SJSU CMPE 180-38 Term Project Report (Team-6)

CMPE 138
Fall 2016
Prof: Kong Li
Department of Computer Engineering,
San Jose State University, San Jose, CA, US

SJSU CMPE 180-38 Term Project Report (Team-6)

Title: A portal for the company, which will be used for buying and selling of Used Car(s) through its branches.

Team #6: Satyateja Pothuru (010760776)
Ishan Pandya (008686899)
Dhiraj Gurnani (010813023)
Anne Sai Pavan Teja (010820524)
Rishiraj Randive (010745059)

Choice of Database Project:

We are going to create a portal for a company, which maintains a transaction history, used for buying and selling the used cars. There will be a company having multiple branches, where the sale deed can be performed. There will be Sellers who can sell their used cars at a particular branch. There will be Buyers who can buy the used cars from a particular branch.

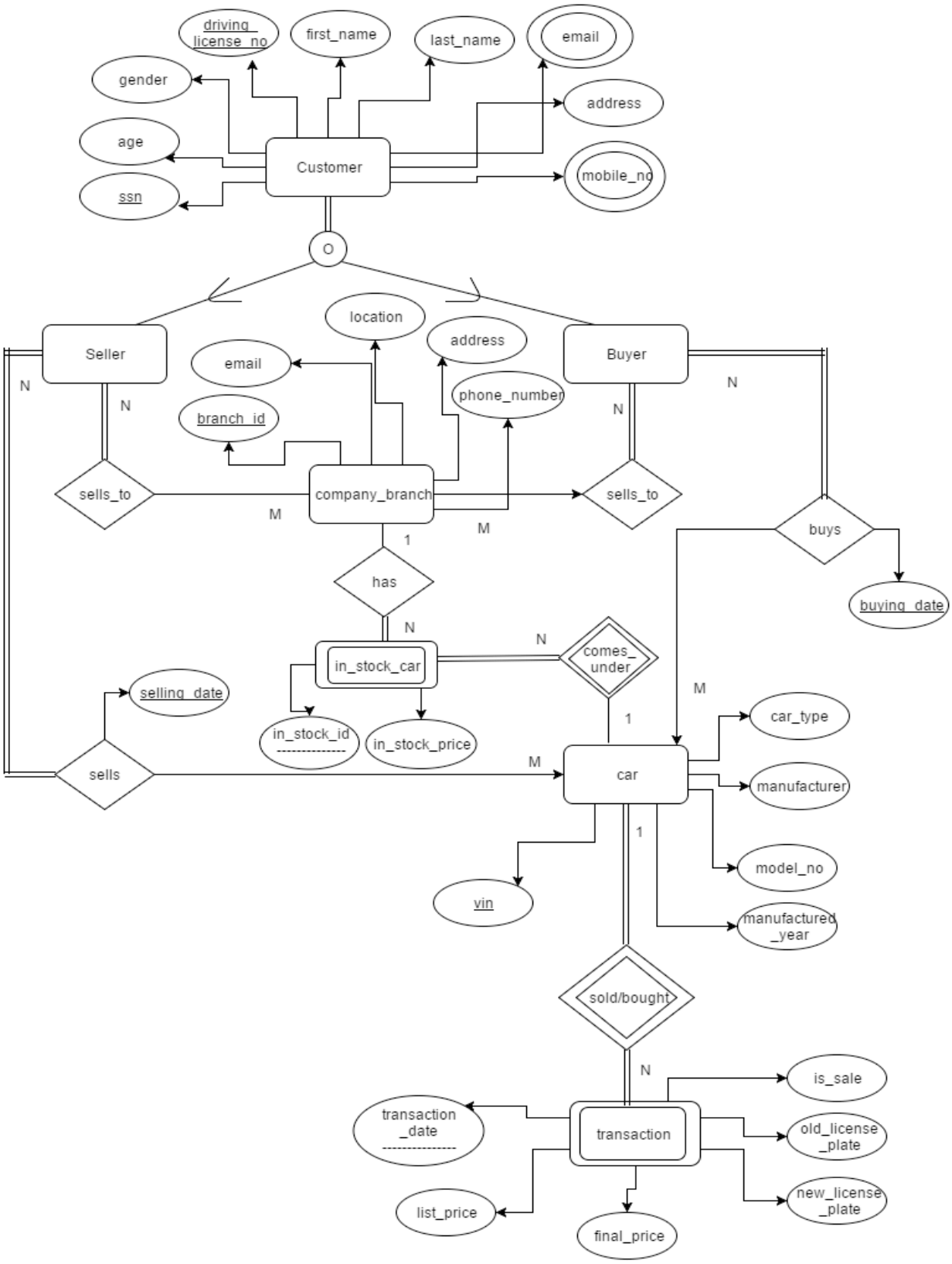
Technologies:

DB Engine – MySQL (InnoDB)
DB application Technologies – Node.js, Angular JS, MySQL commercial RDMS provider

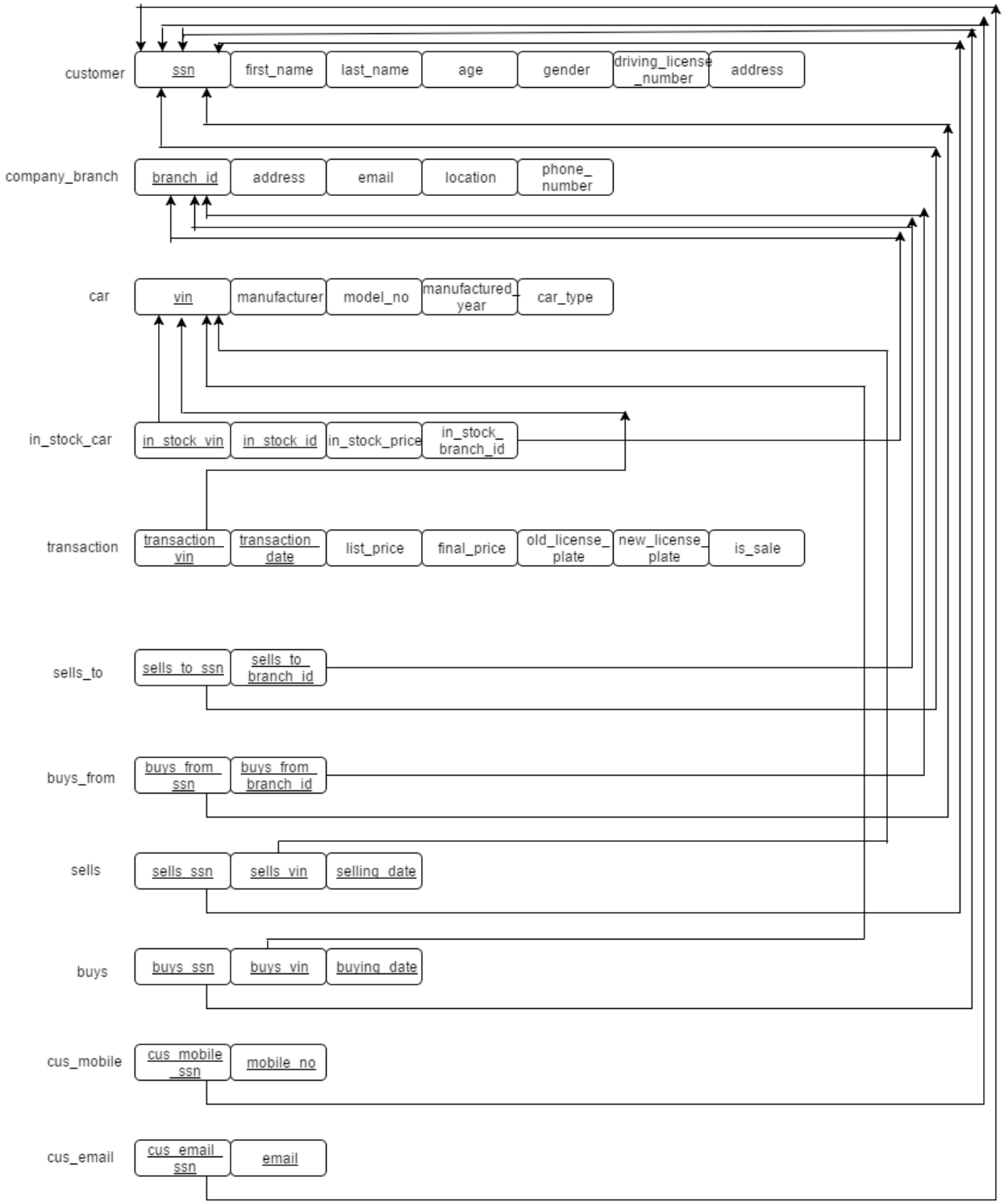
Functionalities:

- All possible operations on branches (Create, Update, Delete)
- Get all Branches available for the Company.
- All possible operations on Customers (Register, Update)
- All possible searches on Customer (Search by: Name, Ssn, License, Email, Phone)
- Get all Customers registered with the Company.
- History of Customer transaction with the company.
- Getting Discount for a Customer based on transaction history with the company. (example: more than 5 transactions then 5% discount)
- All possible operations on Cars (Register Cars purchased from Customers)
- All possible searches on Cars (Search by: Vehicle Id, Manufacturer, Manufactured Year, Model No, Car Type, Price Range)
- Get all In Stock Cars within a particular branch.
- Get all In Stock Cars within the Company.
- Get all Transaction History of a Car with the Company.
- Add new transaction with the branch for selling a Car from Customer
- Add new transaction with the branch for Purchasing a Car from Customer

EER Diagram:


driving_license_no
first_name
last_name
email
gender
address
age
Customer
mobile_no
ssn
O
location
address
Seller
email
Buyer
phone_number
branch_id
sells_to
company_branch
sells_to
buys
buying_date
has
in_stock_car
comes_under
selling_date
in_stock_id
in_stock_price
sells
car
car_type
manufacturer
model_no
vin
manufactured_year
sold/bought
is_sale
transaction_date
transaction
old_license_plate
new_license_plate
list_price
final_price

EER to Relational Mapping (Table Design):



Functional Dependencies and Normalization:

customer (ssn, first_name, last_name, age, gender, driving_license_number, address, email, mobile_no)
{ssn} ->{first_name, last_name, age, gender, driving_license_number, address, email, mobile_no}
Here, email and mobile_no are multivalued attributes. Hence there will be redundancy of data associated because of every multivalued attribute values. Hence, we decompose this table to three different tables to minimize the redundancy.

Table1: customer (ssn, first_name, last_name, age, gender, driving_license_number, address)
Table2: cus_mobile (cus_mobile_ssn, mobile_no)
Table3: cus_email (cus_email_ssn, email)

company_branch (branch_id, address, email, location, phone_number)
{branch_id} -> {address, email, location, phone_number}

Car (vin, manufacturer, model_no, manufactured_year, car_type)
{vin} -> {manufacturer, model_no, manufactured_year, car_type}

in_stock_car (in_stock_vin, in_stock_id, in_stock_price, in_stock_branch_id)
{in_stock_vin, in_stock_id} -> {in_stock_price, in_stock_branch_id}

Transaction (transaction_vin, transaction_date, list_price, final_price, old_license_plate, new_license_plate, is_sale)

{transaction_vin, transaction_date} -> {list_price, final_price, old_license_plate, new_license_plate, is_sale}

Sells (sells_ssn, sells_vin, selling_date)
{sells_ssn, sells_vin} -> {selling_date}

Buys (buys_ssn, buys_vin, buying_date)
{buys_ssn, buys_vin} -> {buying_date}

Normalization Conclusion:

Now, all tables have been normalized to 3NF.

DB Object Specification:

Tables:

i. customer: An entity which can be categorized under buyer/seller at any particular point of time or during a particular transaction, who is interested in purchasing/selling a car.

ii. company_branch: An entity which mediates the transaction between a buyer and a seller, performed based on the available cars in the impound.

iii. car: The vehicle which can be sold/ bought/ impounded.

iv. in_stock_car: Stores details of the vehicles currently available in a branch impound.

v. transaction: Holds the current and previous transactions records.

vi. sells_to, buys_from: Holds records about which customer made transaction in which branch.

vii. sells, buys: Holds records about which car has been traded and at what time.

viii. cus_mobile, cus_email: Normalized multi-valued customer details.

Column Description:

i) Customer:

ssn	Fixed length attribute which is unique for each customer
first_name, last_name	Name of each customer. A name attribute can be derived from these attributes.
driving_license_number	Fixed length unique attribute for each customer with a license.
address	Residence/work address registered under a customer's entry.

ii) Company_branch:

branch_id	Unique id for each company branch.
location, address	Location of a branch
email, phone_number	Contact details for a branch.

iii) Car:

vin	Unique for each car irrespective of manufacturer, model, type, number plate.
manufacturer	Manufacturing company of a car.
model_no	model_no of a car given by its manufacturer.
manufactured_year	Year when a model under a manufacturer has been made.

iv) in_stock_car:

in_stock_vin	The unique car id for an impounded car.
in_stock_id	Unique id provided for each impounded car at a particular time.
in_stock_price	Price listed for an impounded car.
in_stock_branch_id	Lists branch under which a car has been registered under.

v) Transaction:

transaction_vin	Holds the unique car id on which a transaction has been performed.
transaction_date	Date on which a car has been sold/bought by a customer.

list_price, final_price	The proposed and final price on which a deal has been closed.
old_license_plate, new_license_plate	Lists branch under which a car has been registered under.
is_sale	A flag which records if a transaction was buy/sell.

vi) Sells:

sells_ssn	Unique id for a seller.
sells_vin	Unique id for a car.
selling_date	Date on which the car has been sold.

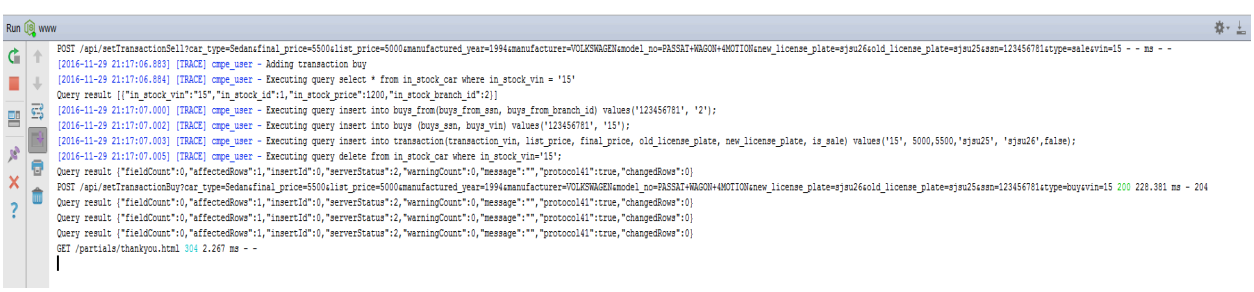
vii) Buys:

buys_ssn	Unique id for a buyer.
buys_vin	Unique id for a car.
buying_date	Date on which the car has been bought.

Functionality involving modification of more than one table:

We have implemented two functionalities named “buy transaction” and “sell transaction” where four and five tables are updated respectively at a single API call. We executed these queries in a single transaction, if any of the query fails, entire transaction is rolled back else transaction is committed successfully. This is implemented in server using Node JS MySQL transactions.

Sample Execution of query for “Sell Transaction” (selling a car to customer)



Sample Execution of query for “Buy Transaction” (buying a car from customer)

```
Run www
[2016-11-29 21:26:15.016] [TRACE] cmpe_user - Adding transaction for sale
[2016-11-29 21:26:15.017] [TRACE] cmpe_user - Executing query select * from sells_to where sells_to_ssn = '123456781' and sells_to_branch_id = '2'
[2016-11-29 21:26:15.019] [TRACE] cmpe_user - Executing query select * from car where vin = '15'
Query result [{"sells_to_ssn":"123456781","Sells_to_branch_id":2}]
[2016-11-29 21:26:15.135] [TRACE] cmpe_user - Transaction sale unsuccessful
Query result [{"vin":"15","manufacturer":"VOLKSWAGEN","model_no":"PASSAT WAGON 4MOTION","manufactured_year":1994,"car_type":"Sedan"}]
[2016-11-29 21:26:15.142] [TRACE] cmpe_user - Executing query insert into sells(sells_ssn, sells_vin) values('123456781', '15');
[2016-11-29 21:26:15.144] [TRACE] cmpe_user - Executing query insert into transaction(transaction_vin, list_price, final_price, old_license_plate, new_license_plate, is_sale) values('15', 4500,4000,'sjsu26', 'sjsu27',true);
[2016-11-29 21:26:15.145] [TRACE] cmpe_user - Executing query insert into in_stock_car(in_stock_vin, in_stock_price, in_stock_branch_id) values('15', 4000, '2');
Query result {"fieldCount":0,"affectedRows":1,"insertId":0,"serverStatus":2,"warningCount":0,"message":"","protocol41":true,"changedRows":0}
Query result {"fieldCount":0,"affectedRows":1,"insertId":0,"serverStatus":2,"warningCount":0,"message":"","protocol41":true,"changedRows":0}
Query result {"fieldCount":0,"affectedRows":1,"insertId":1,"serverStatus":2,"warningCount":0,"message":"","protocol41":true,"changedRows":0}
POST /api/setTransactionSell?car_type=Sedan&final_price=4000&list_price=4500&manufactured_year=1994&manufacturer=VOLKSWAGEN&model_no=PASSAT+WAGON+4MOTION&new_license_plate=sjsu27&old_license_plate=sjsu26&ssn=123456781&type=sale&v
```

DB App Technology:

DB Engine – InnoDB
DB application Technologies – MySQL commercial RDMS provider
Framework – Express JS
Languages - JavaScript, Html, Node.js

DB App Access Technology:

DB is accessed using MySql node modules.

Major modifications from the Proposal:

- New entity named “in_stock_car” have been added as a dependent entity on “cars” entity. This was done to keep track of car details in car entity and, cars in stock details in “in_stock_car” entity so that car details redundancy will be reduced from “in_stock_cars”.
- New attribute named “buying_date” and “selling_date” have been added into “buys” and “sells” entity respectively. This was done to avoid conflicts between entries with same SSN or VIN.

Unique design:

- Entire transaction is done using single form, no need to fill different forms for customer or car details. In the server the transaction handles the query updates.

Potential Improvement:

- Can be improved to make an online portal where users can post the details of car, and others can bid/directly buy the car.
- Having an option to direct buy or bidding like Ebay.

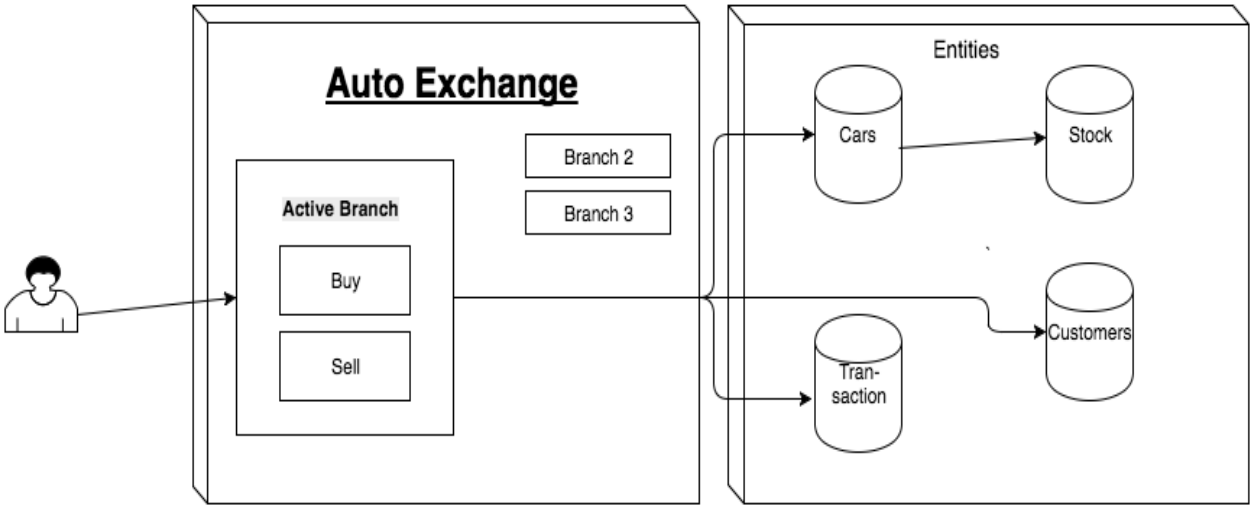
Tasks Performed by each team member:

Tasks	Team member	Completion date
Create, update, and delete branches	Dhiraj	11/20/2016
Get all branches	Pavan	11/21/2016
Customer register and update	Satyateja	11/21/2016
Customer search by name, SSN, license, email, and phone	Dhiraj	11/23/2016
Get customers of company	Pavan	11/21/2016
Get history of customer transactions	Rishiraj	11/25/2016
Get discount information for customer	Rishiraj	11/22/2016
Register cars in company stock	Ishan	11/26/2016
Car search by VIN, manufacturer, manufactured year, model no, car type, and price range	Ishan	11/23/2016
Get all in stock cars with branch	Ishan	11/23/2016
Get all in stock cars with company	Rishiraj	11/25/2016
Get transaction history of car	Dhiraj	11/26/2016
Add new sale transaction	Satyateja	11/22/2016
Add new buy transaction	Satyateja	11/24/2016

Test plan execution:

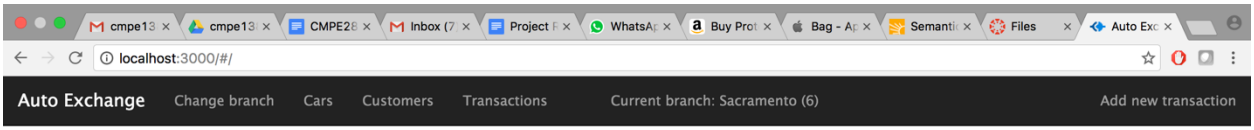
Test	Description	Expected
Add new transaction	On home page click the link "Add new transaction" in upper right corner	Should open a page with input text with "Check" and "New" button
Enter new transaction (sale)	Enter details in the transaction form and click Add	Should save the transaction to DB and update tables, transaction, sells_to, sells, car and in_stock_car. In case the customer is new should add to Customer table as well
Enter new transaction (buys)	Enter details in the transaction form and click Add	Should save the transaction to DB and update tables, transaction, buys_from, buys and delete car entry from in_stock_car. In case the customer is new should add to Customer table as well
Add new branch	Click on "Change branch" link and then click on "Add branch" button	Should open form to fill the branch information and add it to DB. Updates company_branch table
Get all cars	Click on "Cars" link and then click on "Get all cars"	Should show all the cars from the DB which are available across all branches
Get car history	Click on "Cars" link and then click "Get history" button after entering the VIN	Should show all the transactions history associated with Car
Search car	Click on "Cars" link and then click "Search" button after entering the VIN	Should search the car and show the detail
Search customer	Click on "Customers" link and then enter value based on option selected	Should show the detail of customer based on the search criteria
Edit customer	Click on "Customers" link and then click "Edit" button to open the customer update form and fill in the details.	Should update the table customer and if required, cus_email and cus_mobile

Component diagram:



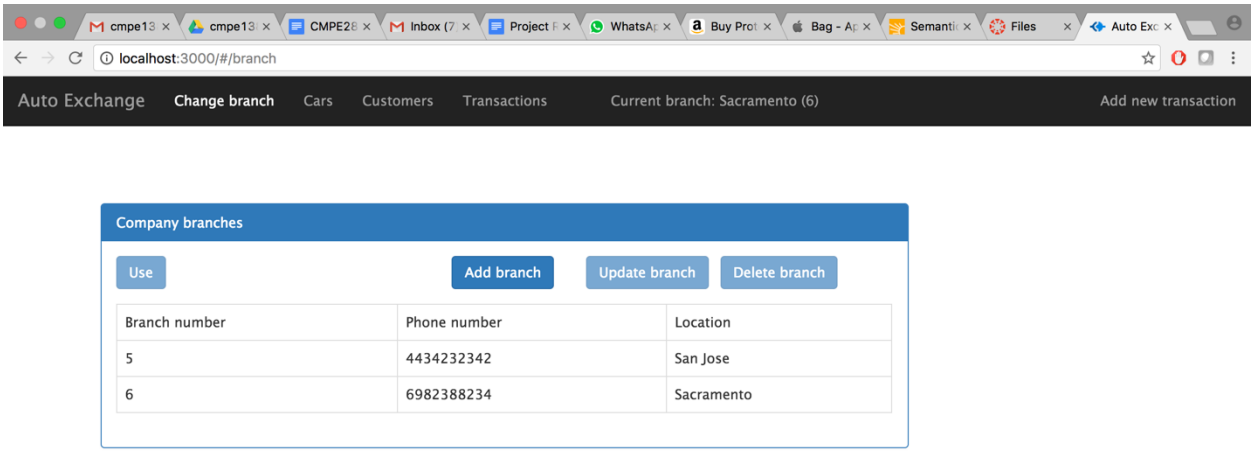
Application screenshots:

Welcome page:

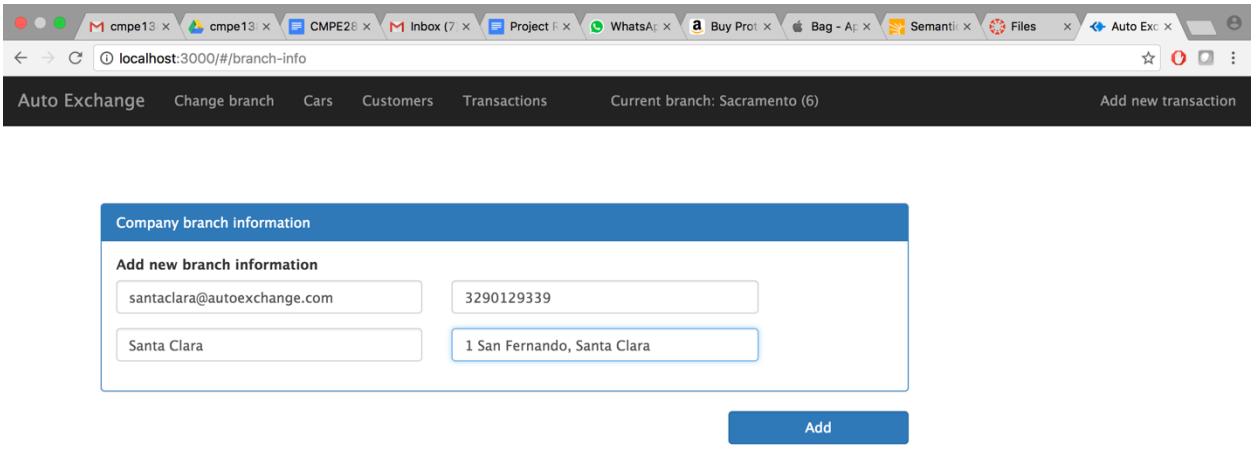


Welcome to Auto Exchange

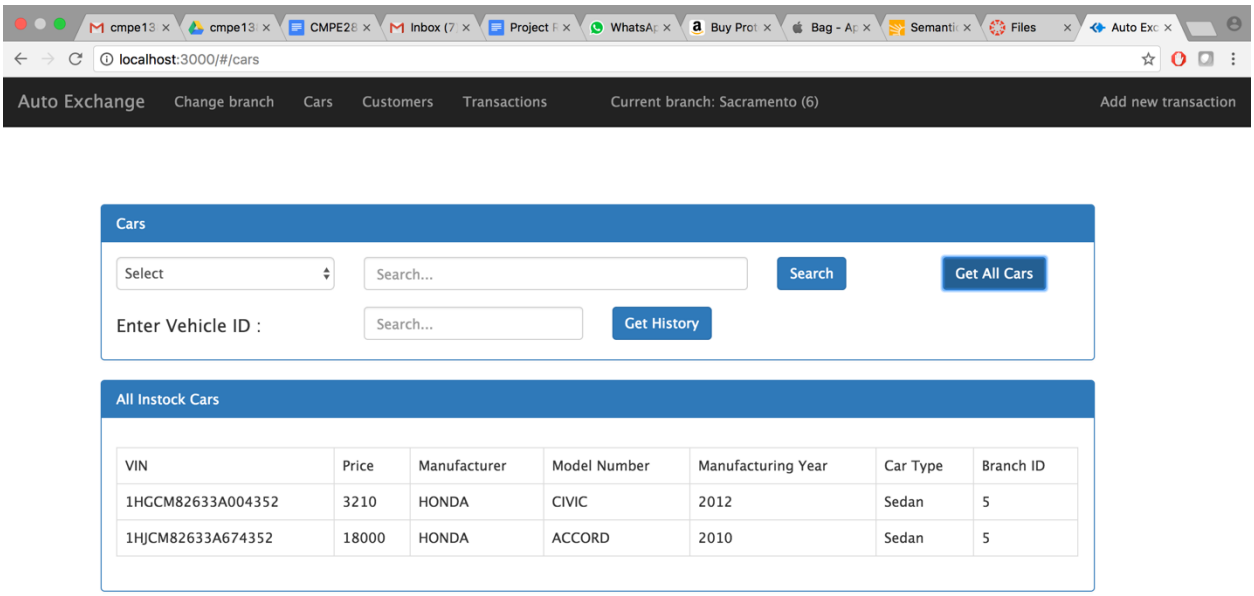
Branch page: (shows available branches, adds new, updates and delete branch)



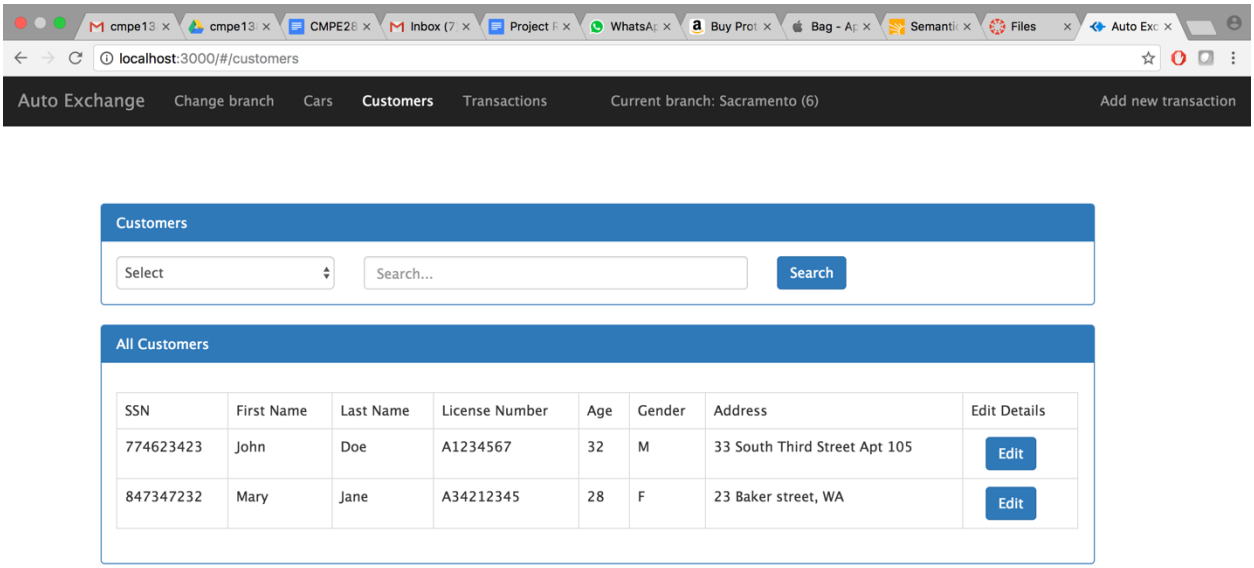
Add new branch: (Adds new branch)



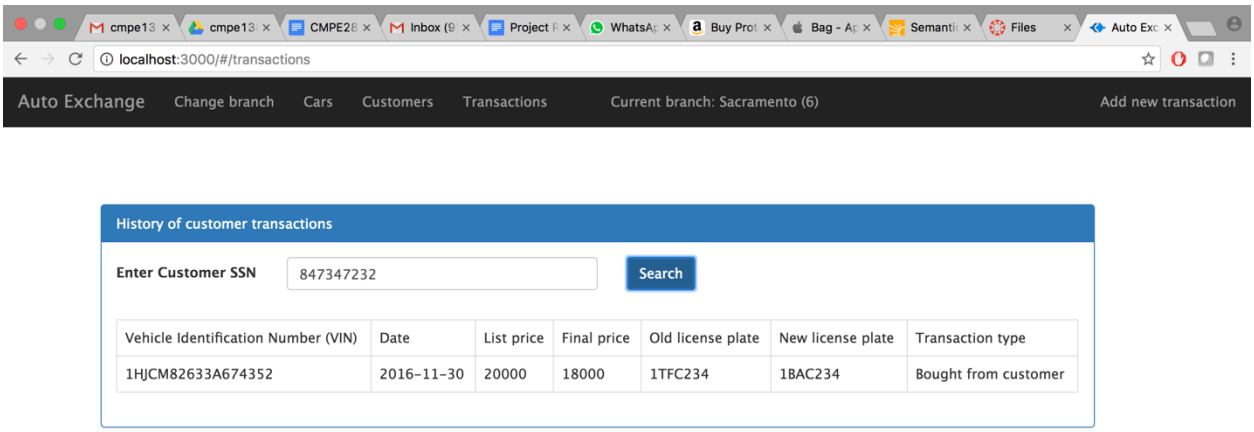
Cars page: (Search cars, get history and get all cars in stock across all branches)



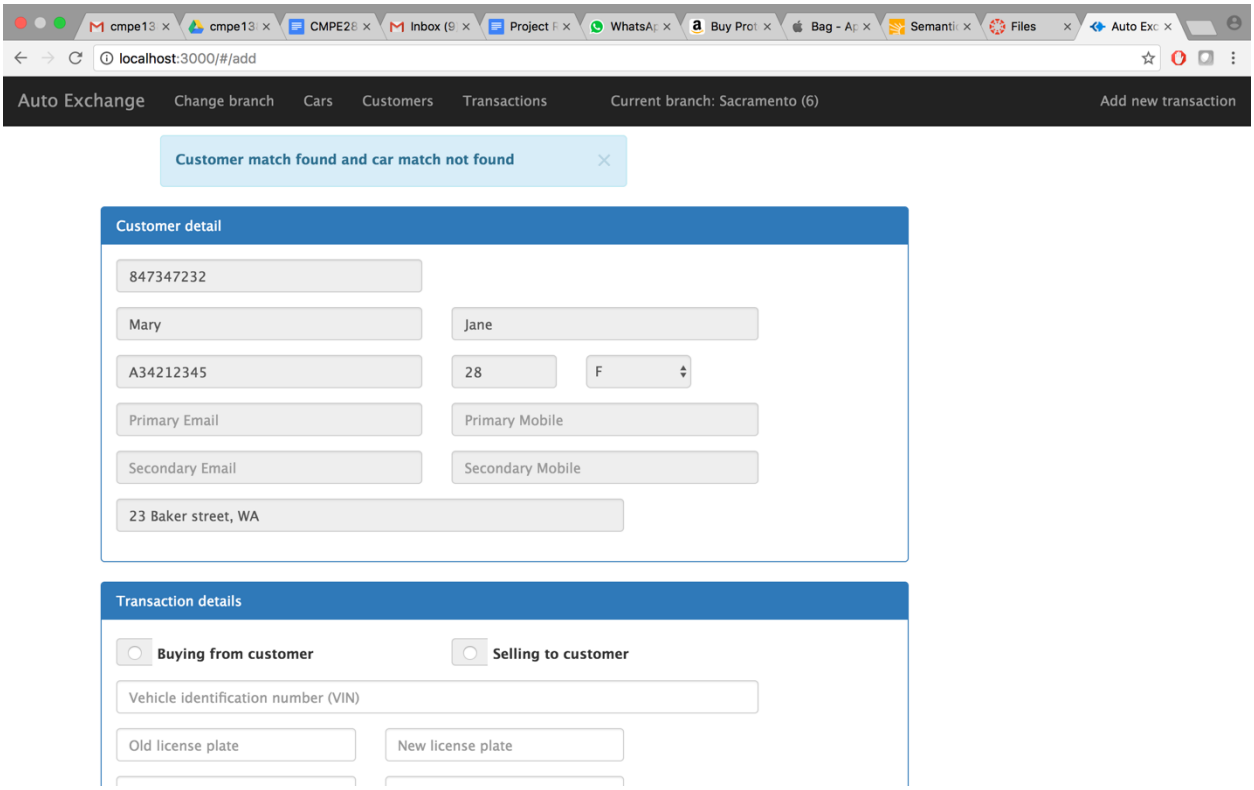
Customers page: (Search customer, and option to edit customer)



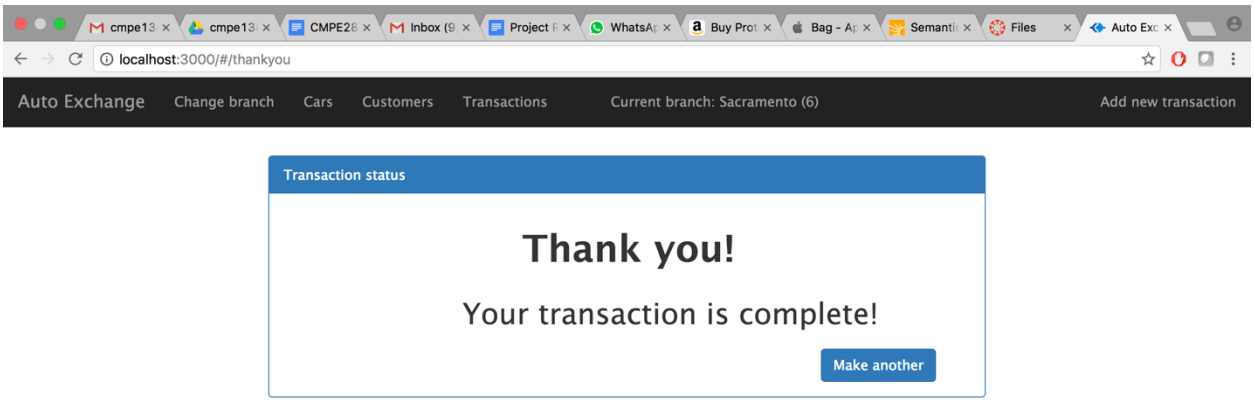
Transaction for customer: (Search customer transaction with SSN)



Add new transaction: (can add Sale and Buy transactions for new/existing customer and car)



Thank you page after transaction added successful:



Customer getting discount for transactions more than 5: