**Design Documentation – Backend and Database Design**

This document outlines the design of both the database and the backend of ‘Team Biscuit Bulk’s project – Biscuit Bulk Barbering.

**Technology used**

The database runs on MySQL – for the scope of this project and based on the developers understanding of SQL, it is a viable option.

The backend of the site is ran using Express.

**Database & queries**

The database is a relational database that follows normal form. Below is a diagram (generated using Workbench) which shows the relations between the tables, the column names and their types.

**Diagram

Description automatically generated**

Throughout the API there are queries to the database which have been made by the developer. They include a range of joins, basic inserts, updates, selects and deletes. There are also transactions being completed in there to allow for the use of previous insert ID’s and also to ensure that foreign keys are being matched correctly and items are inserted at the same time.

**Routes on API**

|  |  |  |
| --- | --- | --- |
| **Route** | **Required Parameters** | **Description** |
| / |  | Returns JSON with title of API |
| /auth | Username : string  Password : string | Login Route – authenticates user and sets cookies for user type, username and loggedin |
| /logout |  | Clears all cookies for client |
| /register | Email : string  Password : string  fName : string  lName : string | Allows user to register on the site as a customer, makes use of transactions to use previous insertID’s and also dictate order of queries being completed |
| /cuts |  | Return all haircuts |
| /book | staffID : int  date : dateTime  time\_start : dateTime  email : string  cutID : int | Customer booking route – checks that there are no clashes with the potential appointment using the customers time starts and ends, uses moment library and functions .isSameOrAfer and .isSameOrBefore to compare with the prebooked appointments |
| /resetPassword | Email : string  currentPassword : string  newPassword : string  conf\_newPassword : string | Allows users to reset their password provided they know their current password. |
| /getAppointments |  | Returns all booked appointments |
| /getAppointments/:custEmail | custEmail : string | Returns specific booked appointments based on customer email that is entered |
| /getCustomers |  | Lists all customers, names, and email addresses |
| /deleteBooking | bookingID : int | Deletes bookings from database based on ID passed in |
| /addCut |  | Adds a haircut to database with the title ‘placeholder’ |
| /manageCut | cutID : int  cutName : string  cutDuration : int  cutLength : string  cutCost : int  cutAvailability : int  cutImg : string | Allows an administrator to manipulate haircuts, change their details etc.  NOTE: All details must be re-entered when editing. |
| /manageUser | Email : string  fName : string  sName : string | Allows users to change their name on their account |

**Security concerns**

There are some security concerns with these routes, given that one can access them easily without authentication. A way to address this would be to implement a time to live table which has the following: userID, random string of characters, expiry time. This would then store the random string of characters as a cookie for the user, and then every time the user attempts to use something which should be restricted a query will be completed on the database checking the stored userID and random string, and the users type.