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

GitHub Username: helmuthb

GDG Vienna Conference Q&A App

Description

We at GDG Vienna are organizing awesome events, and sometimes also larger conferences (e.g. DevFest Vienna). In these events we want to encourage questions, but many people are shy and don't want to ask questions directly.

This app allows attendees in a room to ask questions, upvote existing questions, and optionally even stay anonymously.
Multiple rooms are supported by the app.

Intended User

The app is intended for people attending the conference in person or watching the live stream (if available).

Features

- Allow Login (using Firebase Authentication)
- Show available rooms
- Show existing questions per room
- Allow asking questions (only when logged-in)
- Allow "liking"/upvoting existing question (only when logged-in)

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 Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1: Home

This screen is the start screen when opening the application.

When the user is logged in it is shown accordingly.

Screen 2: Drawer

The main navigation pattern is a drawer menu.

Screen 3: Questions

Asking questions or up-voting existing ones is only possible for logged-in users.

A filled star means the user has upvoted the question, an empty one means that the user has not yet upvoted the question. The FAB allows to add a new question.
The number in the star shows the number of current upvotes.

Questions are sorted by number of upvotes. The assumption is that not so many questions will be actually asked (maybe 5) that it justifies any sort or filtering capabilities.

Screen 4: New Question

Asking questions or up-voting existing ones is only possible for logged-in users. However, a user can opt-out from making their name available together with the question.

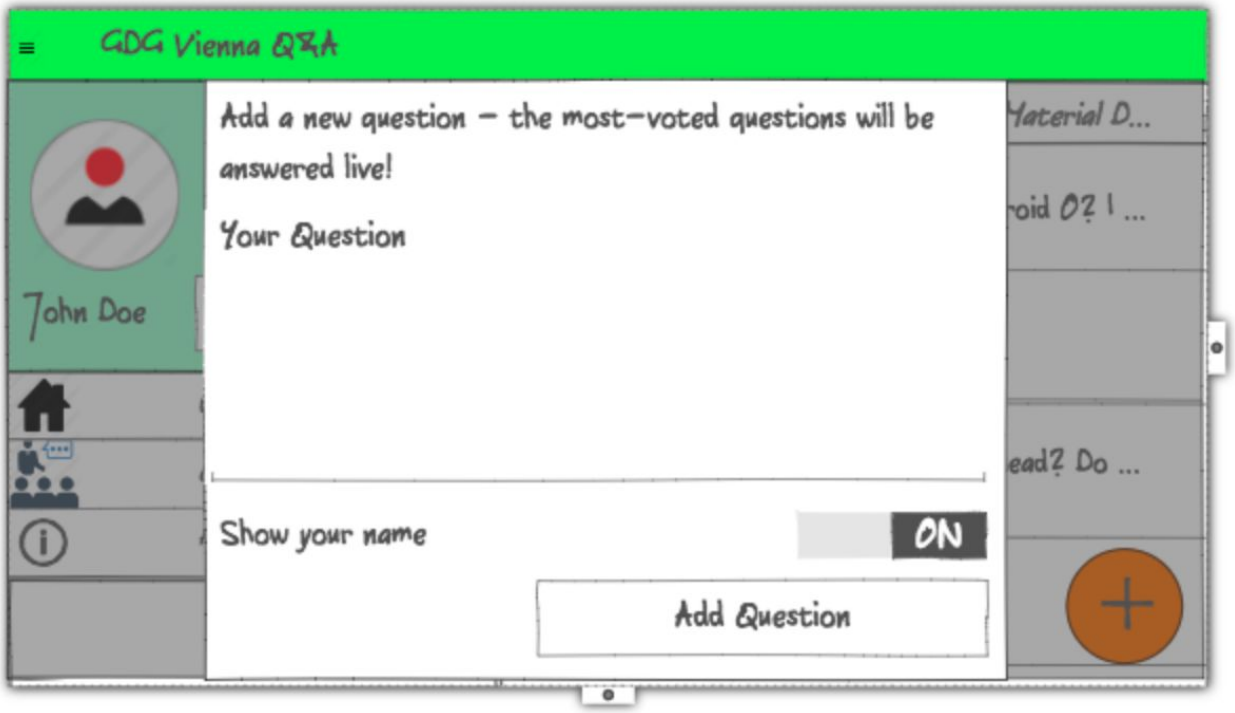
Screen 5: About

This screen shows used libraries, license, link to source code etc.

Screen 6: Tablet UI

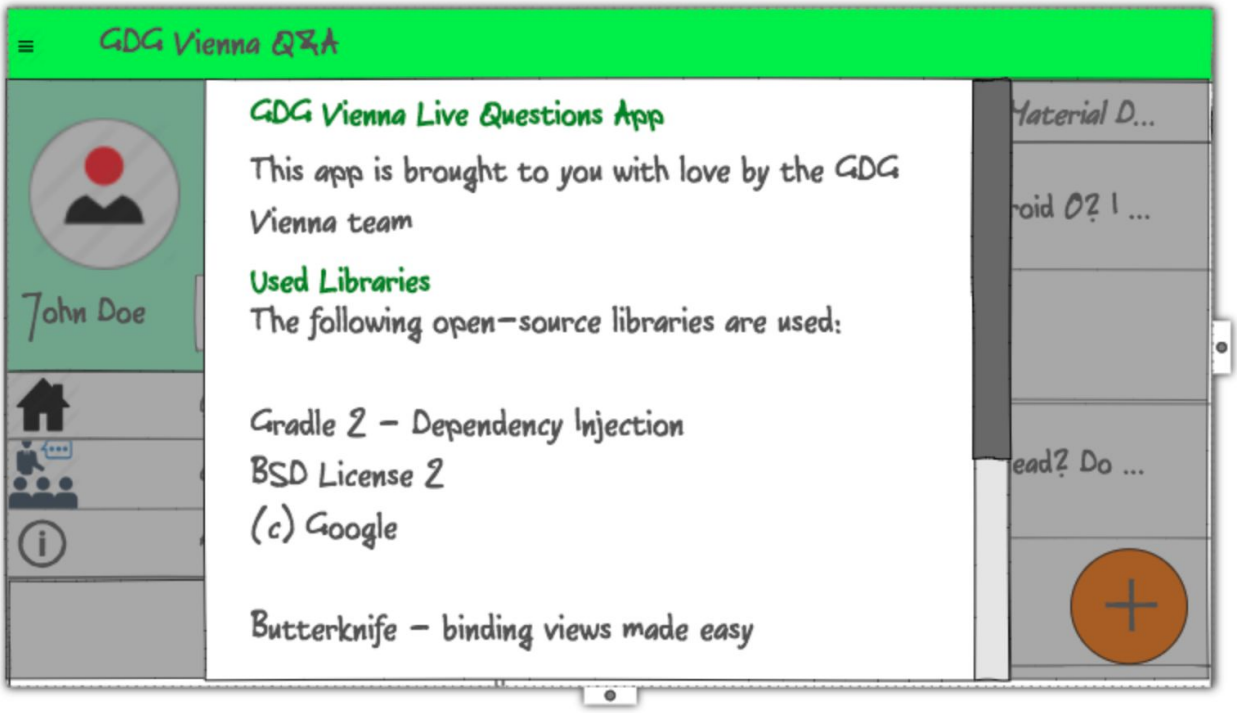
In a screen wide enough the the drawer will be shown always open.

Screen 7: New Question in Tablet UI



In the tablet UI, new questions will be added using a Dialog.

Screen 8: About Screen in Tablet UI



In the tablet UI, the About screen will be shown as a Dialog.

Screen 9: Widget Configuration

The user can add a widget to their home screen. For this they can specify the room for which they want to see the top-voted questions.
Multiple widgets can be added, allowing the user to see the questions from all rooms at a glance.

Screen 10: Widget

The widget shows the top questions from the selected room. It is scrollable. Clicking on the widget opens the activity showing the questions in the corresponding room.

Key Considerations

How will your app handle data persistence?

The app will use a Firebase Realtime Database for the questions & votes as well as the "static" information (room labels, etc).
The questions & votes will be persisted in the cloud using Firebase.

Describe any edge or corner cases in the UX.

The available options depend on whether the user is logged in or not. It is important to make the user aware that without logging in they are missing most of the stuff.

A question asked by a user is automatically "liked" by them. However, a user can "unlike" his own question afterwards.

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

For authentication I want to use Firebase UI as it is probably the most easiest way to provide robust, good-looking & well-designed authentication.
For view-binding I will be using Butterknife as it makes live just easier.
For the overall application architecture the Architecture components from Google, and for dependency injection I plan to use Dagger 2, this should allow for an extensible and testable application architecture.
In addition I will be using support library including RecyclerView and the Design library, for making it easier to have a look matching current platform standards.
For authentication I will add Facebook & Twitter API libraries (as required by Firebase UI).

Describe how you will implement Google Play Services or other external services.

I will be using Firebase, for the following purposes:

- Data storage (Firebase Realtime Database)
- Analytics
- Authentication

Next Steps: Required Tasks

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

Task 1: Project Setup

I will be using Android Studio, latest stable release. The values will be the suggested default values (Min-SDK) and no activity for the beginning.

Firebase cloud setup:

- Setup Firebase project (console.firebase.google.com)
- Setup Facebook, GitHub & Twitter apps as authentication backends
- Configure Firebase authentication

Android setup:

- Configure library dependencies
- Design logo / launcher assets
- Link App with Firebase project in Android Studio

Task 2: Implement UI for Each Activity and Fragment

Here I will create the basic layout XML files needed for the UI screens and dialogs.

- UI for MainActivity
- Drawer UI
- UI for Questions page
- UI for Ask Question
- UI for About page
- UI for Widget
- UI for Widget Configuration

Task 3: Firebase Database

Here I will design the Firebase database used. The idea is to store static information (room labels) in another JSON, the Firebase database only has dynamic information (questions asked, votes) which can be added & voted for by everyone who is authenticated.

The output of this task is a database instance with some sample data, and the security rules needed to support the requirements.

Task 4: First Activity

I will add a first activity to the project, showing the main screen and integrating the drawer menu (all non-functional for the beginning).

Task 5: Authentication

I will add authentication to the app, using Firebase UI. The drawer & main activity will be updated to show / hide relevant elements depending on the authentication state.

Task 6: Access Firebase Realtime Database (using AsyncTask)

I plan to implement the data architecture along the lines of
https://developer.android.com/topic/libraries/architecture/guide.html

Since the scope of the application is small, my idea is to stay with one single shared ViewModel for the application.

Most of the functionality will be provided by Firebase and its SDK. To show the items from Firebase, I will implement an Adapter which will:

- listen on updated data to be available (will be notified by Firebase)
- start an AsyncTask to sort the results, and to identify the changed vs. added entries

Task 7: Create Question List

Here I will create the Fragment to show the list of currently asked questions, using a PagerTabStrip and ViewPager to show the different rooms.

Depending on the login status this will allow the user to up-vote existing questions and also show the ones they have voted for already.

Task 8: Create "Add Question Dialog"

In this step I will create the Fragment to add a new question and to store it in the Firebase Realtime Database.

Task 9: About Page, Splash Activity

I will add an about page and a splash-screen activity. Start time performance will help me decide whether it will be a Placeholder UI or a Branded Launch Screen.

Task 10: Tablet refinements

Here I will modify the activity to show the two-pane setup for tablets and to show the New-Question-UI and About-Screen as a dialog.

Task 11: Widget & Widget Configuration

Finally I will implement the widget and the widget configuration.

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

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"