Temasek Polytechnic

School of Informatics & IT

**Database Application Development (CIA1C06)**

**AY 22/23 Oct Semester**

**Project Part 2**

**(Database Design)**

|  |  |
| --- | --- |
| Name: | JAVEN LAI LE YU |
| Admission Number: | 2202934B |
| Class: | P17 |

Table of Contents

Introduction 3-5

The Entity-Relationship Diagram 6

Data Dictionary 7-15

Reflection 16

# Introduction

This Database is built for a fully-fledged Food Review Website for Singapore, using MySQL.

It allows Users to **review on restaurants if the User is signed in** (using **Reviews** Table). A **review consists of a rating** from 1-5 **and a Comment** restricted to 100 words. The Database is designed to restrict each User to 1 Review per User for each Restaurant.

Users can also **comment** (**Comments** Table) **on Reviews and other Comments**, allowing all Users to voice their complete opinions freely – and have meaningful discussions for everyone to see.

Each Comment can be no more than 100 words, but a User can Comment on the same Comment/Review as much times as he please.

This may lead to spam and fake rumours...

Which is why a **like or dislike system is introduced** (**Upvotes** Table) – creates **a natural moderation system**, collective community of Users, for nonsensical false claims to be identified from truthful reviews and comments – which can later be removed by Moderators.

Users can either Like or Dislike any review or comment.

Only 1 vote can be contributed per User for each Comment or Review. All Likes and Dislikes will be tabulated, using front-end, to produce an Upvote ratio. Upvote ratio = Like – Dislikes.

To provide flexibility, **Users can update their Comments**, Reviews and Votes, **but an ‘edited’ column will log a change** of Comment/Review. This is to ensure transparency – as User may comment something controversial and edit the Comment after it garnered a bunch of hate.

To **save storage space**, **old Comments** and Reviews **after an edit** **will** be overridden and **not saved** – upvotes, along with Comments on the Comment/Review will reveal the truth.

**Accounts** Table

Includes Reviewer (R), Owner (O) and Moderator (M) roles.

* A normal User will receive (R).
* A **Moderator Account has the ability to edit other User’s reviews and comment and terminate User Accounts**.
* An **Owner Account** indicates the Account is **in charge of a Restaurant** on the platform.

Each Restaurant can Only have 1 Owner to for safety and easier accountability for actions, but an Account can own multiple Restaurant.

As (M) holds immense power, it can only be changed using MySQL Workbench (Backend) while the abilities will be configured using front-end.

**Users must have a unique and exclusive name** no one else has. Their **passwords are encrypted/hashed before being stored** – confidential and secured; Database will always be expecting a 60 char hashed string for Account’s Password field.

*What if the Password gets forgotten?*

The **email and phone fields** stored will validate the User, to **provides a secure reset of password** – reset password link to be sent to User’s email or phone accordingly.

**Restaurant** Table

* **plentiful of relevant information** – 100-word description + Option to attach a Menu png.
* **multiple pictures of the Restaurant and its dishes** (Using **Images** Table) can be attached on listing, to entice Users to eat at the Restaurant for their next meal – help Users make informed choices by providing relevant attractions and details for each listing.
* A restaurant can have multiple categories, requiring the need for a (**Categories** Table). When Users are spoilt for choice, they can **filter** based on generic categories, thereby improving User Experience by saving their time and effort.

Categories like ‘Hotpot’, ‘Pizza’ and ‘Spicy’ cannot be filtered from Search results Such categories are useful for classifying restaurants so that Users know more details about a Restaurant from concise Labels.

E.g. A Muslim can instantly tell if the Restaurant is Halal from the Halal Label.

(**Branches** Table) contains all outlets for a Restaurant. Because a Restaurant can have multiple outlets, a separate table is required.

What are the contents required for each branch? That’s for the Owner of the Restaurant decides.

Therefore, a (**Branch\_Details** Table) is needed to enable such flexibility where various outlets can contain different details. Example shown below.

For the same Restaurant,

Tampines Branch displays: phone number, address, manager name, website link.

Bedok Branch displays: phone number, address.

Pasir Ris displays: address

Branch\_Details makes storage for Branches’ contents more optimized since as there will be no unnecessary empty columns if the particulars for that branch does not exist.

A convenient **Favourites and Bookmarks system** (Stored in **Lists** Table), is available for Users:

* **Favourites allow for Users to store the restaurants they like** and will continue to love, indefinitely.
* **Bookmark is essentially a ‘To-Do’ List** -that saves whatever Restaurants they are craving to try.

Bookmarks allow Users to remember what Restaurant they wanted to try, and they can come back to rate and review the restaurant once they are done. After which, the completed listing disappears, but may choose to Favourite so it won’t get lost.

Database **restrict** a User from **adding the same Restaurant more than once** either type of List.

To enhance effectiveness of Restaurant recommendations, User searches are logged, so that we can **recommend the trendiest and most current Restaurants** (Using **Searches** Table). Logging searches is beneficial as we can also know what Users want and expect from the site.

E.g., Users search for ‘Pizza’, but the site has no Pizza Restaurants.

Now, Pizza Restaurants can be added to fulfil the demands and wants of the Users.

# The Entity-Relationship Diagram

Graphical user interface

Description automatically generated with low confidence

Details:

11 Tables

The Field Sizes are calculatedly restricted for security reasons – prevent wrong data from getting stored and SQL injections. VARCHAR is used to allow flexibility – length of input isn’t fixed but cannot exceed file size.

E.g. accounts.phone is restricted to 8 Integers as this site is for Singapore (SG) Users and all SG numbers are 8 digits. ENUM (option 1, option 2) is used where deemed suitable to prevent errors like typo – ENUM() only expects and allow specific inputs, acting as “strict mode”.

Search Queries exceeding 50 chars are most likely unmeaningful as they are likely from User who searched irrelevant content; most restaurant names are within 50 chars.

Most Foreign Keys (FK) are set to CASCADE on delete, as those row(s) of detail are no longer needed when the root parent is deleted.

Some FK do not CASCADE as the data still can be used; hence the row should not be deleted.

E.G. Search Queries are kept even when the User who made the search is deleted.

# Data Dictionary

Table Name: accounts

Description: This table store Accounts and their properties.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idAcccount | INT | AUTO INCRE | P | N | Y | Unique Identification of Account | 1 |
| username | VARCHAR | 45 |  | N | Y | Account’s unique name | Javen |
| password | VARCHAR | 60 |  | N | N | Account’s Password,  hashed 10 times using bcrypt Algorithm. | $2b$10$9kUE3vnBTOoQg  AUvQJN/pukSb03C/Cr89Uh2TWhMs4g3stB/xc1.q |
| picture | VARCHAR | 100 |  | Y | N | File directory of Profile Picture  (optional) | /Javen.png (repetitive file dir. ‘../public/Photos/Profile’ will be configured front end, to avoid redundant storage.) |
| email | VARCHAR | 255 |  | N | N | Email for resetting password  (Can reuse email for Accounts) | javen@gmail.com |
| phone | INT | 8 |  | N | N | (All SG numbers are 8 digits)  Phone Number for resetting password (Can reuse number for Accounts) | 91234567 |
| date | DATE | - |  | N | N | Account Creation date | 2022-11-29 |
| role | ENUM | R, O, M |  | N | N | Status of Account: Reviewer,  Owner, Moderator.  Default Expression:  R | R |

Table Name: restaurants

Description: This table store Restaurants and their properties.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idRestaurants | INT | AUTO INCRE | P | N | Y | Unique Identification of Restaurant | 1 |
| name | VARCHAR | 50 |  | N | Y | Name of Restaurant | McDonalds |
| description | TEXT | 100 |  | N | N | Brief explanation of the Restaurant | A popular fast-food Restaurant that specializes in Burgers and Fries. Can be enjoyed by whole family. |
| idAccount | INT | - | F | Y | N | Owner of Restaurant’s Account | 1 |
| menu | VARCHAR | 100 |  | Y | N | File Directory of image menu for Restaurant | /mcdonalds  .png |
| date | DATE | - |  | N | N | Restaurant Creation date | 2022-11-29 |

Table Name: images

Description: This table store Restaurants’ Images; A Restaurant can have multiple Images.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| IdImage | INT | AUTO  INCRE | P | N | Y | Unique Identification of Image | 1 |
| idRestaurants | INT | - | F | N | N | Which Restaurant this image belongs to | 1 |
| image | VARCHAR | 100 |  | N | N | File Directory of Image | /mcdonalds/  1.png |

Table Name: categories

Description: This table store Categories of a Restaurant, for purposes of search filtering or categorizing restaurants using labels. Any kind of Category can be allowed – which is why didn’t use ENUM() to restrict, but keep within 50 chars for a concise and convenient to read Category.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| IdCategory | INT | AUTO INCRE | P | N | Y | Unique Identification of Category | 1 |
| idRestaurant | INT | - | F | N | N | Which Restaurant this Category belongs to | 1 |
| category | VARCHAR | 50 |  | N | N | Category of the Restaurant | Indian |

Table Name: branches

Description: This table store Branches of a Restaurant

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idBranch | INT | AUTO  INCRE | P | N | Y | Unique Identification of Branch | 1 |
| idRestaurant | INT | - | F | N | N | Which Restaurant this Branch belongs to | 1 |
| branch | VARCHAR | 80 |  | N | N | Name of Branch | Tampines |

Table Name: branch\_details

Description: This table stores the content of each Branch. A branch can have multiple content.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idBranchDetails | INT | AUTO INCRE | P | N | Y | Unique Identification of branch\_detail | 1 |
| idBranch | INT | - | F | N | N | Which Branch this detail belongs to | 1 |
| title | VARCHAR | 50 |  | N | N | What this detail is | Phone |
| Content | VARCHAR | 100 |  | N | N | Content of the detail | +65  9999 6666 |

Table Name: list

Description: This table stores both Favourites (FAV) and Bookmarks (BM) of Users – combined 2 similar Tables into 1. ‘type’ property defines if the row is for FAV or BM list.

*Type must be PK for User to add a Restaurant to both FAV and BM.*

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idAccount | INT | - | P, F  (Composite) | N | Y | Unique Identification of Account | 1 |
| idRestaurant | INT | - | P, F  (Composite) | N | N | Unique Identification of Restaurant | 1 |
| type | ENUM | F, B | P  (Composite) | N | N | Whether this Row represents **F**avourite or **B**ookmark | F |
| datetime | DATETIME | - |  | N | N | Date when added to list | 2022-11-29 16:00:00 |

Table Name: searches

Description: This table stores Searches made and relevant details.

By nature of DB design, only concise searches within 50 chars are logged. Users MUST BE LOGGED IN for their Search to be logged.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idSearch | INT | AUTO  INCRE | P | N | Y | Unique Identification of Search | 1 |
| idAccount | INT | - | F | N | N | Which account made that Search | 1 |
| searchQuery | VARCHAR | 50 |  | N | N | Search input: what was searched | Hai Di Lao |
| Datetime | DATETIME | - |  | N | N | Date of Search | 2022-11-29 16:00:00 |

Table Name: reviews

Description: This table stores Reviews of Restaurants

*Id of Restaurant and Account act as* ***Composite Key to prevent User from reviewing the same Restaurant twice***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idRestaurant | INT | - | P, F | N | N | Which Restaurant the review is for | 1 |
| idAccount | INT | - | P, F | N | N | Which Account left the review | 1 |
| rating | ENUM | 1,2,3,4,5 |  | N | N | Rating of Restaurant:  1-5 Stars | 3 |
| review | TEXT | 100 |  | N | N | Summary of Reviewer’s experience on the Restaurant | Great food and Service! I would come back again. |
| date | DATE | - |  | N | N | Date of posted *or* last edited | 2022-29-11 |
| edited | ENUM | T, F |  | N | N | Has row been edited?  True or False | Default Expression: F |

Table Name: comments

Description: This table stores Comments on Reviews or other Comments.

As Reviews Table doesn’t have a unique PK, the idRestaurant and idAccount is taken as the PK. Since Users can Comment unlimited times, the PK of idComment is suitable.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idComment | INT | AUTO INCRE | P | N | Y | Unique Identification of Comment | 1 |
| idRestaurant | INT | - | F | N | N | CK – to identify which Review this Comment is attached to. | 1 |
| idAccount | INT | - | F | N | N | CK – to identify which Review this Comment is attached to. | 1 |
| accountId | INT | - | F | N | Y | Id of Account who made this Comment | 1 |
| commentId | INT | - | F | Y | N | Id of Comment where this Comment is for/at.  If Comment is on a Review, this column will take its default value of 0; There cannot be a Comment id 0. | Default Expression: 0 |
| comment | TEXT | 100 |  | N | N | A comment | I Agree with your opinion on this Restaurant! |
| date | DATE | - |  | N | N | Date of posted *or* last edited | 2022-29-11 |
| edited | ENUM | T, F |  | N | N | Has row been edited?  True or False | Default Expression: F |

Extra:

commentId is a callback to idComment.

Table Name: upvotes

Description: This table stores Likes and Dislikes of Reviews and Comments.

***Composite*** *is required to* ***ensure a User doesn’t vote twice*** *for a particular Review or Comment.*

If Upvote is on a Review, Comment will use its Default Value 0 – still creates a Unique Composite Key while not affecting Comments - since AUTO INCRE starts from 1 there is no idComment which is 0.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Field Name** | **Data Type** | **Field Size** | **Primary/**  **Foreign Key**  **(P/F)** | **Nullable** | **Unique** | **Description** | **Example** |
| idRestaurant | INT | - | P, F  (Composite) | N | N | CK – to identify which Review Upvote is attached to. | 1 |
| idAccount | INT | - | P, F  (Composite) | N | N | CK – to identify which Review Upvote is attached to. | 1 |
| idComment | INT | - | P, F  (Composite) | N | N | ID of Comment the upvote is for  Column takes default value of 0 if idComment is absent | Default Expression:  0 |
| accountId | INT | - | P, F  (Composite) | N | N | ID of User who voted | 1 |
| vote | ENUM | L, D |  | N | N | Like or Dislike | D |

**Reflection**

**Progress:**

As of 2022-6-12 09:43 PM,

Foodie’s SQL ER diagram and Tables Concept are finalized.

8/11 of Tables (All except last Reviews, Comments and Upvotes) are working as intended.

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

**Reflection:**

MySQL was intriguing and it was fascinating to be able to interact with a ‘portable’ Database on my Laptop with Queries.

It took week(s) for me to splice up a working database model, through trial and error from experimenting, especially with all the complexities that arises from many relationships between Tables.

Optimization, Relevance and Simplicity were the design principles for this ambitious database for my project – an advanced database that can be understood by Others fairly quickly.

Im glad to be learning SQL through DBAV as I was previously only able to fantasize a robust database concept in my mind (for COMT) which I couldn’t build. But now, I am able to design a fully functional database architecture and use it.