NYU Tandon School of Engineering
Computer Science and Engineering
CS6083, Spring 2025

April 14, 2025

Project #1 (due April 27)

You are hired by a startup company to help build the database backend for a new web-based service, similar to Pinterest, that allows people to maintain online "pinboards" with pictures that they find and like and want to share with others. Users can sign up for the service, and can then create one or more pinboards. Later, users can "pin" pictures that they find on the web or upload themselves, and these pictures then become visible on one of their pinboards. Users can also "repin" pictures that they find on other user's pinboards, which adds them to their own boards. Users can also follow other people's pinboards, and can invite users to be their friends. Finally, users can "like" pictures they find on other boards, and can add short comments to other's pictures.

As an example, consider two users, Erica and Timmy. Erica likes to travel, and also loves antique furniture. She signs up and creates two pinboards, "Furniture" and "Dream Vacations". Whenever she sees a picture on the web that she likes and wants to show to her friends, say a picture of a nice sofa on an website, or a picture of a beautiful beach, she pins it to one of her boards. Erica also has friends who often look at her images and sometimes like the pictures or leave comments such as "Cute" or "love it!". Timmy is seven years old, likes dinosaurs and monsters, and when he grows up he wants to become a pirate. He creates boards named "Super Dinosaurs" and "Pirates" and whenever he sees a picture of dinosaurs or pirates (or even better, dinosaurs *and* pirates) he pins it to his boards. He also follows several pinboards by others that have a lot of pictures of monsters and dinosaurs – to do so he defines a 'follow stream" called "Monsters and Dinosaurs" containing pictures from four other boards that he follows. He also sometimes repins some of these pictures so they appear on his own board. For simplicity, we assume that all boards, pictures, pins, and likes are visible to everyone, and that all pictures could be repinned and liked by any other user. However, a user's follow stream is private.

So this describes the basic idea behind the system. (You may also explore services such as Pinterest to get the idea.) In this first part of the course project, you will have to design the relational database schema that stores all the information about users, boards, pictures, friendships, follow streams, repins, likes, and comments. In the second part of the project, you have to design a web-accessible interface that makes this system usable for real users.

You should use your own database system on your laptop or an internet-accessible server. Use a system that supports text operators such as like and contains. Both parts of the project may be done individually or in teams of two students. However, you have to decide on a partner and email the TAs with your names by Monday, April 21. The second part of the project will be due a few days before the final exam. Note that the second project builds on top of this one, so you cannot skip this project.

Before starting your work, you should think about what kind of operations need to be performed, and what kind of data needs to be stored. For example, there should be a login page, a page where a user can sign up for the first time (by supplying an email address and choosing a user name), and a page where users can create or update their profiles. Users should be able to create pinboards, ask other users to become friends, and should be able to answer friend requests. They should be able to pin pictures they find on the web, or which they upload themselves. They should be able to repin and like pictures. For simplicity, we assume that all boards, pictures, pins, and likes are visible to everyone, and that all pictures could be repinned and liked by any other user. However, people may decide to only allow their friends to comment on pictures on their board (this is a setting they can choose for each board). When a user likes a picture, this is counted as a like of the original pin of the picture, not of a particular repinning of the picture. However, comments about a repinned picture are only associated with the repinned and not the original picture. Users may also use keyword queries to search for pictures, by matching against their tags, and the system would then return pictures matching the keywords sorted by either time, relevance, or number of likes.

Some words about pinning and repinning, which will mainly be important for the second part of the project. When a user pins a picture on the web, she should supply the URL of the image, the URL of the page in which the image was found, and a few tags (e.g., "couch, brown, modern', ikea). The system should also download and store the picture itself in the database as a blob (in case the image changes later or is removed from the original site). If the user uploads an image, the system would store the image on its site, assign a URL to the image, and then pin that URL. When a user repins (or re-repins etc.) an image, this does not result in a copy of the picture, but is just a pointer to the picture as it was first pinned, with the same URL and tags, and if the first pinner removes it, it should become inaccessible everywhere it was repinned. Of course, the same picture might be originally pinned by several users, possibly under different URLs, and you do not have to remove such duplicate pictures. Also, ideally images would be pinned using a button on your browser that is provided as a browser plugin, but you do not have to do this as part of this project, so your system will probably require users to paste URLs into a dialog box.

Two more remarks: First, it is recommended to always store time stamps for any action such as pinning, liking, commenting, as real services use such log information for later data mining. Second, you should of course not use database permissions or views to implement user identification. There will not be a separate DBMS account for each user, but the web interface and application itself will log into the database. So, the system you implement can see all the content, but has to make sure at the application level that each logged-in user is identified through the use of cookies in the second part of the project.

Project Steps and Deliverables: In the following, we describe the suggested steps you should take for this project, and the associated deliverables. You should approach this project like one of the design problems in the homeworks, except that the schema may end up being a bit more complicated. You should spend some time carefully designing sample data that allows you to test some of the functionality. Note again that in this first problem, you will only deal with the database side of this project - a suitable web interface will be designed in the second part. However, you should already envision, plan, and maybe describe the interface that you plan to implement. The suggested steps are as follows:

(a) Design, justify, and create an appropriate relational database schema for the above scenario. Make sure your schema is space efficient, and suitably normalized. Show an ER diagram of your design, and a translation into relational format. Identify keys and foreign key constraints. Provide a short discussion of any assumptions that you made in your design, and how they impact the model. Note that you may have to revisit your design if it turns out later that the design is not suitable.

(b) Use a database system to create the database schema, together with key, foreign key, and other constraints.

(c) Write SQL queries (or sequences of SQL queries) for the following tasks.

(1) **Signing Up, Creating Boards, and Pinning:** Write queries that users need to sign up, to login, to create or edit their profile, to create pinboards, to pin a picture, and to delete a pinned picture.

(2) **Friends:** Write queries for asking another user to be friends, and for answering a friend request.

(3) **Repinning and Following:** Write queries for repinning a picture and for creating a follow stream. Also, write a query that given a follow stream, displays all pictures belonging to that follow stream in reverse chronological order.

(4) **Liking and Commmenting:** Write queries to like a picture, and to add a comment to a picture (while making sure the user is allowed to comment on this picture).

(5) **Keyword Search:** Write a query to perform a keyword search for pictures whose tags match the keywords. Use the contain operator to do so.

(d) Populate your database with some sample data, and test the queries you have written in part (c). Make sure to input interesting and meaningful data and to test a number of cases. Limit yourself to a few users and a few messages and threads each, but make sure there is enough data to generate interesting test cases. It is suggested that you design your test data very carefully. Draw and submit a little chart of your tables that fits on one or two pages and that illustrates your test data! Print out and submit your testing.

(e) Document and log your design and testing appropriately. Submit a well-written description and justification of your entire design, including ER diagrams, tables, constraints, queries, procedures (if any), and tests on sample data. Your documentation should be a comprehensive paper, including introduction, explanations, ER and other diagrams, and more (typically about 8-12 pages). This paper will be expanded in the second part of the project.