Socially Constructed Flashcard System

Software Requirement Specification

by
Matthew Bojey
Raffi Kudlac
Ethan Owusu
Duncan Szarmes

Table of Contents **Page**

1 Overview

1.1 Introduction

This document serves as a preliminary software requirement specification for our Flashcard Q and A idea developed by Ethan, Matt, Raffi, and Duncan for COSC 304: Introduction to Database Systems. It contains an overview of our mission and analysis of

system requirements from software design perspective. This will be followed by our database design, design explanations and assumptions about the domain, the ER diagram, and DDL for the project. At the end of this document a prototype website is given that represents what we plan our website to resemble.

1.2 Our Goal

Our goal is to simulate real flash cards online that will be able to quiz users on any subject that they desire but we want to specifically focus on courses that are offered at UBCO. Ultimately we would like to put this on mobile phones so that people can access flash cards anywhere but for this project we are just planning on making a usable website to go along with the database structure. There is also potential for a premium version of the app that people will pay for.

1.3 Constraints

The major constraint of this project will be the time requirement. As this will be our first project using databases we are bound to run into unfamiliar errors and it may take more time than usual to solve them. This is also the first time we have worked in a group of this size so we are bound to run into conflicts but we are all familiar with each others coding styles and we are confident that we can work as a team to overcome any obstacles.

1.4 The Game Plan

We acknowledge our time constraint and our unfamiliarity with databases and as a result we have created a priority list of tasks to get done. First we want to create the database structure with the correct relations connected to each other and storing the right information. Then we will want to create a list of common queries that users will be using. After this we will move on to creating a fully functional website. Depending on how much time we have left over we hope to put all of this on a mobile platform as well.

2 Database Description

2.1 Entity Design

What follows is a short description of each entity in our database along with a justification for its presence. We also state any assumptions that we will be making regarding these entities.

2.1.1 User (Strong)

A user is a person who signs up for the service. This entity will store their email, username, password, a unique user ID, a boolean storing whether the user has paid for premium service or not, the sign-up date and time, as well as the number of decks they have made.

We are assuming usernames and emails will be unique. We included the boolean just in case it becomes a paid service in the future.

2.1.2 Card (Strong)

A card is the fundamental building block of the system. It is made up of a question and an answer, as well as a unique card ID. The card entity will also store if it has been used and the date and time of its last use. Cards will also store which deck they belong to. Cards will be able to be marked by users as favorite cards in hope to make it easier for them to create their own decks.

We are assuming cards will always have a question and answer, and will be close-ended questions.

2.1.3 Deck (Strong)

A deck is a composition of many cards. Each deck will have a unique deck ID stored as a integer, as well as the deck name, the number of uses, and the size of the deck. Decks will correspond to courses.

We are assuming the contents of the deck are related to the subject they’re created under.

2.1.4 Course (Weak)

A course refers to a university course, i.e. COSC **304 - Introduction to Databases.** The entity will store the course name and number. Each course is a division of a subject. There will be courses that we will hard code into the system but when a user creates a deck they will have the option of creating their own course to go with it.

We are assuming each course exists at UBC Okanagan.

2.1.5 Subject (Strong)

A subject refers to a subject of information, i.e. Biology, Chemistry, Computer Science, etc. The only thing this entity will store is the subject name, which we will hard code into the database ourselves.

We are assuming each subject exists at UBC Okanagan.

2.1.6 RecentDeck (Weak)

The idea is to store recent decks used by each user. This entity will store the ID of the deck, the date and time of its last use for each user, and the current card being used by the user. It will only store the 5 most recent decks used by a user.

2.1.7 FavoriteDeck (Weak)

The idea is to store each deck a user favorites. The entity will store the current card the user left off on the last time they used the deck. A user can have as many favorited decks as they want.

2.1.8 Rating (Strong)

A user will be able to rate cards and decks. The entity will store the rating (liked or disliked), a comment, the date the rating was created, and a unique rating ID. This entity will show the validity of a deck. Users will be able to sort deck search results buy deck rating making it easy for them to find valid decks.

We are assuming users will use ratings to locate the most useful decks.

2.2 Entity Summary

Entity Name	**Attributes**
User	userID, username, email, password, decksMade, paid, signupDate
Card	cardID, qText, aText, isUsed, lastUsed
Deck	deckID, deckTitle, deckSize, numOfUses
Course	courseNum, courseName
Subject	subjectName

RecentDeck	whenUsed, currentCard
FavoriteDeck	currentCard
Rating	ratingID, liked, comment, dateCreated

2.3 Relationship Summary

- There is a relationship “favorites” between User and Card. This is because a user may find a card that they like in a deck. The user is then able to place this card in their favorite cards which can be maintained by the user. A user may favorite many cards and a card may be favorited by many users.
- There is a relationship “creates” between User and Course. This is because if a user creates a deck in a course that does not yet exist then that course will be created. A course can only be created once by a single user then it is available for all future users. A user may create many courses if they are the first to make a deck in multiple courses.
- There is a relationship “makes” between User and Rating. A user can create a rating that will be applied to either a card or a deck. A user can create many ratings and each rating is only created by one user.
- There is a relationship “has” between User and RecentDeck. The need for this variable is that a user will use a deck and then may want to use it again soon afterwards. This represents the fact that a user can have up to 5 recent decks but each deck is only recent to one user.
- There is a relationship “has” between User and FavoriteDeck. This represents the fact that a user can have many favorite decks and each favorite deck is may be favorited by many users.
- There is a relationship “is a” between Deck and RecentDeck. This is because a recent deck is a deck. There can be many recent decks, 5 possible per user, but each recent deck is only one deck.
- There is a relationship “is a” between Deck and FavoriteDeck. This is because a favorite deck is a deck. There can be many favorite decks, 1 possible per user, but each favorite deck is only one deck.
- There is a relationship “creates” between User and Deck. This relationship represents the fact that a user can create many decks and a deck is only created by one user.
- There is a relationship “rates” between Rating and Deck. This is caused by the fact that a user can create a rating that applies to a specific deck. This rating is a boolean of the form “like/dislike”. Each deck can have many ratings but each rating applies to only one deck.
- There is a relationship “rates” between Rating and Card. This is caused by the fact that a user can create a rating that applies to a specific card. This rating is also a boolean of the form “like/dislike”. Each card can have many ratings but each rating applies to only one card.

- There is a “has” relationship between Deck and Card. This is as decks are made of cards, as a results a deck may have many cards. Also, a card may belong to many decks.
- There is a “has” relationship between Course and Deck. This represents the fact that a deck belongs to a specific course and a course may have many decks.
- There is a “belongs to” relationship between Subject and Course. This represents the fact that a course belongs to a specific subject and a subject may have many course .
- There is a “belongs to” relationship between Card and RecentDeck. This is so that the currentCard may be remembered between each user session. There is only one currentCard per recent deck but a card may be the current card to many recent decks.
- There is a “belongs to” relationship between Card and FavoriteDeck. This is so that the currentCard may be remembered between each user session. There is only one currentCard per favorite deck but a card may be the current card to many favorite decks.

2.4 ER Diagram

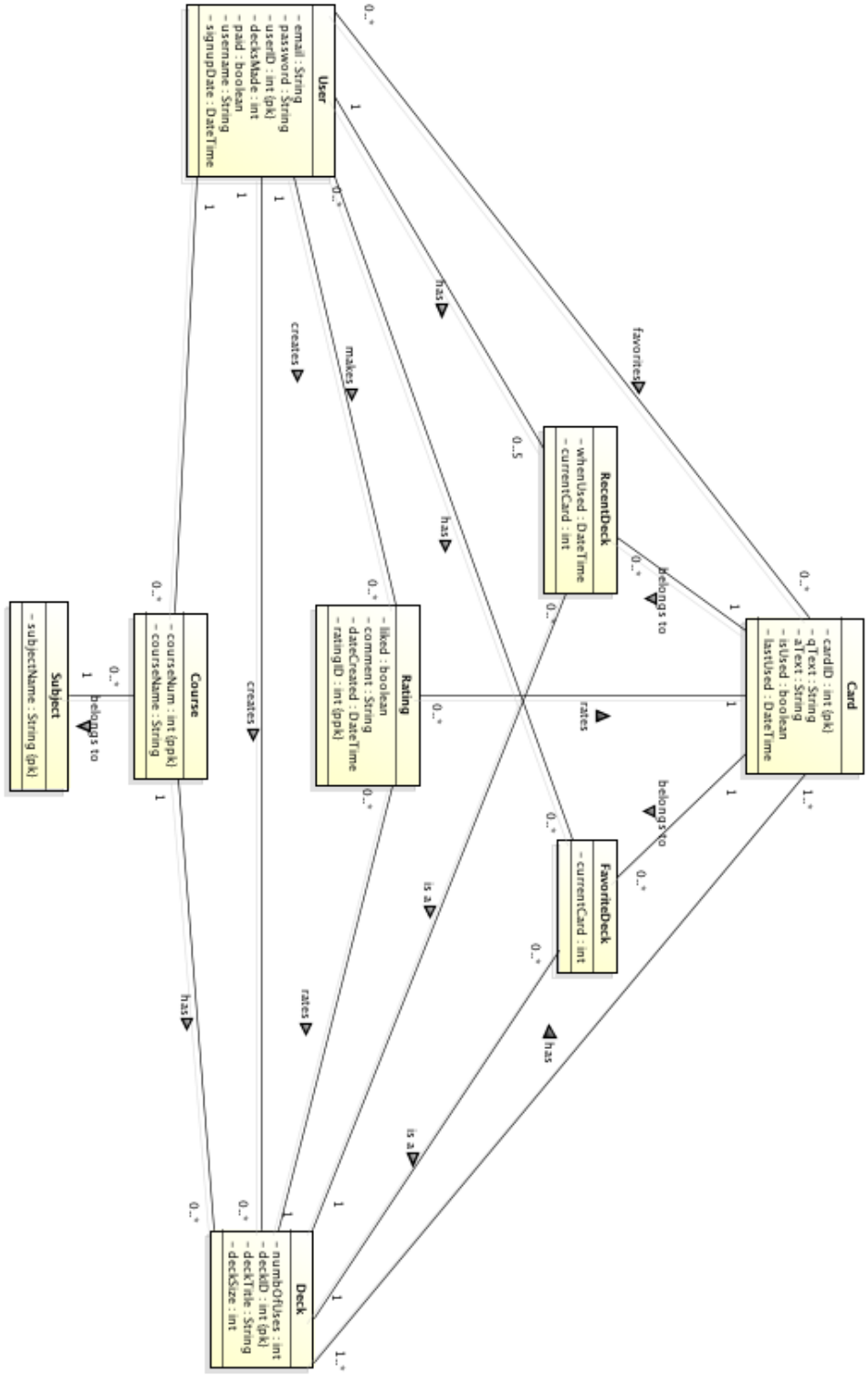

Figure 1: ER Diagram for above described entities and relationships.

2.5 Database DDL

```
CREATE TABLE User(
  userID INTEGER,
  username VARCHAR(16),
  email VARCHAR(30),
  password VARCHAR(16),
  decksMade INTEGER,
  paid BOOLEAN,
  signupDate DATETIME,
  PRIMARY KEY(userID)
 );

CREATE TABLE Deck(
  numbOfUses INTEGER,
  deckID INTEGER,
  deckTitle VARCHAR(30),
  deckSize INTEGER,
  userId INTEGER,
  courseName VARCHAR(255),
  courseNum INTEGER,
  PRIMARY KEY(deckID),
  FOREIGN KEY(userId) REFERENCES User(userID),
  FOREIGN KEY(courseNum) REFERENCES Course(courseNum),
  FOREIGN KEY(courseName) REFERENCES Course(courseName)
);

CREATE TABLE Card(
  cardID INTEGER,
  qText  VARCHAR(255),
  aText VARCHAR(255),
  isUsed BOOLEAN,
  lastUsed DATETIME,
  deckID INTEGER,
  PRIMARY KEY(cardID),
  FOREIGN KEY(deckID) REFFERENCES Deck(deckID)
);

CREATE TABLE Rating(
  comment VARCHAR(255),
  dateCreated DATETIME,
  ratingID INTEGER,
  liked BOOLEAN,
  userID INTEGER,
  deckID INTEGER,
  cardID INTEGER,
  PRIMARY KEY(ratingID),
  FOREIGN KEY (userID) REFERENCES User(userID),
  FOREIGN KEY(deckID) REFERENCES Deck(deckID),
  FOREIGN KEY(cardID) REFERENCES Card(cardID)
);
```

```
CREATE TABLE RecentDeck(
  currentCard INTEGER,
  deckID INTEGER,
  whenUsed DATETIME,
  userID INTEGER,
  cardID INTEGER,
  PRIMARY KEY(deckID),
  FOREIGN KEY(userID) REFERENCES User(userID),
  FOREIGN KEY(deckID) REFERENCES Deck(deckID),
  FOREIGN KEY(currentCard) REFERENCES Card(cardID)
);

CREATE TABLE Subject(
  subjectName VARCHAR(20),
  PRIMARY KEY(subjectName)
);

CREATE TABLE Course(
  courseNum INTEGER,
  courseName VARCHAR(255),
  subject VARCHAR(20),
  PRIMARY KEY(courseNum,courseName),
  FOREIGN KEY(subject) REFERENCES Subject(subjectName)
);

CREATE TABLE FavoriteDeck(
  currentCard INTEGER,
  userID INTEGER,
  deckID INTEGER,
  FOREIGN KEY(userID) REFERENCES User(userID),
  FOREIGN KEY(currentCard) REFERENCES Card(cardID),
  FOREIGN KEY(deckID) REFERENCES Deck(deckID)
);

CREATE TABLE FavoriteCard(
  userID INTEGER,
  cardID INTEGER,
  FOREIGN KEY(userID) REFERENCES User(userID),
  FOREIGN KEY(cardID) REFERENCES Card(cardID)
);
```

3 List of Features

This section details the features that we wish to implement in the system and they relate to the database structure described above.

1. Be able to search all created decks
    - This feature would involve a query to the Deck table. We would like to be able to sort the results by subject, course, author or date. All of this information can be taken from the Deck and Course tables.
2. Be able to access recent decks used
    - This would be done with a query to the RecentDeck table for a particular user.
3. Be able to mark favorite decks and cards
    - This would be done with an insert call to the FavoriteDeck or FavoriteCard table while remembering the current user and the deck or card ID.
4. Be able to access favorite decks and or cards
    - This would be done with a query to the FavoriteDeck table for a user to find decks and a query to the FavortieCard table.
5. Create your own decks
    - This would be done with an insert call to the Deck table. Cards would then be added to the deck by the user one at a time with insert calls to the Card table.
6. Rating of decks and cards from users
    - This would be done by insert new tuples into the Rating table every time that a user rates a deck or card. Ratings will be of the form thumbs up or thumbs down so that a boolean “liked” value can represent the ratings.
7. Create aggregate decks for courses based on user ratings
    - This would be done by sorting all cards from all decks related to a course and choosing the most highly rated cards. This process would be done with a query to the Card table and creating a new deck. The process may need human monitoring to ensure that a deck is not made up of many extremely similar cards.
8. Presenting cards to users at the time that is optimal for data retention
    - Based on data from various studies we see that presenting data at set time intervals can aid in data retention. By using a query to the Card table and finding the time last used the system will be able to try to create these optimal intervals between card viewing.

4 Prototype UI

4.1 Home Page

This is the site theoretical home page. Where users can watch a small video on how the site works. They can sign in or create accounts. Some buttons like the “option” and “View Your Decks” will only appear once the user has signed in.

Figure 2: A prototype of the layout of the home page of the website for the system.

4.2 Search Page

This photo represents rough idea of what we hope the search page of our web site to look like. The main box in the middle holds links to all decks that fit the search parameters. The sorting options in the top left allow the user to sort the currents decks by the available options. The blue box holds a list of subjects that when clicked on will expand into sub categories, such as year and course. As soon as you click on one of these menus, it should send a query to get the appropriate decks.

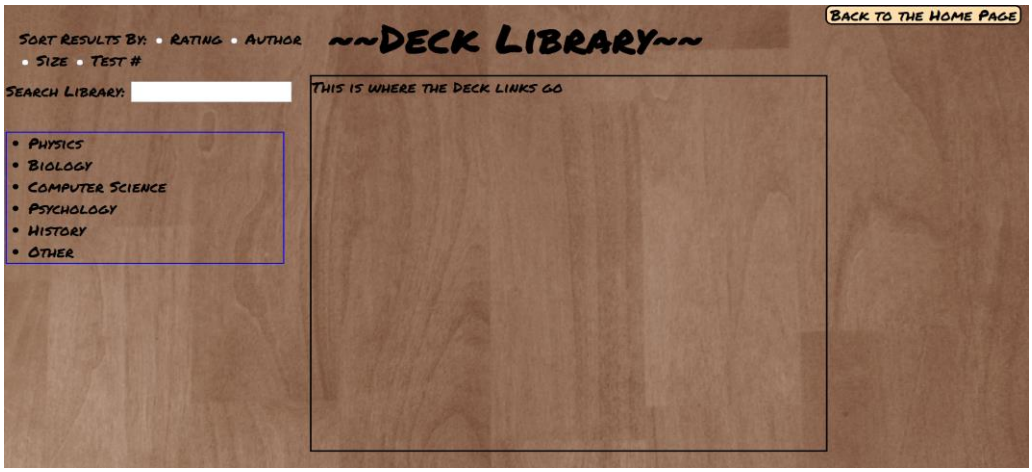

Figure 3: A prototype of the layout of the search page of the website for the system.

4.3 Mobile Platforms

As discussed earlier we also hope to, time permitting, provide mobile access for users of the service. Prototypes of this mobile interface are unavailable at this time.