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: dveter

SportsBuddy

Description

Tired of messaging each of your buddies if they will come to a recreational sport event? SportsBuddy will be a simple app to invite your buddies to an evening football match in just a few taps:

1. Set time of event,
2. choose buddies to invite,

3. choose location where match will take place,
4. send invites,
5. check if they are coming or not.

Intended User

Every recreational team sport player/enthusiast (football, basketball, hockey,...).

Features

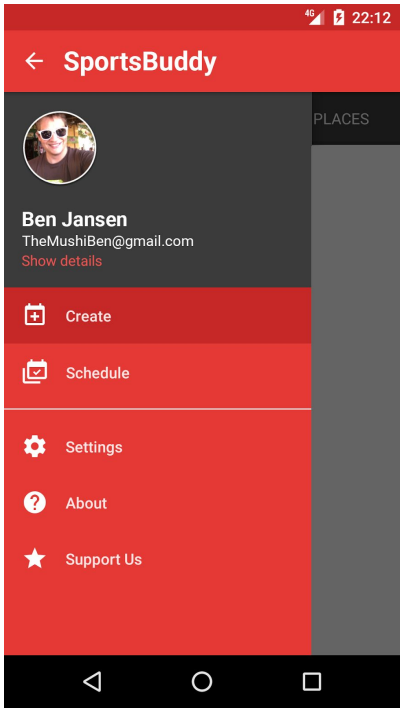
- Get feedback from your team members/buddies without texting or calling every single one
- Immediate overview who will attend next match/recreational event
- Fast invites and responses via notifications

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.

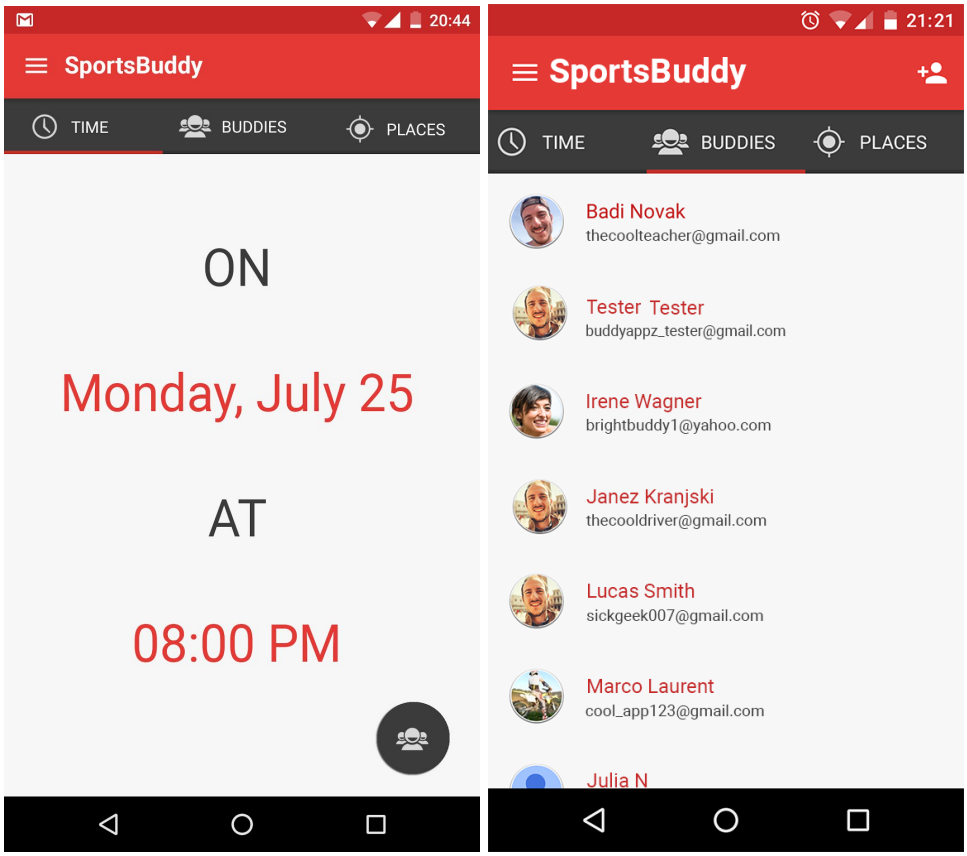
Color scheme may be changed in real app.

Screen 1



User has to log-in first to get his name, image and email address from Google+ or Facebook account. Email address is an unique users ID.

Screen 2



Main view will consists of a viewpager with 3 tabs:

- Time
- Buddies
- Places

Time tab will be used to set date and time of sport event using using Date/Time picker dialogs.

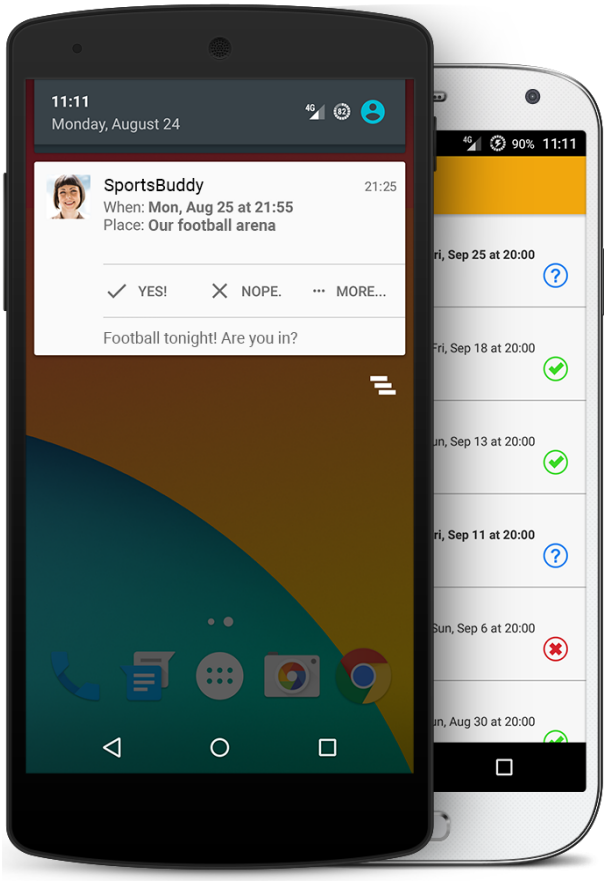
Buddies tab will be used to select multiple buddies user wants to invite. Name and exact email address is needed to add a new buddy.

Places tab will be used to select a single location (arena, playground, stadium,..) where the sport event will take place.

Buddies and Places lists will be managed by user. He will be able to add new buddies/places by tapping “+” icon on the toolbar or deleting them by long press (action mode).
When all required data will be set (time, buddies and place), user will be able to send invites by tapping the FAB.

FAB button will change its functionality according to current user selection. For example, if he didn't select buddies yet, it will slide viewpager to buddies tab automatically on tap. If every required information will be selected/set, FAB will change to "Tap to send" button. After next tap, invites will be sent to users selected on Buddies tab.

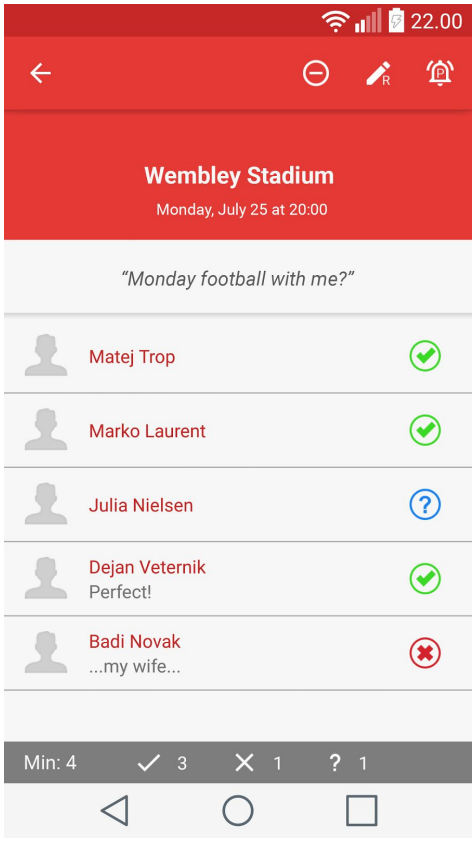
Screen 3



If invited user will have SportsBuddy app installed, simple notification will appear on his phone. Invite will be saved to local database. Users will be able to respond fast by clicking "yes" and "nope" buttons inside notification. If notification itself or "More" button will be tapped, "Invite details" activity will launch (see next screen).

If invited user won't have SportsBuddy app, he will be able to respond using simple links in a "HTML email" invite.

Screen 4



"Schedule" option from left app drawer will show the list of all received/sent invites. Clicking the list item (or notification when invite is received) will open "Invite details" activity with all invited buddies and their statuses listed. Every user will be able to change his own status (response if he is coming or not) and also "ping" other users who didn't respond yet. Everyone will be also able to write a short message. Changes will be synchronized between invited buddies automatically.

Key Considerations

How will your app handle data persistence?

After sign-in, data (buddies, places,...) will be downloaded from app server if user is already registered. The data will be saved into local SQLite database using content provider. Places and buddies tabs will be populated using content provider and loader. Basic user profile will be saved using shared preferences. Invite schedule/history is also persisted in SQLite database and accessed using content provider.

Describe any corner cases in the UX.

Viewpager with 3 tabs will be the main view. User will swipe left/right between main fragments to select desired event data. If back button will be pressed on the main view (no matter on which tab), application will exit. To view schedule, user will use left menu drawer. On Schedule view, back button will navigate to main (tabs) view. From “Invite details” view, back button will navigate to schedule view if opened via schedule (all invites list) or exit application if opened via invite notification.

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

- Google support design to implement material look and feel.
- Google play services plus - Authentication/profile info
- Google play services gcm - Messaging using GCM
- Facebook SDK - Authentication/profile info
- Glide to handle the loading and caching of user/ buddy images.
- Retrofit to handle REST communication with server
- Gson to serialize Java objects
- Schematic - content provider generation (maybe, otherwise I will implement it myself)

Describe how you will implement Google Play Services.

- `com.google.android.gms:play-services-plus` for profile information like user image, name and email address
- `Com.google.android.gms:play-services-gcm for communication between users (buddies)`
- `Com.google.android.gms:play-services-analytics to track usage and in-app navigation (optional)`

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.

App will use backend server (Tomcat + Jersey + MySQL) to link buddies, store places and invites, broadcast GCM messages.

Task 1: Project Setup

Configuration of development environment:

- Android side:
    - Android Studio + latest SDK
    - Generate release signing certificate
    - Desired libraries
    - Google Developer Console API configuration (auth, GCM - project ID, IPs, sha1 keys,..)
    - Extra Gradle configuration if needed

Task 2: Implement UI for Each Activity and Fragment

- Create left side navigation drawer with profile info header
- Build main activity fragment with viewpager and 3 tabs
    - Build “when” fragment with date/time selection dialogs
    - Build “buddies” fragment with recyclerview
    - Build “places” fragment with recyclerview
    - Build “all sport events” fragment with recyclerview
- Build “Event details” activity
    - List (recyclerview) of all responses from other invited users
    - My response
    - FAB to reply if I am coming or not
- About activity
- Manage screen rotations (to restore state of each fragment)

Task 3: Implement sign up

- Import google auth and Facebook SDK libs
- Implement google+ and Facebook authentication
- Save profile data to shared prefs

Task 4: Implement server side (RESTful webservice)

- Setup Java development environment
- Setup Libraries (Gson, Jersey)
- Setup Tomcat
- Setup MySQL database + domain model

App server will run on pa public server. I will provide URL and also server Java code if needed.

Task 5: Implement client classes to enable communication/sync with server

- Register to app server using profile data from task 3 and GCM registration ID
- Pojos, Retrofit Responses objects
- Sync mechanisms if needed

Task 6: Implement GCM notification service

- Handle GCM messages
- Style formatted notification
- Setup broadcast receivers to enable notification actions (buttons)

Task 7: Parse and persist data

- Create methods to store data into local DB

Task 7: Upcoming sports events widget

- Design collection widget
- Implement widget content provider
- Implement remote view logic

Task 8: Other

- Test the app on multiple devices
- Catch exceptions
- 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**"