Mobile Application Development
2024-25

A travel sharing application

Lab5 Persist information with Google Firestore and Manage Feedback via Notifications

Learning objectives

- Connecting to Cloud Firestore database and performing remote operations
- Using Cloud Firestore Security Rules to manage permissions
- Using Sign-in with Google to properly authenticate with Firebase
- Choose a data model suitable for retrieving information according to the app s requirements
- Appling MVVM architecture
- Define the final information architecture of the app

Description

From the Lab1 description:

"A travel sharing app is designed to help users find and participate in experiences proposed/organized by other travelers (no agencies) and connect with travel buddies. This app enables users to discover (explore and filter) and apply to participate in travel experiences organized by other travelers as well as allowing them to propose their travel in turn. Registered users can apply to join trips either individually or for multiple spots. The platform also allows users to manage their published travels, their applications, post reviews and tips about their experiences. Furthermore, users can access past trips log and review travel companions. Lastly, users can receive notifications about last-minute published trips, application received to their published trips, and their current applications status.

The goal of this app is to ***connect travelers and facilitate the discovery, organization, and sharing of community-based travel experiences."***

In this Lab assignment you are asked to implement the following UIs/features:

- **User Authentication and Profiles**
  - User registration and login functionalities
  - Personal profile management (done in Lab2)

- **Feedback via Notifications regarding**

  - Last-minute travel proposal
  - Recommended trips
  - Application (to their proposals) new or pending management
  - Status update on their pending application
  - Review received

The app needs to ensure data persistence, requiring the use of a database. Since much of this data should be accessible to multiple users, a cloud-based database is a suitable option.

In this lab, the objectives are to integrate storage and retrieval of information using Google Cloud Firestore and to implement user authentication. It s possible to obtain efficient and secure support for authentication using the Credential Manager API.

The Google ID helper library allows implementation within the Credential Manager of *Sign in with Google* button, which leads to a sign-in process that follows the industry-standard OAuth protocol for authorization.

Steps

0. To accept the assignment, in Git Hub Classroom as you did in Lab4, use the link provided into Submission rules.

1. Set up **Firestore** in your project (also check slide deck ***a13*** in course materials)

    a. A Google account is required, so create a new one for your group (or use an existing one among your team members.

    b. Create a Firebase Project and Firebase support for the app. Remind that you can use the AndroidStudio integrated wizard for setup. Full instructions for configuring the project can be found here: https://firebase.google.com/docs/android/setup

        i. During step 2 Register your app with Firebase , even if it s generally optional, add information for **Debug signing certificate SHA-1** , since *SHA-1 hash* is required by Firebase Authentication (when using Google Sign In)

2. Define the appropriate **data model(s)** (if you haven t already) and implement data storage and retrieval, linking operations on your model to the data stored in the database

    a. An example of key entities you might need to model include Travel Proposal, Travel activities related to a travel proposal, and Participants to travel proposals

    b. Now is the time to apply the reasoning from Lab 1, where you were asked to conceptualize a possible ER diagram for the entities involved in your system.

    c. Bear In mind the NoSQL nature of FirebaseDB, so model for schema-less collections, and carefully decide when de-normalizing.

3. Set up **Sign in with Google** with Jetpack Compose

    a. Follow the instructions from the official documentation: https://developer.android.com/identity/sign-in/credential-manager-siwg
    https://firebase.google.com/docs/auth/android/google-signin

        i. In the OAuth consent screen section choose *External* as User Type

        ii. Creating an OAuth consent screen page, you can also add specific *scope* you need to access from your application (through Google ID token)

        iii. Add a test user using your new email address

        iv. For depency
        androidx.credentials:credentials
        use latest version available here:
        https://developer.android.com/jetpack/androidx/releases/credentials

        v. For dependency
        com.google.android.libraries.identity.googleid:googleid
        use latest version available here:
        https://mvnrepository.com/artifact/com.google.android.libraries.identity.googleid/googleid

        vi. In *GetGoogleIdOption.Builder()*.setServerClientId(getString(R.string.default_web_client_id))

        Or more generically *.setServerClientId(WEB_CLIENT_ID)* use as *WEB_CLIENT_ID* the **OAuth client ID** from the second OAuth client generated following the guide (the Web application type one) https://console.cloud.google.com/apis/credentials

    b. Implement the sign-in/registration screen you need to support the design defined in Lab1 and enable clickable widgets to behave accordingly.

        i. If necessary, revise and integrate sub-navigation graphs to support it.

4. Remember that you can use Firebase Cloud Storage, to persist large sized files (e.g. pictures of profile and reviews, documents, video)

    a. https://firebase.google.com/docs/storage

    b. In case the free-use dataplan is no more available you can host media data (foto, video) in alternative services like
    Storage | Supabase Docs https://supabase.com/docs/guides/storage
    (Kotlin SDK available
    https://supabase.com/docs/guides/getting-started/tutorials/with-kotlin)
    or a custom FTP self-hosted server.

    c. Useful links (Supabase):

        i. QuickTutorial (Kotlin integrated):
        https://supabase.com/docs/guides/getting-started/quickstarts/kotlin

        ii. Connecting Supabase to Firebase:
        https://supabase.com/docs/guides/database/extensions/wrappers/firebase

            1. Google Authentication:
            https://supabase.com/docs/guides/getting-started/tutorials/with-kotlin#set-up-google-authentication

            2. Build Config:
            https://supabase.com/docs/guides/getting-started/tutorials/with-kotlin#read-and-set-value-to-buildconfig

            3. Dependencies:
            https://supabase.com/docs/guides/getting-started/tutorials/with-kotlin#set-up-supabase-dependencies

            4. Hilt Injection:
            https://supabase.com/docs/guides/getting-started/tutorials/with-kotlin#set-up-hilt-for-dependency-injection

            5. …

    d. YouTube tutorials:

        i. Getting started with Android and Supabase
        https://youtu.be/_iXUVJ6HTHU

        ii. Integrate Supabase In Android App with Jetpack Compose - Android Studio
        https://youtu.be/OwlTKWGSImU

5. Set up and **implement the notifications system.** Implement all the notification features required **above**

    a. Notifications shall at a very least appear in some dedicated view you defined in Lab1.

    b. Carefully consider which is the trigger for the notifications:

        i. E.g. based on action made by some user
            1. Last-minute travel proposal
            2. Application (to their proposals) new or pending management
            3. Review received
            4. Status update on their pending application

        ii. Or e.g. based on some time-related policies
            1. Recommended trips
            2. Last-minute travel proposal

        iii. Or as you may have noticed, some belong to both behaviors

    c. All of (or a subset of) these notifications should be prompted to the users by:
        i. At a very least exploiting snackbars https://developer.android.com/develop/ui/compose/components/snackbar
        ii. [Optional] Using true Push Notification, via the Firebase FCM (firebase cloud messaging), instead of snackbars.
            1. https://medium.com/nybles/sending-push-notifications-by-using-firebase-cloud-messaging-249aa34f4f4c
            2. https://firebase.blog/posts/2023/08/adding-fcm-to-jetpack-compose-app/
            3. https://firebase.google.com/docs/cloud-messaging/android/client

    d. Clicking on the notification shall navigate the user to the corresponding screen/data using the routes defined in the navigation graph potentially leveraging explicit deep link https://developer.android.com/guide/navigation/design/deep-link

        i. [Optional] Ideally that should be implemented via implicit deep linking https://developer.android.com/training/app-links and NavDeepLinkRequest. Have a look at the AndroidStudio App Link assistant https://developer.android.com/studio/write/app-link-indexing to speed up the process.

    e. A **short video** demonstrating the notifications implemented must be included in the assignment submission.

        i. Use app like that one to record screen of real device https://play.google.com/store/apps/details?id=com.hecorat.screenrecorder.free&pcampaignid=web_share

        ii. Or OBS https://obsproject.com/ for recording the emulator

6. If necessary, review your choices on information architecture and integrate all previously developed parts of the app (Lab2-4)

    a. Extension features proposed by the group in Lab1 will be asked to implement for the final assignment. Nonetheless, you may consider to already implementing the necessary data models into the backend.

7. As you did during Lab 1, upload documentation to the repository, including some screenshots showing the main screens of your application.

Submission rules

- The work must be submitted by
    - EVEN groups June 9th, 5:00 pm
    - https://classroom.github.com/a/K6wnJurl

- In addition to the android studio project, you are requested to upload the video of point 5.e and an additional **short video (max 3 minutes)** demonstrating the firebase configuration. The firebase video shall include
    - The firestore Database configuration and dashboard: data (show all the collections, and a few representative entries for each of them), rules and indexes.
    - The authentication dashboard: users, sign-in method
    - Firestore storage dashboard (if used).

- The last commit before the deadline will be evaluated. Alternatively, create a release and label it completed.
- The use of third-party libraries (e.g. to implement custom/advanced widgets) **must be approved** by using the procedure at https://forms.gle/7cHANzcruvnZjZgM7 (you can also access a list of already evaluated libs)

LLMs Disclaimer:

For completing this assignment, the following uses of LLMs are authorized/forbidden:

- Authorized use:
    - Digest of documentation
    - Suggestions about widgets to use for a given purpose (from a description)
    - Suggestions about code architecture
    - Placeholder Data Generation
    - Help with debugging code, by uploading error messages and excerpts of code (about 10/15 lines)
    - Vibe Coding the draft of UI from sketches or descriptions. Also, code related to CredentialManagerAPI is allowed to be vibecoded.

- Forbidden Use:
    - Vibe Coding the draft of UI from sketches or descriptions
    - Any other use not authorized above

During evaluation we will check the code with tools able to detect vibe coding and LLMs generated text. Any misuse not communicated may affect the final mark.

Should you interact with some LLMs for the activity related to carrying out the assignment, the chat(s) must be shared (via link provided by platforms) and link included (full print of the chatpage otherwise, if not possible) in a pdf file “chatbot_interactions.pdf” which must be included in the GitHub commit(s).