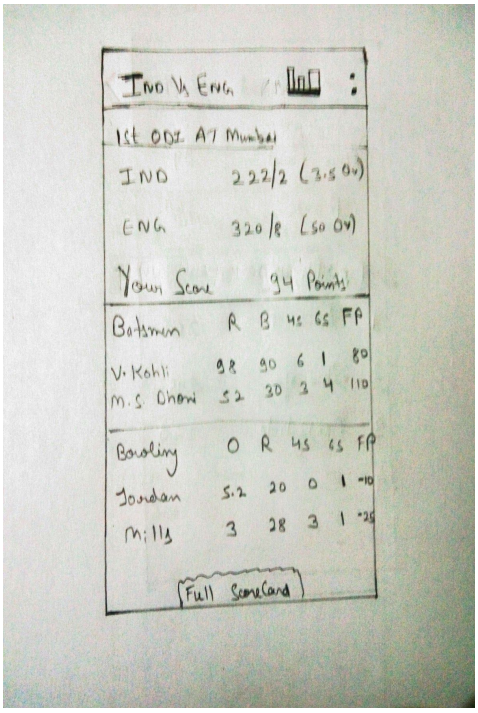
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1: Match List / Home Screen



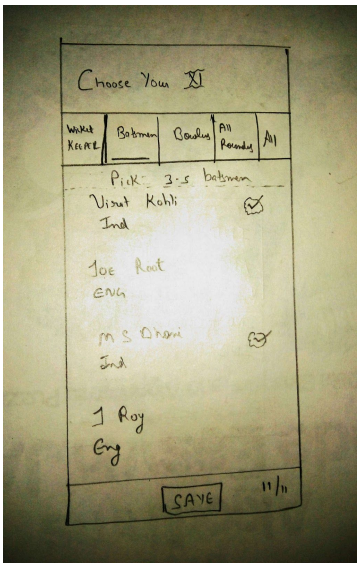
This is the home screen that user sees first on login. It has 3 tabs for the live, upcoming and concluded matches. Every Match in the list shows the 2 team who are playing the match along with date, time and venue of the match. If the match is live or concluded then the list item also shows the score of the corresponding match and if user has made a fantasy team for that match he can also see points he earned for the match. If the match is upcoming match user gets an option to make a fantasy team for the match.

Screen 2: Match Detail/ Scorecard



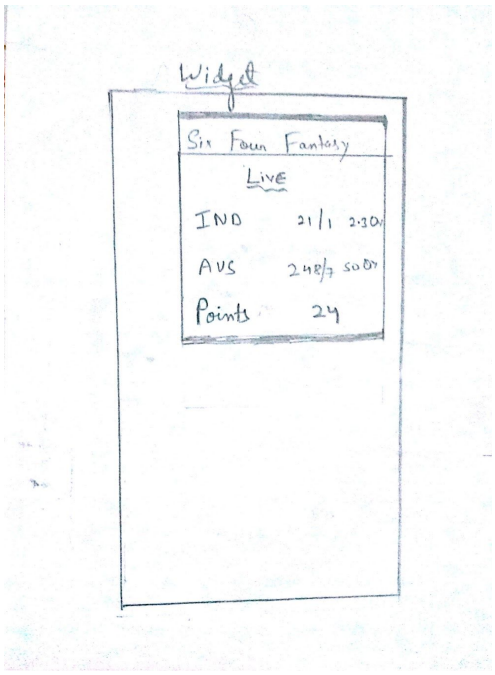
This screen shows the scorecard for the match user clicked in match list plus the fantasy points earned by the user.

Screen 3: Fantasy Team Page



This screen shows up when user clicks on ‘Make Team’ option from a match in the upcoming list. It allows user to actually select 11 players for his team and save it afterwards. User can also return to this screen later on to make any edits to his team. Players on the screen are divided into different categories depicted by the tabs. Once user selects 11 players the save button gets enabled.

Screen 4: Widget



This is the app widget that shows the scores and fantasy points of all the live matches going on currently. User can click on the match to go directly to the match detail screen. Clicking on the title opens up the home page of the app.

Key Considerations

How will your app handle data persistence?

App will use Firebase real-time database for handling user data such as user score and user team while using content providers for user independent data like match list and scorecard.

Describe any corner cases in the UX.

- Once user saves his team, the make team option in the match list screen should change to edit team.
- Save button on Team page should only get enabled if user selects 11 players satisfying all different constraints like exactly 1 wicketkeeper, 3-5 batsmen and minimum 3 bowlers.
- If user press back on the team page without saving the team, give him an warning saying all his changes after last save will be lost.

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

- FirebaseUI: To implement the login UI for the app.
- RxJava and RxAndroid: To use reactive Java and android APIs.
- Retrolambda: To use Java 8 lambda syntax.
- Retrofit: To make http requests with RXAndroid Observables.
- Firebase JobDispatcher: For syncing match and scores data locally from 3rd party APIs.
- Dagger: Dependency Injection
- Android Data Binding: For binding UI elements to model.
- Android Design Support: For taking advantage of material design components like Coordinator Layout.
- Picasso: For loading network images and caching them.

Describe how you will implement Google Play Services.

- Real-time Database: To implement data sync between local and cloud database.
- Authentication: To implement user authentication for the app.
- Notifications: To give users alerts on the upcoming matches of their home team.
- Analytics: To understand user behaviour and demographics.

Next Steps: Required Tasks

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

Task 1: Project Setup

- Configure libraries in build.gradle
- Configure min and max sdk
- Configure Java 8 compile options.

Task 2: Implement UI for Each Activity and Fragment

- Implement login activity with FirebaseUI

- Implement MatchListActivity
    - Coordinator layout with tabs for different match categories.
    - UI for each type of match Item using cardView
    - UI for tab fragments using RecyclerView with different match items layout.
    - Give SignOut and Settings option to action menu.
    - Use Cursor Loader to load match data and live scores.

- Implement Sign Out with FirebaseUI

- Implement SettingsActivity
    - Option to enable/disable notifications

- Implement TeamActivity
    - TabLayout for different player categories
    - RecyclerView with cardView items for each tab
    - Save Button

- MatchDetailActivity
    - Different UI for live, upcoming and finished matches.
    - Scorecard for live and finished matches.
    - Option menu for the leaderboard
    - Use Cursor Loader to watch for changes in scores.

- LeaderBoardActivity UI

Task 3: Model Classes

Design and add model classes for all data elements.

- User Model
- Player Model
- Match Model
- Team Model

Task 4: Data Layer

- Implement content provider for user independent data
- Sync matches and scores by using the Firebase JobDispatcher.
- Implement Firebase DB for user data like fantasy points and user team.

Task 5: Implement RuleChecker

Module to check whether all constraints for a fantasy team has been satisfied by the user.

Task 6: Implement FantasyScorer

Module to calculate and save score for a given match and a team with eleven players.

Task 7: App Widget

- List UI to show live Matches
- Intents to integrate app with widget

Task 8: Google Play Services

- Integrate Notifications
- Integrate Analytics

Add as many tasks as you need to complete your app.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"