



CMPS 312 Project Phase 2 – Data Layer using Supabase Database and Storage (10% of the course grade)



Submission is due by **8am Thursday 4th December 2025**.

1. Deliverables

In this phase, you will extend phase 1 solution to manage the app data and files using Supabase Database and Storage. Your project phase 2 deliverables include:

1. Database Design and SQL Scripts:

- Design the database schema and write SQL scripts to create all required tables in Supabase. Document the DB schema using an ERD.
- Enable Row-Level Security (RLS) to protect user data.
- Use Supabase auto-generated primary keys for all entities (e.g., packageId).

2. Database Initialization: Initialize the database either through in-app code using JSON seed files or through a SQL initialization script.

3. Repository Implementation:

- Implement repositories for reading and writing all entities using Supabase as the data source.
- Perform all filtering and querying on the server side and return only the necessary data.
- Ensure entity lists (e.g., list of packages) automatically refresh after create, update, or delete operations.

4. User Authentication: Implement sign-in using Supabase Authentication with Email & password and an authentication provider (e.g., GitHub or Google).

5. Sentence Multimedia (Images & Videos):

- Extend add/update sentences editor to allow selecting or capturing images/videos.
- Upload multimedia files to Supabase Storage.
- Delete related multimedia files when a sentence is removed.
- Update the app to fetch and display/play sentence multimedia from Supabase Storage.

6. [5pts optional bonus if fully working] AI Sentence Suggestions: Enhance the add/edit word sentence feature by integrating an LLM API (e.g., Gemini API) to auto-generate and display suggested example sentences for a given word.

Design and Testing Documentation

- Supabase 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 Supabase DB Design and Implementation (including Repositories)	45		
2) Database Initialization	5		
3) Signin using Supabase Authentication including Email & password and an authentication provider (e.g., GitHub or Google)	15		
4) Sentence Multimedia: attach images or videos to a sentence and upload them to Supabase Storage. Then use them when accessing sentence resources.	25		
6) Design and Testing Documentation - Supabase 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]			
[5pts optional bonus if fully working, no partial grading] AI Sentence Suggestions: enhance add/edit word sentences by suggesting example sentences returned by an LLM API			
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**.