A DBMS Solution for Catering in Colleges

Submitted by :

* Sahaj Khandelwal : 160001052

* Arjun Srivastava : 160001007

* 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.
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.

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,accordingly.

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 -
    * Their balance is enough.
    * What Subscription are they on.
    * The diner is then verified,according to the database of the Caterer.
* 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.
* A caterer can also update the menu.

App Features

* Menu
* How much you have paid.
* 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 serving was eaten even after the Menu changes.
* 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.)

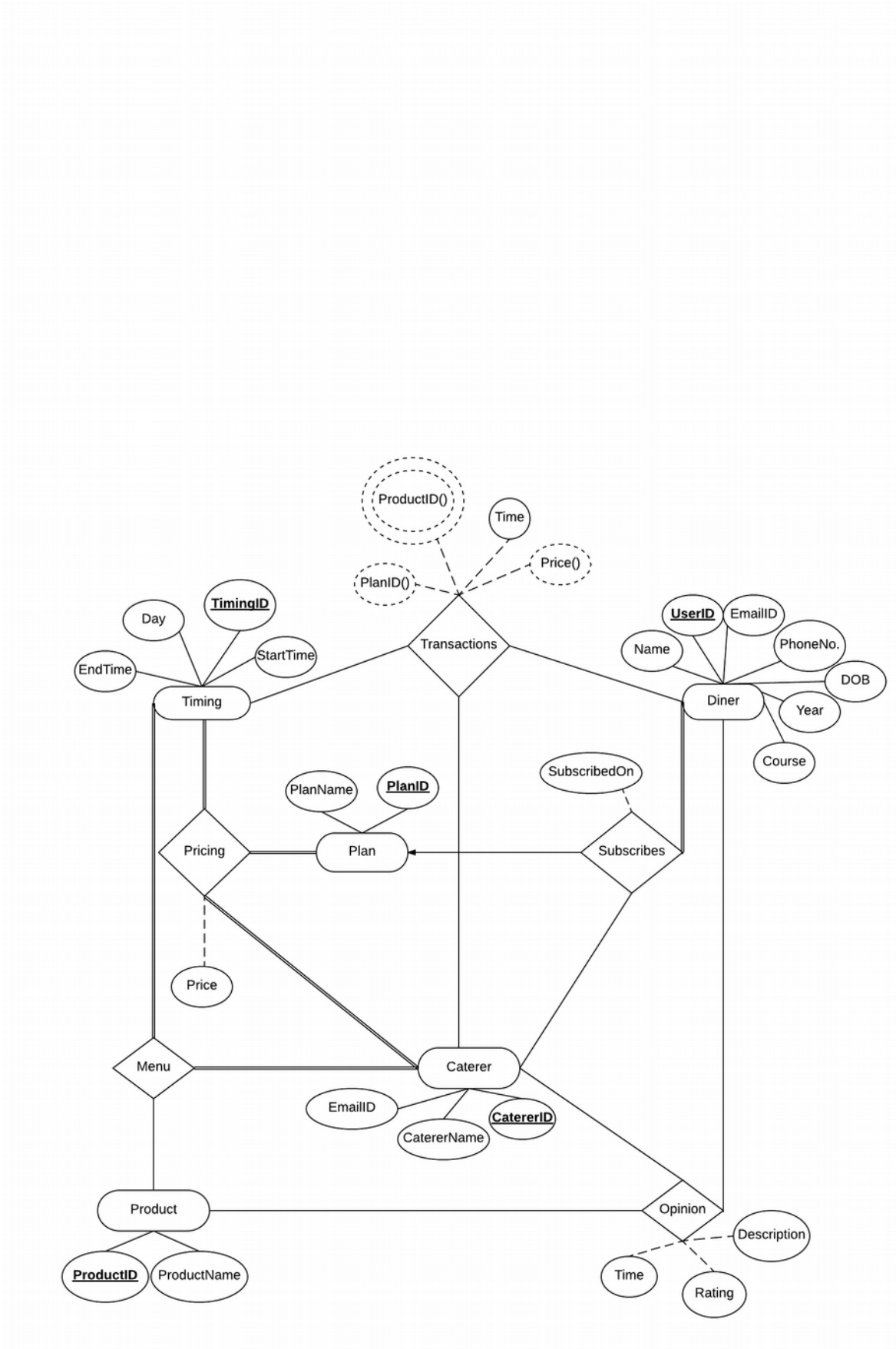
Entities

* Diner
* Plan
* Caterer
* Product
* Timing

Relations

* Transactions(Diner, Caterer, Timing)
* Opinions (Diner, Caterer , Product)
* Subscribes (Diner, Caterer, Plan)
* Menu (Timing, Caterer, Product)
* Pricing (Timing, Caterer, Plan)

The ER Diagram



Tables

* Diner(<u> UserID </u>, Name, DOB, Year, Course, EmailID, PhoneNo.)
* Plan (<u>PlanID</u>, PlanName)
* Caterer (<u>CatererID</u>, CatererName,EmailID)
* Product (<u>ProductID</u>,ProductName)
* Timing (<u>TimingID</u>,Day,StartTime,EndTime)
* Subscribes (<u>UserID,CatererID</u>, PlanID, SubscirbedOn)
* Opinions (<u>ProductID, CatererID, UserID </u>, Rating , Description, Time)
* Menu (<u>TimingID,ProductID, CatererID</u>)
* Pricing (<u> CatererID,PlanID,TimingID</u>, Price)
* Transactions (<u>UserID,CatererID,Time</u>, PlanID(), Price())

Constraints

Foreign Key Dependecies

* 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
* TimingID -> Day
* UserID,CatererID -> PlanID, SubscirbedOn
* ProductID, CatererID, UserID -> Rating , Description, Time
* CatererID,PlanID,TimingID -> Price
* UserID,CatererID,Time -> PlanID , Price
* TransactionID -> ProductID

Creating the Database and the Schemas

* Caterer

```
DROP TABLE IF EXISTS `Caterer`;

CREATE TABLE `Caterer` (
  `CatererID` int(11) NOT NULL AUTO_INCREMENT,
  `CatererName` varchar(32) NOT NULL,
  `EmailID` varchar(30) NOT NULL,
  PRIMARY KEY (`CatererID`),
  UNIQUE KEY `EmailID` (`EmailID`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
```

* Diner

```
DROP TABLE IF EXISTS `Diner`;
CREATE TABLE `Diner` (
  `FirstName` varchar(20) NOT NULL,
  `LastName` varchar(20) DEFAULT NULL,
  `UserID` int(11) NOT NULL AUTO_INCREMENT,
  `RollNo` int(11) NOT NULL,
  `EmailID` varchar(32) NOT NULL,
  `PhoneNo` int(11) NOT NULL,
  `DOB` date DEFAULT NULL,
  `YearOfStudy` int(11) DEFAULT NULL,
  `Course` varchar(20) NOT NULL,
```

```
  `CardDetails` varchar(50) DEFAULT NULL,
  PRIMARY KEY (`UserID`),
  UNIQUE KEY `CardDetails` (`CardDetails`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=latin1;
```

* Menu

```
DROP TABLE IF EXISTS `Menu`;
CREATE TABLE `Menu` (
  `TimingID` int(11) NOT NULL,
  `CatererID` int(11) NOT NULL,
  `ProductID` int(11) NOT NULL,
  PRIMARY KEY (`ProductID`,`CatererID`,`TimingID`),
  KEY `TimingID` (`TimingID`),
  KEY `CatererID` (`CatererID`),
  CONSTRAINT `Menu_ibfk_1` FOREIGN KEY (`TimingID`) REFERENCES
`Timing` (`TimingID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Menu_ibfk_2` FOREIGN KEY (`CatererID`) REFERENCES
`Caterer` (`CatererID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Menu_ibfk_3` FOREIGN KEY (`ProductID`) REFERENCES
`Product` (`ProductID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

* Opinion

```
DROP TABLE IF EXISTS `Opinions`;
CREATE TABLE `Opinions` (
  `Description` text,
  `Rating` int(11) DEFAULT NULL,
  `UserID` int(11) NOT NULL,
  `ProductID` int(11) NOT NULL,
  `CatererID` int(11) NOT NULL,
  PRIMARY KEY (`ProductID`,`CatererID`,`UserID`),
  KEY `CatererID` (`CatererID`),
  KEY `UserID` (`UserID`),
  CONSTRAINT `Opinions_ibfk_1` FOREIGN KEY (`CatererID`)
REFERENCES `Caterer` (`CatererID`) ON DELETE CASCADE ON UPDATE
CASCADE,
  CONSTRAINT `Opinions_ibfk_2` FOREIGN KEY (`UserID`) REFERENCES
`Diner` (`UserID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Opinions_ibfk_3` FOREIGN KEY (`ProductID`)
REFERENCES `Product` (`ProductID`) ON DELETE CASCADE ON UPDATE
CASCADE
```

```
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

* Plan

```
DROP TABLE IF EXISTS `Plan`;
CREATE TABLE `Plan` (
  `PlanID` int(11) NOT NULL AUTO_INCREMENT,
  `PlanName` varchar(30) NOT NULL,
  `PlanLength` int(11) DEFAULT NULL,
  PRIMARY KEY (`PlanID`)
) ENGINE=InnoDB AUTO_INCREMENT=2 DEFAULT CHARSET=latin1;
```

* Pricing

```
DROP TABLE IF EXISTS `Pricing`;
CREATE TABLE `Pricing` (
  `CatererID` int(11) NOT NULL,
  `PlanID` int(11) NOT NULL,
  `TimingID` int(11) NOT NULL,
  `Price` double DEFAULT NULL,
  PRIMARY KEY (`CatererID`,`PlanID`,`TimingID`),
  KEY `PlanID` (`PlanID`),
  KEY `TimingID` (`TimingID`),
  CONSTRAINT `Pricing_ibfk_1` FOREIGN KEY (`CatererID`) REFERENCES
`Caterer` (`CatererID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Pricing_ibfk_2` FOREIGN KEY (`PlanID`) REFERENCES
`Plan` (`PlanID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Pricing_ibfk_3` FOREIGN KEY (`TimingID`) REFERENCES
`Timing` (`TimingID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

* Product

```
DROP TABLE IF EXISTS `Product`;
CREATE TABLE `Product` (
  `ProductID` int(11) NOT NULL AUTO_INCREMENT,
  `ProductName` varchar(32) NOT NULL,
  PRIMARY KEY (`ProductID`)
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=latin1;
```

* Subscribes

```
DROP TABLE IF EXISTS `Subscribes`;
CREATE TABLE `Subscribes` (
  `PlanID` int(11) DEFAULT NULL,
  `UserID` int(11) NOT NULL,
  `CatererID` int(11) NOT NULL,
  `SubscribedOn` date DEFAULT NULL,
  PRIMARY KEY (`UserID`,`CatererID`),
  KEY `PlanID` (`PlanID`),
  KEY `CatererID` (`CatererID`),
  CONSTRAINT `Subscribes_ibfk_1` FOREIGN KEY (`PlanID`) REFERENCES
`Plan` (`PlanID`) ON DELETE CASCADE ON UPDATE CASCADE,
  CONSTRAINT `Subscribes_ibfk_2` FOREIGN KEY (`CatererID`)
REFERENCES `Caterer` (`CatererID`) ON DELETE CASCADE ON UPDATE
CASCADE,
  CONSTRAINT `Subscribes_ibfk_3` FOREIGN KEY (`UserID`) REFERENCES
`Diner` (`UserID`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

* Timing

```
DROP TABLE IF EXISTS `Timing`;
CREATE TABLE `Timing` (
  `TimingID` int(11) NOT NULL AUTO_INCREMENT,
  `Day` varchar(10) NOT NULL,
  `StartTime` time NOT NULL,
  `EndTime` time NOT NULL,
  PRIMARY KEY (`TimingID`)
) ENGINE=InnoDB AUTO_INCREMENT=6 DEFAULT CHARSET=latin1;
```

* Transactions

```
DROP TABLE IF EXISTS `Transactions`;
CREATE TABLE `Transactions` (
  `CatererID` int(11) NOT NULL,
  `UserID` int(11) NOT NULL,
  `Time` time NOT NULL,
  `PlanID` int(11) DEFAULT NULL,
  `Price` double DEFAULT NULL,
  PRIMARY KEY (`CatererID`,`Time`,`UserID`),
  KEY `UserID` (`UserID`),
```

```
    CONSTRAINT `Transactions_ibfk_1` FOREIGN KEY (`CatererID`)
REFERENCES `Caterer` (`CatererID`) ON DELETE CASCADE ON UPDATE
CASCADE,
    CONSTRAINT `Transactions_ibfk_2` FOREIGN KEY (`UserID`)
REFERENCES `Diner` (`UserID`) ON DELETE CASCADE ON UPDATE
CASCADE
  ) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Triggers

```
* CREATE TRIGGER `transaction_values` BEFORE INSERT ON `Transactions`
 FOR EACH ROW BEGIN

DECLARE pid int;
set pid =
(SELECT
  Subscribes.PlanID
FROM
  Subscribes
WHERE
   Subscribes.CatererID = new.CatererID AND
   Subscribes.UserID = new.UserID) ;
SET new.PlanID = pid;

set new.Price =
(SELECT
  Pricing.Price
FROM
  Pricing
WHERE
  Pricing.CatererID = new.CatererID AND
  Pricing.PlanID = pid AND
  Pricing.TimingID = (select
                        TimingID
                    from
                        Timing
                    where
                        Timing.Day=DAYNAME(cast(new.Time as date)) AND
                        Timing.StartTime <= cast(new.Time as time) AND
                        Timing.EndTime >= cast(new.Time as time)
                ));
```

END

* We shall calculate price upon insertion of transaction.
* We shall also calculate the Plan,the user is subscribed to under the Caterer,which is also fired upon insertion of transaction.

* CREATE TRIGGER `update_sub_on` BEFORE INSERT ON `Subscribes`
 FOR EACH ROW
 begin
    set new.SubscribedOn = NOW();
 end

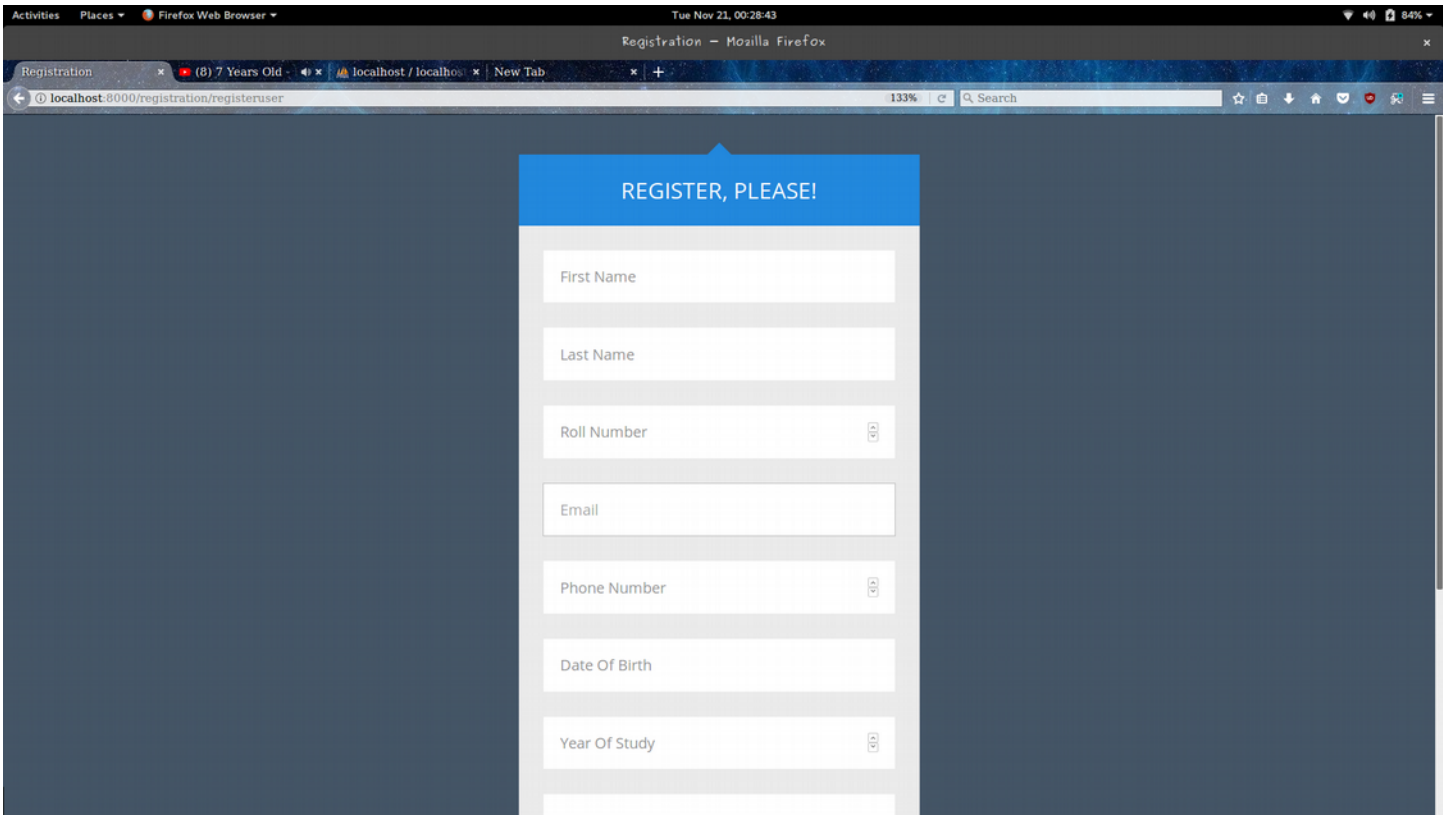
* The above trigger automatically updates the Subscribes table by inserting SubscribedOn value as the current date.

Features provided by the PWA

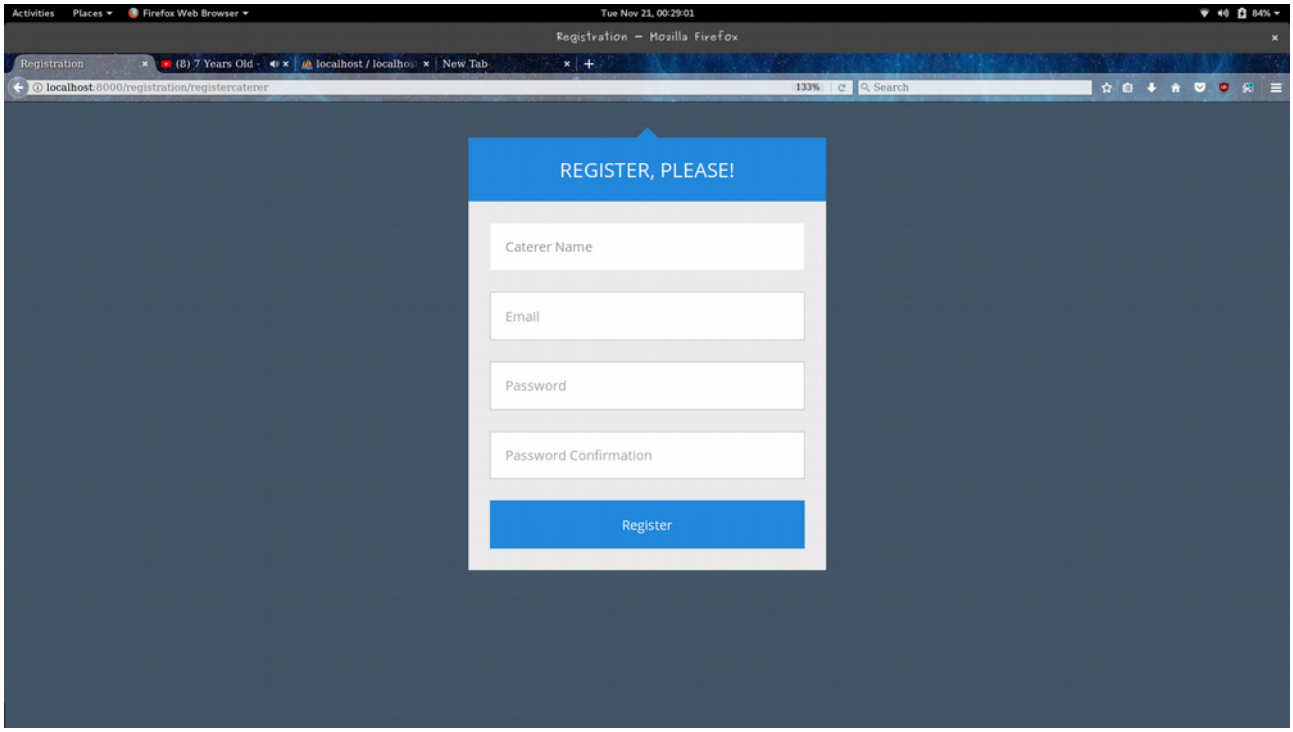
Registration

The caterer/diner will register to the database through the registration page. They will enter the details as required,and, hence will be registered to the database through their EmailIDs.
Whether or not,a particular user is subscribed to any caterer,he/she has to register to the database.

Registration For Diner

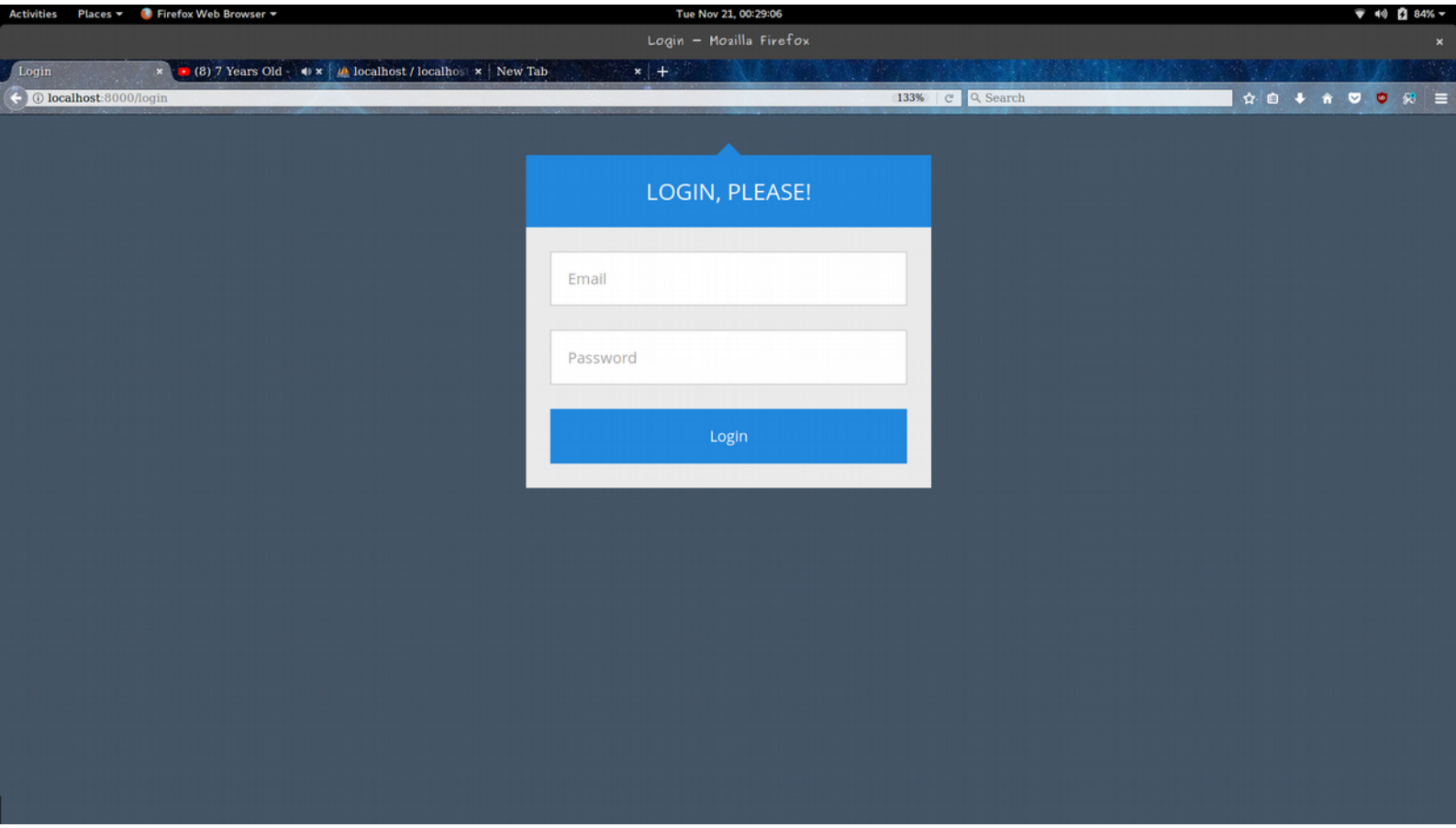


Registration For Caterer

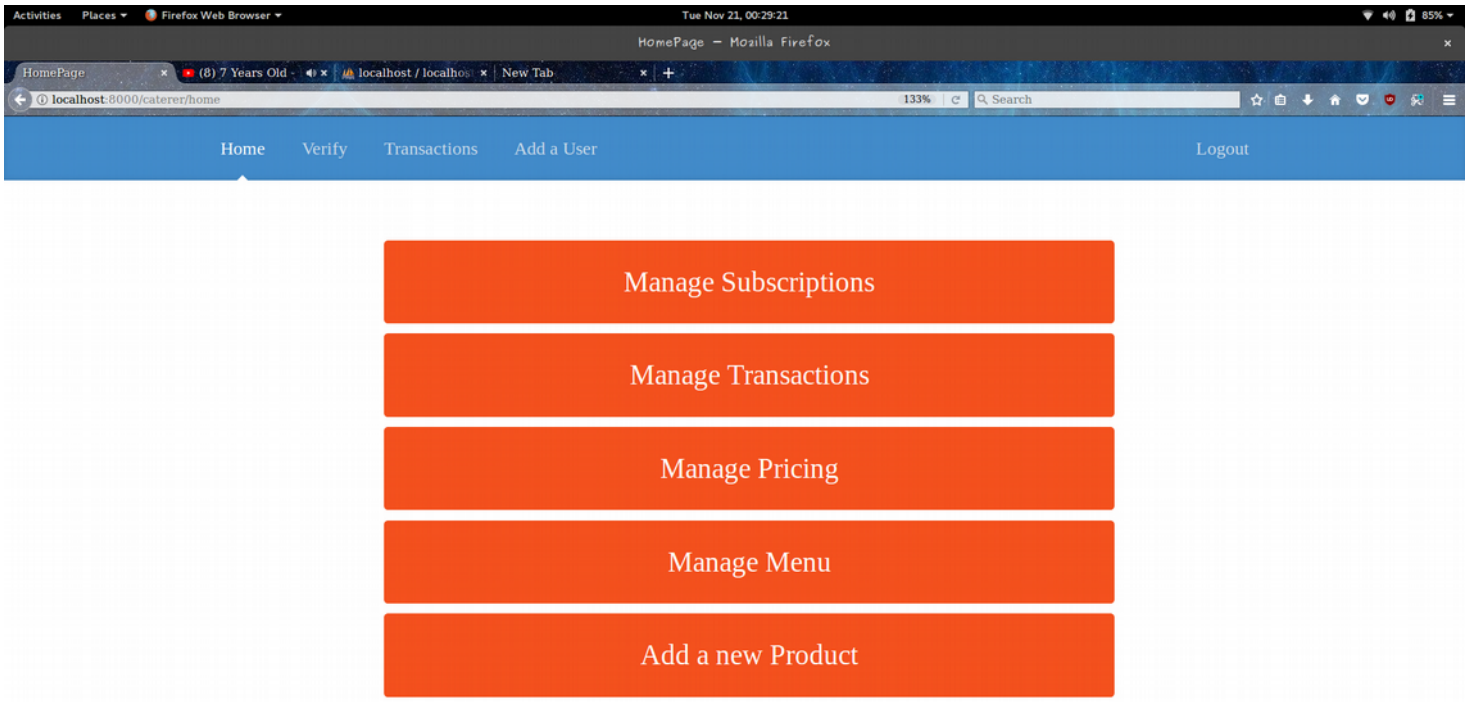


Login Page

The Diner and Caterer,both, are required to log in to their accounts by entering the required details on the login page,i.e., their EmailID and Password.



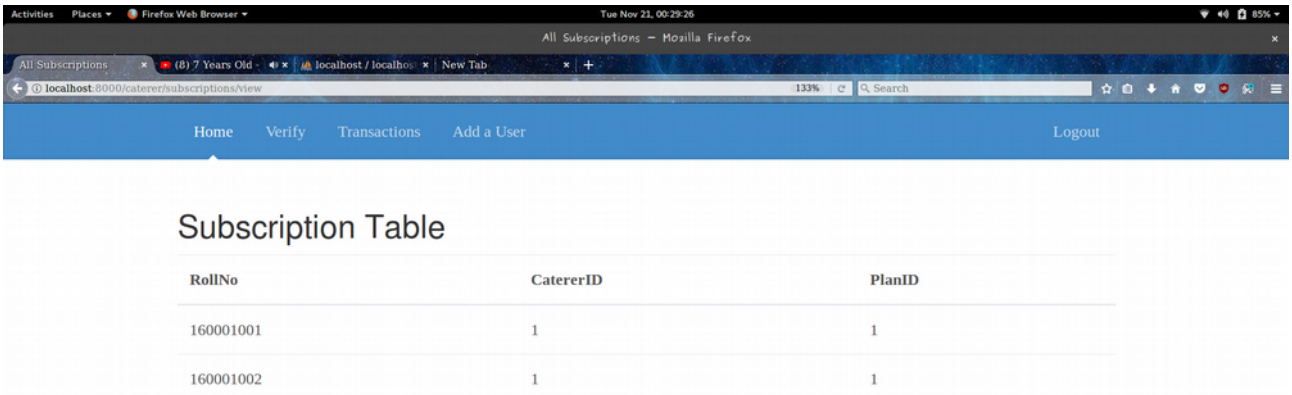
Caterer's DashBoard



Accordingly, the Caterer will perform the desired operations :

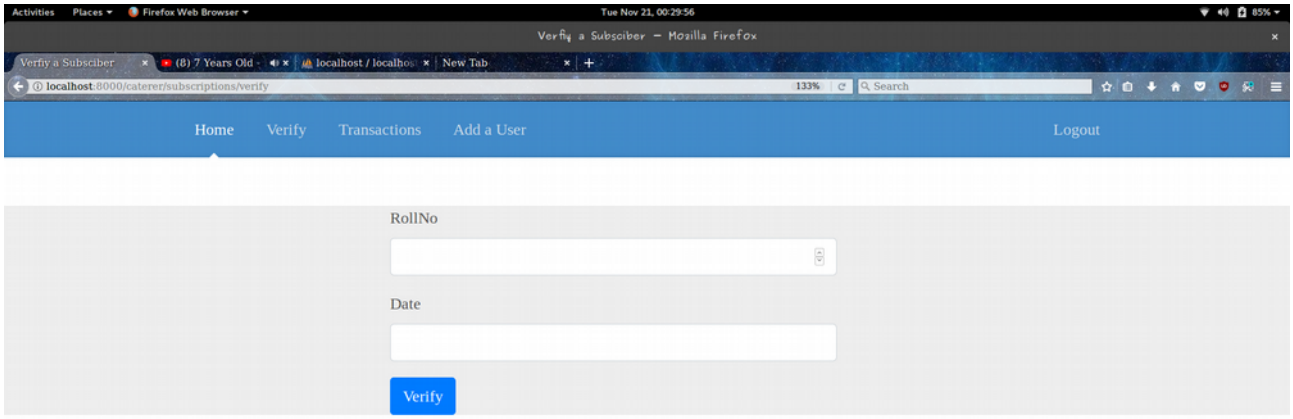
Managing the Subscriptions

The Caterer can view all the subscriptions along with their PlanIDs and the Subscription dates on this page.



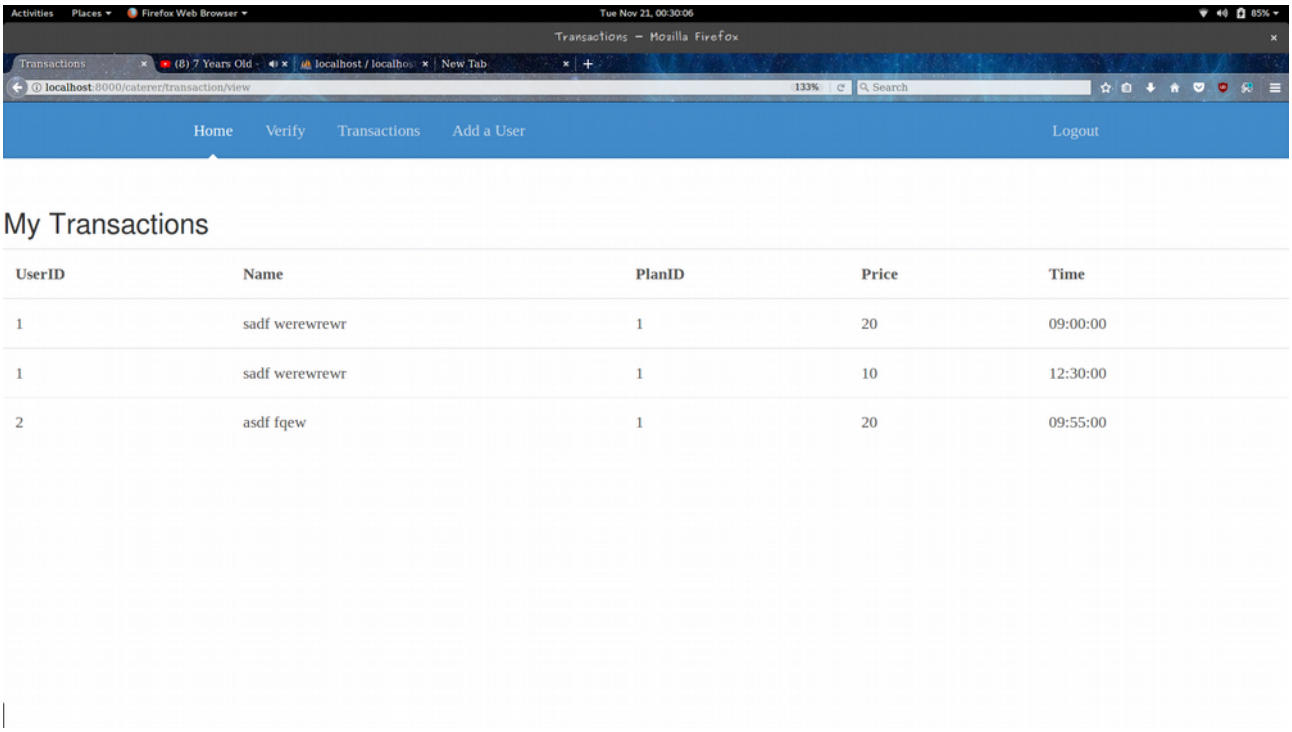
Adding a Subscriber

The Caterer can also add a subscriber to the database ,by entering the required details on this page.



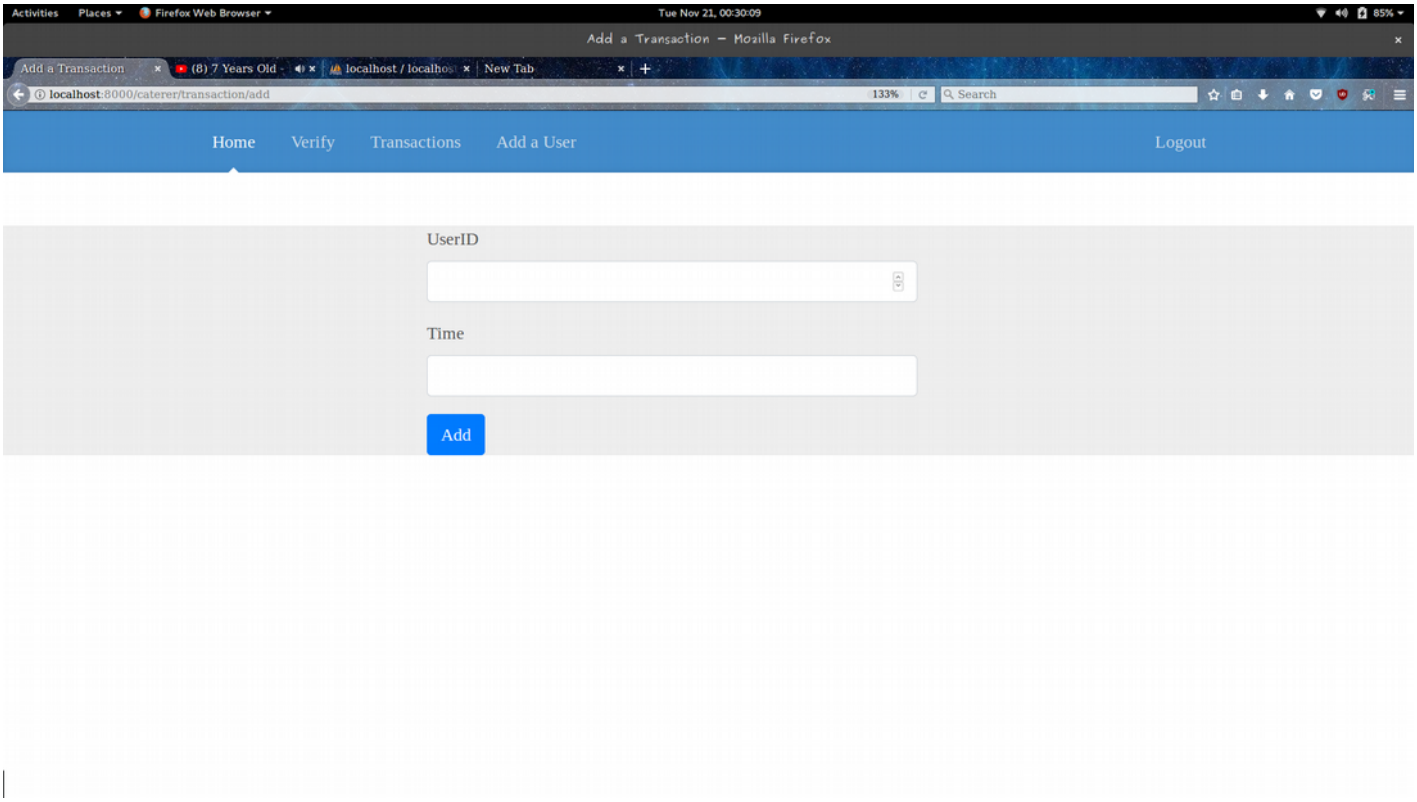
Managing the Transactions

The Caterer can view all the transactions made,along with the roll numbers of the students,their names,PlanIDs,the price it cost and the time, on this page.



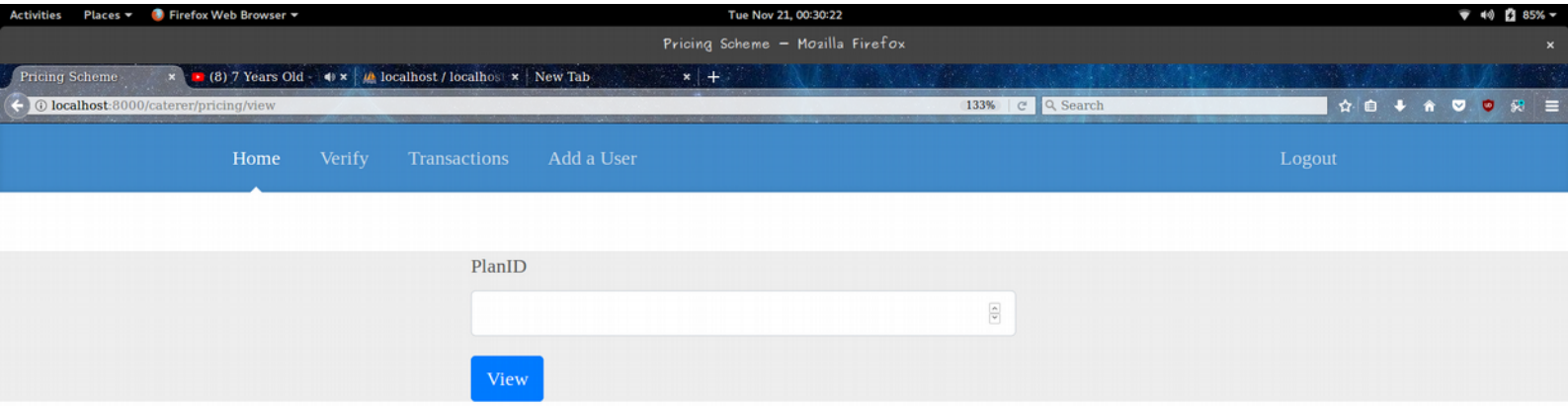
Adding a transaction

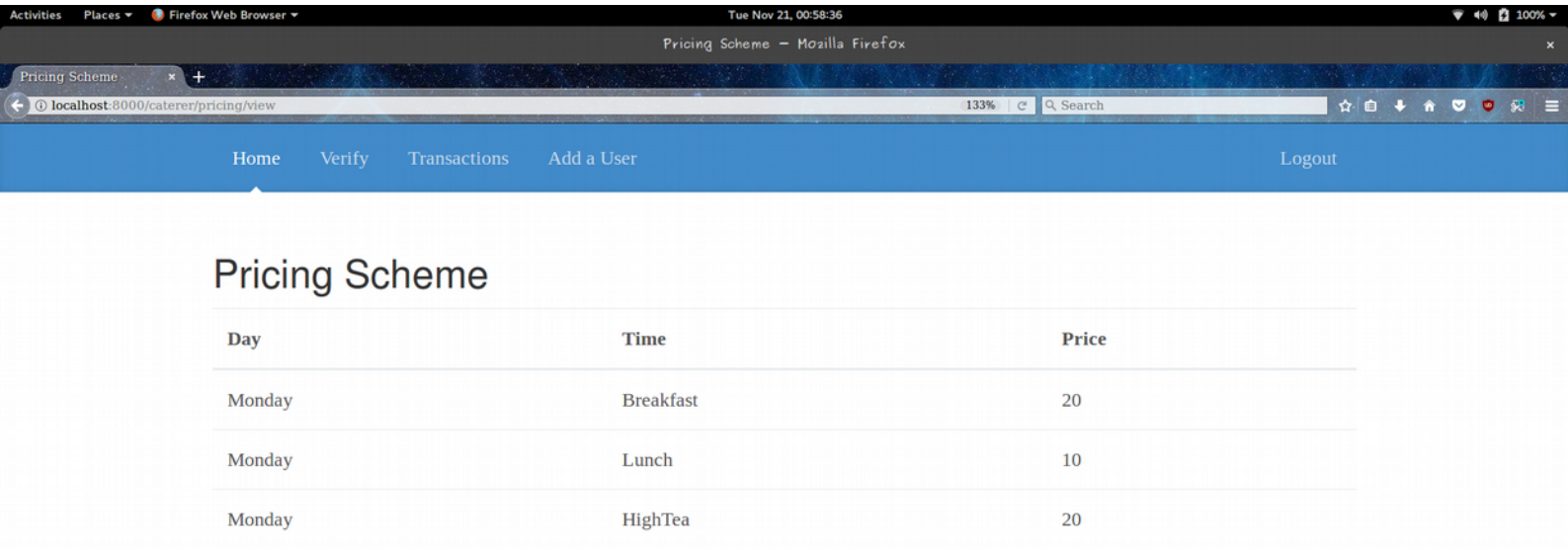
The caterer can add a transaction by entering the user details and the time through this page.



Managing the Pricing

The caterer can view the current pricing scheme,by entering the PlanID through this page.



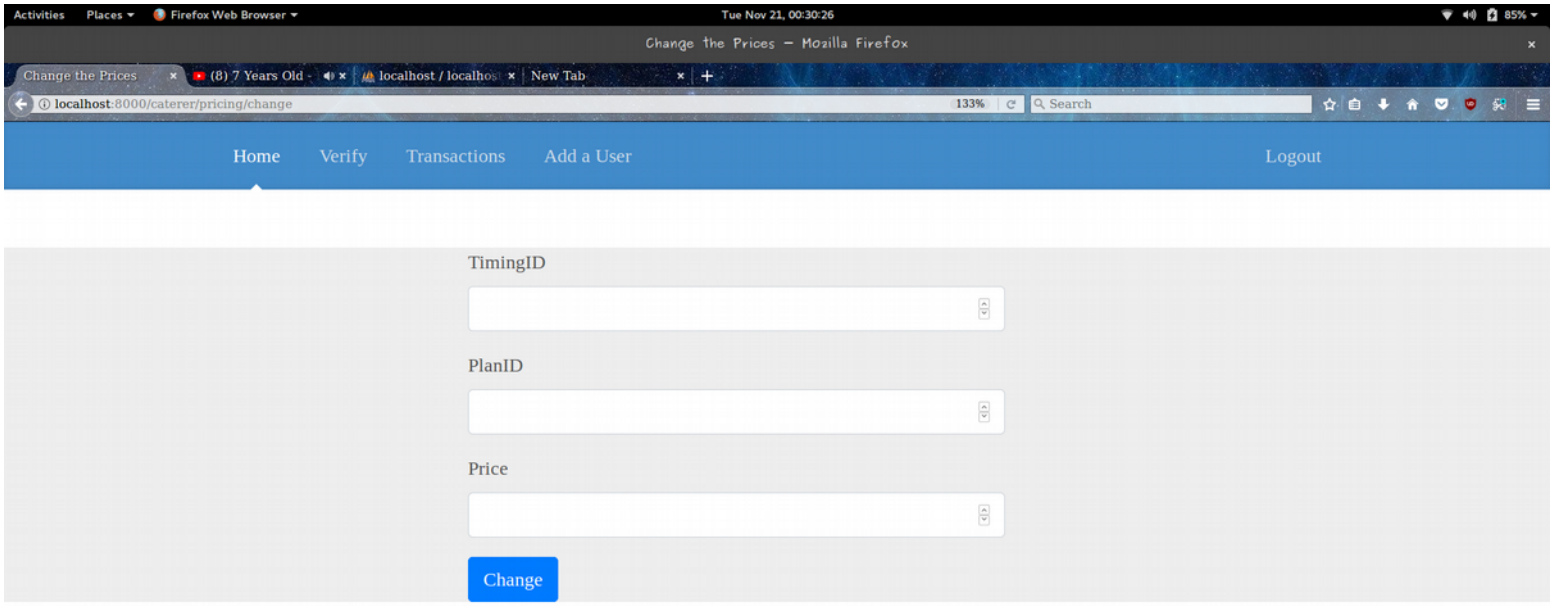


Pricing Scheme

Day	Time	Price
Monday	Breakfast	20
Monday	Lunch	10
Monday	HighTea	20

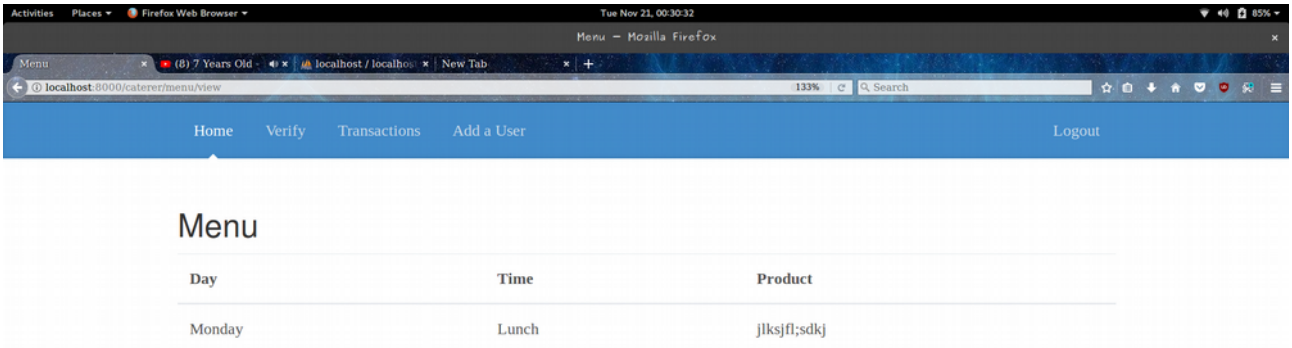
Changing the Pricing Scheme

The Caterer can change the pricing scheme by entering the required details on this page,and it, then, redirects to the Pricing scheme of that particular PlanID.



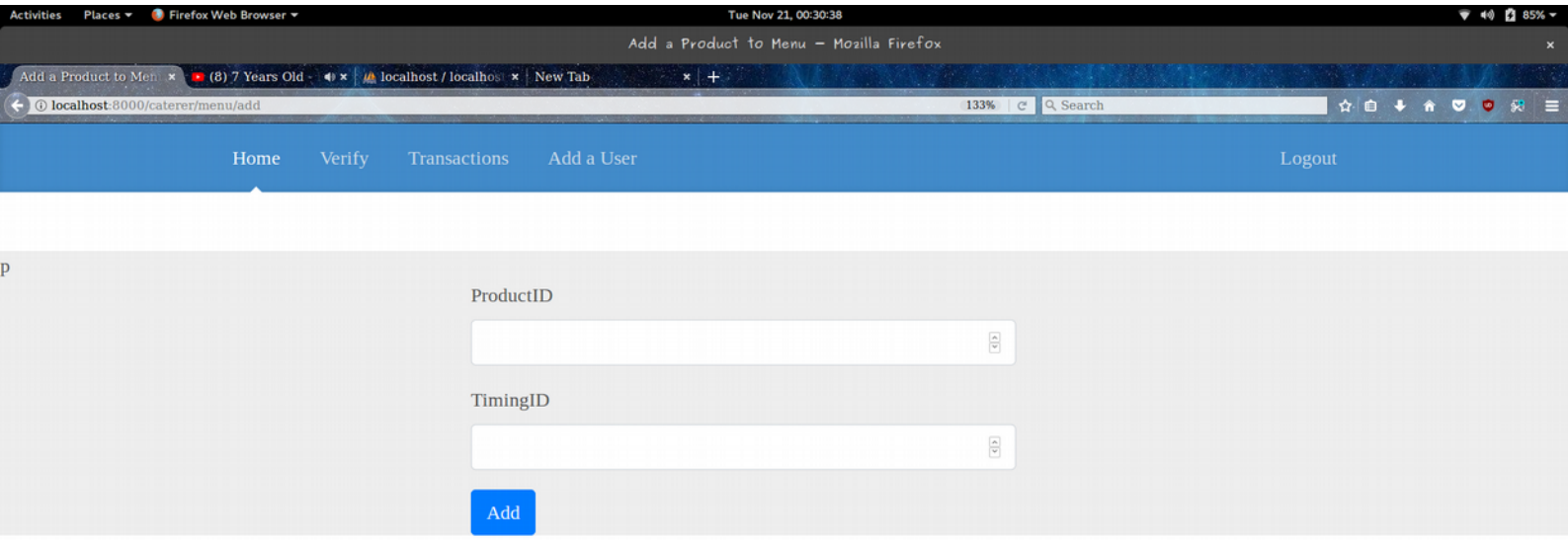
Viewing the Current Menu

The caterer can view the current menu through this page.



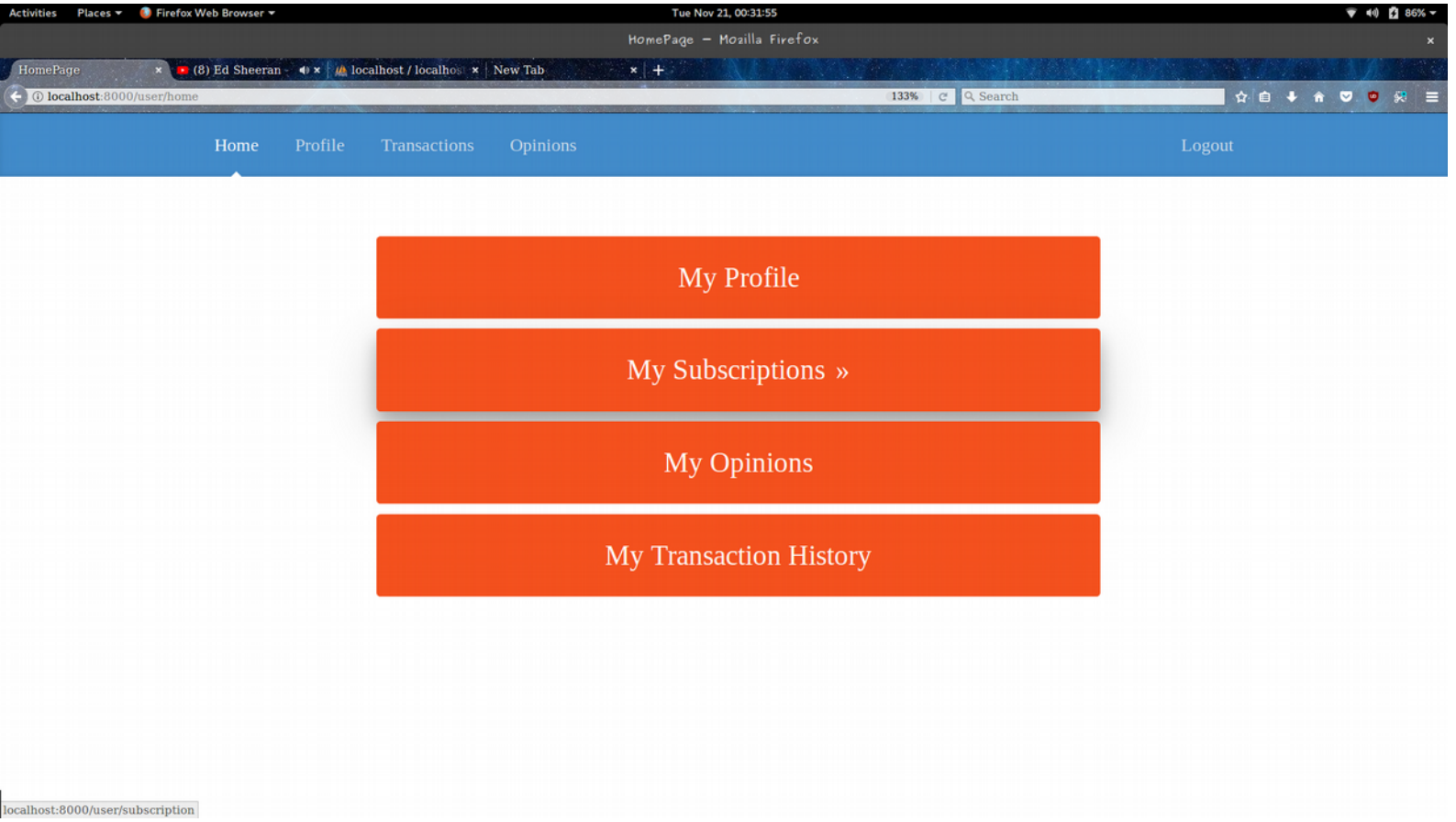
Changing the Menu

The Caterer can also change the menu,by entering the required details on this page.

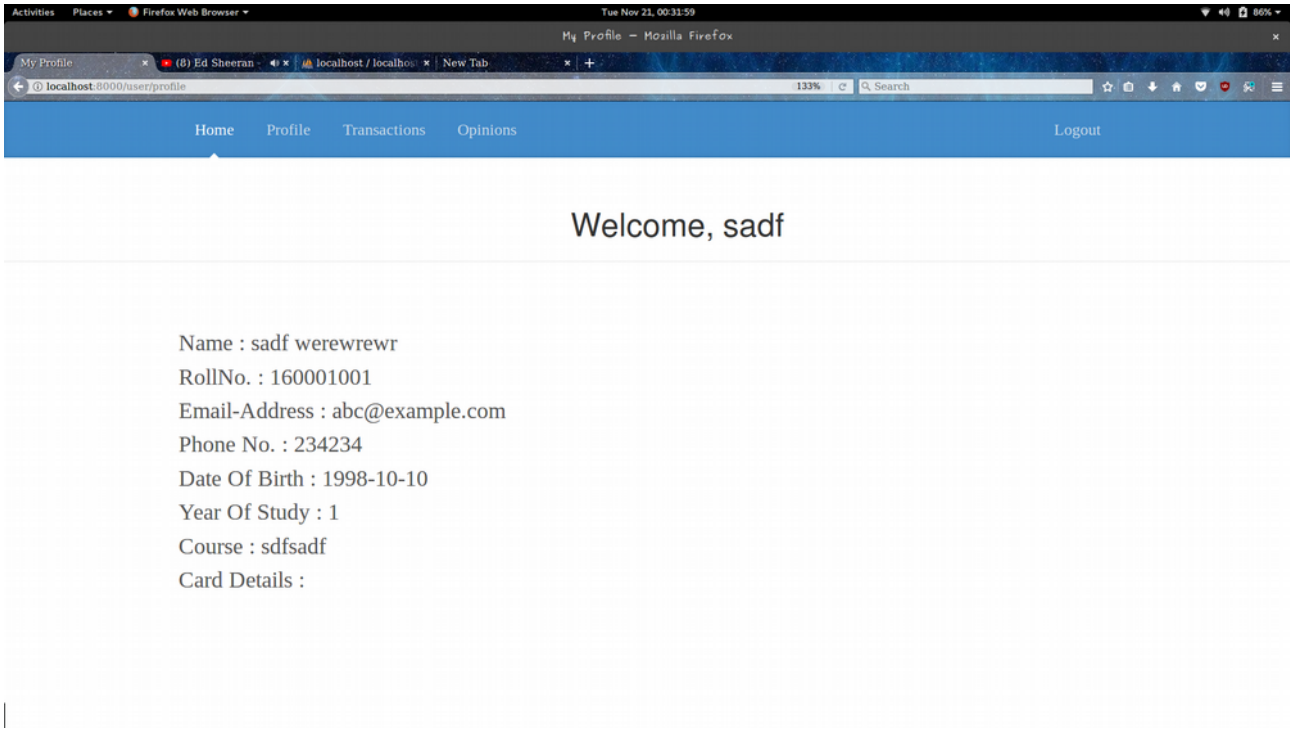


User’s DashBoard

The User can select from the above operations to do,and so,he/she will be redirected to the corresponding page.

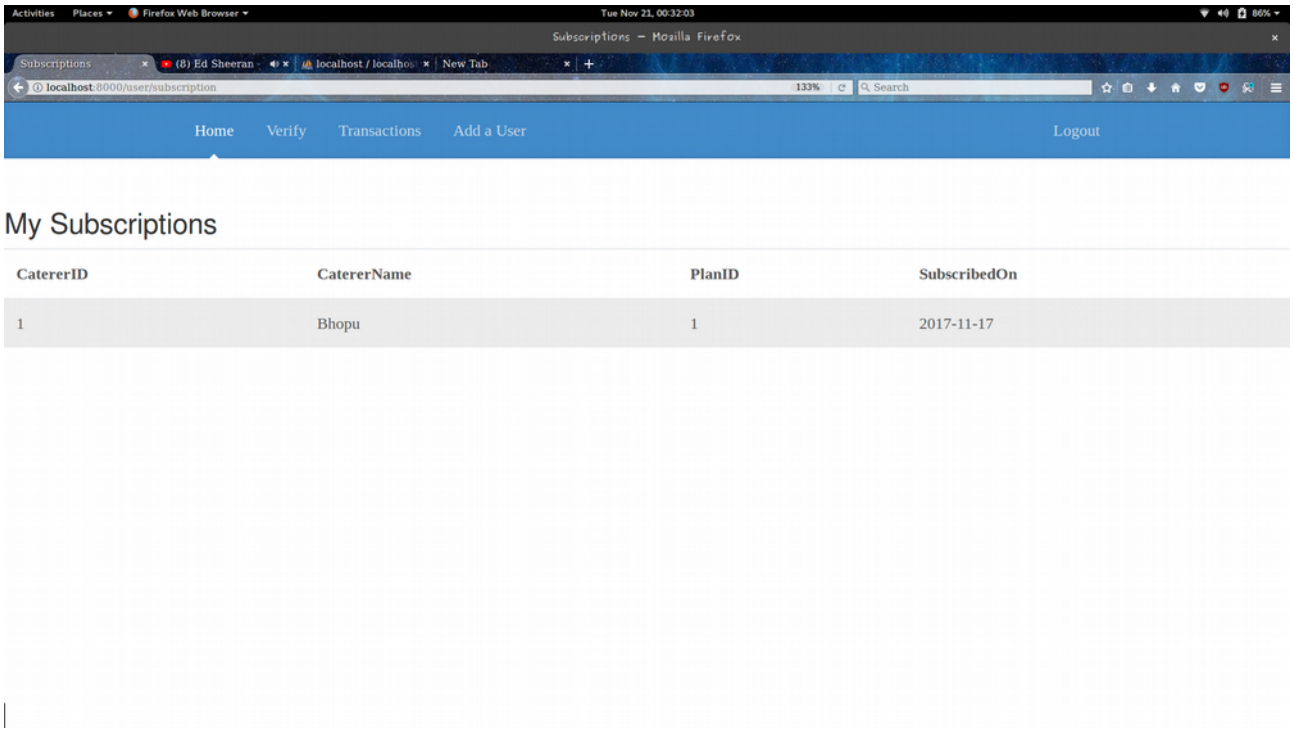


Viewing the Profile/Details



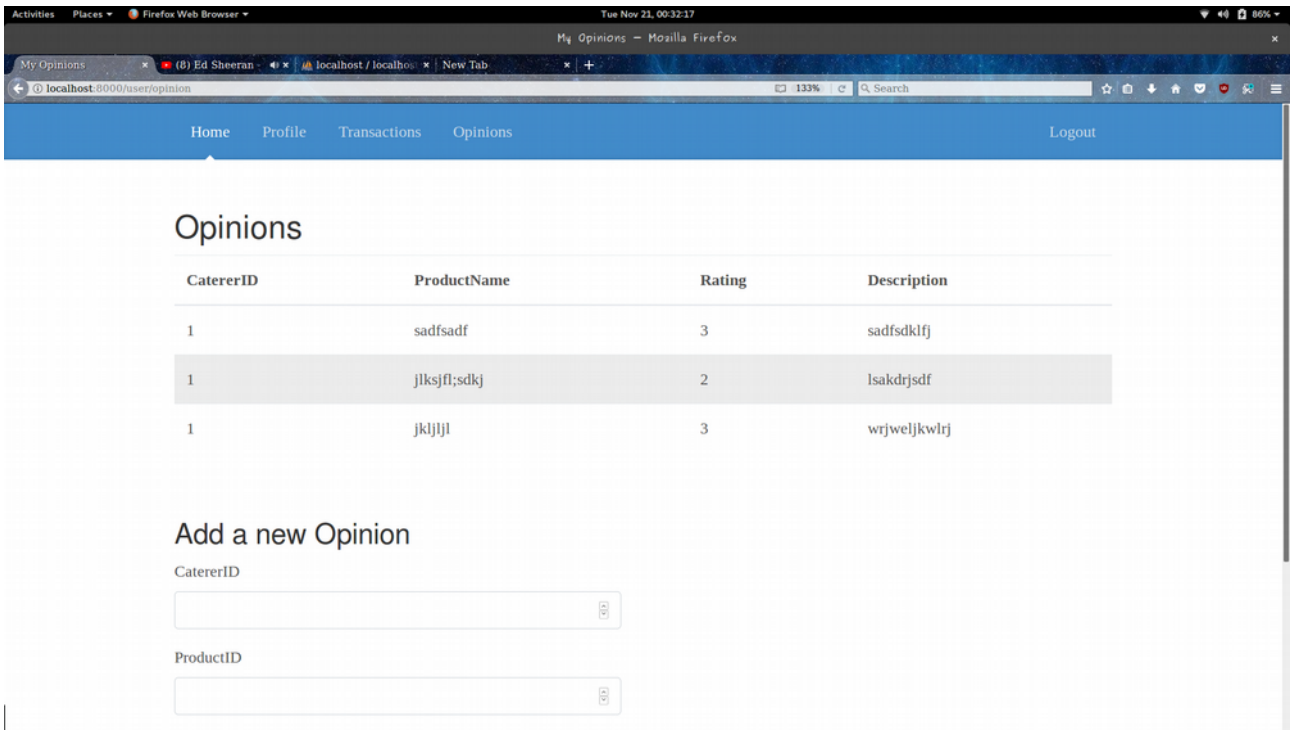
Viewing the Subscriptions

The User can view all his/her subscriptions with the corresponding caterers and the planids, on this page.



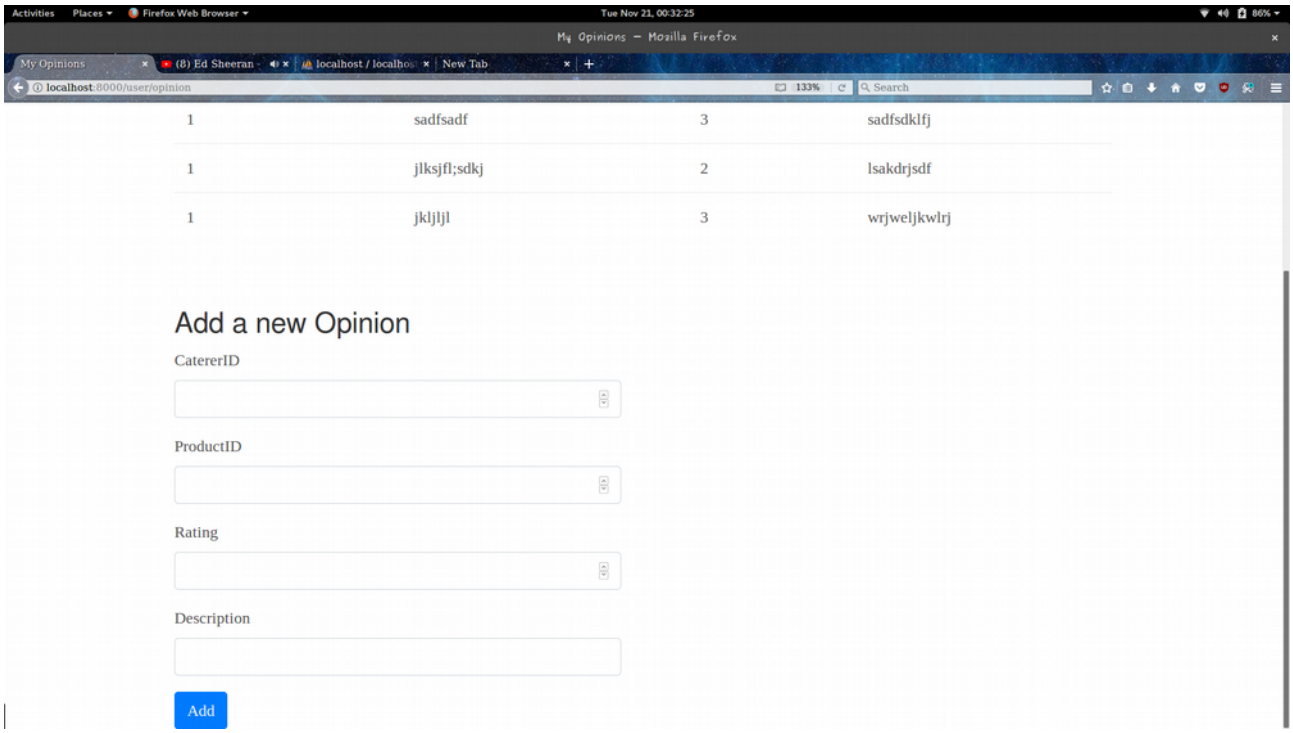
Viewing the Opinions

The Diner can view all the opinions on the products served by the caterer on this page,and rate it.



Adding a Opinion

The diner can add his opinion to the products served by the caterer,on this page.



Viewing the Transactions

The Diner can view all the transactions he/she has been involved in, along with the price it cost him/her, on this page.