CMPS 312 Project Phase 1 – Design and Implementation using on-device data store (10% of the course grade).

The project phase 1 submission is due by 8am Sunday 2nd November 2025. Demos will be organized during office hours in the same week.

1. Requirements

You are requested to design and implement *Lingo* app to aid language learning for both the teachers and learners. The app consists of ***Learning Package Editor*** used by the teachers preferably on a Tablet to create learning packages assign them to student and the ***Lingo App*** used by the students for game-like learning activities.

- *Learning Package Editor* allows the teacher to provide a list of words and associated word definitions and sentence examples. Words and/or sentences in the package can have associated photos, videos, or web links.
- *Lingo App* allows the learner to search and download learning packages. Then the app uses the downloaded package to provide content to the interactive learning activities to study and practice the content of the learning package.

For phase 1, data should be managed in a SQLite database and multimedia files are simply GitHub links. In phase 2, you will use a remote Cloud database (such as Supabase or Firestore) for reading/writing the app data and a remote cloud storage to manage the multimedia files. You will also use the device camera to take photos and videos and upload them to the remote storage. Additionally, you will use an LLM API to facilitates content generation.
The main Lingo use cases are described Table 1.

Table 1. Use cases description

Use case	Brief description
Lingo App	
U1 - List and search learning packages	Get a list of learning packages with the ability to search by keyword or level. In phase 1, you can use json files or SQLite as the data source. Once the list is displayed the learner can select the desired one to download and use as content for the learning games of U2, U3 and U4. If the user is logged in, the user can select to edit or delete a package (only if the package was created by them). Or they can add a learning package.
U2 - Play Flash Cards	Play Flash Cards based on the package content with the ability to loop through them and play/display multimedia content associated with words or sentences (e.g., image, video, or web link) within the app

	using Audio/Video Player and WebView.
U3 - Unscramble Sentences	The learner needs to reorder the words of a sentence to form a meaningful sentence. The user can loop through and play this game for all sentences in the package. The app should validate the user attempt and indicate whether it was successful or not.
U4 - Match Word & Definition	Given a pool of words and their definitions, the learner should match a word and its corresponding definition. The user can loop through and play this game based on the words and the definitions in the package. The app should validate the user attempt and indicate whether it was successful or not.
Learning Package Editor	
U5 – Login	Allow the teacher to login. Login is prerequisite for all the teacher use cases (U6 and U7). No need for a signup screen instead use the provided `users.json` to initialize the `users` table used for authentication.
U6 - Delete Learning Package	The teacher can delete a learning package he/she has created earlier (but should NOT be able to delete packages they haven't authored). The App should ask for the user confirmation before deleting.
U7 - Add/Update Learning Package. This use case includes:	Teacher can add/update the package details as shown the package entity.
U7.1 – Add/Update/Delete Words	Add/Update/Delete the words to be included in the learning package.
U7.2 - Add/Update/Delete Definitions	Add/update/delete one or many definitions for each word.
U7.3 - Add/Update/Delete Sentences	Add/update/delete one or many sentences for each word.
U7.4 - Attach Multimedia	The teacher can attach multimedia content (photos, videos or web links) to a sentence. In phase 1, these will simply be GitHub links to multimedia files. In phase 2, the teacher can attach photos and videos either recorded using the phone's camera or selected from the image gallery.
U7.5 – Save and Publish the Learning Package	Save the package to a Json file saved locally or to a SQLite database. In phase 2 the learning packages will be saved to a cloud database to enable shared access.

The entities class diagram is shown in Figure 1. These are the base entities provided as a guide you may update them or enhance them as needed.

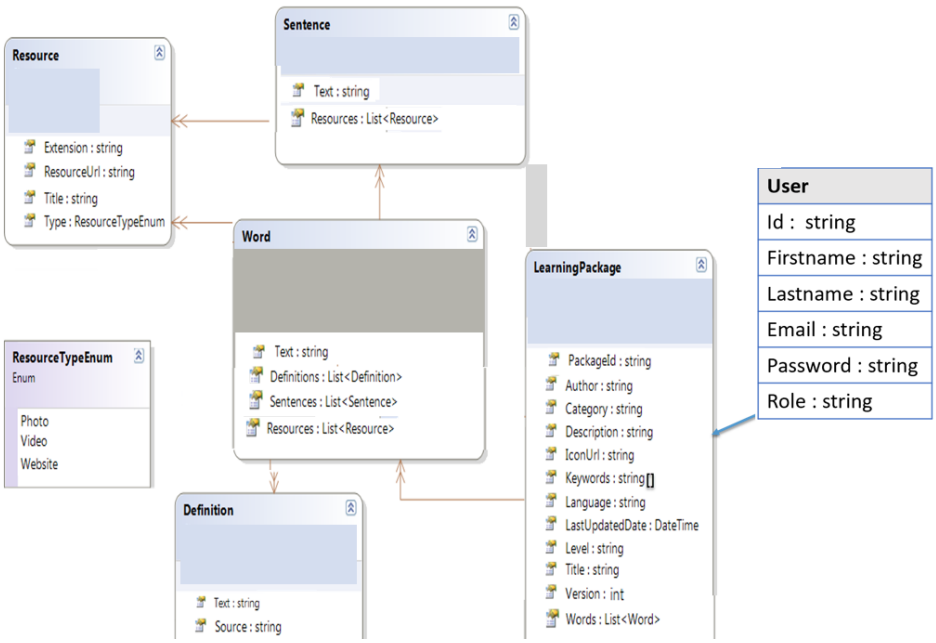

Figure 1. Lingo entities class diagram

Important notes about the entities class diagram:

- The class properties should be lowercase as per Flutter naming convention. They are shown in uppercase in the diagram by the tool used to create this diagram.
- LearningPackage.packageId should be auto assigned and not entered by the user
- LearningPackage.version should be auto-incremented every time the package is updated.
- The learning package should be read/written as an aggregation that could be serialized as single json document.

2. Deliverables

Seek further clarification about the requirements/deliverables during the initial progress meeting with the instructor. Note that further important clarifications maybe modified/added to the project requirements.

1) Application design documentation that includes UI design and the Repositories class diagram. **During the weekly project meetings with the instructor, you are required to present and discuss your design and get feedback**.
2) Implement **responsive UI** for each use case following design best practices. The UI should be fully working using learning package data loaded from a json file and/or SQLite database. Remember that 'there is elegance in simplicity'!
3) Design and implement the app navigation. It should be fully working, and the user can navigate from one activity to another in intuitive and user-friendly way.

Formatted: Font: Bold, Complex Script Font: Bold, Highlight

4) Implement the entities and repositories using Dart. They should be fully working. Create some test Learning Package data to ease testing. First test them using a main function that displays the results to the console before using them in the UI.
5) Document the testing of UI and repositories using screen shots illustrating the testing results.
6) Every team member should submit a description of their project contribution. Every team member should demo their work and answer questions during the demo.

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

3. Grading rubric

Criteria	%	Functionality*	Quality of the implementation
1) Entities & Repositories Class Diagram.	5		
2) Design and implementation	**90**		
U1 - List and search learning packages	8		
U2 - Play Flash Cards	10		
U3 - Unscramble Sentences	10		
U4 - Match Word & Definition	10		
U5 - Login	5		
U6 - Delete Learning Package	5		
U7 - Add/Update Learning Package, including:	10		
U7.1 – Add/Update/Delete Words	8		
U7.2 - Add/Update/Delete Definitions	8		
U7.3 - Add/Update/Delete Sentences	8		
U7.4 - Attach Multimedia (as Web links)	8		
5) Testing documentation using screen shots illustrating the testing of UI and Repositories.	5		
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 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.