GitHub Username: pallucandy

CricBuddy

Description

CricBuddy allows once to know the current matches i.e. matches scheduled for that day. They can check the live scores and inning requirements.

Intended User

This app is intended for Cricket Lovers.

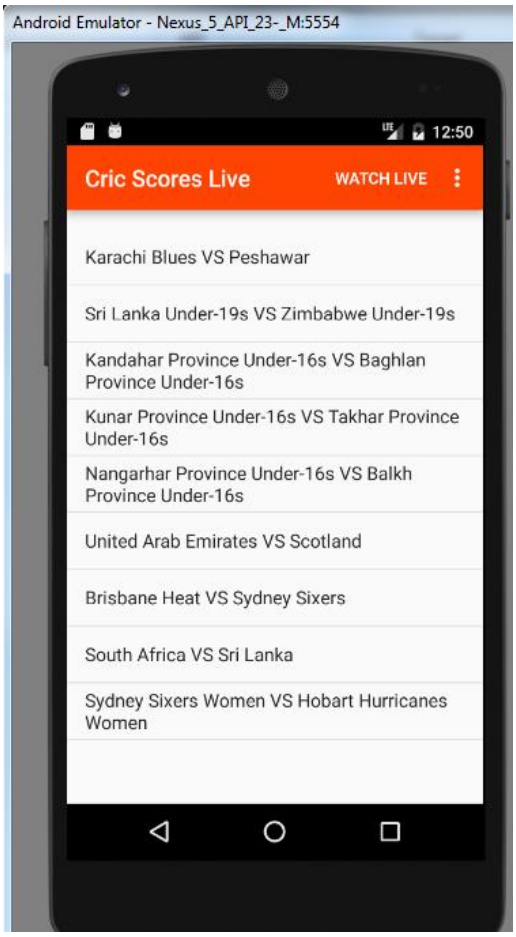
Features

- Know which matches are scheduled for the day
- Get more details on the matches like scores, schedule

User Interface Mocks

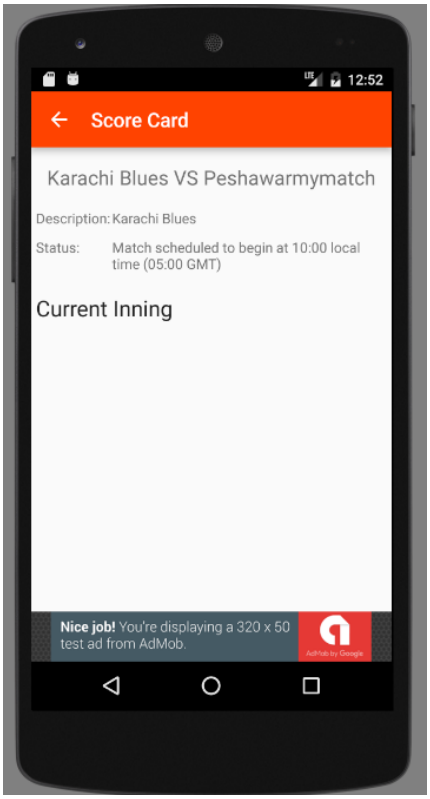
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1



Main activity showing the list of matches scheduled for that day

Screen 2



Screen showing the details of the selected match.

Screen showing the widget which would display the either the current matches or the score.

Key Considerations

How will your app handle data persistence?

I would be using the API's provided by cricapi to get the required data. Then I will build my own content provider for storing data and inflate it from API using Intent service.

Describe any corner cases in the UX.

- User can use back button from the details view to get to the Main screen
- The details view is refreshed automatically
- The screen auto adjusts on rotation
- App supports standard system Back button navigation
- All the screens can be dismissed on using standard back button
- When the app is relaunched from All Apps, the app restores the app state to the previous state.

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

- Google Support Library (AppCompat, Design, CardView, RecyclerView) for using Material Design UI components, font styles etc.
- Google Play Services library for integrating AdMob for showing some.
- Google Play Services to track user activity.
- Butterknife to bind views to fields and methods

Describe how you will implement Google Play Services.

- Set up Google AdMob ads. We will be using one interstitial ad when we launch the details View for the match.
- Set up Google Analytics and create some methods for sending user events

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

- Add library dependencies to build.gradle file

Task 2: Implement UI for Each Activity and Fragment

- Build SplashScreen Activity
- Build UI for MainActivity
- Build UI for ScoreCardView
- Build UI for WidgetView

Task 3: Create the required models

- Create the Model, MatchModel and MatchDetailsModel
- Use these models to add data parsed from the json response
- Create the content provider.
- Create the Loader to load the data to the activity.

Task 4: Sync the data

- Create the SyncAdapter class to keep the data updated data
- Implement the onPerformSync()

Task 4: Implement the widget

- Implement the widgetProvider
- Can use the Content Provider to get data into the widget
