Tribhuvan University

Faculty of Humanities and Social Sciences

A Project report on:
CardsQL
- A flashcard revision/quiz web-app

Submitted to:

Department of Computer Application,

Himalaya College of Engineering,

Chyasal,Lalitpur

In partial fulfillment of the requirements for the Bachelors in Computer Application

Submitted by:

Sujal Gurung 6-2-378-82-2020

Nischal Karki 6-2-378-67-2020

December 24, 2023

Under the Supervision of

Er. Himal Chand Thapa

Tribhuvan University

Faculty of Humanities and Social Sciences

Himalaya college of Engineering

LETTER OF APPROVAL

This is to certify that this project prepared by Sujal Gurung, Nischal Karki entitled “**CardsQL”** in partial fulfillment of the requirements for the degree of bachelor’s in computer application has been evaluated. In our opinion it is satisfactory in scope and quality as a project for the required degree.

………………………………….. **SIGNATURE of Supervisor** Himal Chand Thapa Lecturer Himalaya College of Engineering Chyasal, Lalitpur	**……………………………………** **SIGNATURE of HOD/ Coordinator** Himal Chand Thapa Coordinator Himalaya College of Engineering Chysal, Lalitpur
………………………………… **Internal Examiner**	………………………………… **External Examiner**

Abstract

Traditional study techniques of just reading through books, course materials are **passive** and don't engage our minds too much. Students may not be able to retain studied information for long periods of time, due to the way our brains work.

Flashcards are an alternative solution for retaining information by remembering facts or answers to questions. The reasoning being that retreiving previously learned information from our memory engages our brains, strengthens the neural connections to that info and thus, we can remember it easier the next time.

This reflects real world scenarios like exams and tests how much of our learning we actually remember. Practicing flashcards from time to time can be a simple but effective form of revision. CardsQL implements this in an easy-to-use web interface with some additional quality of life features.

Keywords: efficient learning, flashcards, revision, self-hosted, web-app

Acknowledgement

We are grateful to the Department of Computer Application at Himalaya College and Tribhuvan University for giving us the opportunity to work on this project as a partial fulfillment of our Bachelors studies. Special thanks go to some of our friends and seniors for assisting us in our times of need. Lastly, we would like to extend our gratitude towards our project Supervisor, Coordinator and Examiners for taking the time to go through our work, evaluating it and providing feedback.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Introduction

Flashcards are a simple way for students to revise topics. On a piece of paper, we write a question on one side and then the answer on the next side. The following is an example of what a flashcard might look like:

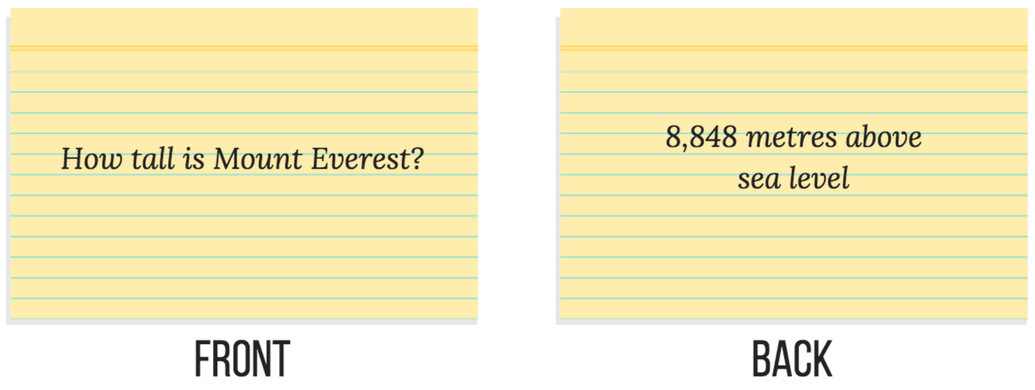


Figure 1.1.1: Example Flashcard

To utilize this flashcard, we would read the question & try to remember the answer. Then, we flip the card over & see how well we remembered it. CardsQL is a web app that digitises this technique. The name is derived from a combination of Flashcards & SQL. It allows you to create cards by entering the text on the front side (usually, the question) and the text on the back(the answer). Each card is stored in a database along with some metadata. Users can then practice/review created cards, where CardsQL will show only the question at first and prompt the user to remember the answer. After attempting to remember it, users can press a button to reveal the answer.

Users can then rate how well they remembered it, and this rating is sent to an algorithm to decide when the card should be reviewed again. This algorithm, along with a few scientific theories related to making such a revision technique effective, is discussed in Chapter 2. Thus, instead of re-reading the same study materials, users can just create cards once and study them regularly, while retaining that knowledge for much longer. The proof for flashcarcds allowing for longer memory retention is also discussed in that chapter.

CardsQL maintains review dates for cards so there is no need to study every card everyday. It sets review date such that when we revise a card on that day, it isn't too hard nor is it too easy. As such, users olny need to use CardsQL a little bit every day to revise effectively.

1.2 Problem Statement

- **inefficient traditional studying techniques**
- **inconvenient to manage physical flashcards**
  - Physical flashcards will eventually get damaged or wear out. It may also be impractical to store or carry around a large number of cards.
  - Storing cards digitally allows them to be physically secure & portable.
- **cumbersome to schedule, keep track of next repition for cards**
  - After reviewing/ recalling a card, we need to schedule when to practice it again. We not only need to keep track of its schedule, but also remember to review it on that day. If we forget to do so, then keeping track of overdue cards adds another layer of unneeded complexity.

1.3 Objectives

- **Create an easy to use flashcard program.**
- **Promote regular study habit among students by making it effortless and quick to do.**
- **Improve rate of retaining studied knowledge among users**

1.4 Scope and Limitaiton

1.4.1 Scope

CardsQL would be appropriate for the following types of users:

- Students ranging from high schooolers to Doctorates, as they have to understand and memorize a lot of concepts.
- People learning a new language. Applications like **Dulingo, Anki** that use question-answer based learning are popularly used for this purpose.
- People, in general, that want to retain their knowledge.

1.4.2 Limitation

- Cards are stored locally on users' computer so there are no cloud backup or multi-device synchronization features. Users can implement it manually if they wish using services like **Dropbox** or **Syncthing**.
- Initial setup process of installing dependencies and running a web server from the command line may be jarring for non-technical users.
- Databse backups haven't been implemented due to time constraints.
- Currently, all flashcards are stored in a single table in the database. This may become unorganized after a large number of cards are added.

1.5 Report Organization

The project is explained across 5 major chapters in this report. This section is part of Chapter 1, which serves as an introduction to the project. It explains what CardsQl basically does and the probelsm it sets out to achieve. The chapter also talks about suitable users for CardsQL and some of its current limitations to get a better understanding of the project. Chapter 2 describes some theories, scientific concepts, and an algorithm related to the project. These provide a scientific basis for the effectiveness of CardsQL in enhancing learning. It also includes an analysis of two similar existing products, and lists out their advantages and disadvantages that differ it from this project.

Chapter 3 goes through the results of the initial planning and design phase of the project, such as identifying requirements and figuring out various designs for making CardsQL work. These phases were carried out before starting development on the project so that development would go smoothly later. Chapter 4 similarly describes how the plans were implemented and briefly goes through the inner workings of the project. It also lists some test measures put in place to ensure that the application works smoothly and users don't face any hindrances. Chapter 5 concludes this report, talking about the objectives it meets and future improvements that can be made.

2 Background Study and Literature Review

2.1 Background Study

2.1.1 Active Recall

Cognitive researchers have found that trying to recall facts strengthens the relevant neural connections in our brain & thus, allows us to remember it for longer periods of time. This process is specifically called **Active Recall** & is proven to be more effective than passive studying[1].

2.1.2 Spaced Repetition

Hermann Ebbinghaus, a German psychologist, concluded after extensive research that as time passes, our ability to rememeber a piece of information slowly decreases[2]. He called this the forgetting curve[2].

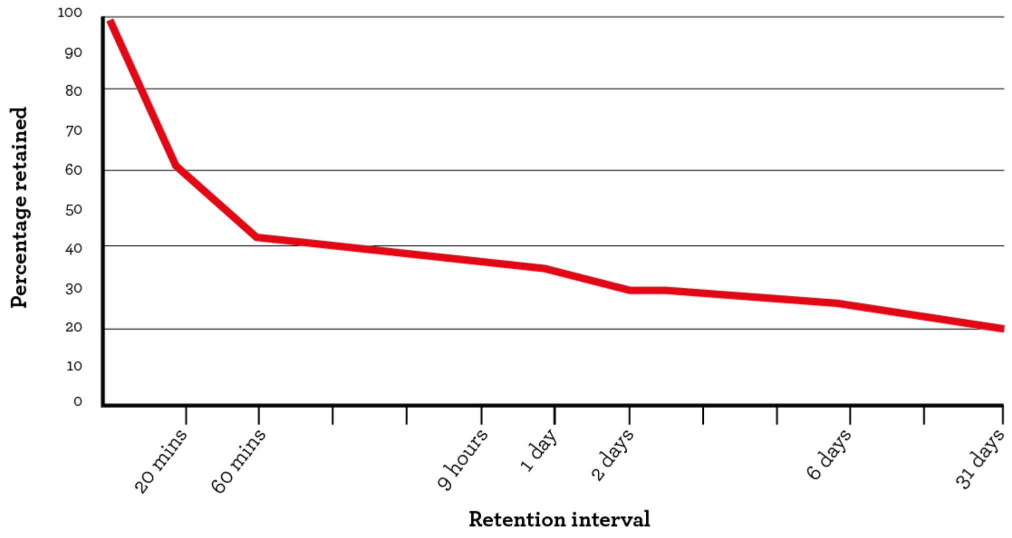


Figure 2.1.1: Ebbinghaus' forgetting curve

Ebbinghaus discovered that performing Active Recall at increasing time intervals would increase memory retention & thus, counter the forgetting curve. This is termed as **Spaced Repetition**[2].

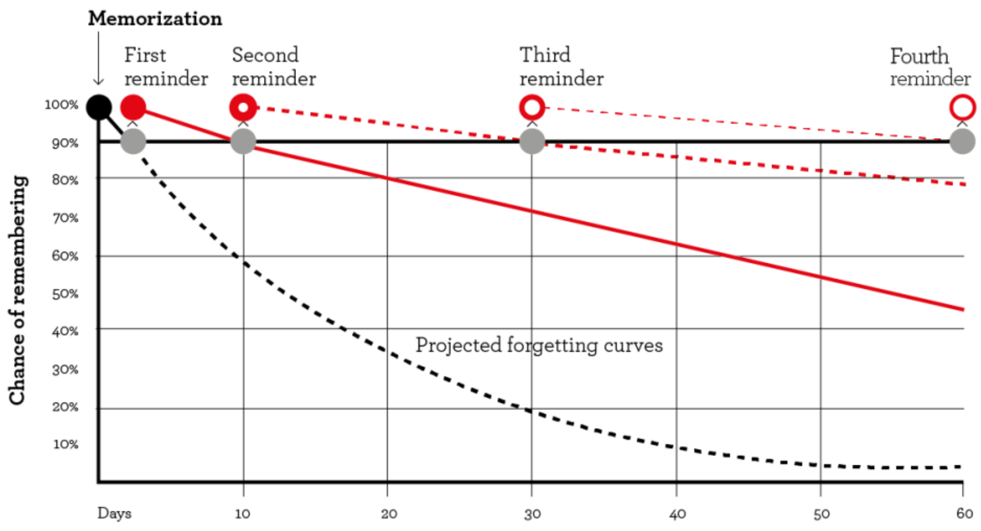


Figure 2.1.2: Ebbinghaus' forgetting curve countered by Spaced Repition

2.1.3 SM-2 algorithm

SuperMemo is a learning software that implements the afore-mentioned methods. It uses an underlying algorithm for calculating when the next repetition/ revision date for a flashcard should be. The algorithm differs across versions but the SM-2 algorithm[3] is popular among other flashcard software. A slightly modified version of SM-2 (used in CardsQL) is described below.

While practicing flashcards, users can rate how well they were able to remember the answer. The rating/ quality of a review is represented by `Q`.

Table 1: Meanings of flashcard rating values

rating(Q)	meaning
0	Forgot
1	Partially remembered
2	Remembered after some effort
3	Remembered easily

SM-2 also tracks three properties for all cards:

- The repetition number `N`, which is the consecutive number of times the card has been successfully recalled (meaning $Q \geq 2$). Rating a card below 2 will thus reset `N` to 0.
- The interval `I`, which is the number of days after which the card should be reviewed again (to negate the forgetting curve). CardsQL sets next review date = previous review date(i.e. today) + `I`.
- The easiness factor `EF`, which loosely indicates how "easy" the card is (More precisely,

it determines how quickly the interval grows). The initial value of `EF` for all cards is 2.5.

Due to the formula used, `EF` value should be `>=1.3` so that a card's review isn't scheduled too frequently and isn't too easy. Similarly, it should be `<=2.5` so that it isn't scheduled so far into the future that we've forgotten the answer completely by then.

The main algorithm is as follows:

```
input:  user rating(Q),  repetition number(N),  easiness factor(EF),  interval(I)
output: updated values of N, EF, I

if Q >= 2 (i.e. correct response) then
    if N = 0 then  I = 1
    else if N = 1 then  I = 6
    else I = round(I × EF)
    end if
    increment n
else (incorrect response)
    N = 0
    I = 1
end if

EF = EF + (0.1 − (4 − Q) × (0.09 + (4 − Q) × 0.03))
if EF < 1.3 then  EF = 1.3
else if EF > 2.5 then  EF = 2.5
end if
return (N, EF, I)
```

These properties of a card are considered as that card's **metadata** and are stored alongside the card's contents in the database. When revising cards on a particular day, CardsQL will show you cards that are scheduled for that day or older(for overdue cards).

2.2 Literature Review

In a study from 2011[4], researchers divided students into four distinct groups, each responsible for studying the same study materials and subsequently undergoing assessments to test their understanding. The study instructed each group to study in different ways as follows:

1. The first group was tasked with a single reading of the material.

2. The second group was required to read the material **four** times.
3. The third group was instructed to read the material and then create a mind map.
4. The fourth group was directed to read the material **once** and then recall as much information as possible.

Both in the verbatim test (*where participants were asked to recall facts*), and the inference test (*designed to test understanding of concepts*), the active recall group exhibited notably superior performance compared to the other groups.

The results show that just testing ourselves once on what we learned yields better results than studying something four times, while also taking far less time to do so. This also proves the previously discussed scientific concepts. Thus, testing ourselves using flashcards is a far more effective way to study.

2.2.1 Study of existing system

Two popular flashcard apps are:

1. Quizlet
   (a) Pros
      - pre-made flashcards for subjects
      - emphasis on mobile version UX which allows users to revise anytime, anywhere
      - utilizes machine learning from anonymous user-data to create custom study plans for users

   (b) Cons
      - free version has ads & lacks advanced features
      - can't be used offline on free version
2. Anki
   (a) Pros
      - Free & Open Source Software (FOSS)
      - supports sync between multiple devices

- highly customizable with user-defined card types & community-made plugins

(b) Cons

- complex from start; CardsQL can act as gateway/ introduction to flashcards. Users can transition to Anki later
- might have to spend a lot of time customizing the program, adding plugins, to get a good experience

3 System Analysis and Design

3.1 System Analysis

3.1.1 Requirement Analysis

1. Functional requirements

   Note: *As CardsQL is meant for personal use, it doesn't have admin, multiple users etc.*

   - can add cards
   - can revise due cards
   - can edit text & review date of existing cards

2. Use Case diagram

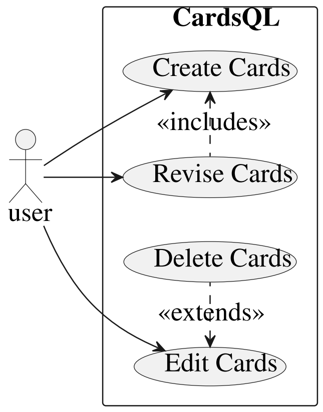


Figure 3.1.1: Use case diagram for CardsQL

3. Non-Functional requirements

   - **offline access to all features**

     This is achieved by both hosting php server & storing data on user's computer. Users don't need an internet connection after downloading the app files and setting up php and sqlite dependencies.

   - **simple to use**

     The first thing a user sees is the card creation interface. Interface for reviewing and editing cards is also straightforward with suitable form labels and button text that instruct user what to do.

- **Abstraction: User doesn't need to understand the algorithm, or how cards are stored**

  Achieved by providing simple, intuitive GUI for all user actions. All the complex factors and variables used in the `SM-2` algorithm are hidden and only the next scheduled date is shown, which can also be modified to their liking. SQL queries for CRUD operations have been setup so that they only need to press the appropriate buttons to make changes.

3.1.2 Feasibiliity Analysis

1. **Technical**

   CardsQL is not too difficult to implement from a technical standpoint because it uses:

   - plain HTML, CSS for the front-end
   - basic JavaScript and PHP for the busienss logic

   As there aren't any heavy web frameworks used, the application can easily run in the browser on even weak computers. A lightweight way for hosting a local web server is using php's built in `-S` command line flag. SQLite, the RDBMS used by CardsQL, also only requires a single a single database file on the user's computer so it negates the need for maintaining a server for users to connect to.

2. **Operational**

   - Because of the self-hosted architecture, the app will work offline without needing to connect to a central server. Thus, there is no need to designate manpower to keep it operational post-launch.
   - Users are sure to adopt the app as it is more convenient than paper flashcards, while still being easier to pick up and learn than the advanced programs discussed in **Study of existing system**

3. **Economic**

   CardsQl is viable from an economic standpoint as:

   - There are no additional costs for web hosting, server maintenance etc.

- There were no development costs as weonly used existing hardware & freely-licensed tools.
- The app is distributed freely to help users & doesn't have any profit incentives.

3.1.3 Data Modeling

Note: *The database is only used for storing cards along with their metadata. As it is meant to be used by a single-user , there is no need to implement or store login credentials.*

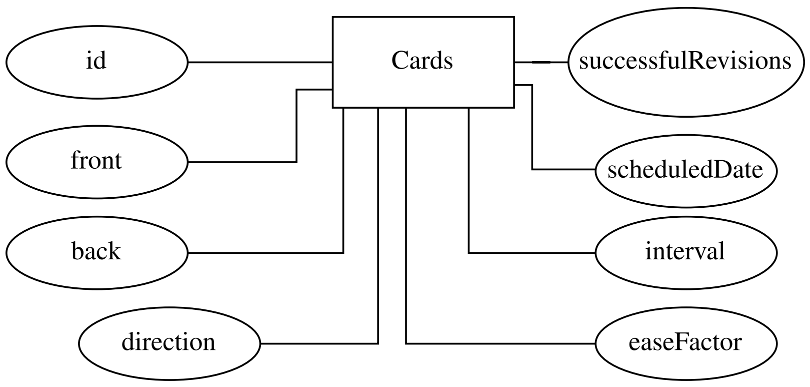


Figure 3.1.2: Entity-Relationship diagram for CardsQL

3.1.4 Proces Modeling(DFD)

The following Level 0 Data Flow Diagram/ Context Level Diagram shows a very abstract, simplified overview of CardsQL. There is only one external actor(user), who sends data to the system by adding or editing cards. Card data can be retrieved from the system and shown to the user when editing or practicing cards.

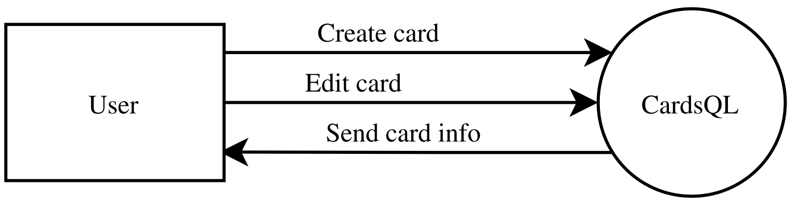


Figure 3.1.3: Context Level diagram for CardsQL

The Level 1 DFD below expands on the previous diagram by breaking down the system into a series of processes and data stores. It also shows how data flows between processes, data stores and users.

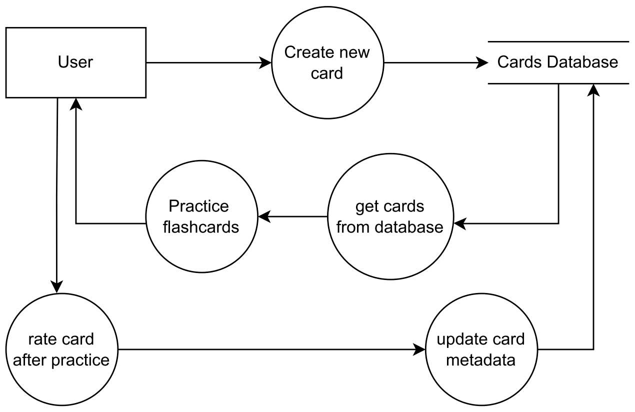


Figure 3.1.4: Level 1 Data Flow Diagram for CardsQL

3.2 System Design

3.2.1 Architectural Design

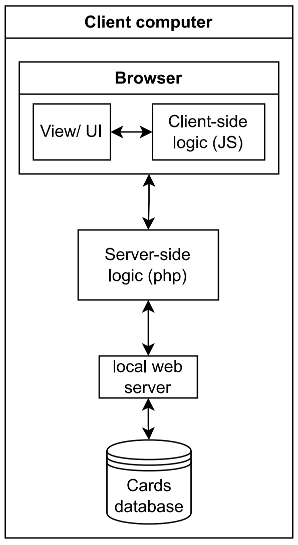


Figure 3.2.1: Architectural design for CardsQL

3.2.2 Database Schema Design

Note: *In SQLite,* `text` *and* `varchar` *data type are the same and allocate space according to value's length.* `real` *data tpye refers to floating-point numbers*

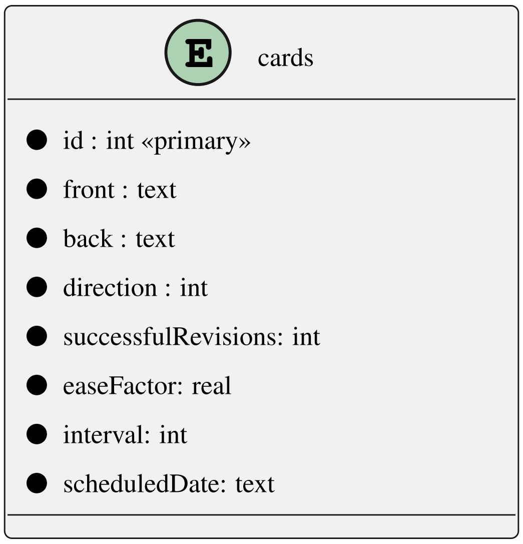


Figure 3.2.2: Database schema for CardsQL

3.2.3 Interface Design

Due to the app's simple nature, User Interface(UI) designs were not made beforehand. Styling was decided on during development, and applied after implementing correct business logic.

3.2.4 Physical DFD

The following Data Flow Diagram goes more in-depth than the previous diagram, and reflects how some processes are performed/implemented technically. The user will only see simple, easy to use forms and tables for submitting data. This data is sent to the local web server through HTTP requests and parsed by PHP code to perform operations accordingly. These operations may include inserting new cards, updating a card's text contents or metadata. All of these involve executing SQL queries on the database using submitted data. This is also done with PHP code that runs on the server.

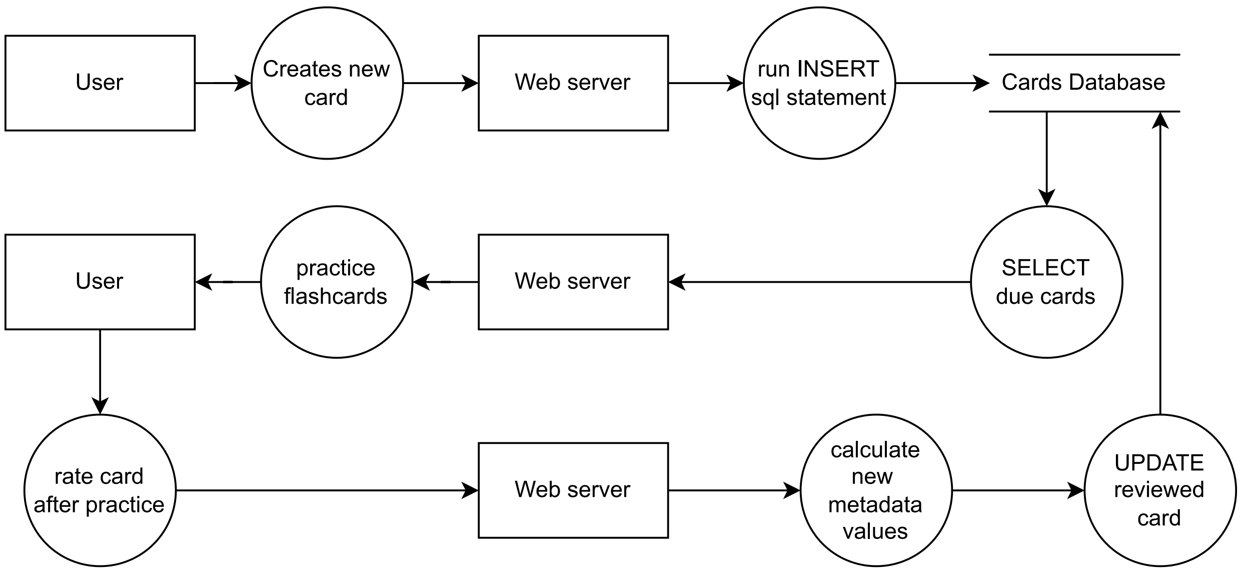


Figure 3.2.3: Physical DFD for CardsQL

4 Implementation and Testing

4.1 Implementation

4.1.1 Tools Used

The following Computer Aided Software Engineering(CASE) tools were used during the project's development:

1. Presentation layer

    - HTML was used for structuring the webpage and its contents.
    - Plain CSS, i.e. without any frameworks, was used for appplying styling and decorations.

2. Logic layer

    - JavaScript is used for client-side interactivity & DOM manipulation such as showing card answer only when button is clicked, populating edit dialog's values according to selected card etc.
    - PHP is used for server-side scripting, i.e., mainly for performing Create, Read, Update, Delete (CRUD) operations on the database. Installing the language also allows us to use its basic, local web server, which is suitable for running CardsQL. This was used during development instead of dedicated server software like Apache.

3. Data layer:

    - SQLite is a lightweight Relational Database Management System (RDBMS) used for our database purposes. It uses a single database file on the user's computer so it negates the need for maintaining an SQL server.

4. Miscellaneous:

    - Git was used for version control, while Github served as a remote backup for the code.

4.1.2 Implementation Details of Modules

The different possible actions and how they are implemented are as follows:

1. **Add flashcards**

   The homepage allows a user to add a new flashcard by specifying the text contents of its front and back side in an html form. Upon submitting the form, the contents are sent to the web server using an Hyper Text Transfer Protocol (HTTP) request. PHP code intercepts it and inserts a new row in the database table. Each row in the table represents a flashcard. Metadata values for newly created card such as the `Interval`, `EaseFactor` are set using default constraints defined in the table schema.

2. **Review flashcards**

   In the review page, php code retrieves flashcards that have a due date of today or older, one at a time. Only the question(content of front side) is shown first and the user is prompted to recall the answer. They can press a button to show the correct answer and rate how well they remembered it on a scale of 0 to 3. The rating is sent to the web server and php code uses it to update the card's metadata using the `SM-2` algorithm discussed previously. This process continues until no due cards are left.

3. **Edit flashcards**

   The edit page initially shows a table of all the cards in the database. Clicking on a row will bring up a modal dialog for editing the selected card. The dialog has a form whose values are populated with the selected card using javascript. Confirming edits will submit the form and php code will execute `UPDATE` SQL statement on the database accordingly. The dialog also has a delete button which will tell php to execute a `DELETE` SQL statement instead.

4.2 Testing

4.2.1 Test cases for Unit(Manual) Testing

The test cases are successfull because of the following measures put in place during development:

- Test 1 is successfull because of `required` attribute in HTML form and `NOT NULL` constraint in dtabase schema.

Table 2: Unit test cases for CardsQL

S.N.	Test case	Input	Result	Behave
1	Text fields should be filled while creating or editing cards	Empty	Can't submit	True
2	While practicing cards, $0 \leq$ `rating value` ≤ 3	-	-	True
3	$1.3 \leq$ `EaseFactor` ≤ 2.5 for algorithm effectivenes	-	-	True
4	Card should be shown even if `scheduledDate` has already passed	set card to past date	still shown	True

- For Test 2, form for submitting rating only has inputs with specified set of values(0-3). In case its value is somehow outside that range, the follwing php code is put in place to keep it inside the 0-3 range:

  ```
  $rating = $rating < 0 ? 0 : ($rating > 3 ? 3 : $rating);
  ```

  If rating is below, it's set to 0 and if it is above 3 it's set to 3.

- For Test 3, code similar to above is used to keep `EaseFactor` value in check.

- Test case 4 is possible due to condition while querying the database. This behavior comes in handy if a user misses reviewing a card on its scheduled date. They can just review/ practice it on a later day, and the algorithm schedules it to `x` number of days from that day.

5 Conclusion and Future Recommendation

5.1 Lesson learnt/ Outcome

From completing this project, we deepened our knowledge of HTML, CSS, Javascript and PHP languages. Some of the language features we learned were native `<dialog>` elements in HTML, using `PDO` for connecting with any type of database in PHP, etc. This knowledge will no doubt be useful for a long time in our career. Mainly, we experienced the process of turning a project idea into a Minimum Viable Product and all the associated Software Development Life Cycle (SDLC) phases. This gave me a small taste of how real-world projects are worked upon and will surely help me in our career.

5.2 Conclusion

Thus, we believe CardsQl has met the following goals it set out to achieve:

- Provide a simple introduction to using flashcards, active recall & spaced repititon for learning
- Eliminate the need to constantly read or make notes on the same topics
- Help make studying an effortless, daily habit

We hope that students and other learners can study effectively through CardsQL and be able to retain what they learn for longer periods of time. Students and programmers might also take inspiration to build their own flashcard app, either simple or complex. Building projects like this is a great way to learn new things as well as solidify previously learned programming concepts. As such, we are grateful to have completed this project as part of our coursework.

5.3 Future recommendation

Some improvements that can be made to CardsQL are:

- Add different types of cards like fill in the blanks(cloze) and list with multiple answers
- Setting direction in which to show a card:
  By default, the question is asked first and then the answer is shown. Some flashcard

software allow setting different directions like backwards direction, where, for example: a term's definition may be shown and the user has to remember the term. This provides a new way to test our knowledge.

  Currently, users can specify direction while creating cards, but it hasn't been implemented in the review page.

- Allow grouping cards by subject or topic
- Create tutorial or documentation for users to understand how to use it.
- Allow setting daily target/limit so that users are motivated to at least meet the target as well as not be overwhelmed with too many cards in a day.

6 Apendix

6.1 Screenshots

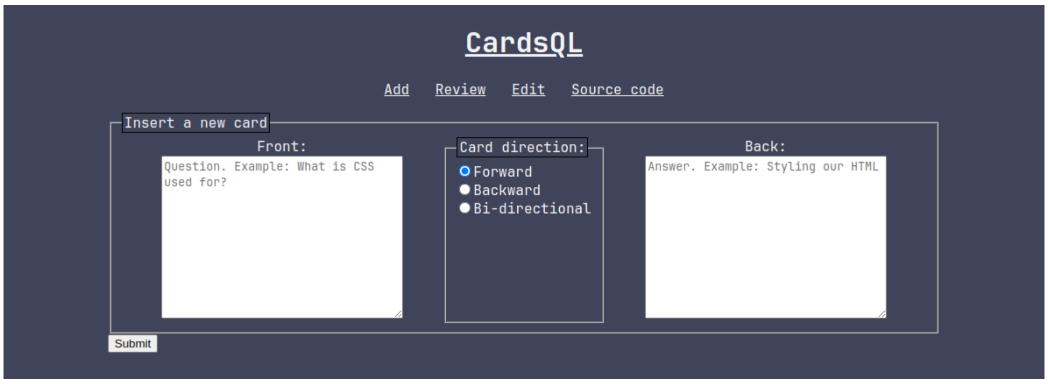
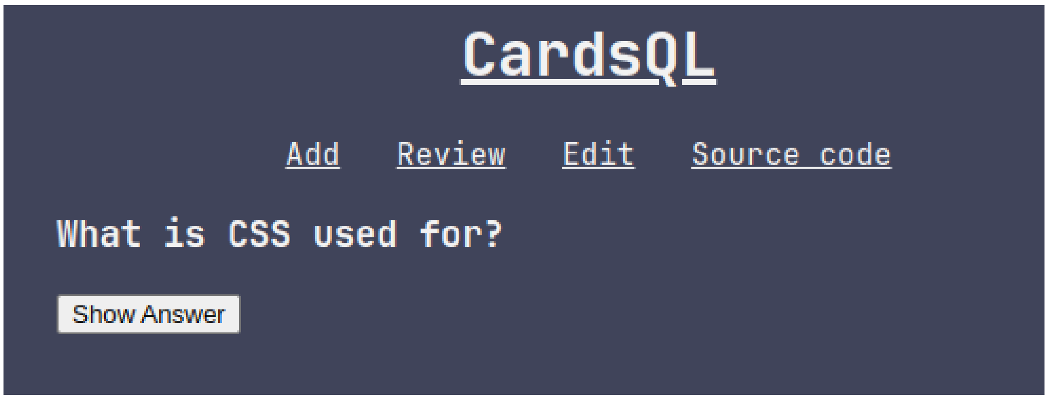
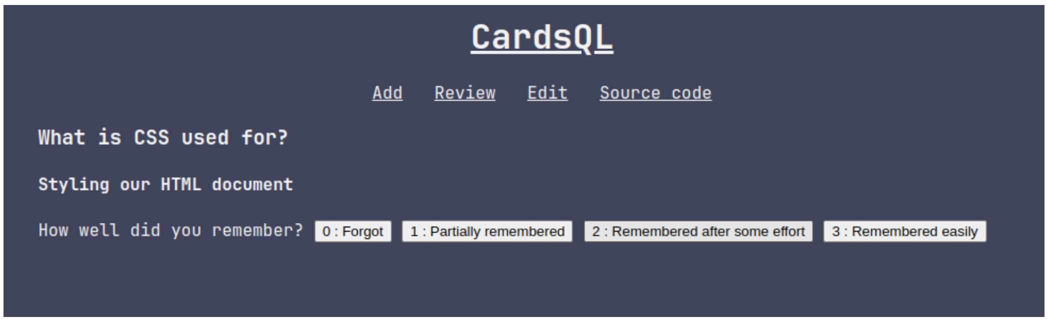


Figure 6.1.1: Interface for adding new cards



What is CSS used for?

Figure 6.1.2: Initial interface while practicing cards

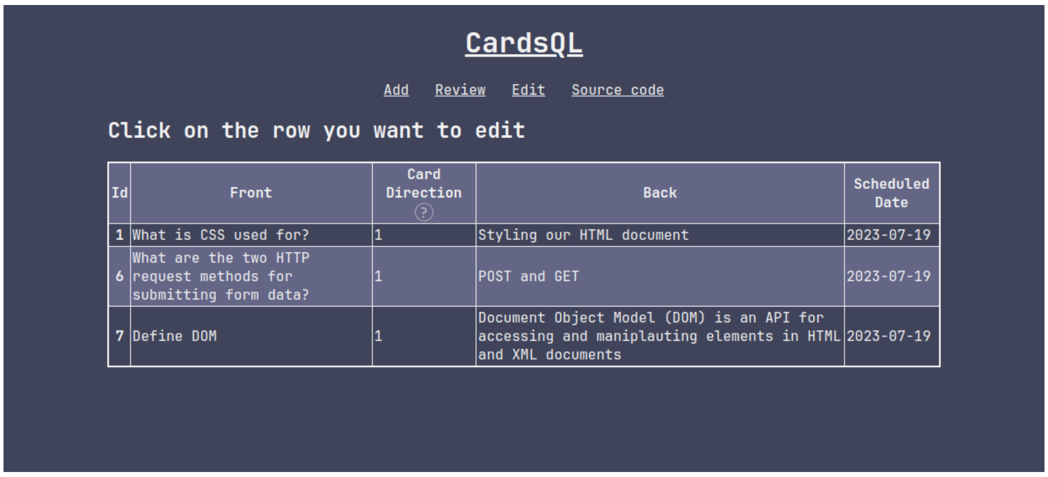


What is CSS used for?

Styling our HTML document

How well did you remember?

Figure 6.1.3: Interface for rating how well you remembered a card's answer



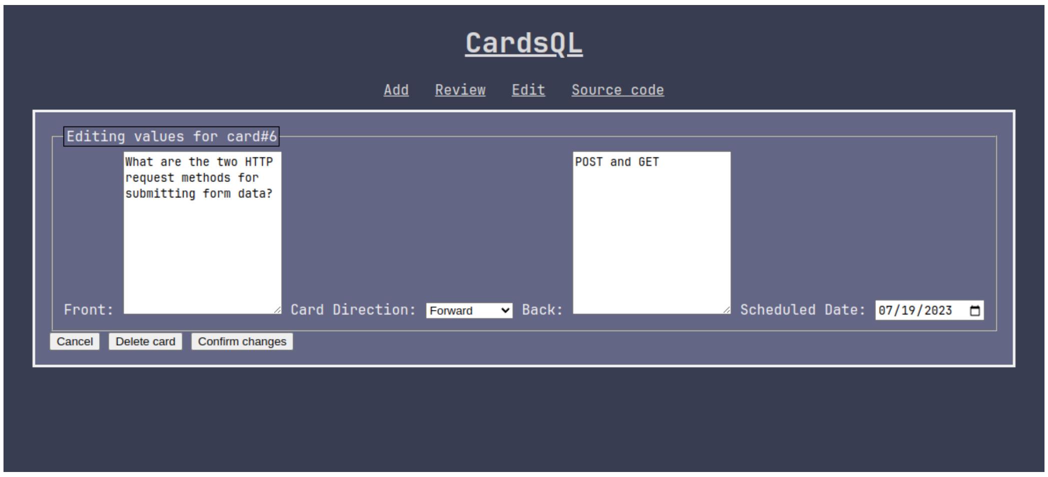
CardsQL

Add Review Edit Source code

Click on the row you want to edit

Id	Front	Card Direction ?	Back	Scheduled Date
1	What is CSS used for?	1	Styling our HTML document	2023-07-19
6	What are the two HTTP request methods for submitting form data?	1	POST and GET	2023-07-19
7	Define DOM	1	Document Object Model (DOM) is an API for accessing and maniplauting elements in HTML and XML documents	2023-07-19

Figure 6.1.4: Initial interface for selecting a card to edit



CardsQL

Add Review Edit Source code

Editing values for card#6

Front: What are the two HTTP request methods for submitting form data?

Card Direction: Forward

Back: POST and GET

Scheduled Date: 07/19/2023

Cancel Delete card Confirm changes

Figure 6.1.5: Interface for editing a card

7 References

[1] A. Abdaal, “How to study: Active recall.” https://aliabdaal.com/activerecallstudytechnique/.

[2] S. Parrish, “The spacing effect: How to improve learning and maximize retention.” https://fs.blog/spacing-effect/; Farnam Street Media Inc.

[3] P. Woźniak, “Application of a computer to improve the results obtained in working with the supermemo method.” https://www.super-memory.com/english/ol/sm2.htm.

[4] J. D. Karpicke and J. R. Blunt, “Retrieval practice produces more learning than elaborative studying with concept mapping.” https://www.science.org/doi/10.1126/science.1199327.