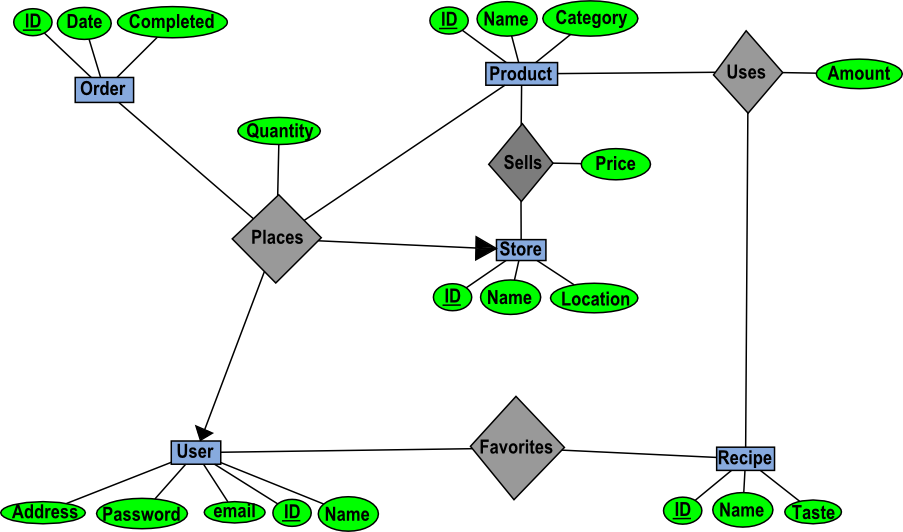
**Project Description:**

* A web-based application where users can buy groceries online and have them delivered to their doorstep.
* Users can choose products from stores around their home.
* Users can order items based on the ingredient list of recipes and can keep a list of favorite recipes.
* The system can provide suggested recipes and products based on their order history
* The system can automatically suggest recipes based on users’ favourite

**ER Diagram:**



**ER Diagram Description:**

* **Entities:**
  + **Orders** (Key: ID number): Orders made by users, has the date the order was placed and whether or not the order is completed.
  + **Product** (Key: ID number): Products that are available at certain stores, has a name and a product category.
  + **Store** (Key: ID number): Store from which products can be purchased, has a name and location.
  + **Recipe** (Key: ID number): Recipe the user can base purchases off of, has a name and a taste/flavor category.
  + **User** (Key: ID number): The user’s account from which they interact with the system, has a name, email, password, and address.
* **Relationships:**
  + **Places:** A user ‘places’ an order. Has a quantity for the product being purchased. Relates a User (one), Store (one), Product (many), and Order (many).
  + **Sells:** A store ‘sells’ a product. Has a price for the product it sells. Relates a Store (many) to a Product (many)
  + **Uses:** A recipe ‘uses’ a product. Has an amount for the product being used. Relates a Recipe (many) to a Product (many).
  + **Favorites:** A user ‘favorites’ a recipe. Relates a User (many) to a recipe (many). (‘favorites’ here means ‘likes’)
* **Domain Constraints:** 
  + **Quantity** (In relation ‘Places’): Must be between 0-100.
  + **Price** (In relation ‘Sells’): Must be between $0.01-$200.
  + **Order:** An order should only be from one user through one store

**Functionalities:**

* User can search/choose products from stores and place order
* User can view their order history in their account
* User can add product to wishlist for later use
* User can update their account info(like email, address)
* User can search recipe based on different tastes/ ingredients
* User can checkout products for delivery from wishlist
* (Note that these are all just examples of how these functionalities will be used. Many of the applications functions will involve a combination of multiple of these concepts.)
* **Inserting Tuples:**
  + User placing an order inserts an order.
  + Users are created by new accounts being established.
  + Products, stores, and recipes can all be added by administrators.
* **Deleting Tuples:**
  + Products, stores, and recipes can all be removed by administrators.
  + Users are deleted by account cancellations.
* **Modifying Tuples:**
  + Orders are modified to update their completion.
  + Store, product, and recipe information can all be modified by administrators.
  + Users can update account information.
* **Joins:**
  + Users can see a compiled list of the orders they have placed.
* **Projections:**
  + Involved when displaying information of tuples to users
* **Selections:**
  + Users can search for product, stores, and recipes with certain properties.
  + Users can search their order history.

**Queries** (Examples of common queries, related to some functionalities listed above)**:**

* (This query searches for products under 10$)

**SELECT** Product.name

**FROM** Product, Sells

**WHERE** (Sells.price < $10) AND

(Sells.productID = Product.ID)

* (This query searches for recipes that use an item entered by the user)

**SELECT** Recipe.name

**FROM** Recipe, Uses

**WHERE** (Product.name = “(user input here)”) AND

(Product.ID = Uses.productID) AND

(Uses.recipeID = Recipe.ID)

* (This query finds all the items for a particular order number)

**SELECT** Product.name

**FROM** Places, Product

**WHERE** (Places.orderID = (order number here)) AND

(Places.productID = Product.ID)

**User-Interface:**

* A web-based interface

**Languages:**

* Java
* PostgreSQL