CMPS 312 Project Phase 2 – Data Layer using Firestore, Firebase Storage and local SQLite Database (10% of the course grade)

Submission is due by 8am Thursday 12 December 2024. Demos will be organized on the same day.

1. Deliverables

In this phase, you will extend the phase 1 solution to manage the app data using Firestore, Firebase Cloud Store and local SQLite Database. The shared data that changes overtime need to be stored and managed online such as Customer, Invoice, Payment, Cheque, Cheque Deposit. The remaining static data (data that rarely change such as *Cheques Return reasons*) should be stored and managed in a local database to improve performance and reduce the cost of using Firestore.

Your project phase 2 deliverables include:

Part 1 - Cloud Firestore Database Design and Implementation

1. Design Cloud Firestore database to manage the App data.
2. Implement the repository methods to read/write entities using Cloud Firestore as the data source. All **data filtering should be done on the server** using Cloud Firestore queries and only the required data should be retrieved.
   Also, it is important to ensure that the list of entities (such as Customer, Invoice, Payment and Cheque Deposit) is properly refreshed after add/update/delete operations.
3. Enhance add/update cheque payment to allow the user to attach a cheque image from the gallery or take an photo using the Camera.
   When adding/updating the cheque payment to Firestore you need to upload the associated cheque file to Firebase Cloud Storage. Also, when you delete a cheque you should delete its associated image file from Firebase Cloud Storage.

   Also, when viewing the cheque image it should be downloaded from Firebase Cloud Storage.
4. Initialize the Firestore database: when the app starts, check if Firestore database is empty then initialize it with data from *the json files under the **assets** folder*.
5. Document your database design in a schema diagram.

Important notes: When adding entities (e.g., a Customer) to Firestore you should let Firestore auto-set the entity id (e.g., auto-assign the **customerId** when adding a Customer).

Part 2 - Manage static data using local SQLite

1. Implement the needed entity annotations to manage static data in a local SQLite database using Floor package.
2. Implement the Data Access Objects (DAO) and repositories to read/write from SQLite using Floor package.
3. If the database is empty, populate the database with the data from the json files under the assets folder.
4. Document your database design in a schema diagram.

Part 3 – Signup and Signin using Firebase Authentication

Implement signup and authentication using Firebase Authentication. Upon sign-up a user account should be created on Firebase Auth (using the user email and password) and the user details should be stored in **users** collection in Firestore.

Design and Testing Documentation

- Firestore database schema diagram and SQLite database schema diagram.
- Write a testing document including screenshots of conducted tests illustrating a working implementation.
- Every team member should submit a description of their project contribution. Every team member should participate in the solution demo and answer questions during the demo.

Important notes:

- Continue posting your questions to the Questions channel on Teams.

- Do not forget to submit your design and testing document (in Word format) and fill the *Functionality* column of the grading sheet provided in the phase 2 Word template.

- Push your implementation and documentation to your group GitHub repository as you make progress.

- You need to test as you go!

- Seek further clarification about the requirements/deliverables online and during office hours.

2. Grading rubric

Criteria	**%**	**Functionality***	**Quality of the implementation**
1) Cloud Firestore DB Design and Implementation (including Repositories)	60		
2) Manage static data using local SQLite (including Repositories)	20		
3) Signup and Signin using Firebase Authentication	10		
4) Design and Testing Documentation - Firestore database schema diagram and SQLite database schema diagram. - Testing documentation: with evidence of working implementation using snapshots illustrating the results of your solution testing (you must use the provided template).	10		
6) **Discussion of the project contribution** of each team member [-10pts if not done]			
Total	100		
Copying and/or plagiarism or not being able to explain or answer questions about the implementation	-100		

* **Possible grading for functionality** - ***Working*** (get 70% of the assigned grade), ***Not working*** (lose 40% of assigned grade and ***Not done*** (get 0). The remaining grade is assigned to the quality of the implementation.

In case your implementation is not working then 40% of the grade will be lost and the remaining 60% will be determined based on of the code quality and how close your solution to the working implementation.

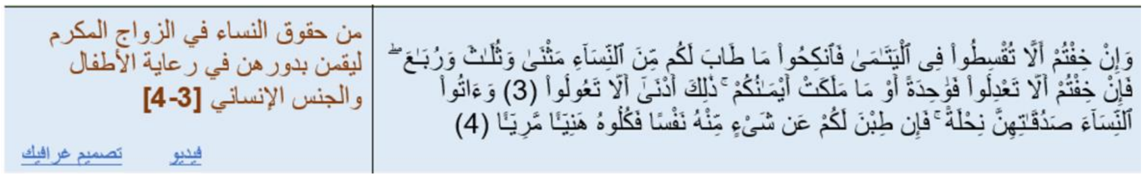
Solution quality also includes meaningful naming of identifiers (according to Flutter/Dart naming conventions), no redundant code, simple and efficient design, clean implementation without unnecessary files/code, use of comments where necessary, proper code formatting and indentation.

Marks will be reduced for code duplication, poor/inefficient coding practices, poor naming of identifiers, unclean/untidy submission, and unnecessary complex/poor user interface design.

Basair Project

For Basair project the same deliverables are expected as explained above. All Basair data should be managed in a local SQLite database except the Mihwar resources are stored and managed in a Firestore database.

- If the database is empty, populate the database with the data from the json files under the assets folder or using data fetched from online resources.
- Provide the ability to add/update/delete resources associated with a Mihwar. The resource could be a link, a PDF document, YouTube video or an image such as a graphical illustration. The resource details should be stored and managed in a Cloud Firestore Database. The images and PDF documents should be uploaded to Firebase Cloud Storage.
- In the Quran Viewer allow access to the resources associated with the Mihwar.



من حقوق النساء في الزواج المكرم ليقمن بدورهن في رعاية الأطفال والجنس الإنساني [4-3] تصميم غرافيك فيديو	وَإِنْ خِفْتُمْ أَلَّا تُقْسِطُوا۟ فِى ٱلْيَتَـٰمَىٰ فَٱنكِحُوا۟ مَا طَابَ لَكُم مِّنَ ٱلنِّسَآءِ مَثْنَىٰ وَثُلَـٰثَ وَرُبَـٰعَ ۖ فَإِنْ خِفْتُمْ أَلَّا تَعْدِلُوا۟ فَوَٰحِدَةً أَوْ مَا مَلَكَتْ أَيْمَـٰنُكُمْ ۚ ذَٰلِكَ أَدْنَىٰٓ أَلَّا تَعُولُوا۟ (3) وَءَاتُوا۟ ٱلنِّسَآءَ صَدُقَـٰتِهِنَّ نِحْلَةً ۚ فَإِن طِبْنَ لَكُمْ عَن شَىْءٍ مِّنْهُ نَفْسًا فَكُلُوهُ هَنِيٓـًٔا مَّرِيٓـًٔا (4)

- Signin is required when the user want to add/update/delete resources associated with a Mihwar. Implement signup and authentication using Firebase Authentication. Upon sign-up a user account should be created on Firebase Auth (using the user email and password) and the user details should be stored in **users** collection in Firestore.