The Giving Challenge

Kyle Parker
P7: Capstone Stage 1

Table of Contents

GitHub Username: kyleparker (https://github.com/kyleparker)

The Giving Challenge

Description

Problem

Most of us lead busy, hectic lives and tend to focus on our own immediate needs and responsibilities. We stay in our own lane. This perspective isn't selfish or misguided, but it can cause us to miss out on opportunities to share of ourselves with others who are in need. Whether it's a warm meal on a holiday, supplies for that first day or school, or a stranger who just needs a smile from a passer-by. Giving back doesn't have to be a chore, an act of kindness doesn't require your wallet, and helping others doesn't mean you go without. Opportunities surround us each and every day, and finding a way to give back that matches your abilities and capabilities, or receiving encouragement from friends, family, colleagues or classmates in your communities may make the difference in providing for others who are in need.

Proposed solution

The Giving Challenge was created to help alleviate some of the obstacles associated with giving back to your community and finding the encouragement from others to share your time, talents and treasures with those around you. Through a daily giving challenge, the app reduces the hassle of searching for and finding charitable organizations, easy pay-it-forward acts of kindness, and ideas for other outlets for your offerings. The Giving Challenge builds on a successful web and blog presence from Stephanie L. Jones, a.k.a. Giving Gal, and is a companion to a forthcoming book written by Stephanie entitled *The Giving Challenge: 40 Days to a More Generous Life*.

Giving back shouldn't be a chore or yet another thing on your already hectic schedule, so The Giving Challenge focuses on the little things that can make a big difference in someone else's life. As you master the easy ones, you'll have the opportunity to challenge yourself and others to do more!

Whether you need a plan to guide your giving or confident that opportunities will present themselves, The Giving Challenge is there to provide a way to keep you on track while sharing the journey with others. Use the app to take a picture of your gifts, view the progress of those in your communities, create your own challenges, and receive encouragement to keep going.

Once the momentum starts, you'll be amazed at how easy it becomes – you'll see possibilities wherever you go. Going out to eat? Leave a little extra tip for the waitress putting herself through

college. Have some toys you don't play with anymore? Stop by a church or toy-drive to brighten another child's day. See a parent carrying a baby? Hold the door for her, the baby isn't the only thing mom or dad is probably carrying.

Shy or outgoing, rich or just making it by, busy or carefree, there are countless ways you can make a difference in someone else's life, and through The Giving Challenge you'll find ways to make that a reality. Challenge yourself, challenge those people in your own life, and watch the difference it can make when you give.

Intended User

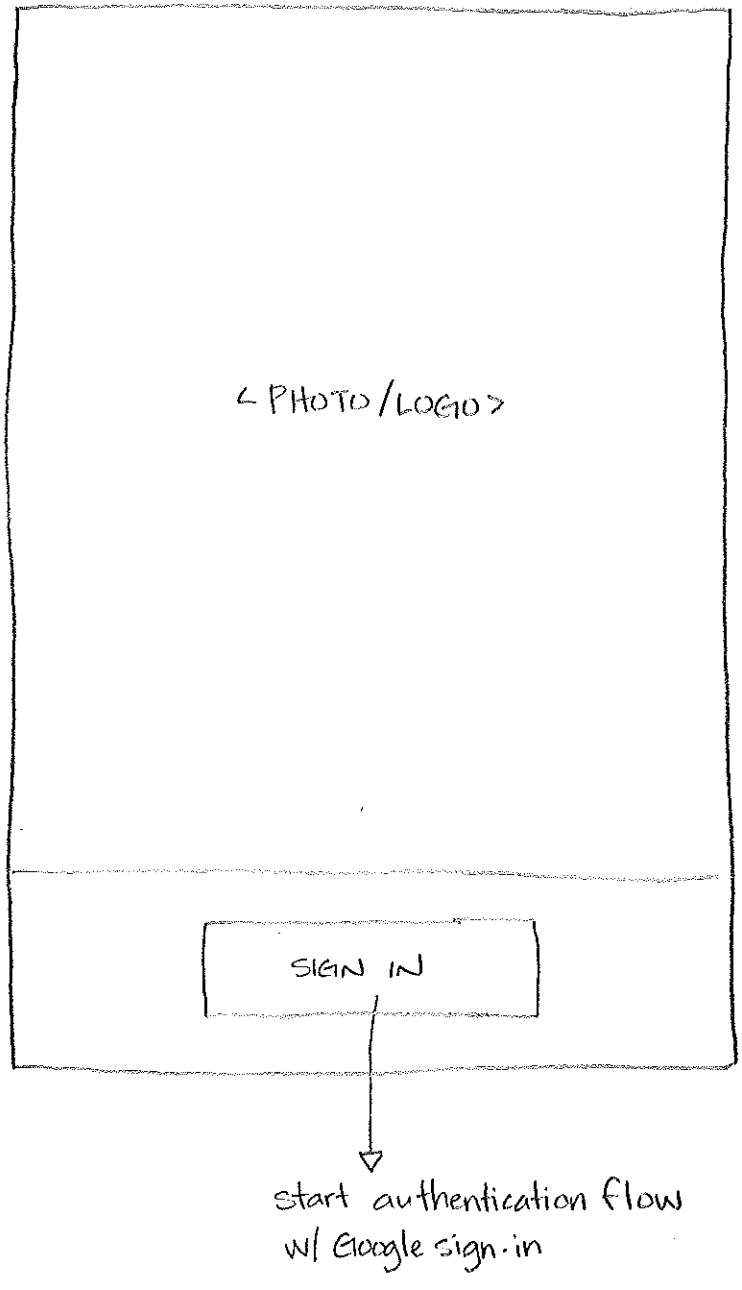
The intended user for the first phase of the app is women, aged 25-55. Many of those women have children, so phase 2 may possibly include a coloring book and challenges that would target both populations. Stephanie believes that kids who are raised as givers will continue to give once they become adults.

Features

- View daily challenges based on a "push" from community members or random selection from challenge list
- View current and upcoming challenges for yourself and others in your communities
- View challenges based on location
- View community members
- View profile for yourself and others in your communities
- Invite others to join a challenge
- Accept a challenge from a community member
- Build communities based on family, friends, colleagues, classmates, neighbors and other affinity groups
- Take pictures of activities to document your challenge
- Create new challenges

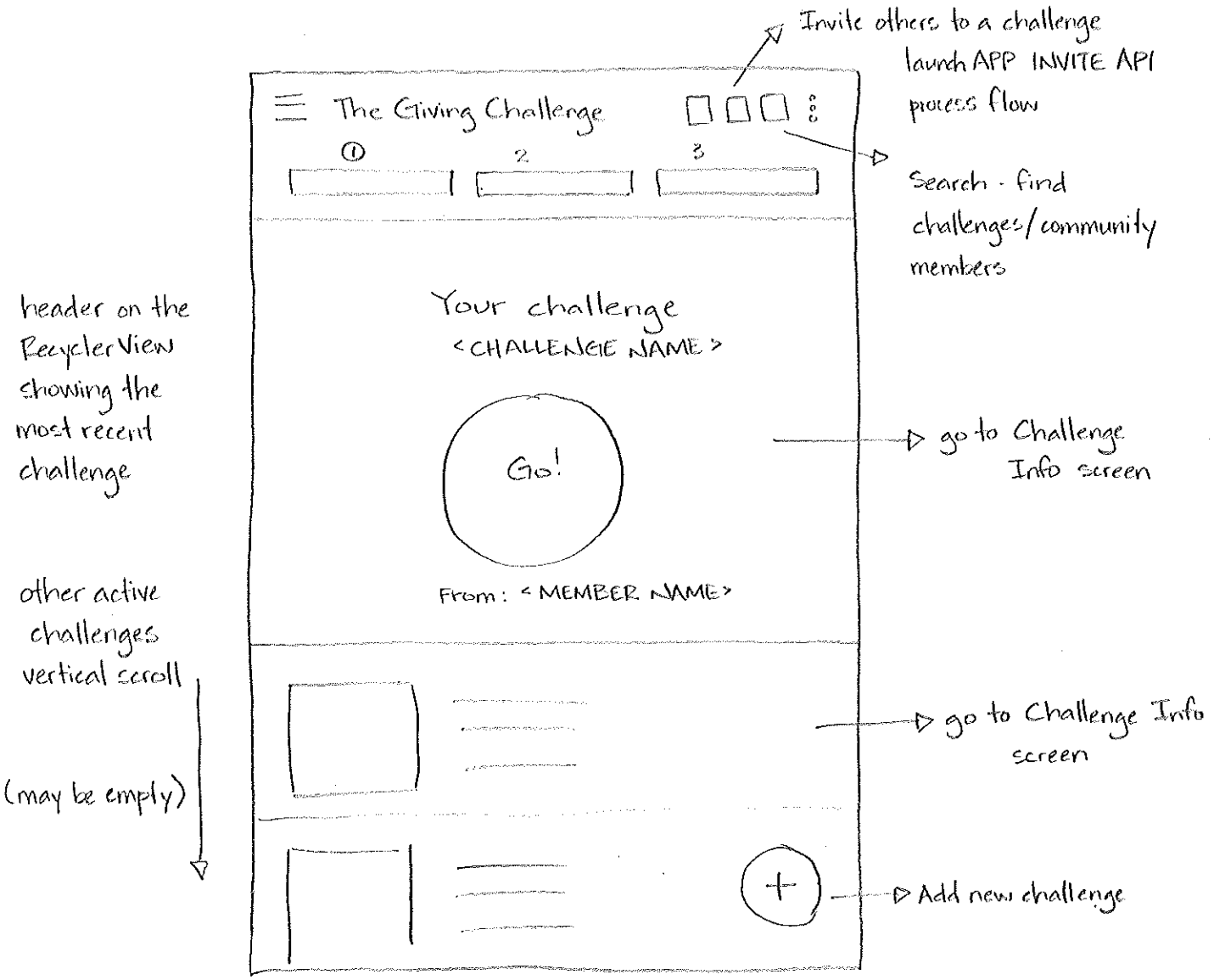
User Interface Mocks

Account Activity



MainActivity

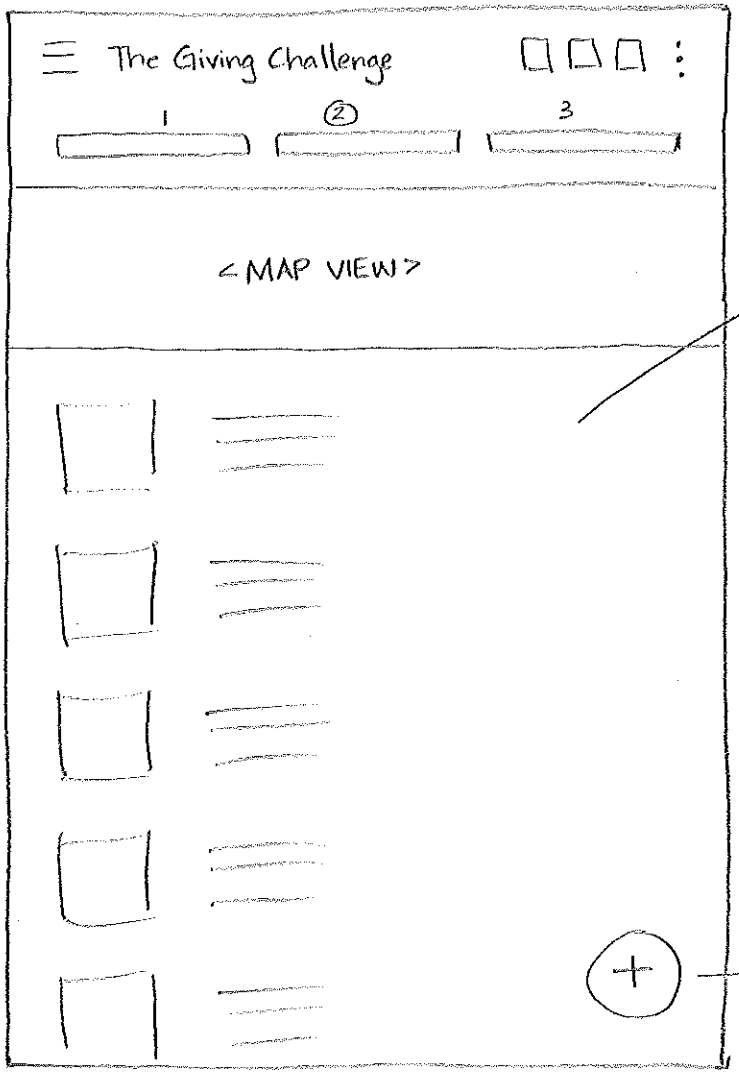
ViewPager w/ three fragments



① Challenge Me!

MainActivity

ViewPager w/ three fragments

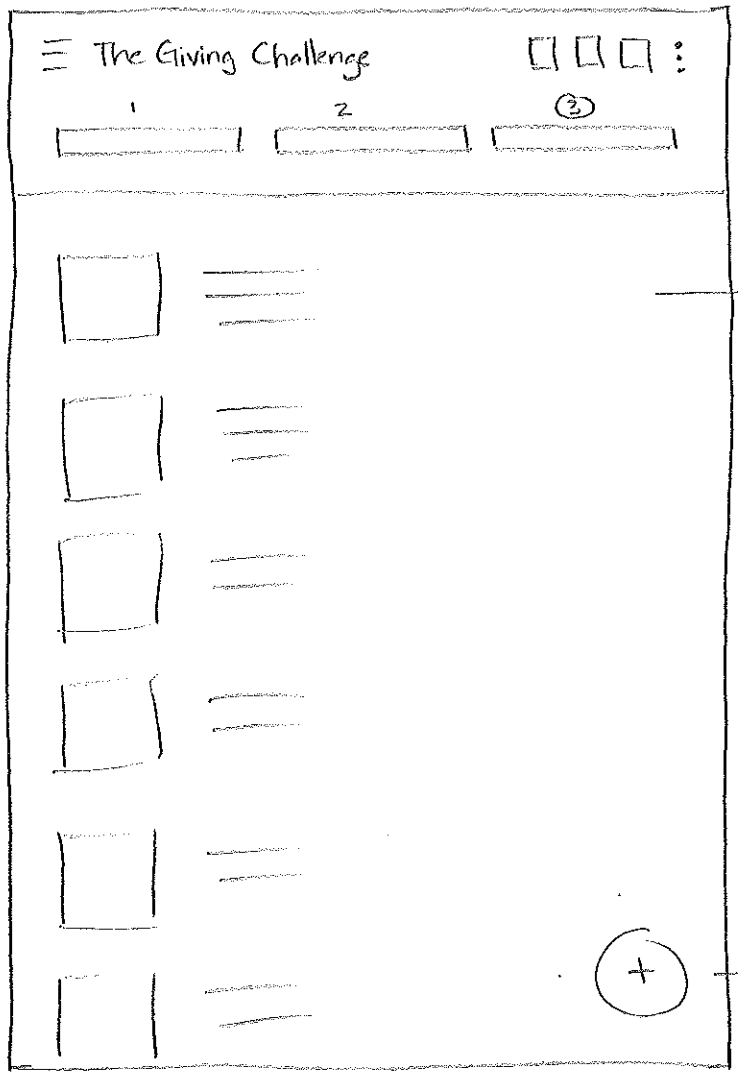


display additional challenges
- Nearby
- Community activity
- Community membership

(2) More Challenges

Main Activity

ViewPager w/ three fragments



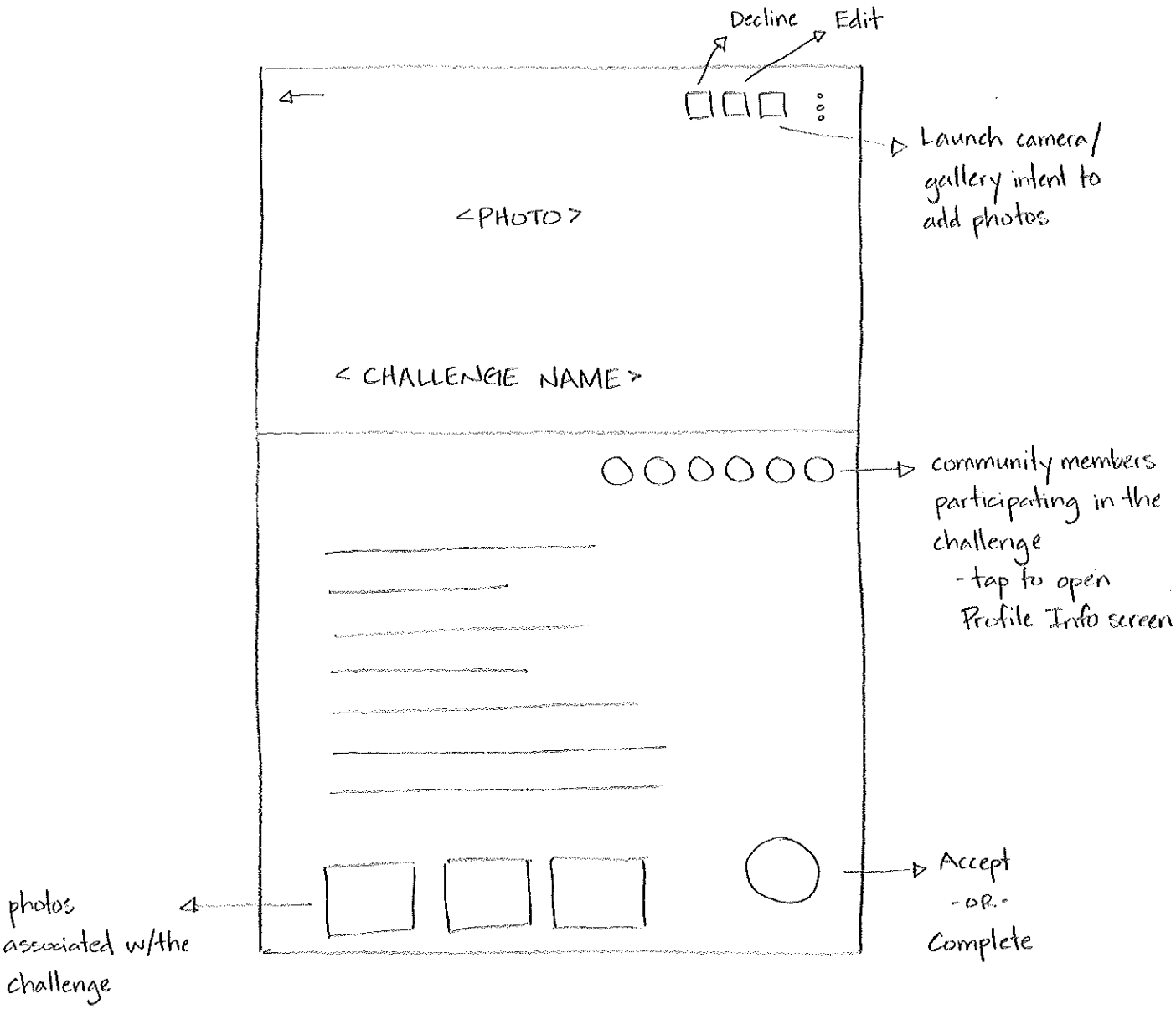
list of community members organized by name

→ go to Profile Info screen

→ add challenge

③ Community

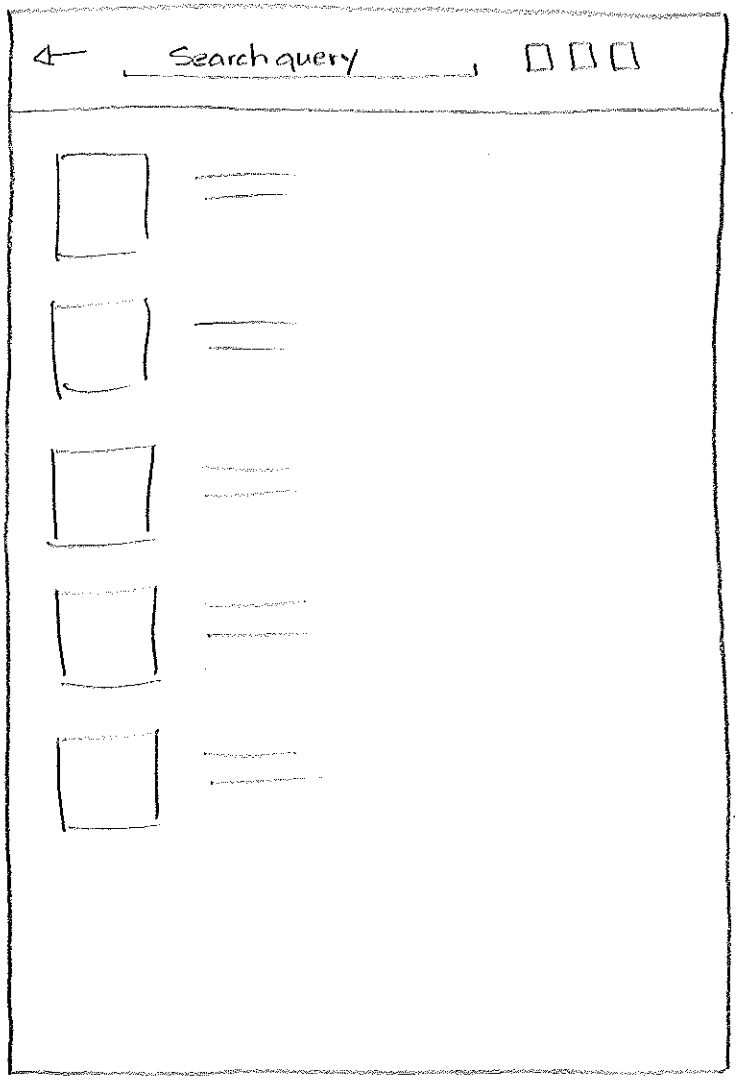
Challenge Info



Search Results Activity

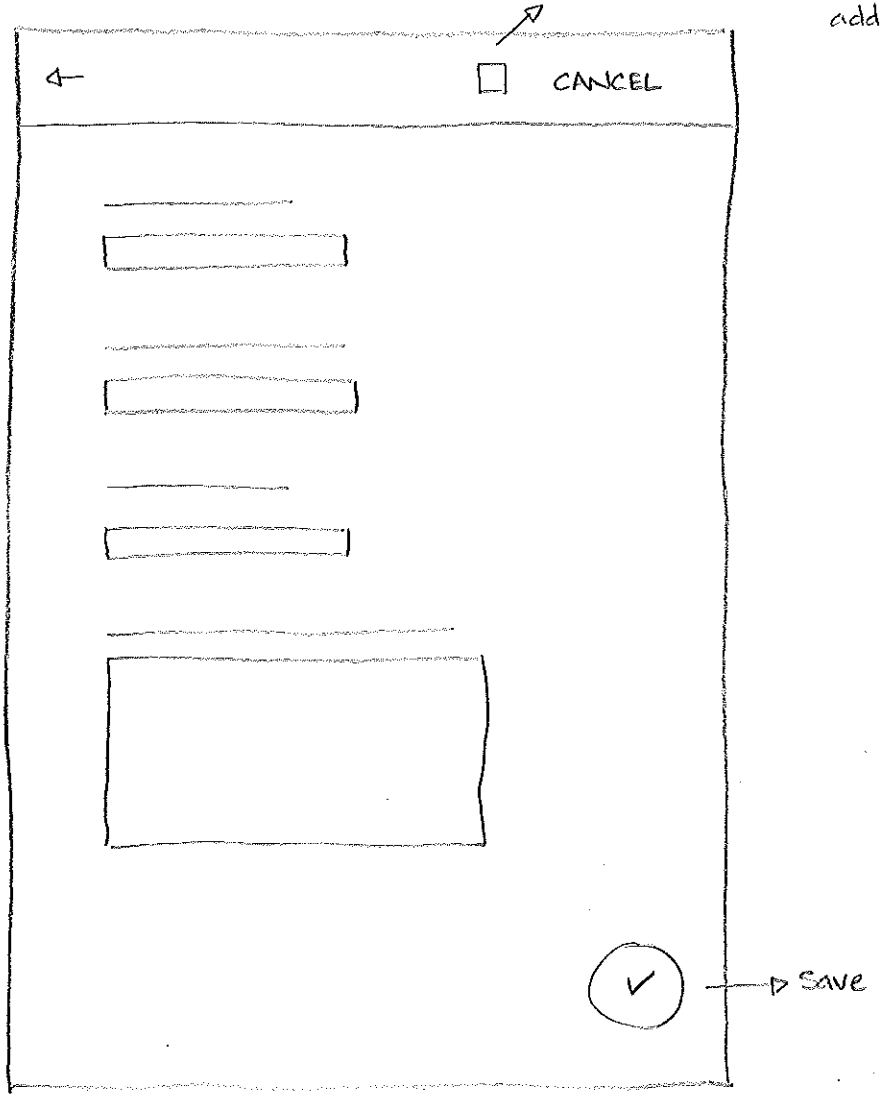
search results based on query

- section headers for challenges, communities & members



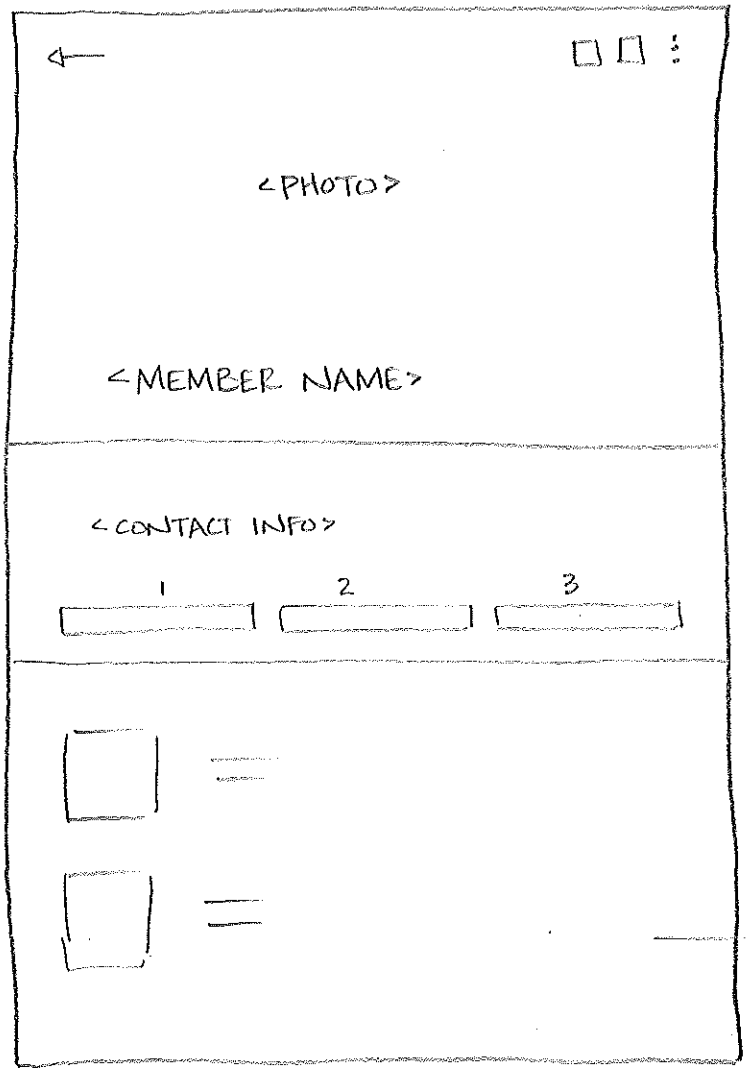
Challenge Edit Activity

create a new challenge or edit an existing one



- popup confirmation on "save"
- ability to invite others to join the challenge

Profile Info Activity



ViewPager w/ three fragments displaying the member's challenge activity

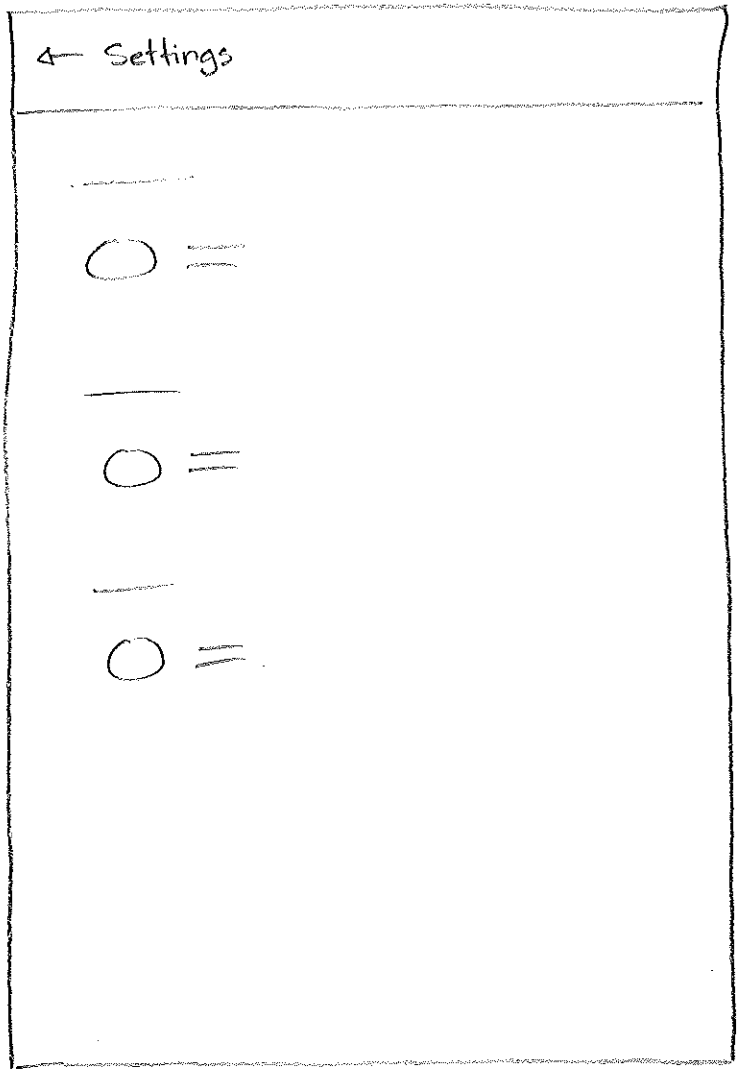
Challenges
(1) Active / Participated
(2) Invited
(3) Upcoming

go to Challenge Info screen

Settings Activity

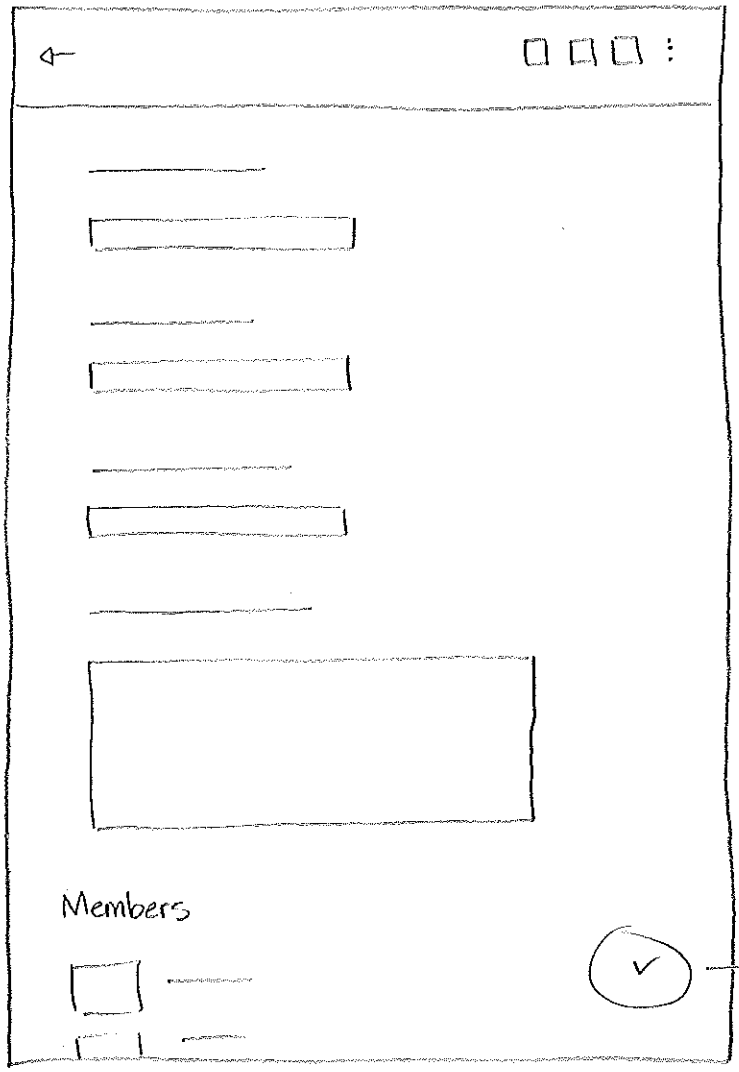
Standard Settings activity

- functionality : options TBD



Community Edit Activity

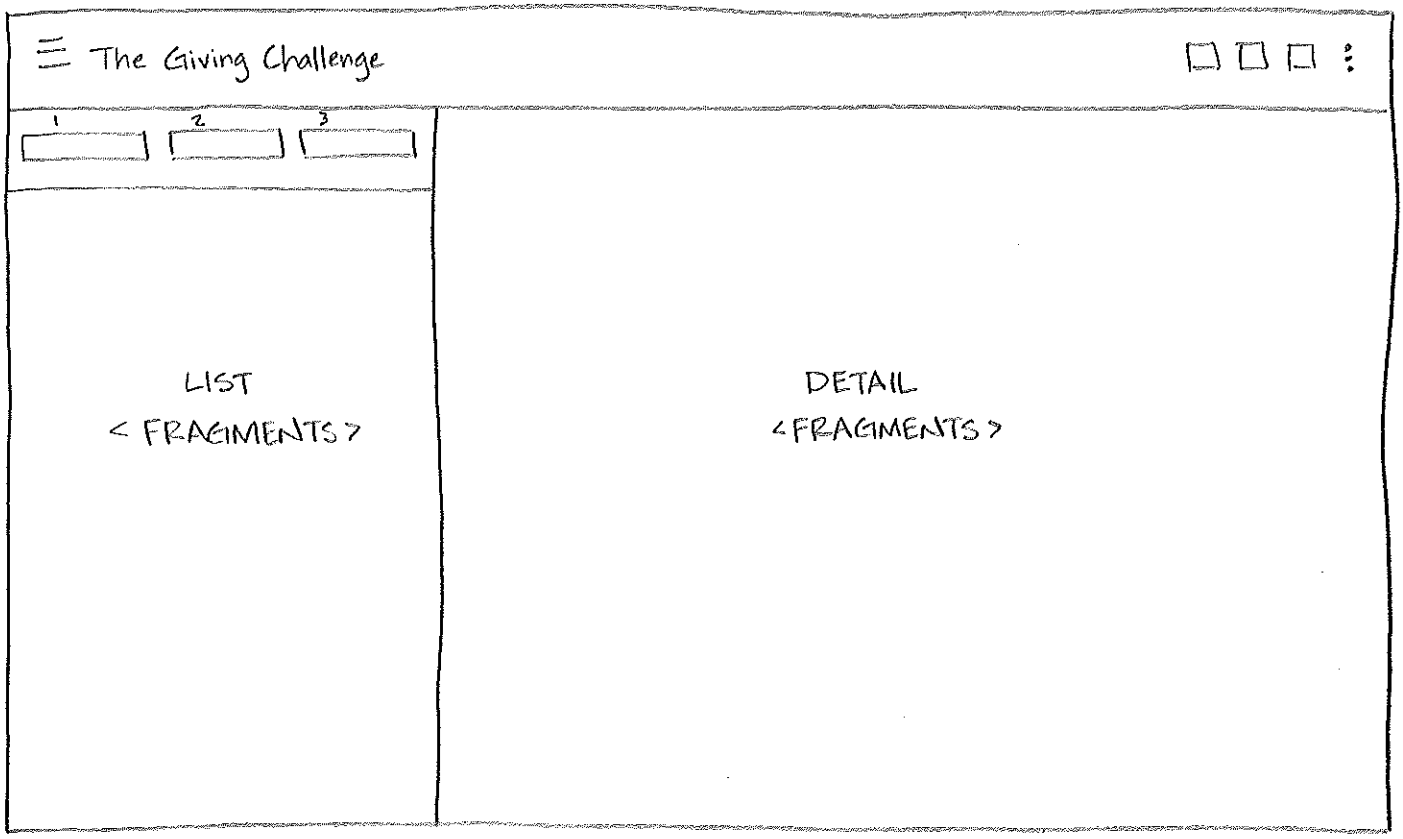
edit existing or create new community



members of the community

- show confirmation on "save"
- invite others to join the community

Tablet layout



1 Challenge Me

2 More Challenges

3 Community

- other screens like "edit" and "info" activities will appear as DialogFragments over the MainActivity

Key Considerations

How will your app handle data persistence?

Data for the app will originate from and be stored in multiple locations, depending on the form and function.

- Initially, all challenge data will reside in a Firebase cloud db and will be maintained via a secondary web interface (not included in this design documentation). As the app matures and grows, this may migrate to a different solution based on cost and usage statistics.
- As users download and sign into the app, the data will be synced to the device via a SyncAdapter. The SyncAdapter will maintain the data consistency between the device and Firebase db.
    - When new data is added to the Firebase db, a silent notification will be sent to the device via Google Cloud Messaging. Once received by the device, the SyncAdapter will be initiated to update the local sqlite database and refresh the app display.
- On the device, there will be a content provider that utilizes a local sqlite database. Data in the sqlite database will consist of the following:
    - User data
        - signed in user
        - community group(s) member profile
    - Challenge data
        - past and current challenges
        - associations between users and challenges (both signed in user and community members)
    - Community data
        - Communities the user is a part of
        - Other members of those communities
- Shared preferences will be used as needed on the device to facilitate certain functions within the app.
- The app will utilize the Android backup provider to store certain preferences in the cloud to easily migrate data from device to device, or in the event the user has multiple devices.

The primary partner in this venture - Stephanie Jones, owner of GivingGal.com - will provide the initial set of challenges available to users. These challenges will highlight content that appears on her web site, as part of the “500 Days of Giving” challenge, and her forthcoming book, *The Giving Challenge: 40 Days to a More Generous Life.*

Describe any corner cases in the UX.

The app will be developed according to best practices, as defined by the Android Developer documentation, and utilize all of the standard UI interactions for navigation, layout, button/action item placement, etc. The primary screen will utilize a ViewPager with fragments, which include RecyclerViews to browse and view the data. Navigation will be present in the toolbar, including a “back to home” button, and a floating action button. The nav drawer will provide secondary navigation to other areas of the app.

Notifications will be utilized for this app and take the user to the appropriate screen, based on the information available in the notification.

The user may leave the app via a “share intent” function, which will behave according to the Android Developer documentation guidelines.

Other considerations

- Device rotation: the app will make use of saved instance state to retain all applicable data in the event the user rotates the device. The retained state will also be used to determine what, if any, methods/calls need to be fired - certain calls will only occur during an onCreate event, and will not be repeated if the device is rotated.
- Error handling for content provider and shared preferences: the app will confirm and validate values saved via the content provider or shared preferences to ensure accurate and expected behavior when retrieving the data.
- Device offline/no data connection: the app will verify the user has a data connection prior to making calls to any cloud services, when retrieving data like images or other network requests. If the data connection is not available, a message will be displayed and the call will exit gracefully.
- GPS not enabled: the app will check to ensure GPS is enabled when using the map interface. A message will be displayed to the user that the app cannot determine their location without GPS and provide an easy way to jump to the settings to enable that feature.
- Save before exiting: the app will check to determine whether an edit screen has been modified and if the user has tapped the back button. If this case occurs, a dialog box will appear prompting the user to save their changes, ignore changes or cancel.

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

Google Play Services: Maps and Location

- display upcoming challenges on the map, based on location to find additional challenges
- display previous challenges on the map, based on location for the user to see the extent/coverage of their giving

Google Play Services: Authentication
- handle authentication using a standard Google account, thereby eliminating the need to create a separate username/password for the app

Google Play Services: Analytics
- gather usage statistics

Google Play Services: App Invite
- allow users to invite and challenge others
- the App Invite will include a deep link activity that will take invitees directly to the appropriate screen in the app. If the app has not been installed, the invitee will be prompted to download it from the Play Store.

Google Cloud Messaging
- send notifications to device - initiate SyncAdapter, provide challenge updates, invitation notifications, etc.

Universal Image Loader
- handle downloading and caching image for display within the app

OkHttp
- networking communication with cloud services

Firebase
- cloud database provider

Design Support Library: Palette and CardView
- UI and layout functionality

SuperSLiM
- add sections and headers to the RecyclerView

Next Steps: Required Tasks

Task 1: Project Setup

- Create new project using Android Studio
- Select phone/tablet platform with a minimum SDK of API 16, JellyBean 4.1
- Add Blank Activity to the project
- Configure build.gradle for the app
    - add necessary libraries from Google and third-party developers
    - add necessary build types for release and debug
- Begin generating necessary resource folders and files
    - styles
    - strings
    - dimens
    - drawables
    - integers
    - keys
- Create project folder structure

    - content
        - content provider and related utilities
    - layout
        - db table layouts and schemas
    - services
        - cloud based services and sync adapter
    - ui
        - activities and fragments
    - utils
        - various utilities and helper classes, adapters, etc
- Configure AndroidManifest.xml
    - register services, content providers, sync adapters, activities, and permissions
- Create necessary property via Google Analytics console
    - add Google Analytics to the project
- Create necessary permission and API registrations on Google API Console web site

Task 2: Implement UI for Each Activity and Fragment

Build UI for the following activities:

- MainActivity
- AccountActivity
- ChallengeInfoActivity
- ProfileInfoActivity
- ChallengeEditActivity
- ProfileEditActivity
- CommunityMemberListActivity
- CommunityEditActivity
- SettingsActivity
- SearchResultActivity

Build UI for the following fragments:

- ChallengeMeFragment
- MoreChallengesFragment
- CommunityFragment
- ChallengeInfoFragment
- ProfileInfoDialogFragment

Task 3: Create data layer

- Create content provider
- Create database helper
- Create provider utils to query the content provider
- Create table layouts for local sqlite database

- Create objects that implement Parcelable and Comparable
- Create SearchView provider and SearchSuggesetion helper methods
- Update AndroidManifest.xml to register providers

Task 4: Create AccountActivity

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Implement Google Play Services sign-in authentication
- Save credentials to local sqlite db
- Set shared preferences
- Sync data to Firebase cloud db
- If signed in, start the MainActivity, otherwise start the authentication process flow based on the API implementation

Task 5: Create MainActivity

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Verify user has authenticated - if not, start the AccountActivity
- Set up ViewPager and TabLayout for the three fragments displayed on this screen
    - Challenge Me!
    - More Challenges
    - Leaderboard
- Set up the nav items in the sliding nav drawer
    - Profile
    - Community
    - Settings
- Set up the header view for the sliding nav drawer
    - Profile pic
    - Display name
- Set up the action items and intents for the toolbar
    - Add new challenge
    - Search
    - Invite others
- Implement the SearchView
- Connect to GoogleApiClient
- Implement the FAB - launch the App Invite process flow using the App Invite API

Task 6: Create ChallengeMeFragment

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed

- Query local database using the content provider to retrieve all challenges associated with the user
- Using the content provider, load an array of objects based on the data retrieved from the db
- Display the results using a RecyclerView with a header view
- The RecyclerView header will show the most recent challenge given to a user
    - This view will use a different layout than the items within the adapter
- Create an adapter to display each item in the RecyclerView
- Add an onClick listener to each item - this will start the ChallengeInfoActivity intent
- Implement the Universal Image Loader for the images in the header view and the adapter items

Task 7: Create MoreChallengesFragment

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Query local database using the content provider to retrieve all challenges available
- Using the content provider, load an array of objects based on the data retrieved from the db
- Display the results using a RecyclerView
- Create an adapter to display each item in the RecyclerView
- Implement SuperSLiM to add a section header in the adapter to group challenges by location, community member activity, community activity
- Add filter functionality (action item in the toolbar) to filter the results and only display the selected groups
    - Implement using a dialog to select the appropriate filter, then refresh the adapter
- Implement the Universal Image Loader for the images in the adapter items
- Consider implementing a map view using the Google Maps API to display challenges based on location

Task 8: Create CommunityFragment

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Query local database using the content provider to retrieve all people in the user's communities with their giving/challenge activity
- Using the content provider, load an array of objects based on the data retrieved from the db
- Display the results using a RecyclerView
- Create an adapter to display each item in the RecyclerView
- Consider using SuperSLiM to group people based on community membership
- Implement the Universal Image Loader for the images in the adapter items
- Consider adding a filter to view the list based on specific community types/names

    - Implement using a dialog to select the appropriate filter, then refresh the adapter

Task 9: Create ChallengeInfoFragment

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Query local database using the content provider to retrieve the challenge info based on the ID
    - ID passed to the fragment via the extras bundle
- If the user comes to the fragment via the ChallengeMeFragment:
    - Implement FAB to mark the challenge as completed
- If the user comes to the fragment via the MoreChallengesFragment
    - Implement FAB to accept the challenge
- Implement the Universal Image Loader for the images in the adapter items
- Launch the camera or gallery intent to add photos to the challenge info
    - This will create a photo library highlighting the user's experiences during the challenges, and also serves as a visual record of the participation

Task 10: Create SearchResultActivity

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Query local database using the content provider to retrieve the results based on the query
    - Query passed to the activity via the extras bundle
- Using the content provider, load an array of objects based on the data retrieved from the db
- Display the results using a RecyclerView
- Create an adapter to display each item in the RecyclerView
- Implement the Universal Image Loader for the images in the adapter items

Task 11: Create ChallengeEditActivity

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- If editing an existing challenge, query the local database using the content provider to retrieve the challenge info based on the ID
    - ID passed to the activity via extras bundle
    - Using the content provider, load an object based on the data retrieved from the db
    - Display all existing challenge info
- If creating a new challenge, display a blank form for the user to supply the necessary info
- Implement FAB to save the challenge info and update/insert the data in the local sqlite db using the content provider
- Initiate the SyncAdapter to sync the challenge data to the Firebase db

Task 12: Create ProfileInfoDialogFragment

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Query local database using the content provider to retrieve the profile info for the selected user based on the ID
    - ID passed to the activity via the extras bundle
- Final implementation to be determined:
    - Either create a ViewPager to display the challenges for the user (participated, invited and upcoming)
    - Or display in a RecyclerView with section headers
- Determine if challenges should be displayed for community members or just the user logged into the app - should the lists of challenges be private for the members?
- Implement the Universal Image Loader for the images in the adapter items
- Launch camera/gallery intent and allow the user to change the profile picture

Task 13: Create CommunityMemberListActivity

Determine if this activity is needed with the CommunityFragment on the main activity.

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Query local database using the content provider to retrieve all community members related to the user
- Using the content provider, load an array of objects based on the data retrieved from the db
- Display the results using a RecyclerView
- Create an adapter to display each item in the RecyclerView
- Implement the Universal Image Loader for the images in the adapter items
- Implement SuperSLiM to add a section header in the adapter to group community members by community type

Task 14: Create CommunityEditActivity

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- If editing an existing community, query the local database using the content provider to retrieve the community info based on the ID
    - ID passed to the activity via extras bundle
    - Using the content provider, load an object based on the data retrieved from the db
    - Display all existing community info and members
- If creating a new community, display a blank form for the user to supply the necessary info

- Implement FAB to save the community info and update/insert the data in the local sqlite db using the content provider
- Initiate the SyncAdapter to sync the community data to the Firebase db
- Implement App Invite API to invite others to join the community
- Display community members in a RecyclerView below the basic community info
- Create an adapter to display each item in the RecyclerView
- Implement the Universal Image Loader for the images in the adapter items

Task 15: Create SettingsActivity

- Create layout and resource files (strings, drawables, dimens, etc)
- Define constants and utility/helper methods as needed
- Specific options for the SettingsActivity to be determined
- Update shared preferences and/or local database using the content provider, depending on the options implemented

Task 16: Create SyncAdapter

- Create SyncAdapter to update data between the device and Firebase cloud db
- Register adapter in the AndroidManifest.xml
- Retrieve challenges from the Firebase cloud db to populate the lists on the MainActivity
- To be determined: sync on initial load of the app, then refresh via SyncAdapter as new challenges are added
    - Use GCM to notify the app and initiate the SyncAdapter

Task 17: Create tablet layouts

- Create layout and resource files (strings, drawables, dimens, etc) to display challenge fragments in a list/detail tablet layout
- Define constants and utility/helper methods as needed

Task 18: Create DeepLinkActivity

- Define process flow for incoming deep links
- Handle logic and display