Database Design of A Food Donation Platform

DATABASE DESIGN AND IMPLEMENTATION OF A FOOD DONATION PLATFORM

PREPARED FOR

Dr. Nkemeni Valery

CEF440 - Internet programming and Mobile programming

PREPARED BY GROUP 15

Randy Susung Nesinyu Kwalar - FE20A101

Niba Godfaith Cedric Fuh - FE20A081

Achale Ebot Oma - FE20A002

Agyingi Jan Royal - FE20A005

Ataba Emmanuel Junior - FE20A013

Table of Contents

INTRODUCTION

Software design is a creative activity in which you identify software components and their relationships, based on a customer's requirements. Implementation is the process of realising the design as a program.

Overview

Let's start with an overview of what is software design in software engineering. The process of creating software methods, functions, objects, and the general structure and interaction of your code such that the resulting functionality meets the needs of your users is known as software design.

Project Scope

The project scope for addressing the food donation problem includes the following:

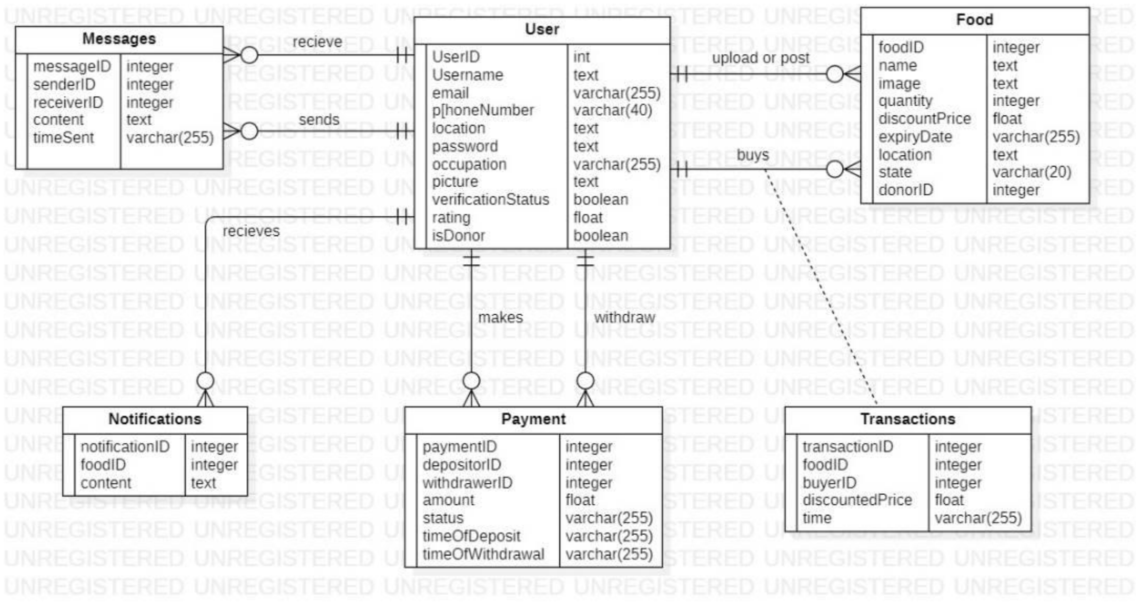
- Identifying food donors: The project will identify individuals or organisations that have excess food and are willing to donate it.
- Partnering with food banks or organisations: The project will partner with food banks or organisations that address hunger and food insecurity, to facilitate the distribution of donated food to those in need.
- Ensuring food safety: The project will ensure that all donated food is safe and of good quality by adhering to safety regulations and guidelines.
- Developing a communication plan: The project will develop a communication plan to effectively communicate the benefits of food donation, how the system works, and how to get involved.

- Setting up an escrow payment system: The project will set up a reliable and efficient payment system to deliver the money to the donor when the food is gotten.
- Monitoring and evaluation: To ensure the project's success, there will be continuous monitoring and evaluation to track progress towards project goals, and identify areas that may need improvement.
- Raising awareness: The project will raise public awareness about the issue, to encourage more people to donate food and reduce food waste.

Conceptual Design

Entity-Relationship Diagram

Below is the Entity-Relationship Diagram (ERD) depicting the entities, attributes, and relationships.



Messages

messageID	integer
senderID	integer
receiverID	integer
content	text
timeSent	varchar(255)

User

UserID	int
Username	text
email	varchar(255)
p[honeNumber	varchar(40)
location	text
password	text
occupation	varchar(255)
picture	text
verificationStatus	boolean
rating	float
isDonor	boolean

Food

foodID	integer
name	text
image	text
quantity	integer
discountPrice	float
expiryDate	varchar(255)
location	text
state	varchar(20)
donorID	integer

Notifications

notificationID	integer
foodID	integer
content	text

Payment

paymentID	integer
depositorID	integer
withdrawerID	integer
amount	float
status	varchar(255)
timeOfDeposit	varchar(255)
timeOfWithdrawal	varchar(255)

Transactions

transactionID	integer
foodID	integer
buyerID	integer
discountedPrice	float
time	varchar(255)

Entities And Fields

User

Fields: userID (primary key), username, email, phoneNumber, location, password, occupation, picture, verificationStatus, rating, isDonor.

This table stores user information such as username, email, password, location, and profile picture.

Food

Fields: foodID (primary key), name, image, quantity, discountPrice, expiryDate, location, state.

This table stores food information such as name, image, quantity, discount price.

Messages

Fields: messageID (primary key), senderID, receiverID, content, timeSent.

This table stores messages information such as the content, the sender and receiver of the message.

Payment

Fields: paymentID (primary key), depositorID, withdrawerID, amount, status, timeOfDeposit, timeOfWithdrawal.

This table stores payment information such as the amount, the depositor and withdrawer and the time it was done.

Notifications

Fields: notificationID (primary key), foodID, content.

This table stores notification information such as the content and the user it is meant for.

Transactions

Fields: transactionID (primary key), foodID, buyerID, discountedPrice, time.

This table stores transaction information such as the food to be purchased, the buyer and withdrawer and the time it was done.

Relationships between Entities

User - Food

There exists a one-to-many relationship between users and their selected foods.

A user can either buy or upload several foods but a food could be purchased or uploaded by only one user.

In the case of a user buying food, another table is created called `transactions` that will hold a detailed record of the process.

User - Messages

There exists a one-to-many relationship between users and messages.

A user can either send or receive several messages but a message could only be sent or received by only one user.

User - Payment

There exists a one-to-many relationship between users and payment.

A user can either make or receive several payments but a payment could only be made or withdrawn by only one user.

User - Notification

There exists a one-to-many relationship between users and notification.

A user can receive many notifications but a notification is only intended for one user.

Logical Design

Below is the physical database schema mapping the UML design to database tables and columns

User Table

Column Name	Data Type	Constraints
userID	SERIAL	PRIMARY KEY
username	TEXT	NOT NULL
email	VARCHAR(255)	NOT NULL
phoneNumber	VARCHAR(40)	
location	TEXT	
password	TEXT	NOT NULL
occupation	VARCHAR(255)	
picture	TEXT	
verificationStatus	BOOLEAN	NOT NULL
rating	FLOAT	
isDonor	BOOLEAN	NOT NULL

Food Table

Column Name	Data Type	Constraints
foodID	SERIAL	PRIMARY KEY
name	TEXT	NOT NULL

image	TEXT	NOT NULL
quantity	INTEGER	NOT NULL
discountPrice	FLOAT	
expiryDate	VARCHAR(255)	
location	TEXT	NOT NULL
state	VARCHAR(255)	NOT NULL
donorID	INTEGER	FOREIGN KEY (user.userID)

Messages Table

Column Name	**Data Type**	**Constraints**
messageID	SERIAL	PRIMARY KEY
senderID	INTEGER	FOREIGN KEY (user.userID)
receiverID	INTEGER	FOREIGN KEY (user.userID)
content	TEXT	NOT NULL
timeSent	VARCHAR(255)	NOT NULL

Payment Table

Column Name	**Data Type**	**Constraints**
paymentID	SERIAL	PRIMARY KEY
depositorID	INTEGER	FOREIGN KEY (user.userID)
withdrawerID	INTEGER	FOREIGN KEY (user.userID)
amount	FLOAT	NOT NULL
status	VARCHAR(255)	NOT NULL

timeOfDeposit	VARCHAR(255)	NOT NULL
timeOfWithdrawal	VARCHAR(255)	

Notifications Table

Column Name	**Data Type**	**Constraints**
notificationID	SERIAL	PRIMARY KEY
foodID	INTEGER	FOREIGN KEY (food.foodID)
content	TEXT	NOT NULL

Transactions Table

Column Name	**Data Type**	**Constraints**
transactionID	SERIAL	PRIMARY KEY
foodID	INTEGER	FOREIGN KEY (food.foodID)
buyerID	INTEGER	FOREIGN KEY (user.userID)
discountedPrice	FLOAT	
time	VARCHAR(255)	NOT NULL

Physical Design and Implementation

Creation of 'user' table

```
CREATE TABLE user (
        userID SERIAL PRIMARY KEY,
        username TEXT NOT NULL,
        email VARCHAR(255) NOT NULL,
        phoneNumber VARCHAR(40),
        location TEXT,
        password TEXT NOT NULL,
        occupation VARCHAR(255),
        picture TEXT,
        verificationStatus BOOLEAN NOT NULL,
        rating FLOAT,
        isDonor BOOLEAN NOT NULL
);
```

Creation of ‘food’ table

```
CREATE TABLE food (
        foodID SERIAL PRIMARY KEY,
        name TEXT NOT NULL,
        image TEXT NOT NULL,
        quantity INTEGER NOT NULL,
        discountPrice FLOAT,
        expiryDate VARCHAR(255),
        location TEXT NOT NULL,
        state VARCHAR(255) NOT NULL,
        donorID INTEGER REFERENCES user(userID)
);
```

Creation of `messages’ table

```
CREATE TABLE messages (
        messageID SERIAL PRIMARY KEY,
        senderID INTEGER REFERENCES user(userID),
        receiverID INTEGER REFERENCES user(userID),
        content TEXT NOT NULL,
        timeSent VARCHAR(255) NOT NULL
);
```

Creation of `payment’ table

```
CREATE TABLE payment (
        paymentID SERIAL PRIMARY KEY,
        depositorID INTEGER REFERENCES user(userID),
        withdrawerID INTEGER REFERENCES user(userID),
        amount FLOAT NOT NULL,
        status VARCHAR(255) NOT NULL,
        timeOfDeposit VARCHAR(255) NOT NULL,
        timeOfWithdrawal VARCHAR(255)
);
```

Creation of ‘notifications’ table

```
CREATE TABLE notifications (
        notificationID SERIAL PRIMARY KEY,
        foodID INTEGER REFERENCES food(foodID),
        content TEXT NOT NULL
);
```

Creation of ‘transactions’ table

```
CREATE TABLE transactions (
        transactionID SERIAL PRIMARY KEY,
        foodID INTEGER REFERENCES food(foodID),
        buyerID INTEGER REFERENCES user(userID),
        discountedPrice FLOAT,
        time VARCHAR(255) NOT NULL
);
```

After executing these SQL statements in your PostgreSQL database, the necessary tables and columns will be created, allowing you to interact with the database using appropriate CRUD (Create, Read, Update, Delete) operations.

Security and Access Control

Measures that were taken to ensure security and access control include:

- **Data Validity:** Enforcing non-null constraints and appropriate data types for fields to ensure data integrity.

- **Foreign Key Constraints:** Enforcing referential integrity by ensuring that foreign keys reference valid primary key values in their respective tables.

- **Unique Constraints:** Ensuring that usernames and email addresses are unique to avoid duplicate accounts.

- **Messaging Rules:** Implementing rules to handle message sending, such as ensuring that only users involved in a conversation can send messages and validating message content.

- **Privacy and Security:** Implementing appropriate security measures to protect user data, including hashing and securely storing passwords, protecting user conversations, and securing sensitive information.

- **User Consent and Privacy Settings:** Implementing mechanisms for obtaining user consent to store and process their data, as well as providing privacy settings for users to control the visibility of their profiles and information.

- **Reporting and Moderation:** Implementing reporting mechanisms to handle abusive or inappropriate behaviour, and enforcing moderation policies to maintain a safe and respectful environment.

These measures ensure the platform's functionality, data integrity, user privacy, and overall user experience.

Conclusion

In conclusion, the design and implementation of a PostgreSQL database is complex and requires expertise in database administration, programming, and performance testing. However, a reliable and scalable database system will lead to the development of high-performing food donation platform software that meets the needs of its users.