A DBMS Solution for Catering in Colleges

Submitted by :

* Arjun Srivastava : 160001007
* Sahaj Khandelwal : 160001052

Project Guide : Mrs. Aruna Tiwari

Problem Statement

We provide a solution to the vows of the people in IITI. The current system includes a tedious task of verifying individual QR codes, and there is no record available to Our system proposes a method to turn it into transactions through the SmartCards,and hence in our solution, we make a PWA(Progressive Web Application),and change the payment method to SmartCards.

Web-app features

- Tell the user about upcoming meals.
  - What all the meal contains.
  - Meal price
  - Meal rating
- Show meal history.
  - All the meals the user has had.
  - And how much it cost him/her.
- Subscriptions.
  - Show the diner all of his/hers subscriptions with the corresponding caterers.
- Reviews.
  - Every diner has the facility to rate the meal. They can express their opinions on the caterer and the meal served through the ratings.

Payment

- Payment Device
  - Will use the SmartCard to make a transaction. It will also provide the time and the caterer.

Caterer

- Billing
  - We will provide the relevant transactions to the caterer.
  - He will deduct money monthly.
  - He will be able to see which meals are popular and how many people have them.
  - We also provide a price calculator which allow them to fix prices for meals.

User prerequisites

Every student has to register with the system in order to be eligible to dine at the mess. As a user registers, he/she provides -

- Name
- Date of birth

- Year
- Course (B.Tech/M.Tech/PhD/Guests)
- Subscription (Monthly/Daily/Semester)
- Email ID
- Phone numbers

What the user wants

- Every time a diner goes to eat,they walk in and press their Smart card against the reader.
- The computer automatically detects the right amount, and it checks the following -
  - If the person has already had the meal today.
  - Their balance is enough.
  - What Subscription are they on.
- And then, on successful verification of the diner, the transaction is recorded, with instant calculation of the price.

Caterer prerequisites

- Every caterer has to register to the system and will obtain a unique CatererID.
- Then the caterer has to opt for the plans to be made available for subscription to the diners, along with the price details.
- The caterer then, has to list out the products,that will be available to be served,and hence, update the menu.
- The caterer will also notify about the timings at which the diners can avail the dining facilities.

What the caterer wants

- The caterer will have the access to the transactions made every time a diners avails the facility.
- The caterer is privileged with the facility of changing the pricing scheme, along with surge pricing.
- A caterer can also update the menu.

App Features

- Balance
- Menu
- How much you have paid.
- How many people are currently in the mess(tentatively).
- They get a notification on the app whenever they make a transaction.
- They are allowed to rate and review a meal.
- They can also know the average rating of the meal before they even leave for it.

Identifying constraints

- The Caterer can only append to the tables,but can't delete the previously existing entries. This will allow us to show which servings were eaten previously, as the Menu might keep changing as per the mess committee and caterer's will.
- We will also calculate and insert price with the a transaction so that we don't have errors when the caterer updates the prices, later on.
- We will allow a person to have just one meal from a caterer at a time (no double dinners.)

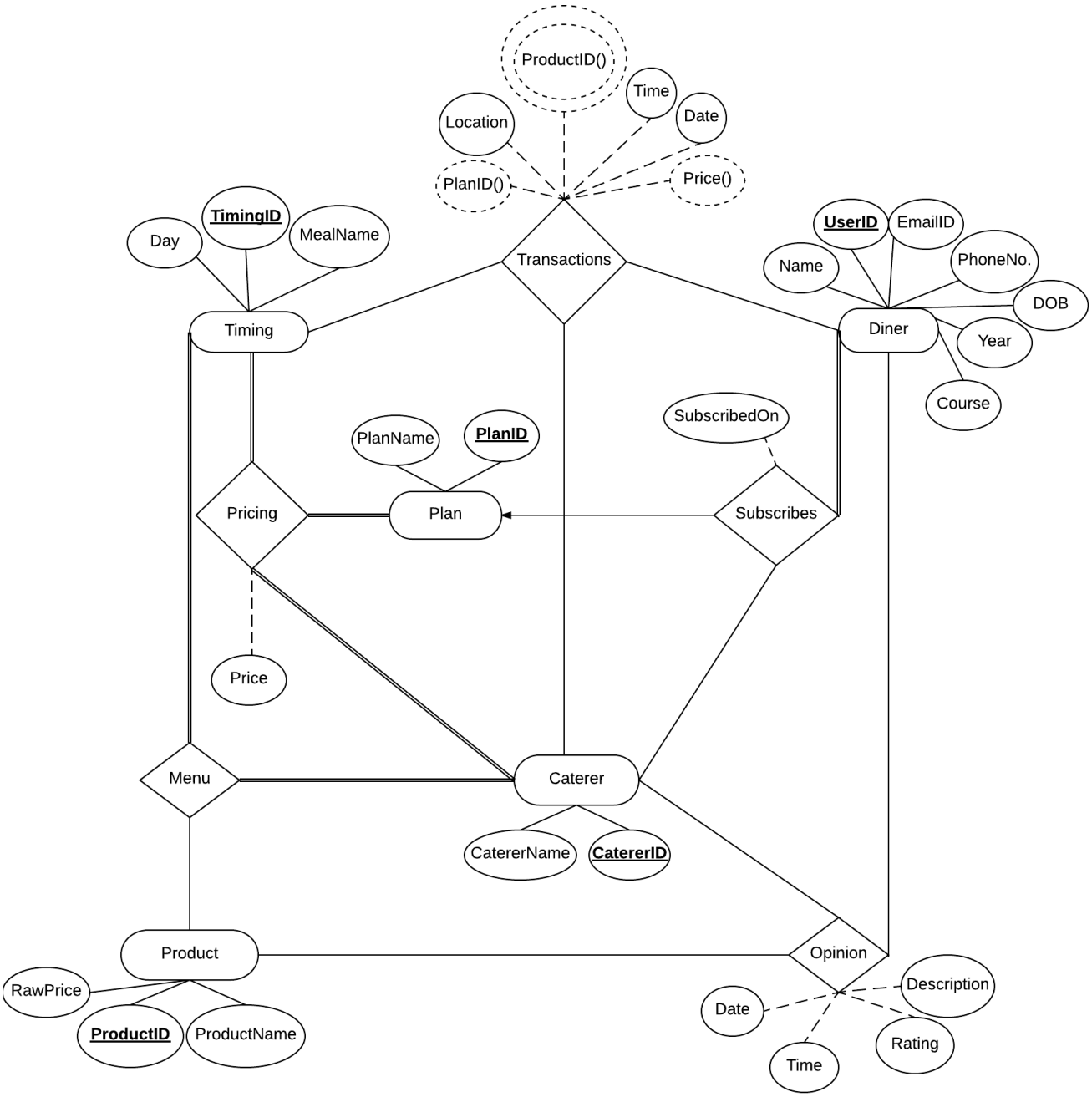
Entities

- Diner
- Plan
- Caterer
- Product
- Timing

Relations

- Transaction(Diner, Caterer, Timing)
- Opinions (Diner, Caterer , Product)
- Subscribes (Diner, Caterer, Plan)
- Menu (Timing, Caterer, Product)
- Pricing (Timing, Caterer, Plan)

The ER Diagram



Tables

- Diner(UserID , Name, DOB, Year, Course, EmailID, PhoneNo.)
- Plan (PlanID, PlanName)
- Caterer (CatererID, CatererName)
- Product (ProductID,ProductName, RawPrice)
- Timing (TimingID,Day,MealName)

- Subscribes (UserID,CatererID, PlanID, SubscirbedOn)
- Opinions (ProductID, CatererID, UserID , Rating , Description, Time, Date)
- Menu (TimingID,ProductID, CatererID)
- Pricing (CatererID,PlanID,TimingID, Price)
- Transaction (UserID,CatererID,TimingID, TransactionID, Date , Time, Location , PlanID(), Price())
- Served(TransactionID, ProductID)

Constraints

Foreign Key Dependencies

- Subscribes.CatererID references Caterer.CatererID
- Subscribes.UserID references Diner.UserID
- Subscribes.PlanID references Plan.PlanID
- Opinions.ProductID references Product.ProductID
- Opinions.CatererID references Caterer.CatererID
- Opinions.UserID references Diner.UserID
- Menu.CatererID references Caterer.CatererID
- Menu.ProductID references Product.ProductID
- Menu.TimingID references Timing.TimingID
- Pricing.CatererID references Caterer.CatererID
- Pricing.PlanID references Plan.PlanID
- Pricing.TimingID references Timing.TimingID
- Transaction.UserID references Diner.UserID
- Transaction.CatererID references Caterer.CatererID
- Transaction.TimingID references Timing.TimingID
- Served.TransactionID references Transaction.TransactionID

Primary Key Dependencies

- UserID Name -> DOB, Year, Course, EmailID, PhoneNo.
- PlanID -> PlanName
- CatererID -> CatererName
- ProductID -> ProductName, RawPrice
- TimingID -> Day, MealName
- UserID,CatererID -> PlanID, SubscirbedOn
- ProductID, CatererID, UserID -> Rating , Description, Time, Date
- CatererID,PlanID,TimingID -> Price
- UserID,CatererID,TimingID -> TransactionID, Date , Time, Location , PlanID , Price
- TransactionID -> ProductID

Triggers

- We shall calculate price upon insertion of transaction.
- We will check if the Diner's subscription is valid whenever we goes to eat, if not valid, we will give error.