**Report**

**Problem description**

1. Topic: self-pick-up store
2. Background(difficulty): With the growing number of students and staff at UIC, demand for daily necessities has risen. To meet this need, stores are adapting by establishing self-pickup locations.

3. Goal(benefits) of Self-Pickup Stores:

a. Reduced Staffing Requirements: By eliminating the need for cashiers, shelf organizers, and on-site staff, these stores significantly cut labor costs and relieve employees from the pressure of dealing with customers.

b. Advantages of Interaction-Free Payment: Self-pickup minimizes the time customers spend waiting in line and provides a convenient option for those who prefer not to engage in person-to-person interaction.

c. User privilege management: muti-level employee for company design, employee in different level can have different manipulation function

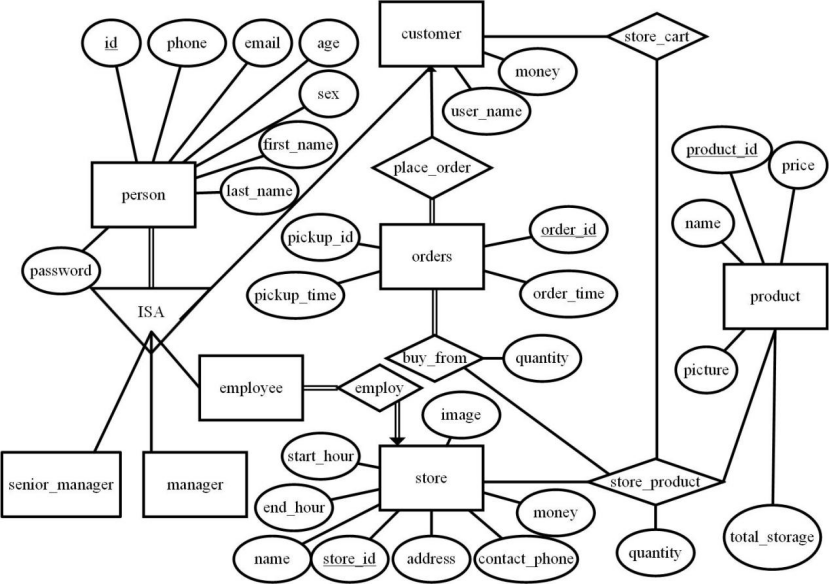
1. **Front end functions**
2. **C-end**

|  |  |
| --- | --- |
| Page | Function |
| Customer registration | 1. Customer can register his/her information to get an identity to login |
| Customer login | 1. Customer use its own identity and password to login |
| Select store page | 1. Customer can enter his/her favourite store to shop (view for product detail) |
| Product detail page | 1. Display the detail information of total products in selected store 2. Click “add to cart” button to add the product into related shopping cart, and disabled the button for out-of-stock products |
| Shopping Cart | 1. Display all products have collected in shopping cart 2. Select box for customer to choose the product to check 3. Count total price of the selected product 4. Page turning & render |
| Order check out page | 1. Check if the store have enough inventory and if the customer have enough money to buy the product |
| Payment completed page | 1. Display the information after order have checked out |
| Personal information page | 1. User can view and update his/her own information such as phone, gender, age and email 2. User can click the “view order” button and jump to “customer individual order page” |
| Customer individual order | 1. User can view all the orders which have been checked out successfully |

1. **B-end**

|  |  |
| --- | --- |
| Page | Function |
| Employee login | 1. Check users’ privilege and level (like a position in a company) 2. User login in have different level |
| Add manager | 1. the only way to add employee into the specified store |
| View pick-up order records | 1. All the users can confirm pickup-time when current customer has picked up all the products in order 2. Manager can view the all the orders in total stores 3. Employee can only view the order in the related store he/she works for |
| View store inventory | 1. Manager can view and update each store inventory 2. Employee can only view the inventory he/she works for |
| View all order records | 1. View orders which checked out in one selected duration time, or sorted by customer |
| View total customer information | 1. Manager can view all information of the customer 2. User can search for identified customer 3. Key word searching 4. Page turning & render |
| View total employee information | 1. Manager can view all information of the employee 2. User can search for identified customer |
| Product information | 1. User can add, update or delete product 2. Upload product image |
| Personal information page | User can view and update his/her own information such as phone, gender, age and email |

1. **Design of database**
2. A person can both be a customer or a position in the store(senior manager, manager, employee), and it should have the attribute of id, phone, email, age, sex, fisrt name, last name, password.
3. A customer should have user name and money.
4. A customer can have many shopping cart with different store and different product.
5. A customer can have many orders.
6. A order should collect the information of customer, order id, order check out time, pick up if(for customer to pick up in each store) and pick up time(when customer have picked the product up).
7. A store should have basic information of id, name, address, contact phone, money, end hour, start hour and image.
8. A store can have many products with quantity.
9. A product should have the basic information of id, name, price picture and the total storage.
10. **ER diagram**
11. **ER Diagram**



1. **Explanation**

**[1] Entities**

1. person

|  |  |
| --- | --- |
| Attribute | Explanation |
| id | Primary key of person |
| phone | Phone of person |
| email | Email of person |
| age | Age of person |
| sex | Sex of person |
| first\_name | First name of person |
| last\_name | Last name of person |
| password | Password of account when login |

1. senior\_manager: The manager with higher permission than manager.
2. manager: The manager of the store and inventory.
3. employee: The employee of the store.
4. customer

|  |  |
| --- | --- |
| Attribute | Explanation |
| user\_name | User name of customer |
| money | The rest money of account |

1. orders: Orders placed by customer.

|  |  |
| --- | --- |
| Attribute | Explanation |
| order\_id | Primary key of orders. |
| pickup\_id | The number is used when customer go to pick up the order items. |
| pickup\_time | Record of pickup time |
| order\_time | Record of order\_time |

1. store

|  |  |
| --- | --- |
| Attribute | Explanation |
| store\_id | Primary Key of store |
| name | Name of store |
| contact\_phone | Contact phone of store |
| start\_hour | The start of working hour of store |
| end\_hour | The end of working hour of store |
| address | The address of store |
| money | The money of the store |
| image | The online cover image of store |

1. product

|  |  |
| --- | --- |
| Attribute | Explanation |
| product\_id | Primary Key of product |
| name | Name of product |
| price | The price sold to customer |
| picture | The online cover picture of product |
| total\_storage | The storage of product in total inventory |

1. **Relationships**
2. ISA: A person can be a staff or a customer or both of them, but they can only have one staff identity.
3. employ: The employee is employed by one store and one store can employ many employee. All employee is employed and all stores have employees. Therefore, it is a many-to-one relationship with both total participation.
4. place\_order: The orders is placed by customer. All the orders is placed by customer but not all customer has placed an order. A customer can place many orders but an order can only be placed by one customer. Therefore, it is a many-to-one relationship with total participation at many side.
5. store\_product: The store restore some product. Not all products are restore in store and not all stores restore the product. A store can restore many products and a product can be restored in many stores. Therefore, it is a many-to-many relationship with both partial participation.

|  |  |
| --- | --- |
| Attribute | Explanation |
| quantity | The quantity of product restored in a store |

1. buy\_from: All products in orders is bought from a store. An order can include many products and a type of product in store can be included in many orders. Therefore, it is a many-to-many relationship with total participation on orders side.

|  |  |
| --- | --- |
| Attribute | Explanation |
| quantity | The quantity of product in the orders |

1. store\_cart: All products stored in shopping cart is from a store. A customer’s shopping cart can have multiple products and a product in store can be in many shopping cart. Therefore, it is a many-to-many relationship with both partial participation.
2. **Schemas and functional dependencies**

**1 Entity Set**

* **person**:  
  + **Explain:**

    - As no functional dependency in contains an extraneous attribute, and each left side of functional dependency in is unique:
    - Decompose the schema using every FD in . Since this FD contains all attributes in the original schema, it cannot be further split.
    - There are no partial or transitive dependencies, so this schema satisfies 3NF.
* **customer**:  
  + **Explain:**

    - There are no partial or transitive dependencies, so this schema satisfies 3NF.
* **employee**:  
  + **Explain:**

    - There are no partial or transitive dependencies, so this schema satisfies 3NF.
* **manager**:  
  + **Explain:**
    - Since is empty (no functional dependencies apply), the table is constructed directly from the candidate key , ensuring the schema satisfies 3NF.
* **senior\_manager**:  
  + **Explain:**
    - Since is empty (no functional dependencies apply), the table is constructed directly from the candidate key , ensuring the schema satisfies 3NF.
* **product**:  
  + **Explain:**

    - There are no partial or transitive dependencies, so this schema satisfies 3NF.
* **store**:  
  + **Explain:**

    - There are no partial or transitive dependencies, so this schema satisfies 3NF.
* **orders**:  
  + **Explain:**

    - There are no partial or transitive dependencies, so this schema satisfies 3NF.

**2 Relationship Set**

* **store\_product**:  
  + **Explain:**

    - There are no partial or transitive dependencies, so this schema satisfies 3NF.
* **buy\_from**:  
  + **Explain:**

    - There are no partial or transitive dependencies, so this schema satisfies 3NF.
* **store\_cart**:  
  + **Explain:**
    - Since is empty (no functional dependencies apply), the table is constructed directly from the candidate keys , ensuring the schema satisfies 3NF.

1. **Primary keys**

|  |  |  |
| --- | --- | --- |
| Schema | Primary key | Description |
| person | id | Identify each person, parent class of “customer”, “employee”, “manager” and “senior\_manager” |
| customer | id | Identify each customer |
| employee | id | Identify employee |
| manager | id | Identify each manager, different to the senior manager |
| senior\_manager | id | Identify each senior manager |
| product | pid | Identify each product |
| store | sid | Identify each store |
| orders | oid | Identify each order |
| store\_product | pid, sid | Each tuple should be identify by one pid and one sid |
| buy\_from | oid, pid, sid | Each tuple should be identify by one oid, one pid and one sid |
| store\_cart | id | Each tuple should be indentify by one person, one pid and one sid |