# Project Requirements and Database Design

## Project Requirements:

ImageOptim asked us to redesign their online license distribution store, with several pre-defined requirements in mind as they were unhappy with their current solution. This platform would be client-facing and accessible to anyone wishing to purchase a license on offer. The client was working as a sole trader and simply wanted a way to efficiently sell software licenses to an end-user.

Our platform was to include variable pricing for companies with different sizes (by employee count) as well as two different license types, being either annual or perpetual. The client wanted us to include an invoice with each successful purchase and give the user the option to view it on demand.

The client specifically mentioned that the user would not be required to make an account to complete a purchase, but they would have the ability to access an account if they wished. Upon each purchase, the user would be sent a confirmation email which would include a way to access their newly created account (if they do not have one already), whether this be by a password or by a link was up to us, as well as a full breakdown of their order, including price, products purchased, and the specification for each product. This is so that the user can have on-demand access to their licenses as well as being able to view their order history.

Another requirement was to include a way for prices pre-negotiated between the client and the user, to be applied to a product. The country specified by the user in the order details would only matter at the point of sale, this included VAT status and pricing. The client was clear that they wanted us to store all order information, except sensitive information like card details, as this would be dealt with using a stripe integration, which was a nice to have requirement.

A single administrator account was also required, this account would be able to view stored information about current users as well as every current issued license and its owner. After our second meeting with the client, we found that there were no changes with the user’s requirements and so we continued to develop the final product.

## Database Design

### Conceptual Model

### 

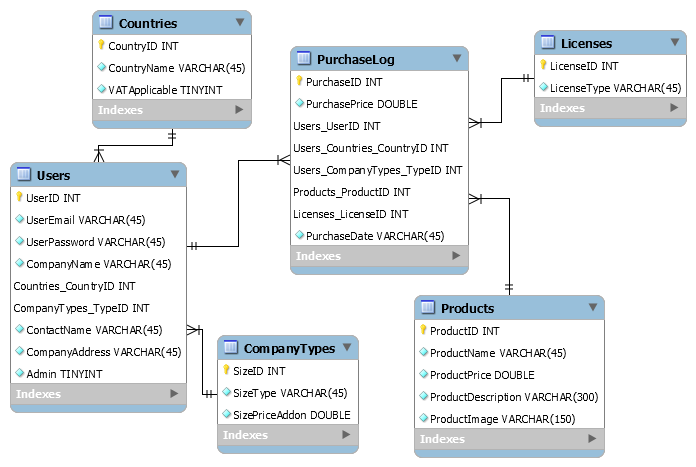
Our initial model was based on the client’s requirements and was reflected in our original database creation script. We created all tables in the database using SQL keywords ‘CREATE TABLE IF NOT EXISTS’ to ensure there was only one version of the table.

The ‘UserStorage’ table was needed so we could uniquely identify each user and associate each of their orders with the correct user. We needed separate tables for ‘CompanyTypes’, ‘Products’, and ‘Licenses’ as these values might change. Each of these tables included a primary key of data type ‘INT’, allowing us to use as foreign keys in ‘PurchaseLog’.

We used the ‘CONSTRAINT’ keyword to ensure that no invalid key could be used in the ‘PurchaseLog’ when we referred to primary keys from other tables. The ‘PurchaseLog’ was supposed to include all the information about each order but was erroneous in doing so as the ‘UserID’ was not included meaning the client requirement was not fulfilled. [[1]](#footnote-1)

We decided to use a camel case naming convention for our database design, and the data types we used for each attribute were suitable. To fulfill our requirements, we implemented relationships between the tables where they would be most necessary. However, we found errors in the normalization and relationships between the tables, which led us to redesign our database.

### Minimum Viable Product Schema

­­

After realizing the errors in the conceptual model, we redesigned the database to ensure that it was in 3rd normal form and contained the correct one-to-many relationships and no transitive dependencies. [[2]](#footnote-2) We renamed some of the tables and columns to conform to naming standards, better meaning, and to avoid confusion during development.

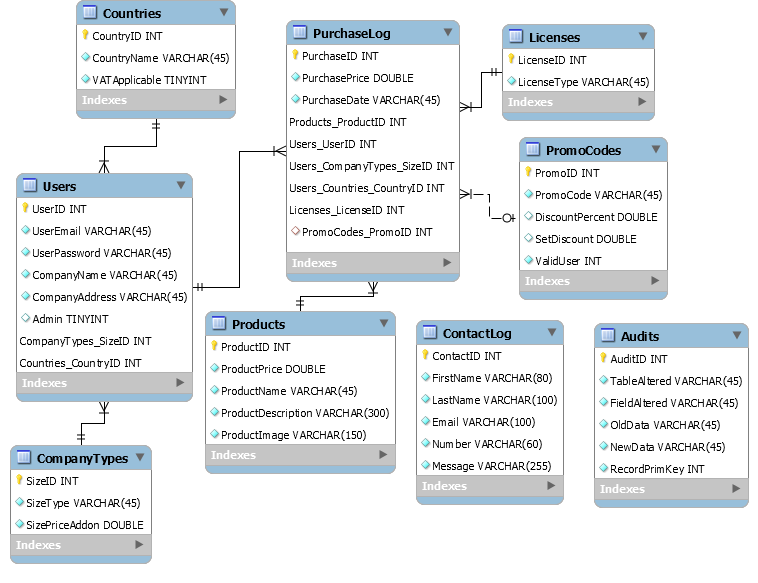
We created a new table ‘Countries’ to separate the column ‘Country’ from the ‘Users’ table and achieve 3rd normal form and allow for the addition of more countries and allow for consistency and referential integrity between records in the database. New foreign keys were added to ‘Users’ and the ‘PurchaseLog’ using the same method used in the first model.

The new foreign keys in the ‘Users’ table were used to maintain referential integrity and ensure that there was no repeated data in the ‘User’ table. The addition of the foreign key ‘UserID’ in the ‘PurchaseLog’ was implemented to meet the client requirement of being able to view all orders made by a single user. Within the ‘Users’ table, we amended the ‘UserID’ column creation code to include ‘AUTO\_INCREMENT’ and the ‘Admin’ column to include a ‘DEFAULT’ value as this saved us time in development and ensured there were never database insertion errors.

‘PurchaseDate’ was added to the ‘PurchaseLog’ as this was a client requirement, where the client wished to be able to view the exact time and date that a purchase was made.

To add sample data into the database, we used the ‘INSERT INTO… VALUES’ keywords, allowing us to test our platform and carry out the required tests to ensure we had met our client requirements.

### End of Project ER Diagram



In our final database design[[3]](#footnote-3), we have included new tables, not related to others and provide features beyond the client’s requirements including the ‘ContactLog’ table, used to store customer comments.

To accommodate for future client needs, we created the Audits table which allows the client to view all changes made to the database and in what table those changes were made. Attributes within this table have been given the keyword ‘NOT NULL’ to ensure that no erroneous inserts can be made.

We also created a new table to accommodate one of the client’s requirements, negotiated prices. This table ‘PromoCodes’ had a many-to-one relationship with the purchase log table and its foreign key within the purchase log was able to contain a ‘NULL’ value where the other attributes were not. This was since not all records within the purchase log would be making use of a negotiated price and therefore did not require a value to identify a promo code.

Stored procedures were implemented to reduce server load and make the database insertion process more efficient. We did this by using ‘PROCEDURE’ followed by inputs defined by ‘IN’ before then writing our insertion statement. A ‘DELIMITER’ was also defined to make sure MYSQL treated the entire procedure as a statement.

1. [Git Tag - Intial DB Model](https://git.cardiff.ac.uk/c2063799/SDS2_ImageOptim_Team10/-/tags/Initial-DB) [↑](#footnote-ref-1)
2. [Git Tag - Release 1](https://git.cardiff.ac.uk/c2063799/SDS2_ImageOptim_Team10/-/tags/Release-1) [↑](#footnote-ref-2)
3. [Git Tag - Demo Release](https://git.cardiff.ac.uk/c2063799/SDS2_ImageOptim_Team10/-/tags/Demo-Release) [↑](#footnote-ref-3)