ITECH3108 Dynamic Web Programming

Assignment 1 - Dessert Mingle

Overview

This is an individual assignment in which you are required to develop a dynamic web application as described below.

Important: This assignment specification is generated just for you. Do not distribute this specification.

Timelines and Expectations

Percentage value of task: 20%

Due: Refer to Course Description

Learning Outcomes Assessed

The following course learning outcomes are assessed by completing this assessment:

- **K3.** Detect opportunities for increasing security and privacy of web applications
- **S1.** Develop client/server web applications using client-side and server-side code
- **S2.** Connect to and manipulate a database management system programmatically using server-side code
- **A1.** Design, develop, test, and debug client/server web applications to provided specifications

Assessment Details

For this assignment, you will create a web-based online dating system, vaguely similar to "Plenty of Fish" or "Ok Cupid".

Your platform is dedicated to matching up people with a shared interest in a certain topic - for your assignment the topic is *Dessert Mingle*, a place for dessert enthusiasts to meet.

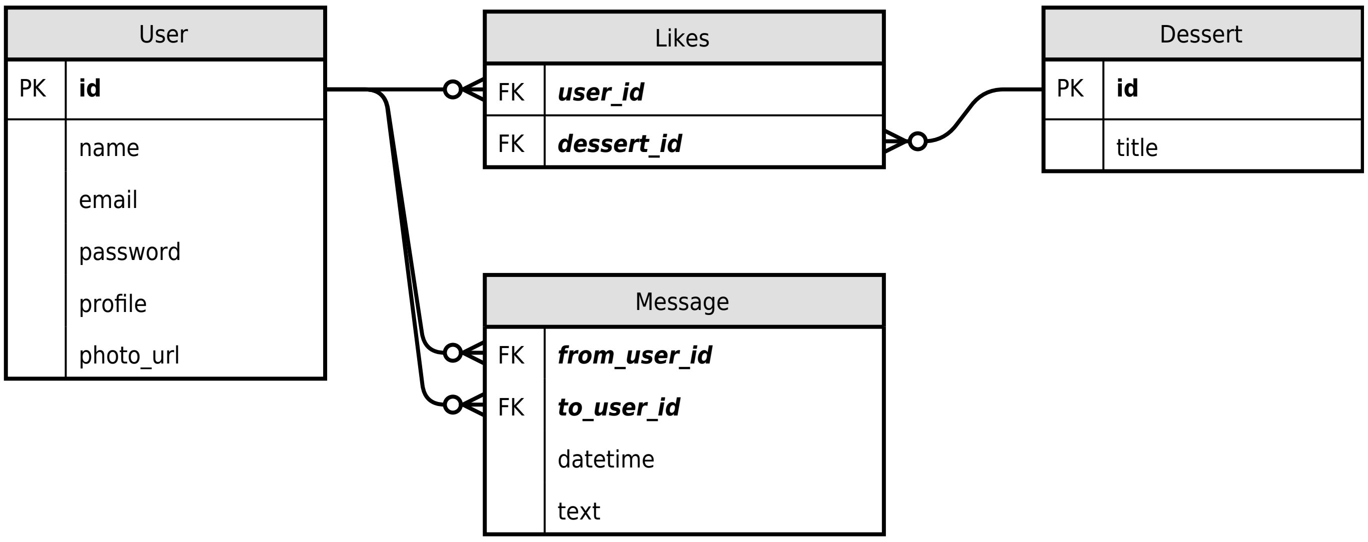
Users can create accounts, indicate a set of desserts that they like, and find people to message who like similar desserts.

You must implement this platform using PHP and MySQL or MariaDB, with some JavaScript for validation.

Database Structure

The web application uses a relational database. The database has the following structure:

- User (id, name, email, password, profile, photo_url)
- Dessert (id, title)
- Likes (*user_id*, *dessert_id*)
- Message (*from_user_id*, *to_user_id*, datetime, text)



Primary keys are indicated with underlines or bold formatting, and foreign keys are italicized.

Each record in the `User` table represents a member of the site who is looking for love. Members can write a short piece of text about themselves (stored in the `profile` field), and include the URL of a photograph (stored in the `photo_url` field).

Each record in the `Dessert` table represents a dessert that a user can indicate that they like.

The `Likes` table stores information about which `Dessert` records each member has indicated that they like. These will be used to match with other members of the site who like similar dessert.

If a record `(1, 2)` exists in `Likes`, this means the user with id `1` likes the dessert with id `2`.

Finally, Dessert Mingle should allow members to send messages to each other. The `Message` table tracks messages sent between members, along with the date and time that the message was sent.

The following constraints should be applied when implementing the application:

- The `user_id` and `dessert_id` fields in the `Likes` table form a compound primary key;
- The `from_user_id` and `to_user_id` fields in the `Message` table form a compound primary key, and both refer to the `id` field in the `User` table - you may choose to use a `message_id` instead if you prefer;
- The `datetime` field should be stored as a MySQL *datetime* type;
- The `password` field should be a VARCHAR of 255 characters. The `name`, `email` and `photo_url` fields should be VARCHAR of a length that you determine to be reasonable and sufficient;
- The User `profile` and Message `text` fields should be either VARCHAR or TEXT.

Please note that you are free to **extend this database schema** if required.

Initial Data

When the database is created, it should be populated with data of your own invention, appropriate to the theme (keep it clean).

You should have at least:

- **5** desserts; and
- **5** users, each of whom likes at least **2** desserts (see note below)

One of the users *must* be you (even if you are not looking for love or even interested in desserts). Use your student id - 30344274 - for the username, and your real name and email address. Invent other users as necessary - perhaps use characters from your favourite movie or band.

One of your users must also have the username **tutor** and the password **guest**.

You do not need to include any messages in this initial data.

Include this data as part of your written report.

Database creation DDL

Create an SQL file that creates the MySQL database, creates the four tables above, and populates them with your initial data.

Use your student id and course code as the database name as follows:

`ITECH3108_30344274_A1.`

Passwords should be hashed using, *at minimum*, the `crypt()` PHP function. Prefer to use the PHP `password_hash()` function to generate password hashes.

For the password 'guest', the following hash may be used in your database:

```
PASSWORD = '$2y$10$12347244303000099999uHDLKr5JAAbT7wZC8f9cgmN2el5UVKii'
```

It is acceptable for all initial users to share the same password for testing.

Use of MD5 or SHA1/2 for password hashes is not acceptable.

Write SQL queries that display all of the initial data using SELECT statements, and **list these queries in your report**.

User accounts

Write PHP and HTML to allow new users to **sign up**. The form should request a username, email address and password. The password **must be hashed** before storing it in the database.

Using PHP, **validate** that the username is unique, and the password is at least 5 characters (before hashing).

Write PHP code to allow users to **log in** and **log out**. This will require the use of sessions and/or cookies.

Setting interests

Allow users to **edit their profile** - changing their profile text and photo URL.

Write PHP and HTML code to allow a user to choose items from the `` table that they like.

You may implement this using whatever User Interface approach makes sense to you - perhaps either a set of checkboxes, a set of buttons, or using a search box and some JavaScript.

Finding matches

Create a page listing **potential matches**.

Write PHP and HTML that shows the logged-in user a list of other users who might be a good match based on their overlapping interest in desserts.

- A user with whom the logged-in user shares *no items in common* should not appear in the list.

- A user with whom the logged-in user shares *all items in common* should appear higher in the list than a user with whom they only share a few interests.

It is up to you determine an appropriate way to sort this list. You may choose to adjust your database schema, although it's not required.

Each displayed match should include the match's name, photo and profile text.

Messages

Write PHP and HTML to allow a logged-in user to send a message to another user of the site. Sending a message should be implemented by adding an entry to the `Message` table.

Create a **messages page** that displays the messages sent and received by the logged-in user, including the date and time for each message.

Adjust your matches page so that **users no longer appear as a match once they have been messaged**.

Aggregate data

Create a page that displays the following information, using SQL aggregation such as COUNT and SUM, subqueries or nested SELECT statements, inner joins and (left or right) outer joins.

- The total number of messages sent on the site;
- The **top 3 most-liked **, ordered in descending order by number of likes;
- The most popular day of the week to send messages;

Bonus challenge task (optional!) - Moderation

Extend the data model and write code to implement moderation features:

- Users can **report users** for violating site rules;
- A moderator can **review reported users**, sorted by number of reports, weighted by the reporting user's *reputation* (descending order)
- Users found to be in violation of site rules should have their accounts disabled, but should not be deleted
- Users who incorrectly report users for being in violation when they are not should have reputation reduced, so their reports are less important in future
- Users who correctly report other users should have their reputation increased, so their reports are more important in future

There are **no partial marks** awarded for this bonus task – you must complete all features to attain the bonus marks.

It is possible to attain full marks for this assignment without completing this challenge task.

Further details

Documentation

Include a written report containing:

- The SQL queries you used to test your database
- A list of parts of the assignment you have completed or not completed.
- Details of specific assistance you received from people other than your lecturer or tutor, and the names of those assisting.
- Anything interesting or cool you'd like to draw your marker's attention to.

Assignment support

This assignment is supported by the first 5 lectures and the first 6 labs. Work on the assignment should be spread over several weeks after the relevant lab has been mastered.

Submission

All files should be submitted to Moodle by the due date and time. Check with your tutor as to whether a hard copy is required in addition to the electronic submission.

Marking Criteria/Rubric

Refer to the attached marking guide.

Feedback

Feedback will be supplied through Moodle. Authoritative marks will be published through fdlMarks

Plagiarism

Plagiarism is the presentation of the expressed thought or work of another person as though it is one's own without properly acknowledging that person. You must not allow other students to copy your work and must take care to safeguard against this happening. More information about the plagiarism policy and procedure for the university can be found at http://federation.edu.au/students/learning-and-study/online-help-with/plagiarism.

Marking Guide: Assignment 1

Feature	Criteria	Maximum	Obtained
Initial data	Requirements satisfied	1	
Creating the database	Table structure, data types, field lengths, initial data entry	1	
User accounts	Account sign-up	1	
	Validation that password meets complexity requirements (at least 5 characters)	1	
	Log in and Log out	1	
	Inappropriate password hashing (MD5, SHA1 or plain-text passwords)	(-2)	
Profile and Interests	Edit profile text and photo URL	1	
	Select interests	2	
Matches	List of users based on overlapping interests	3	
	List sorted appropriately	1	
	Messaged users do not appear in matches	1	
Messages	Send a message to a user	2	
	List messages from other users	2	
Aggregate data	Top 3 most-liked items	1	
	Most popular day to send messages	1	
Bonus optional task - Moderation	Meets specification (user report, moderator list, reputation system) (no partial marks)	3	
Documentation	Initial data and test queries	1	
	Completion of tasks, Assistance statement (lose 1 mark each if not included)	(-2)	
Quality of code	Layout, structure, indentation	(-1)	
	Appropriate and consistent naming scheme	(-1)	
	Valid HTML5	(-1)	
	Total:	**23**	