PSIRT

CSC 263 Final Project
RAM - Roya, Aaron, Michael

Roles

Roya - Frontend (HTML), Styling (CSS), Backend (PHP), Client Functionality, Registration Functionality

Michael - Frontend, Backend, MySQL DB Design/Setup, Handler Functionality, Landing Functionality

Aaron - Frontend, Backend, Sitter Functionality, Login Functionality

Case Analysis

When analyzing the case study, we thought it would be best to split up the application into three dashboards (views):

Client:

- Create orders
- View sitter requests
    - Approve or deny
- Add comments to orders
- Update status of orders
- View order history

Handler:

- See all orders
- Suggest sitters and create sitter requests
- Add/remove available services

Sitter:

- View current jobs
    - Add comments
    - Mark as done/Submit service report
- View previously completed jobs/service reports

This allowed us to keep the application usage for different roles separate, as well as ensuring that different roles were not able to access features that were not intended for them.

Additionally, it also made the development process easier as we all chose a specific view to work on, then coming together to combine them into one singular application.

Schema Design

(With PK and FK)

User (userID [PK], firstname, lastname, type, address, phoneNumber, username, password, emailAddress, ipAddress)

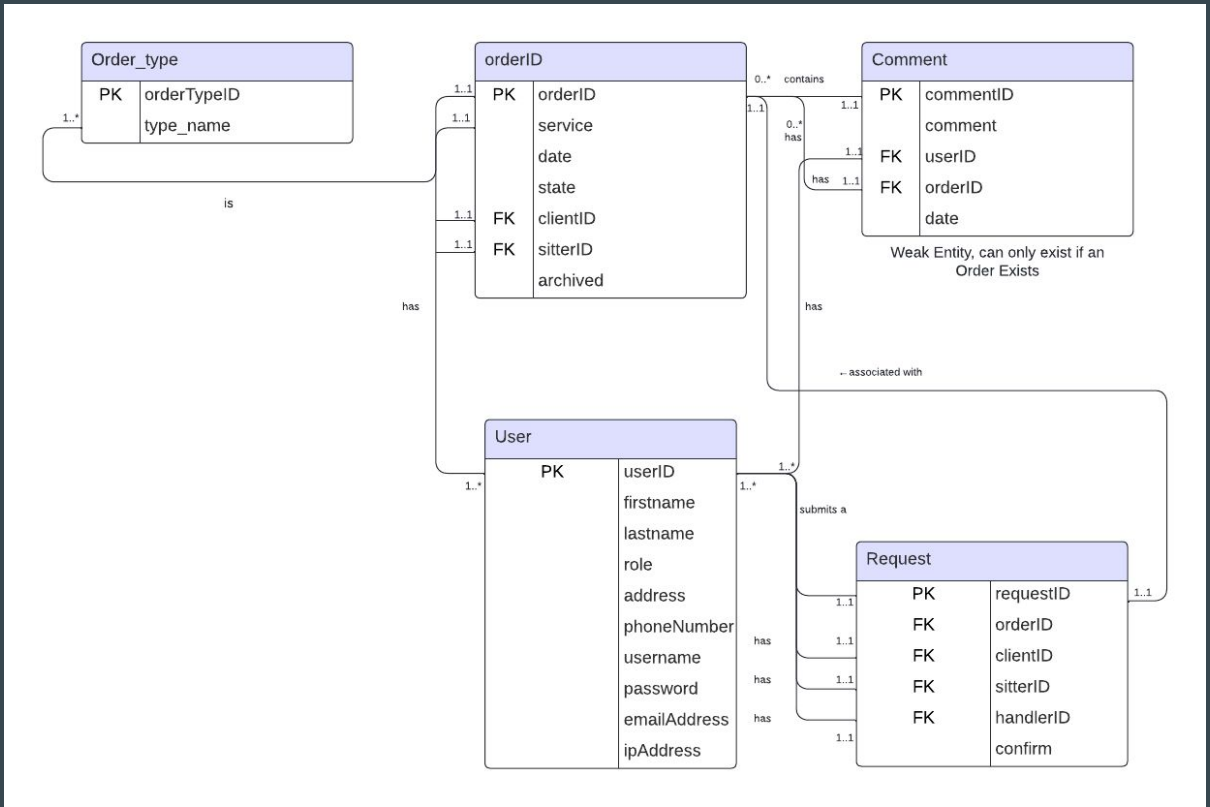
Order (orderID [PK], service, dat, state, clientID [FK], sitterID [FK], archived)

Comment (commentID [PK], comment, userID [FK], orderID [FK], date)

Request (requestID [PK], orderID [FK], clientID [FK], sitterID [FK], handlerID [FK], confirm)

Order_type (orderTypeID [PK], type_name)

Entity Relationship Diagram



Normalization (UNF-1NF)

Unnormalized Form (UNF):

UNF will represent the initial structure of the entities and their attributes.

User (userID [PK], username, password, firstname, lastname, type, address, phoneNumber, emailAddress, ipAddress)

Order (orderID [PK], service, date, state, clientName, clientAddress, clientPhoneNumber, sitterName, sitterAddress, sitterPhoneNumber, archived)

Comment (commentID [PK], comment, userName, orderID [FK], date)

Request (requestID [PK], orderID [FK], clientName, clientAddress, clientPhoneNumber, sitterName, sitterAddress, sitterPhoneNumber, handlerName, handlerAddress, handlerPhoneNumber, confirm)

Order_type (orderTypeID [PK], typeName)

First Normal Form (1NF):

User (userID [PK], firstname, lastname, type, address, phoneNumber, username, password, emailAddress, ipAddress)

Order (orderID [PK], service, dat, state, clientID [FK], sitterID [FK], archived)

Comment (commentID [PK], comment, userID [FK], orderID [FK], date)

Request (requestID [PK], orderID [FK], clientID [FK], sitterID [FK], handlerID [FK], confirm)

Order_type (orderTypeID [PK], type_name)

In this step, we removed repeating groups and made sure each attribute contains only atomic values.

User: no changes were made, the attributes were already atomic

Order: moved client and sitter information to a separate table to avoid repeating groups, added clientID and sitterID as FK

Comment: no changes were made, the attributes were already atomic

Request: moved client, sitter, and handler information to separate tables, added clientID, sitterID, and handlerID as fk

Order_type: no changes were made, the attributes were already atomic

Normalization (2NF-3NF)

Second Normal Form (2NF):

User (userID [PK], firstname, lastname, type, address, phoneNumber, username, password, emailAddress, ipAddress)

Order (orderID [PK], service, dat, state, clientID [FK], sitterID [FK], archived)

Comment (commentID [PK], comment, userID [FK], orderID [FK], date)

Request (requestID [PK], orderID [FK], clientID [FK], sitterID [FK], handlerID [FK], confirm)

Order_type (orderTypeID [PK], type_name)

No changes needed for 2NF: all non-key attributes are fully functionally dependent on the primary key.

Third Normal Form(3NF):

User (userID [PK], firstname, lastname, type, address, phoneNumber, username, password, emailAddress, ipAddress)

Order (orderID [PK], service, dat, state, clientID [FK], sitterID [FK], archived)

Comment (commentID [PK], comment, userID [FK], orderID [FK], date)

Request (requestID [PK], orderID [FK], clientID [FK], sitterID [FK], handlerID [FK], confirm)

Order_type (orderTypeID [PK], type_name)

No changes needed for 3NF: there are no transitive dependencies.

Demo Time!

Thank you for your attention! Any questions?