**Database Technology:**

The database technology that I chose is mySQL relational database located in phpMyAdmin in Apache. I chose this technology because I am hosting my web application on an AWS EC2 Linux instance that is running the Apache server. The Apache server also includes phpMyAdmin and a mySQL database. My service layer is mainly using PHP, so the use of an Apache server was ideal for implementation. The use of a relational database was appropriate since I have a many-to-many relationship between the renters and rentees tables. Also, most of the data in the application will be updated frequently as renters rent out items periodically.

**ER Diagram**

Graphical user interface, text, application

Description automatically generated

**Renters**: The renters table consists of renter information. A renter is a person who rents out an item on the platform. As soon as a user posts an item for rent on the platform, their information is inserted from rentees/users table to the renters table. Renters has a one to many relationship with the transactions table. This means that each renter can be part of many transactions but each transaction can only have one renter, which makes sense. Renters also has a one to many relationship with the items table since each renter can rent out multiple items but each item can only be rented out by one renter. Its fields consist of renter\_id, which is the primary key, name which represents the full name of the renter, username and password of the renter, and street address, city, zip and phone of the renter. This is the required information to post an item as a renter.

**Rentees:** The rentees/users table consists of everyone that signs up for an account on the platform. In addition, it represents a person who rents an item from a renter on the platform. Rentees has a one to many relationship with the transactions table since each rentee can have many transactions but each transaction can only have one rentee. Its fields consist of rentee\_id, which is the primary key, name which represents the full name of the rentee, username and password of the rentee, and street address, city, zip and phone of the rentee. Rentees and renters has a many to many relationship with each other since each rentee can rent items from multiple renters and each renter can rent their items to multiple rentees. To manage this many to many relationship, a linking table between the two is used.

**Transactions:** The transactions table represents the list of rental transactions that take place on the platform. This table helps manage the many-to-many relationship between the renters and rentees table by serving as the linking table between them. The transactions table is the heart of the application as it records all the transaction details. Transactions has a one to many relationship with both renters and rentees table since each transaction can only have one renter and rentee. It has a many to one relationship with the items table since an item can be rented out multiple times. The fields include transaction\_id, which is the primary key of the table, the renter\_id which represents the person renting out the item, the rentee\_id which represents the person renting the item, the item\_id which represents the item being rented out, the item\_name and the rent\_duration which represents how long the item is being rented for.

**Stretch Feature Modification:** In order to accommodate the stretch feature of allowing customers to buy an item, the transactions table includes an additional column called rent\_or\_buy. This column will help differentiate between records that are for rental transactions and for buying transactions

**Items:** The items table consists of the item details of all the items on the platform. An item is added to the table when a new item is posted by a renter from the new posting page. It has a many to one relationship with the renters table since many items can belong to one renter. It has a one to many relationship with the transactions table since each item can be part of many transactions. It has a many to one relationship with the categories table since many items can be part of any one category. The fields include the item\_id, which is the primary key of the table, the renter\_id which represents the renter who is renting out the item, the item name, price, category\_id, the rent\_duration which represents the available rental duration periods for the item, the payment method which represents the accepted payment methods for the item, the contact\_method which represents the contact methods available for the renter of the item, the description of the item and the url of the item, which represents the url path for the item image file, which is used to display the picture of the item across the platform.

**Stretch Feature Modification:** In order to accommodate the stretch feature of allowing customers to buy an item, the items table includes an additional column called buy\_price. This helps record both the rental price and the buy price for an item. These two prices could then be used to populate both the rental item view page and the buy item view page.

**Categories:** The categories table consists of all the item categories that are hosted on the platform. Items related to these categories may be rented out on the platform. The table has a one to many relationship with the items table since each category can have many items. Its fields consist of category\_id, which is the primary key of the table and category\_name which represents the name of the category.