ITECH3108 Dynamic Web Programming

Assignment 1 - Watch Basement

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 student id 000000. 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 an **online marketplace** - a bit like Facebook Marketplace or eBay.

Your marketplace, *Watch Basement* is dedicated to community members selling or exchanging watches with other local members.

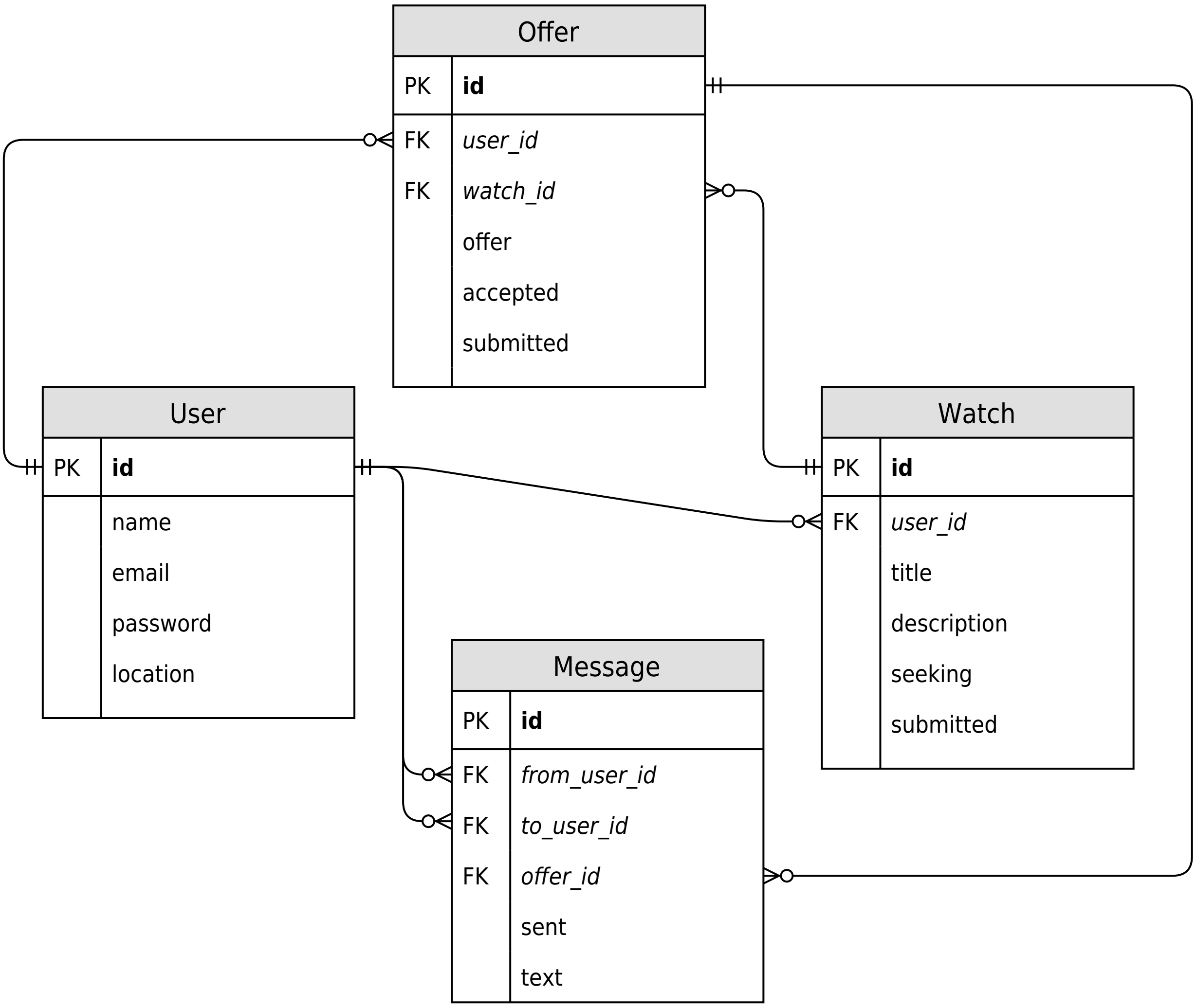
Users can create accounts, browser watches that are up for sale or exchange, make an offer, and message the winner to arrange the exchange.

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, location)
- Watch (id, *user_id*, title, description, seeking, submitted)
- Offer (id, *user_id*, *watch_id*, offer, accepted, submitted)
- Message (id, *from_user_id*, *to_user_id*, *offer_id*, sent, 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. Members can add watches to the site, and review and make offers on the watches added by others.

Each record in the `Watch` table represents a watch that a user has put up for sale or exchange. The `seeking` field is free text and indicates what the user is willing to accept in exchange for the item.

The `Offer` table stores offers made by other users for a particular watch. Users may make multiple offers for a single item.

Finally, Watch Basement should allow members to send messages to each other to arrange the exchange. The `Message` table tracks messages sent between members about a particular offer, along with the date and time that the message was sent.

The following constraints should be applied when implementing the application:

- The User `password` field should be a VARCHAR of 255 characters. The `name`, `email`, and `location` fields should be VARCHAR of a length that you determine to be reasonable and sufficient;
- The `Message.text` and `Offer.offer` fields may be either VARCHAR or TEXT.
- The `Watch.submitted`, `Offer.submitted`, and `Message.sent` fields should all be be MySQL

TIMESTAMPs.

- The `Offer.accepted` field can be either a Boolean indicating whether this offer was successful, or a nullable TIMESTAMP indicating when the offer was accepted.
- Only one offer may be accepted per watch - this is not enforced by the database design but must be enforced by the application. You may change the schema to make it enforced by the database if you prefer.

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** users;
- **3** watches;
- **5** offers, at least one of which is submitted by the **tutor** user, and accepted by the seller; and
- **3** messages about a single accepted offer

One of the users *must* be you (even if you are not interested in watches). Use your student id - 000000 - for the username, and your real name and email address.

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

Invent other users as necessary - perhaps use characters from your favourite movie or band.

Include this data as part of your written report.

Database creation DDL

Create a single 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_000000_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$12300000000000999999999upbG63KDvHpam5TTLh.6xxCvYZFPYZcq'
```

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.

Viewing items

Write PHP and HTML to allow users to view a list of all items currently offered, including any offers made for each. The list should be **sorted** so the most recent items are listed first.

Watches with accepted offers should still be displayed, but highlighted in some way, including highlighting the successful offer.

You may implement this using whatever User Interface approach makes sense to you.

Making and Accepting Offers

Write PHP and HTML that allows users to make an offer on a particular watch.

This should only work on items that don't have accepted offers.

Users should be able to review offers made on their watches, and accept one.

Messages

Write PHP and HTML to allow the users in an accepted exchange to message each other.

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

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 watches** on the site;
- The **top 3 most-wanted watches**, ordered in descending order by number of offers made;

Bonus challenge task (optional!) - Moderation

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

- Users can **report users** for violating site rules (for example, by posting inappropriate offers);
- 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)	
Viewing items	List of all items	2	
	Unavailable and accepted offers indicated	1	
Making offers	Users can make an offer on an item	2	
	Offers only on available items	1	
	Users can review offers made on their items	2	
Messages	Send a message to a user	2	
	List messages from other users	2	
Aggregate data	Total items on the site	1	
	Top three most-wanted items	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:	**20**	