GitHub Username: Richa-Bansal

App Name: WannaJoin

Description

This app is for parents who want an easy and informal way of letting their kids' friends, classmates, etc., know of their plans and also be aware of others' plans without the overhead of calls, emails, etc. It also enables on the spur of the moment group play/activities.

For example, if you plan to go to a neighborhood park with your kid and would like company but are unsure of other parents' availability or don't have the time to call, you can simply enter the park name and hours you plan to be there and share it with a select group of friends. They can decide if/when they want to join and notify you if they plan to be there.

On the other side, if you want to take your kid to a park but would like company. With this app you can quickly look up if any of your kids' classmates or friends are at the park or going their soon and then simply decide to join them (and optionally notify them).

Intended User

Parents with young kids.

Features

Main features of the app:

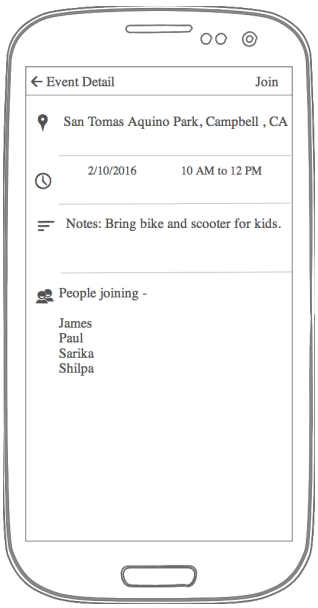
- Save outdoor activity plans and share it with others.
- View other people's outdoor activity plans (provided they shared).
- Ability to join others and see who else is joining.
- Create group of friends/neighbors/class parents etc.

User Interface Mocks

Screen 1: Events List

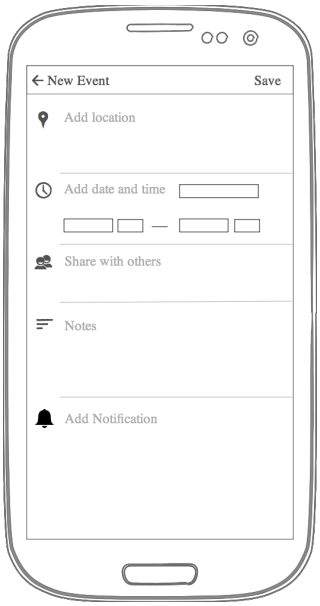
The screen will show a list of all current and future events. It will show event author, date, time, location, number of other people who will join, and a button to join. Users will be able to click on any list item to see event details (Owner of the event will see Edit Events screen instead of Event Details). The screen will also have a floating action button to create a new event.

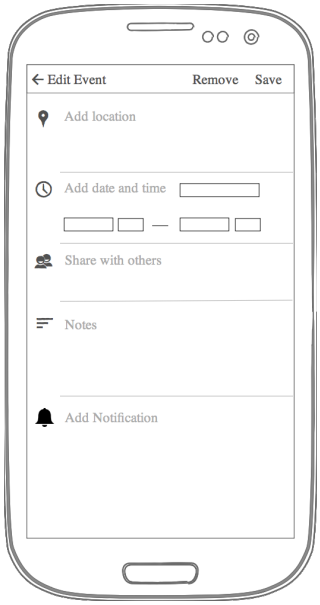
Screen 2: Event Details



Event Details screen will show details of selected event. It will show event author, date, time, location, number of people going, author notes and a button to show author that you may join as well.

Screen 3: New Event/Edit Event





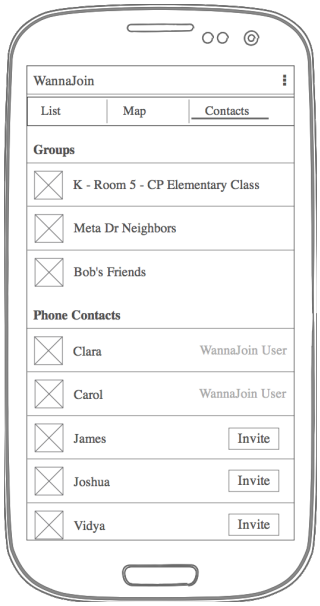
New Event screen will allow users to create/edit events with location, date and time information. Users will be able to add notes and share the event with other WannaJoin app users.

Screen 4: Map (Optional)

Similar to Events List screen, the Map screen will show a list of all current and future park events. Events will show up as markers on map. Each marker will show event author, date, time, location, number of other people who may join, and a button to join.

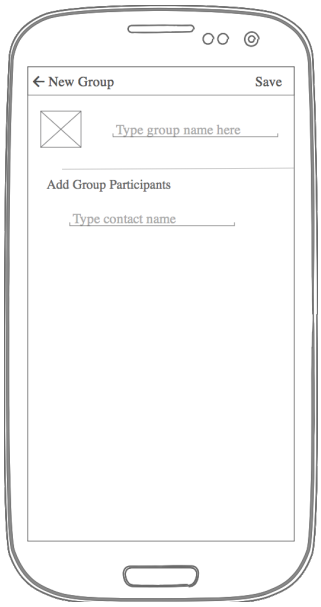
Note: Map screen is an optional screen that may not make the first phase of the app.

Screen 5: Contacts



Contact screen will show a list of all phone contacts and groups (created via WannaJoin app). It will show contacts who are existing WannaJoin app user and will provide an option to invite other contacts to use WannaJoin app. The screen will provide an option to create a group via a toolbar menu item. Clicking a group will open Edit Group screen.

Screen 6: New Group



New Group screen will allow users to create a new group of friends/neighbors/class parents etc. The screen will ask users to enter group name and select contacts from their phone contact list. Users will then be able to share their events with entire group easily.

Screen 7: Edit Group

Edit Group screen will allow users to edit group name, group icon/image, and add/remove group participants.

Screen 7: Settings

Settings screen will allow users to update their default settings. There will be following settings:
- Notifications for new events(yes/no)
- Account setting
  - current account

Screen 8: Login

Login screen will allow users to login with their google account. The screen would also capture user's phone number.

Key Considerations

How will the app handle data persistence?
- Android contact provider for getting phone contact data
- Firebase to store WannaJoin users and groups
- Firebase to store current and future events data.

Corner cases in the UX.
App should be able to handle following cases gracefully.
- User device's google play service version is lower than required
- No available events
- No internet connection
- No WannaJoin contacts
- number of concurrent users more than 100 (Firebase has a limit on number of concurrent users with their basic plan)

Libraries to include
- FirebaseUI library: FirebaseUI is an open-source library for Android that allows developers to quickly connect common UI elements to the Firebase database for data storage, allowing views to be updated in realtime as they change, providing simple interfaces for common tasks like displaying lists or collections of items.

Next Steps: Required Tasks

Task 1: Project Setup

Create a new project in Android Studio and setup Firebase using following steps:

1. Create a new app in Firebase and copy corresponding URL.

2. Update build.gradle and add firebase dependency and packaging option (to avoid duplicate file error)

```
dependencies {
  compile 'com.firebase:firebase-client-android:2.3.1+'
  }
  and
  android {
        ...
        packagingOptions {
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/LICENSE-FIREBASE.txt'
        exclude 'META-INF/NOTICE'
         }
  }
```

3. Update AndroidManifest file and add internet permission

```
  AndroidManifest.xml
  <uses-permission android:name="android.permission.INTERNET" />
```

4. Set firebase context at application level or activity level.

```
  @Override
  public void onCreate() {
  super.onCreate();
  Firebase.setAndroidContext(this);
  // other setup code
  }
```

5. Add firebase url in gradle.properties file

```
  UniqueFirebaseRootUrl = “https://<>.firebaseio.com"
```

6. Add following in app’s build.gradle

```
  buildTypes.each {
  it.buildConfigField 'String', 'UNIQUE_FIREBASE_ROOT_URL', UniqueFirebaseRootUrl}
```

7. Add a constant in constants file

```
  public static final String FIREBASE_URL =
  BuildConfig.UNIQUE_FIREBASE_ROOT_URL;
```

8. Above constant can be used to instantiate Firebase reference in the app.

```
  Firebase ref = new Firebase(Constants.FIREBASE_URL);
```

Task 2: Implement UI for events creation and listing

1. Create event list, new event, edit event and event details screens as shown in the mock ups above.

2. Events data needs to be stored in Firebase using apis listed here:
   https://www.firebase.com/docs/android/api/
3. Setup Firebase UI library in the build.gradle file.
   /* Firebase UI */
   compile 'com.firebaseui:firebase-ui:0.2.2'
4. Use FirebaseListAdapter to create an adapter for events list screen.
5. Use various Firebase listeners to get latest data from Firebase and update UI. Follow the documentation here: https://www.firebase.com/docs/android/api/

Task 3: Implement login functionality

1. Implement login based on google account and design a way to capture user's phone number. Follow Firebase documentation here: https://www.firebase.com/docs/web/guide/user-auth.html
2. Store user information such as email and phone number in Firebase.
3. User phone number would be used when showing list of phone contacts. It would provide a way to identify users who use the app vs users who haven't created an account yet.

Task 4: Update events list UI

1. Update events list UI to show events associated with current user.
2. Update edit/remove privileges based in current user. Only owner of an event should be allowed to edit or remove an event.

Task 5: Implement UI for contacts and groups list

1. Create contacts screen, new group and edit group screens as shown in the mock ups above.
2. Use android contact provider to display phone contacts. Use user information in Firebase to identify current users of the app and users who haven't installed the app yet.
3. Add a menu option to create new group.
4. New group screen should allow users to create groups of existing WannJoin app users.
5. Contact screen should display all groups created by current user. Clicking on any group should launch edit group screen.

Task 6: Implement event sharing functionality

1. Add event sharing functionality on New Event and Edit Event screens. User should be able to add group (or groups) or individual contacts (only existing app users) to their events. Events would then be visible to all contacts with whom it was shared.

Task 7: Implement Map UI (Optional)

1. Create map UI centered at user's current location.
2. Events that are associated with the current user should show up as markers on the map.
3. If user clicks on any marker then event detail screen should open up.