Mobile Application Development 2023-24

A teamwork management application

Lab5 – Persist information with Google Firestore and manage user login

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, exploiting the navigation feature

Description

The complete app specification from the Lab1 description:
"*A teamwork management app is designed to facilitate collaboration, communication, and coordination among team members to enhance users' overall productivity and efficiency.*
By using this app, a user may manage her/his participation in one or more teams, get and set information about tasks to be performed by team members, document her/his own progress and achievements, report contributed efforts, and gather feedback from other team members as well as analytic information derived from collected data.

The goal is to streamline communication, enhance collaboration, and improve overall team performance by providing an easy-to-use app that keeps track of all the details of the team activity.

The app will support the following features:
- *User* ***Authentication*** *and Profiles*
    - ***User registration and login functionalities***
    - *Personal profile management*

- *Managing teams*
    - *Ability to create, edit, and delete teams*
    - *Sending invitations to new members via deep links and/or QR-Codes*
    - *Accessing the list of **team members**, **showing profile pictures and roles***
    - *Option to send direct messages to other team members or to the team as a whole*
    - *Ability to set the degree of one's participation in a team or to withdraw from it*
- *Managing tasks of a team*
    - *Ability to create, edit, and delegate tasks to team members*
    - *Ability to define recurring tasks*
    - *Task categorization and tagging for easy identification*
    - *Real-time task status tracking (e.g., pending, in-progress, on-hold, completed, overdue)*
    - *Filter and sort options for easy navigation*
- *Task Details:*
    - *Detailed view of a selected task*
    - *Task title, description, assigned team member(s), and due date*
    - *Task state and history*
    - *Comments section for team communication*
    - *Attachments and links to related documents*
- *Feedback and Performance:*
    - *Performance metrics and analytics for individual team members*
    - *Visual representations of team achievements*"

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

1. To accept the assignment, use GitHub Classroom as you did in previous labs: https://classroom.github.com/a/8qmHTW81

2. Set up **Firestore** in your project

    a. A Google account is required, so create a new one for your group.

   b. Create a *Firebase Project* and Firebase support for the app.
      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)

3. Define the appropriate **data model** (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 Team, Message, and Task

   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

4. 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

      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(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)

    b. Design a *sign-in pane* to allow unauthenticated users to start using the application

    c. Define appropriate *access rules* on Cloud Firestore to ensure users can only access their data

5. If necessary, review your choices on **information architecture** and integrate all previously developed parts of the app

6. 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 June, 17 23:59
- The design and the code of the user interface will be evaluated.
- The last commit before the deadline will be evaluated. Alternatively, create a release and label it “completed”.