Tribhuvan University

Faculty of Humanities and Social Sciences

A Project proposal on:
Project 2

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

Pawan Sharma Humagain 6-2-378-68-2020

July 27, 2024

Under the Supervision of

Er. Himal Chand Thapa

Contents

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:

diagrams/flashcard-example.png

To utilize this card, 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.

2 Problem Statement

- **inefficient traditional studying techniques**

Cognitive researchers have found that trying to recall such 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(,)**. Another technique used in tandem with Active Recall is **Spaced Repitition[[?]]**. It dictates that your ability to remember a fact deteriorates over time. By performing Active Recall at specific intervals, we can re-strengthen our memory retention.

diagrams/forgetting-curve-spaced-repitition.png

Figure 2.1: Ebbinghaus' forgetting curve countered by Spaced Repition

Thus, instead of studying entire topics repeatedly, we can just revise small key points & be more effective.

- **inconvenient to manage physical flashcards**

    - Physical cards 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 its next Spaced Repitition. We not only need to keep track of its schedule, but also remember to review it on that day. If we forget, then keeping track of overdue cards adds another layer of unneeded complexity.

    - CardsQL automates all these processes. After reviewing a card, we can rate how well we remembered the answer. This rating is used to determine the next review date for that card. This date is set such that by the next review, we haven't completely forgotten the answer, nor is it too easy.

    - When revising on a particular day, CardsQL will show you cards that are scheduled for that day or older. Thus, overdue cards are handled easily.

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

- allow creating different types of flashcards & categorizing them by subject/ tag

- automate scheduling of next review

4 Methodology

4.1 Requirement Identification

This project makes use of the **Waterfall development** approach in order to set concrete requirements & build towards achieving them.

4.1.1 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
        - utilize 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 (FOSS)
        - supports sync between multiple devices
        - highly customizable with user-defined card types & community-made plugins

    (b) Cons

        - complex from start
        - CardsQL should act as gateway/ intro to flashcards. can use Anki later
        - might have to spend a lot of time customizing the program, adding plugins, to get a great experience

4.1.2 Requirement Collection

1. Functional requirements

   Note: *As CardsQL is meant for personal use, it only has one type of user instead of admin, multiple users etc.*
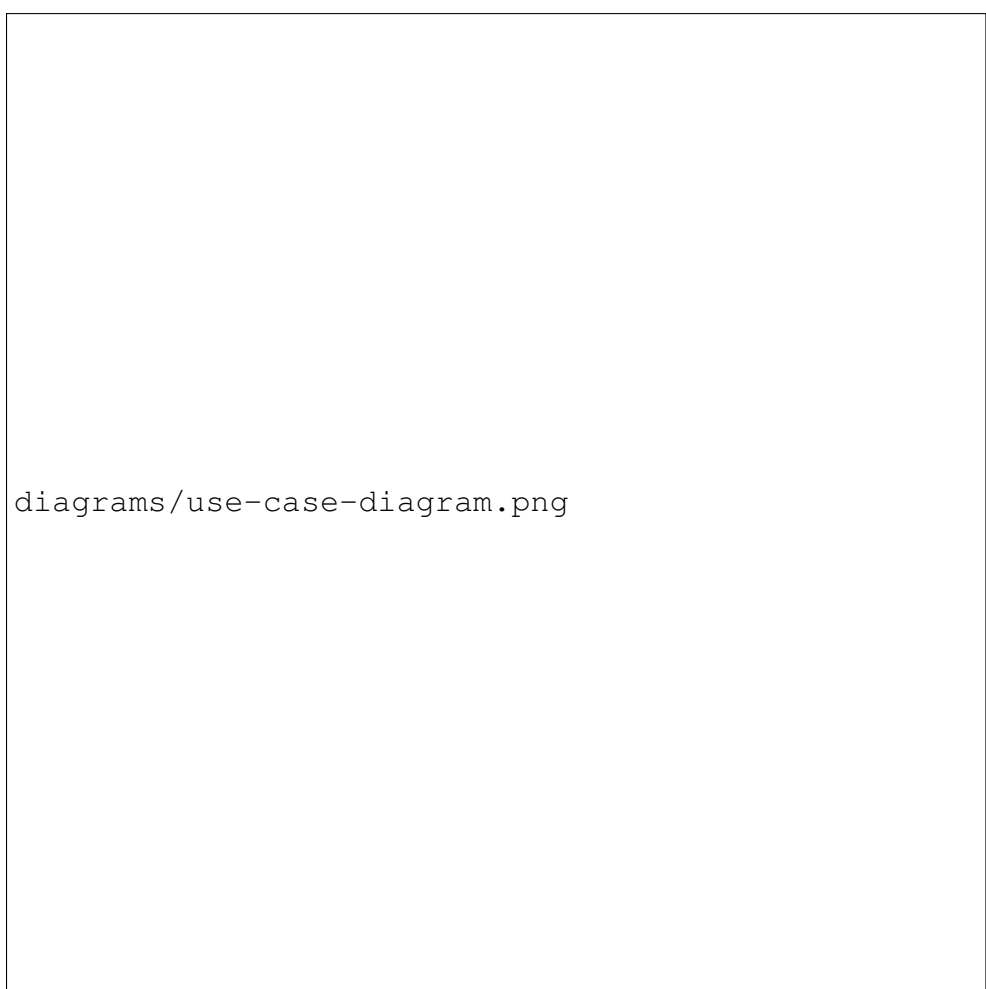
   (a) User

   - can add different types of cards

   - can revise due cards

   - can revise cards regardless of due date (for pre-exam practice)

   - can edit text, type & review date of existing cards

   - can reset review date for all cards
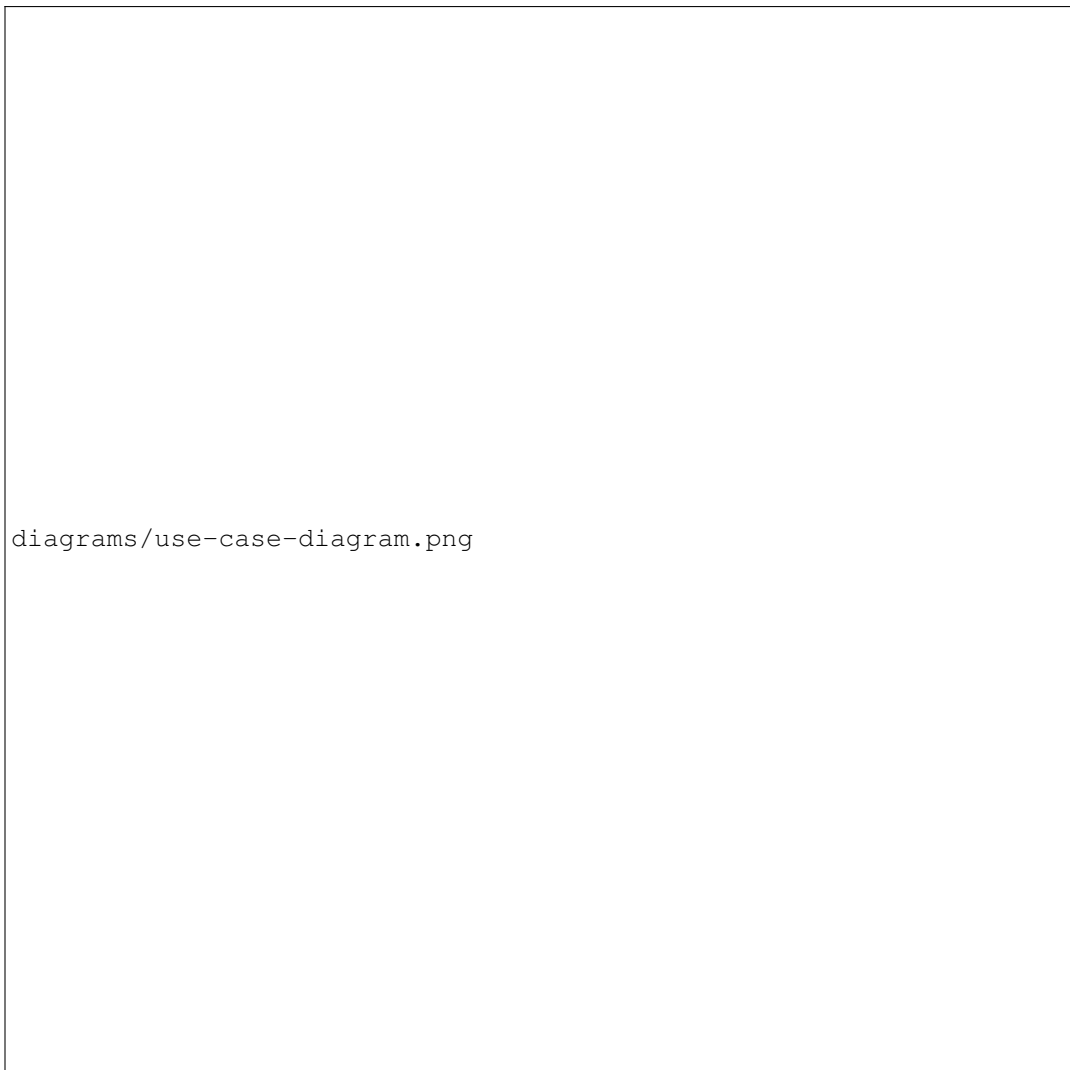
2. Non-Functional requirements

   - offline access to all features achieved by hosting sql server & storing data on user's computer

   - simple to use; 1st thing user sees is just card creation interface

   - not have too many due cards (set maximum limit)

   - regular data backups

3. Use Case diagram

diagrams/use-case-diagram.png

diagrams/use-case-diagram.png

Figure 4.1: Use case diagram for CardsQL

4.2   Feasibility Study

4.2.1   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 database file on the user's computer so it negates the need for maintaining a server for users to connect to.

4.2.2   Operational

- Because of the serverless 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

4.2.3   Economic

CardsQl is viable from an economic standpoint as:

- The project was willingly built by the devloper for free.

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

- There were no development costs as the app was builton the developer's existing hardware & using freely-licensed tools.

- The app is distributed freely to help users so there is no potential profit or loss.

4.3 High level design of System

As the following are high level representations of the system, they aim to provide a basic understanding of the system and thus, leave out intricate implementation details.

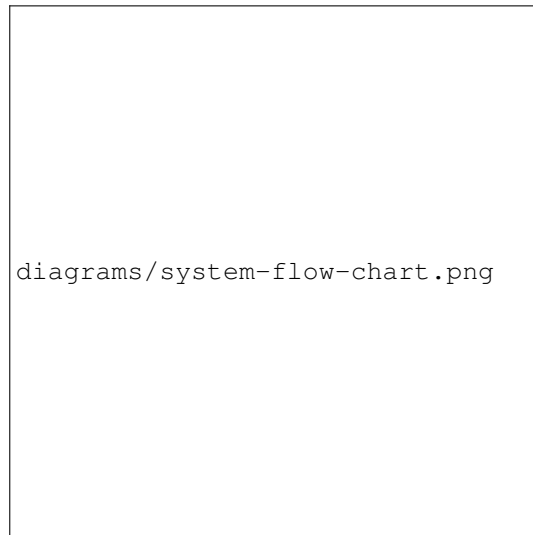### 4.3.1 System Flow Chart

```
diagrams/system-flow-chart.png
```

Figure 4.2: System flow chart

4.3.2 Methodology/ Working Mechanism

As stated previously, CardsQL does not have different types of users so all the following actions can be done by the user.
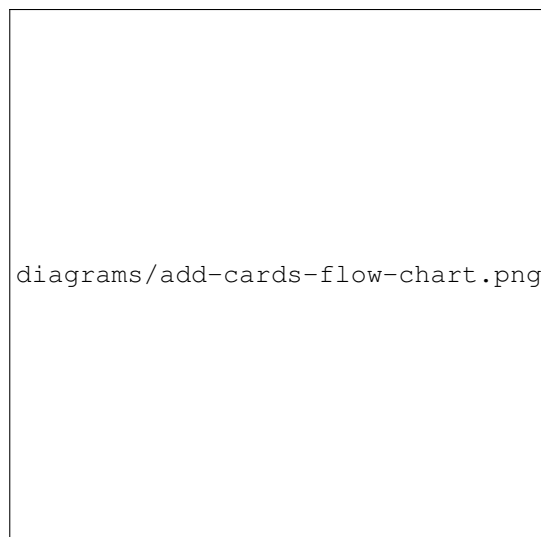
1. Add Cards

diagrams/add-cards-flow-chart.png

diagrams/add-cards-flow-chart.png

Figure 4.3: Flow chart for adding cards

2. Edit Cards

   Card details can be edited or the entire card can be deleted

diagrams/edit-cards-flw-chart.png
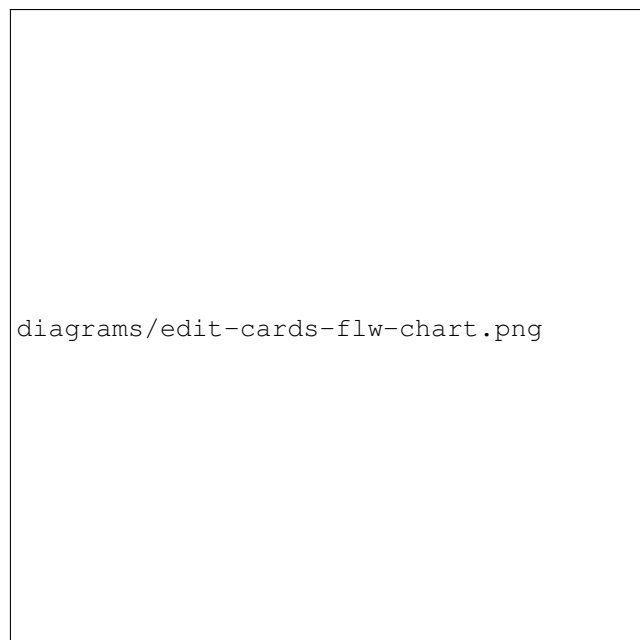
diagrams/edit-cards-flw-chart.png

Figure 4.4: Flow chart for editing cards

3. Revise Cards

diagrams/review-cards-flow-chart.png

diagrams/review-cards-flow-chart.png

Figure 4.5: Flow chart for revising cards

5 Gantt Chart

diagrams/gantt-chart.png

Figure 5.1: Gantt chart based on Waterfall model

6   Expected Outcome

- 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 a daily habit

7   References