CMPS 312 Project Phase 1 – Design and Implementation (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 حِكَايَتِي (Hikayati) an interactive story reading app that supports children's language learning through engaging, multimedia storytelling. It enables teachers to create stories enriched with text, images, and comprehension quizzes, assisted by AI-generated content and Text-to-Speech (TTS) narrations in English, French, and Arabic.

Learners can browse and download stories, listen with synchronized text highlighting, and answer quizzes to assess their understanding. By combining storytelling, audio narration, and AI assistance, Hikayati makes reading more interactive, enjoyable, and effective for multilingual learning.

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 facilitate content generation.
The main Hikayati use cases are described Table 1.

Table 1. Use cases description

Use Case	Brief Description
S-U1: List & Search Stories	The learner can list available stories and searches by title, language (English, French, or Arabic), or reading level (KG1, KG2, Year 1, Year 2, Year 3, Year 4, Year 5, Year 6).
S-U2: Download Story Package	The learner downloads the selected story package, which includes text, images, audio, and quiz data, for offline use.
S-U3: Play Story (Read-Along)	The learner listens to a story while the app highlights text word-by-word or phrase-by-phrase in sync with the audio narration.
S-U4: Take MCQ Quiz	The learner answers multiple-choice questions related to the story to test comprehension and receives instant feedback.
T-U5: Teacher Login	The teacher logs in to access story creation, editing, and publishing features.
T-U6: Create or Edit Story	The teacher writes or edits story text, a story may have sections, adds images to story sections, adds or replaces audio, sets metadata (title, author, language, level, keywork), and prepares the story for learners. The teacher can attach, update, or remove images and audio files associated with the story sections.
T-U7: Add or Edit MCQs	The teacher creates or modifies multiple-choice questions linked to each story for learner assessment.
T-U8: Upload / Publish Story	The completed story package, including text, images, audio, and quizzes, can then be uploaded to the server for learner access.

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.

Phase 1 — Required (Minimum Viable Product)

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 local database (e.g., SQLite) or JSON repository for story, quiz, and user data. 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 screen to another in intuitive and user-friendly way.
4) Implement the entities and repositories using Dart. They should be fully working. Create some test story package including the story text, images, and quiz 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.

Phase 2 — Enhancements

1. **Cloud Integration**
   - Replace local data storage with a cloud database (e.g., Firebase or Supabase).
   - Enable teachers to upload media files directly from the device.
2. **Full LLM and TTS Integration**
   - T-U9: LLM-Based Content Suggestion: The app connects to live LLM APIs for AI-assisted story images and MCQ generation.
   - T-U10: Generate natural, high-quality narration in English, French, and Arabic using a TTS service API.
3. **Read-Along Synchronization**
   - Implement accurate word or phrase synchronization using timestamps or speech marks.
   - Provide accessibility options (e.g., text size adjustment, color contrast).
4. **Offline Mode and Localization**
   - Allow learners to access downloaded stories and audio offline.
   - Support RTL layout for Arabic UI.

3. Grading rubric

Criteria	%	**Functionality***	Quality of the implementation
1) Entities & Repositories Class Diagram.	5		

- Correctness and completeness of class relationships. - Proper use of naming conventions and entity-repository design.			
2) Design and implementation	**90**		
S-U1: List & Search Stories	12		
S-U2: Download Story Package	8		
S-U3: Play Story (Read-Along)	8		
S-U4: Take MCQ Quiz	15		
T-U5: Teacher Login	5		
T-U6: Create or Edit Story	20		
T-U7: Add or Edit MCQs	15		
T-U8: Upload / Publish Story	7		
3) Testing documentation with evidence using screen shots illustrating the testing of UI and Repositories.	5		
4) **Project contribution** and individual responsibilities of each team member clearly documented [-10pts if not done]			
Total	100		
Copying and/or plagiarism or not being able to explain or answer questions about the implementation	-100		

Grading Notes

- **Working Functionality:** A feature that works end-to-end earns up to **70%** of its allocated marks for functionality; the remaining **30%** is awarded for overall solution quality.
- **Partially Working Features:** Receive a **40% penalty** for that item. The remaining **60%** will be based on how close the solution is to full functionality and on its quality.
- **Not Implemented:** Awarded **0 marks**.

Solution Quality considers:

- Modularity, simplicity, and robustness of the design.
- Clear and intuitive UI/UX.
- Meaningful and consistent naming of identifiers (following Flutter conventions).
- Clean, efficient implementation without redundancy.
- Proper use of comments, formatting, and indentation.

Marks will be reduced for:

- Overly complex or poorly designed user interfaces.
- Code duplication or poor coding practices.
- Weak or inconsistent naming conventions.
- Messy, disorganized submissions.