Tribhuvan University

Faculty of Humanities and Social Sciences

A Project report on:

CardsQL
- a flashcard revision/quiz 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

September 28, 2023

Under the Supervision of **<Supervisor name>**

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.

Acknowledgement

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. The following is an example of what a flashcard might look like:

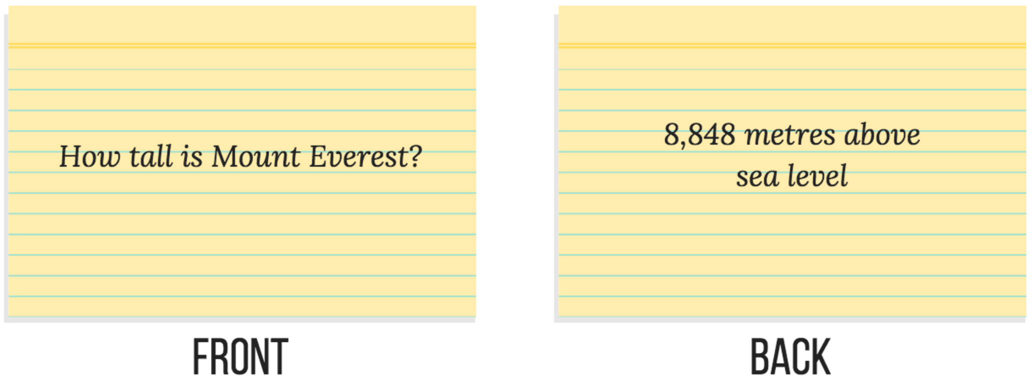

Figure 1: Example Flashcard

To utilize this flashcard, we would read the question & try to remember the answer. Then, 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.

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

- provide as easy of an entry point to flashcard programs by keeping things simple
- provide intuitive interface so that user doesn't have to understand how the system works in order to utilize it
- automate scheduling of next review. Users just have to use it & practice flashcards regularly.
- allow editing creadted cards' contents & review date

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 TODO Report Organization

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 Repitition

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.

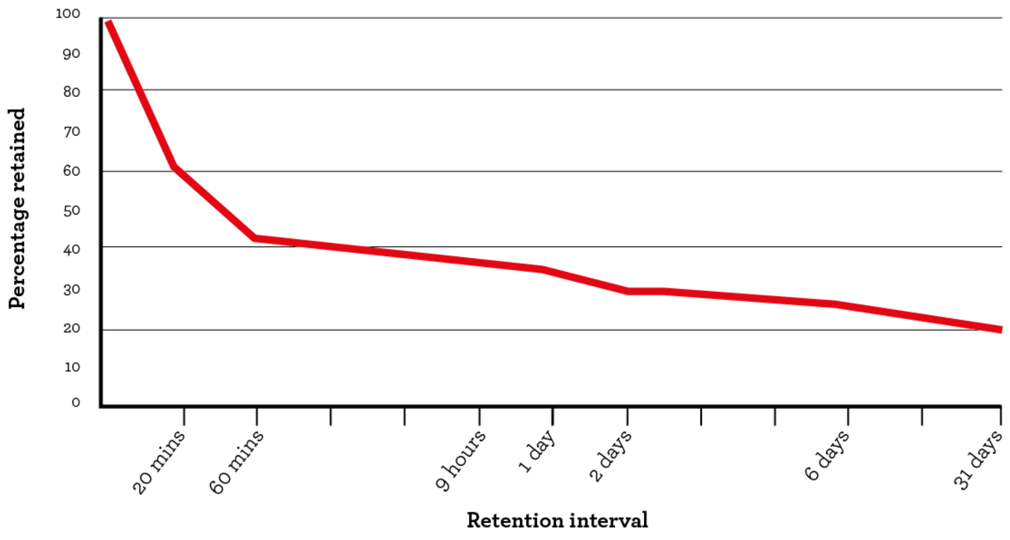

Figure 2: Ebbinghaus' forgetting curve

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

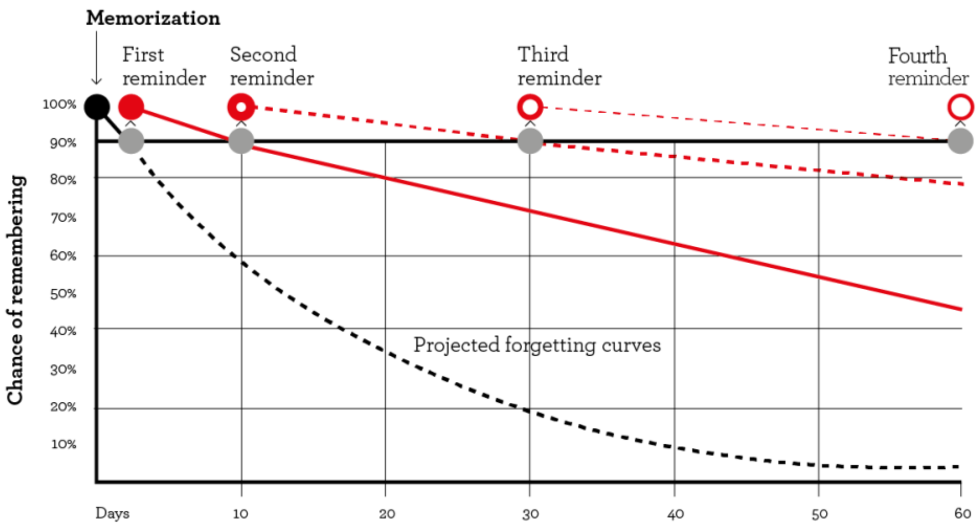

Figure 3: Ebbinghaus' forgetting curve countered by Spaced Repition

2.1.3 SM-2 algorithm[3]

SuperMemo is a learning software that implements the afore-mentioned methods. It uses an underlying algorithm for calculating when the next repitition/ revision date for a flashcard should be. The algorithm differs across versions but the SM-2 algorithm 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(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)
```

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

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 only has one type of user instead of admin, multiple users etc.*

   (a) User

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

2. Use Case diagram

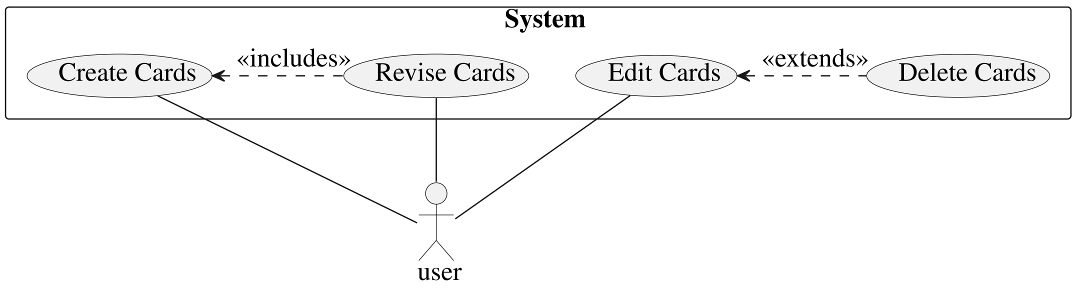

Figure 4: Use case diagram for CardsQL

3. Non-Functional requirements

   - **offline access to all features**
     Achieved by hosting php server & storing data on user's computer
   - **simple to use**
     First thing user sees is just card creation interface
   - **User shouldn't have to review too many cards in a day**
     User can set daily card goal\/limit

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 JS, PHP for the busienss logic
   - SQLite, a lightweight RDBMS, for the database. It uses 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 at all times after downloading it. Thus, there is no need to designate manpower to ensure the app stays operational after launch.

- Users are sure to adopt the app as it is designed to be more convenient than paper flashcards. Anyone should be able to learn to use it, compared to other more advanced flashcard 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 the app was built using existing hardware & freely-licensed tools.
   - The app is distributed freely to help users & doesn't have any profit incentives.

3.1.3 Data Modeling

3.1.4 Proces Modeling(DFD)

3.2 System Design

3.2.1 Architectural Design

3.2.2 Database Schema Design

3.2.3 Interface Design

3.2.4 Physical DFD

4 Implementation and Testing

4.1 Implementation

4.1.1 Tools Used

4.1.2 Implementation Details of Modules

4.2 Testing

4.2.1 Test cases for Unit Testing

4.2.2 Test cases for System Testing

5 Conclusion and Future Recommendation

5.1 Lesson learnt/ Outcome

5.2 Conclusion

5.3 Future recommendation

6 Apendix

7 References and Bibliography

[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.