**K. L. E. SOCIETY’S**

**B. V. B. COLLEGE OF ENGINEERING &TECHNOLOGY, HUBLI – 580031**

**(An Autonomous Institution)**



**DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING**

DBMS Course Project Report on

**YAMAHA SHOWROOM DATABASE APPLICATION**

Submitted by

**Mr. Dileep Hegde 2BV14IS037**

**Mr. Bharat Banavalikar 2BV14IS032**

**Mr. Sandeep Dm 2BV14IS081**

**Mr. Rishabh 2BV14IS076**

**Theme: Application development**

**CONTENTS**

1. **INTRODUCTION**
2. **PROPOSAL PHASE**
3. **DESIGN PHASE**
4. **IMPLEMENTATION PHASE**
5. **README**
6. **SNAPSHOTS**
7. **INITIAL LETTER**
8. **FINAL LETTER**
9. **FEEDBACK**

**INTRODUCTION**

Trivial methods of database management were used in Yamaha Showroom Karwar. Ms Excel was used to store the database of the showroom, which made it insecure also it was difficult to manage. All the data was on a signal computer with no backup measures. Static bill was used which is difficult to maintain. To encounter these problems there was a need of a smart system which could keep the database secure, easily maintainable and systematic.

We developed an online application which has three views, namely customer view, employee view and admin view. The default view is customer view wherein all the basics information (bikes, scoters, service schedules, warranty policy, parts, feedback, about and contact details) about the site is available.

The privilege of editing/adding data to the database is given to employee and admin also admin has unique privilege of editing/adding employee from the database.

This application is put online hence it can be easily accessed from any computer, resolving the problem of using single computer to feed data. It also resolves the problem of back up. As the application is online application, the backup will be also available online. Application has login facility which provides security to the database.

**Proposal Phase Report**

**Category: Application Development**

**Title: Yamaha Showroom Database Application**

**List of Team Members: Bharat, Sandeep DM, Rishabh, Dileep**

**Team Leader: Bharat Banavalikar**

**Responsibilities:**

Dileep will be designing er diagram with cardinality ratios and relationships between the entities.

Sandeep DM will work on drawing data dictionary table and did the other er diagram.

Bharat as a team leader will design schema for the optimal er diagram and the report writing on the whole.

Rishabh will be designing the outlook of the entities, data flow diagram and the presentation.

**Problem Description:**

Storing, linking and fetching of data becomes tedious with files. Requirement of a database over files in terms data integrity and data security is the call of the hour. Hence this motivated us to develop an app for Yamaha showroom

Problem Statement -A database management application for the Yamaha showroom located in Karwar. This real time application eliminates the tedious historic method of data management.

**Requirements:**

Retrieval: Data can be easily retrieved from the database according to the customer’s/client’s requirements.

Data manipulation: the application allows the client to insert new record or update the existing record according to his/her requirement for any of the given entity.

Invoice generation: the total estimated bill for the purchase is generated which includes the price of the bike and discounts (if available) and hence the total price.

Graphical representation of statistics: the application effectively helps to analyze the data in the form of pictorial representation such as –a bar graph for the number of bikes sold for a particular month/year.

Availability of vehicles – the customer can easily surf for his/her desired vehicle in store. The customer is prompted to visit the showroom on a specified day if the vehicle is presently not in store.

Service schedules- the customer is updated about his/her free services and its validity.

**Design Questions to be answered**

**Question 1:** From the problem description, identify the entities that need to be represented in the database, the attributes of each entity, the relationships between the entities, and the cardinality ratios of each relationship.

1. Customer

* Customer\_name
* C\_Address
* C\_SSN
* Ph\_no
* Email\_id

1. Employee

* E\_name
* E\_id
* E\_designation
* E\_ph\_no
* E\_email
* E\_address
* E\_salary

1. Vehicle

* V\_name
* V\_id
* Model\_no
* V\_price
* Vehicle\_description – speed,mileage,cc,type

1. Dealer

* D\_name
* D\_id
* D\_address
* No\_of\_vehicles
* D\_ph\_no

1. Showroom

* S\_name
* S\_id
* Location

Employee services customer- m:n

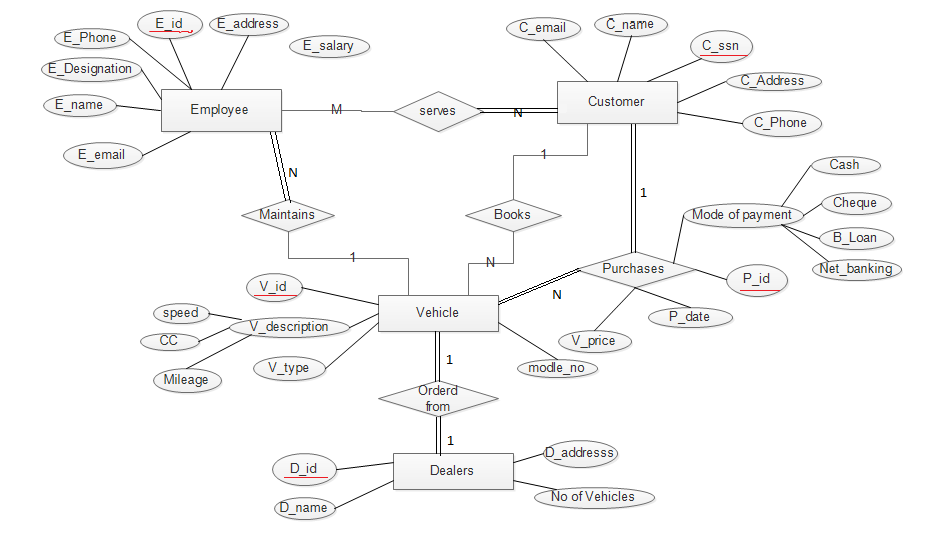
Employee works for showroom-m:n

Customer books vehicle-1:n

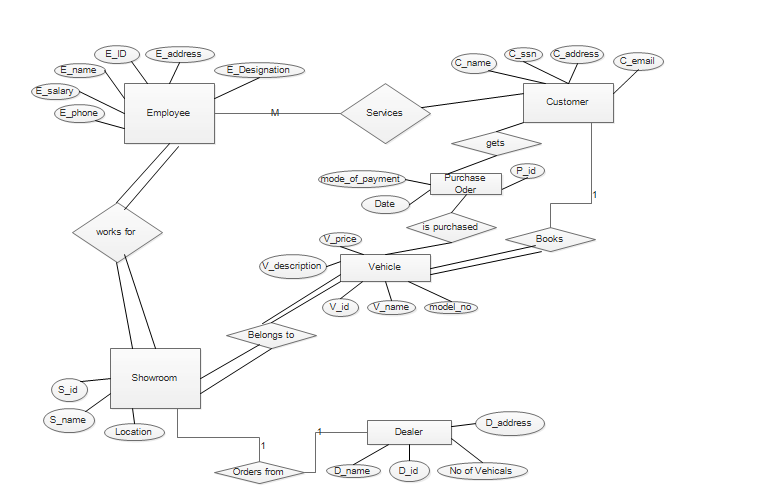
Vehicle belongs to showroom-n:m

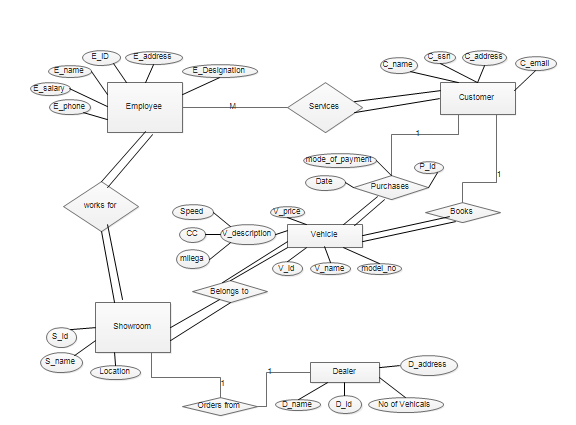
Showroom orders from delaer-1:1

**Question 2:** Draw an Entity-Relationship Diagram illustrating the information you have identified in Question 1.

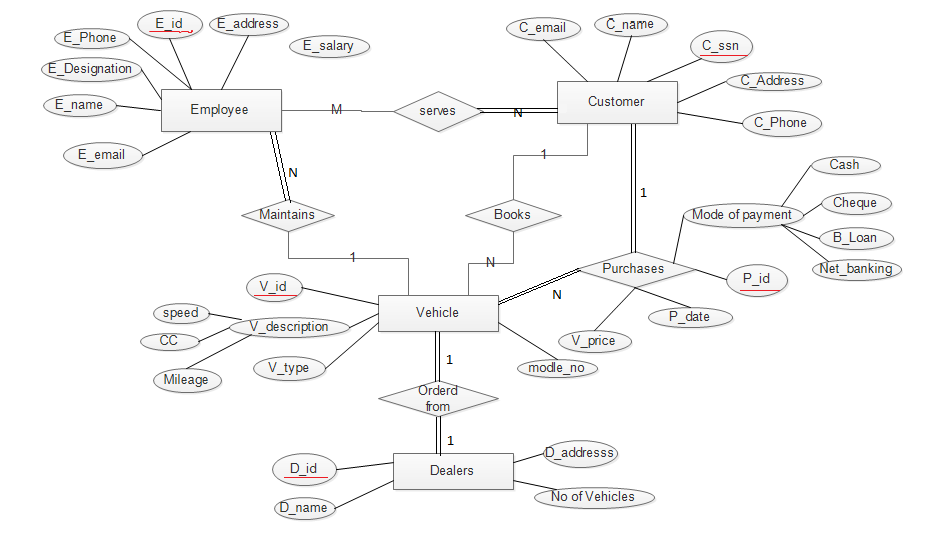


**Question 3:** Draw **alternate**Entity-Relationship Diagram illustrating the information you have identified in Question 1 that you think are most likely to occur.





**Question 4:** Choose the **optimal**Entity-Relationship Diagram from the designs provided above and justify why you think this is an optimal solution for your identified problem specification.



This is an optimal solution for our identified problem specification because it can decrease the work for the employee if they work in our database application as it maintains all the details need to be carried out for them and even it will be convenient for the customers to book the vehicle online and purchase it from the respective showroom.

**Question 6:** Draw an ER to Relation Mapping illustrating the information you have identified in Question 4.

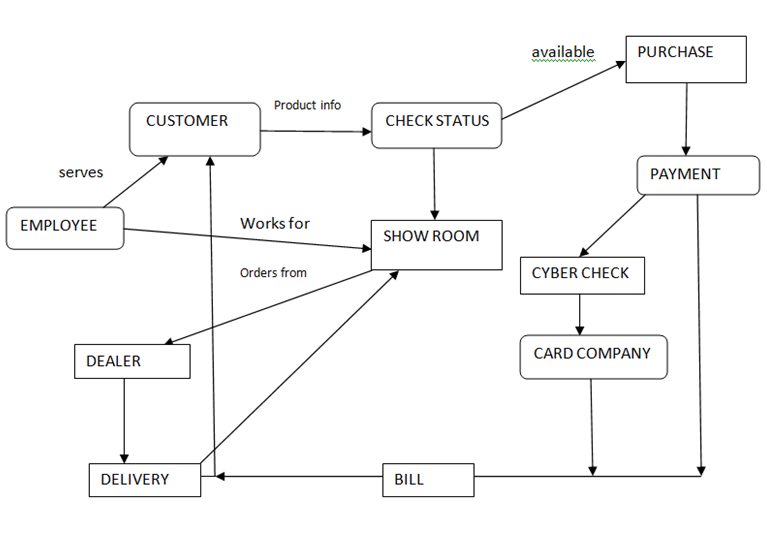


**Question 7:** Draw a Data Dictionary illustrating the information you have identified in Question 6.

Format for data dictionary is as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Object (Entity)** | **Name (Attribute)** | **Type**  **(Data type)** | **Description** | **Primary Key** | **Foreign Key** |
| Customer | Customer\_name | String | Name of the customer | No | No |
| Customer | C\_Address | String | Address of the customer | No | No |
| Customer | C\_SSN | Integer | Unique Identification number for the customer | Yes | No |
| Customer | Ph\_no | Integer | Phone number of the customer | No | No |
| Customer | Email\_id | String | Email\_id of the customer | No | No |
| Employee | E\_name | String | Name of the customer | No | No |
| Employee | E\_id | Integer | Unique Identification number for the employee | Yes | No |
| Employee | E\_designation | String | Designation of the employee | No | No |
| Employee | E\_ph\_no | Integer | Phone number of the employee | No | No |
| Employee | E\_email | String | Email\_id of the employee | No | No |
| Employee | E\_address | String | Address of the employee | No | No |
| Employee | E\_salary | Integer | Salary of the employee | No | No |
| Employee | C\_ssn | string | Social securiety  Of Customer | No | Yes |
| Vehicle | V\_name | String | Name of the vehicle | No | No |
| Vehicle | V\_id | Integer | Vehicle Id | Yes | No |
| Vehicle | Model\_no | Integer | Model no of the vehicle | No | No |
| Vehicle | V\_price | Integer | Price of the vehicle | No | No |
| Vehicle | Vehicle\_description – speed,mileage,cc,type | String | Description of the vehicle | No | No |
| Vehicle | C\_ssn | string | Social Securiety of customer | NO | Yes |
| Dealer | D\_name | String | Name of the dealer | No | No |
| Dealer | S\_id | string | Id of the showroom | No | Yes |
| Dealer | D\_id | Integer | Dealer Id | Yes | No |
| Dealer | D\_address | String | Address of the dealer | No | No |
| Dealer | D\_ph\_no | Integer | Phone no of the dealer | No | No |
| Showroom | S\_name | String | Name of the showroom | No | No |
| Showroom | V\_id | string | Vehical ID | No | Yes |
| Showroom | E\_id | string | Employee id | No | Yes |
| Showroom | S\_id | Integer | Showroom Id | Yes | No |
| Showroom | Location | String | Location of the showroom | No | No |

**Question 8:** Draw a Data Flow Diagram illustrating the information you have identified in Question 1. OR Draw a Use Case Diagram illustrating the information you have identified in Question 1.



**Submission Date:**

**Design Phase Report**

**Category: Application Development**

**Title: Yamaha Showroom Database Application**

**List of Team Members: Bharat Banavalikar, Sandeep DM, Rishabh Verma, Dileep Hegde**

**Team Leader: Bharat Banavalikar**

**Responsibilities:**

Bharat has normalized the relations and optimized the schema and also added tools as a part of the application.

Sandeep has designed the main structure of the application and the design phase report.

Dileep has designed the login page along with the sign up page.

Rishabh has done the data insertion and manipulation part of the application.

**Requirements Specification:**

Retrieval: Data can be easily retrieved from the database according to the customer’s/client’s requirements.

Data manipulation: the application allows the client to insert new record or update the existing record according to his/her requirement for any of the given entity.

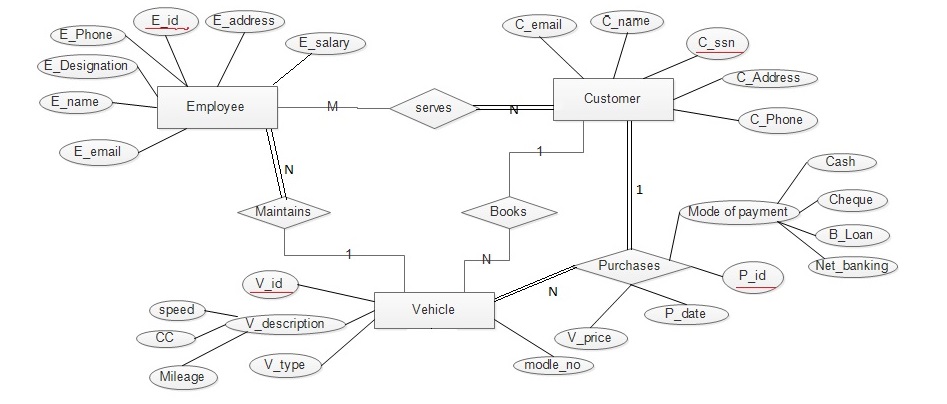
Invoice generation: the total estimated bill for the purchase is generated which includes the price of the bike and discounts (if available) and hence the total price.

Graphical representation of statistics: the application effectively helps to analyze the data in the form of pictorial representation such as –a bar graph for the number of bikes sold for a particular month/year.

Availability of vehicles – the customer can easily surf for his/her desired vehicle in store. The customer is prompted to visit the showroom on a specified day if the vehicle is presently not in store.

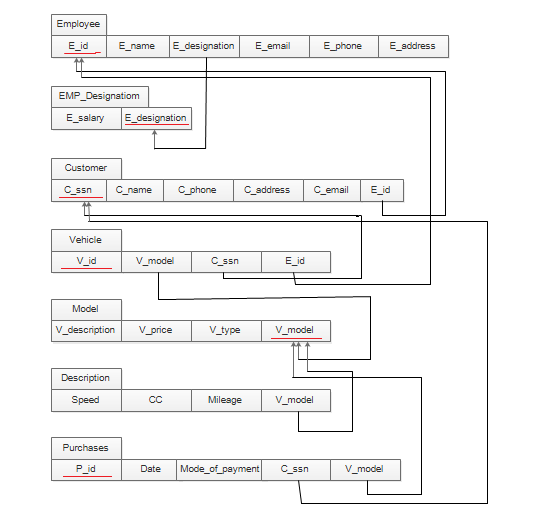
Service schedules- the customer is updated about his/her free services and its validity.

**ER Design:**



The entities Vehicle Showroom and Dealer are removed as there was only one dealer and only one showroom. Hence separate table is not required for those entities.

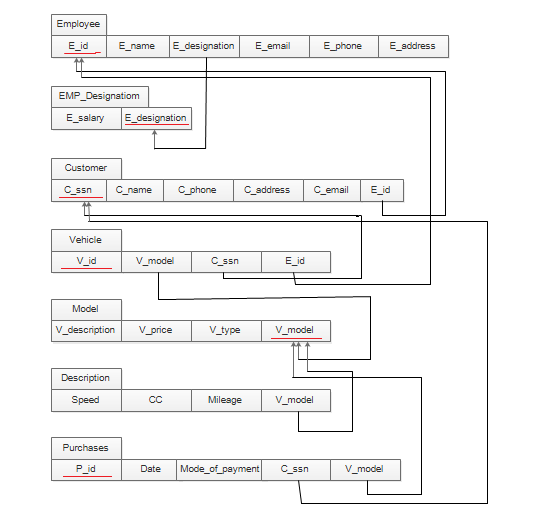
**ER to Relation Mapping:**



**ER to Relation Mapping:Data Dictionary**:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Object (Entity)** | **Name (Attribute)** | **Type**  **(Data type)** | **Description** | **Primary Key** | **Foreign Key** |
| Customer | Customer\_name | String | Name of the customer | No | No |
| Customer | C\_Address | String | Address of the customer | No | No |
| Customer | C\_SSN | String | Unique Identification number for the customer | Yes | No |
| Customer | Ph\_no | Integer | Phone number of the customer | No | No |
| Customer | E\_id | String | Unique Identification number for the Employee | No | Yes |
| Customer | C\_email | String | Email id of the customer | No | No |
| Employee | E\_name | String | Name of the employee | No | No |
| Employee | E\_id | Integer | Unique Identification number for the employee | Yes | No |
| Employee | E\_designation | String | Designation of the employee | No | Yes |
| Employee | E\_ph\_no | Integer | Phone number of the employee | No | No |
| Employee | E\_email | String | Email id of the employee | No | No |
| Employee | E\_address | String | Address of the employee | No | No |
| EMP\_Designation | E\_salary | Integer | Salary of the employee | No | No |
| EMP\_Designation | E\_designation | String | Designation of the employee | Yes | No |
| Vehicle | V\_id | String | Unique Identification number for the vehicle | Yes | No |
| Vehicle | V\_Model | String | Model no of the vehicle | No | Yes |
| Vehicle | C\_ssn | String | Unique Identification number for the customer | No | Yes |
| Vehicle | E\_id | String | Unique Identification number for the employee | No | Yes |
| Model | V\_type | String | Gear/without gear | No | No |
| Model | V\_price | Integer | Price of the vehicle | No | No |
| Model | V\_model | String | Unique Identification for the vehicle model | Yes | No |
| Model | V\_description | String | Description about the vehicle | No | No |
| V\_description | Speed | Integer | Speed of the vehicle | No | No |
| V\_description | CC | Integer | Cylinder capacity of the vehicle | No | No |
| V\_description | Mileage | Integer | Number of kms travelled in 1 litre | No | No |
| V\_description | V\_id | String | Unique Identification number for the vehicle | No | Yes |
| Purchases | P\_id | String | Unique Identification number for the bill | Yes | No |
| Purchases | Date | Date | Date of purchase | No | No |
| Purchases | Mode\_pay | String | Mode of the payment | No | No |
| Purchases | C\_id | String | Unique Identification number for the customer | No | Yes |
| Purchases | V\_model | String | Unique Identification number for the vehicle model | No | Yes |

**Question 1: Normalization:** Are all the relations in your chosen schema in 3NF? Are they in BCNF? Explain your answers. If any of your relations are not in BCNF, normalize them to BCNF. If you choose to normalize your relations only till 2NF or 3NF, explain your reasons (e.g., the amount of redundancy introduced is limited or some other valid reason).

****

In the employee relation there was functional dependency E\_designation(non-prime attribute) which could determine E\_salary , therefore EMP\_Designation is made a separate relation and hence both the relation Employee and EMP\_Designation are in BCNF now.

In Vehicle table there was FD V\_model (non-prime attribute) which could determine V\_description, V\_price and V\_type. Therefore Model is made as a separate relation containing V\_description, V\_price and V\_type. Hence the relation Vehicle is now in BCNF.

The table Models had an attribute V\_description which was compound attribute. A separate relation Description is made to make the values atomic. Hence the relation Description is in BCNF now.

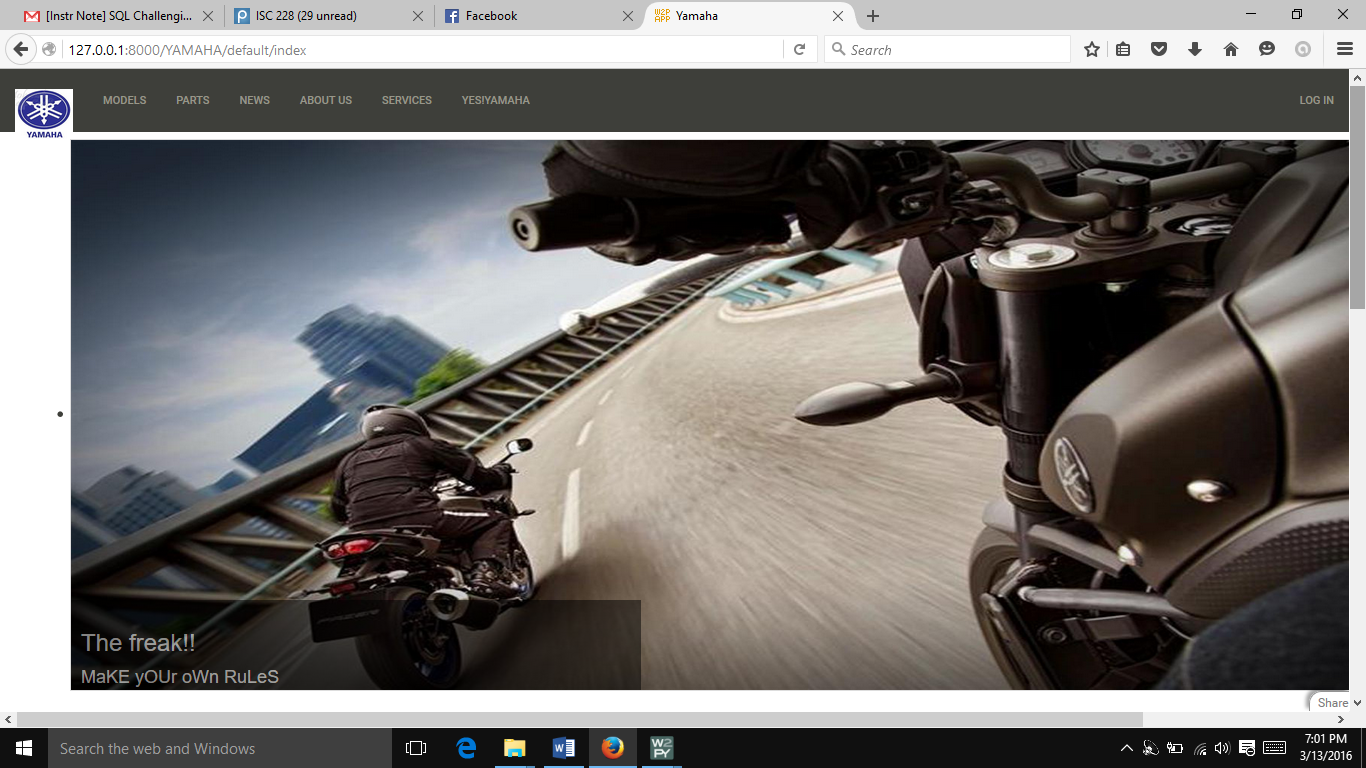
All the other relations are already in BCNF.

**Question 2:** Choose the **optimal** normalized schema from Question 1 and justify why you think this is an optimal solution.

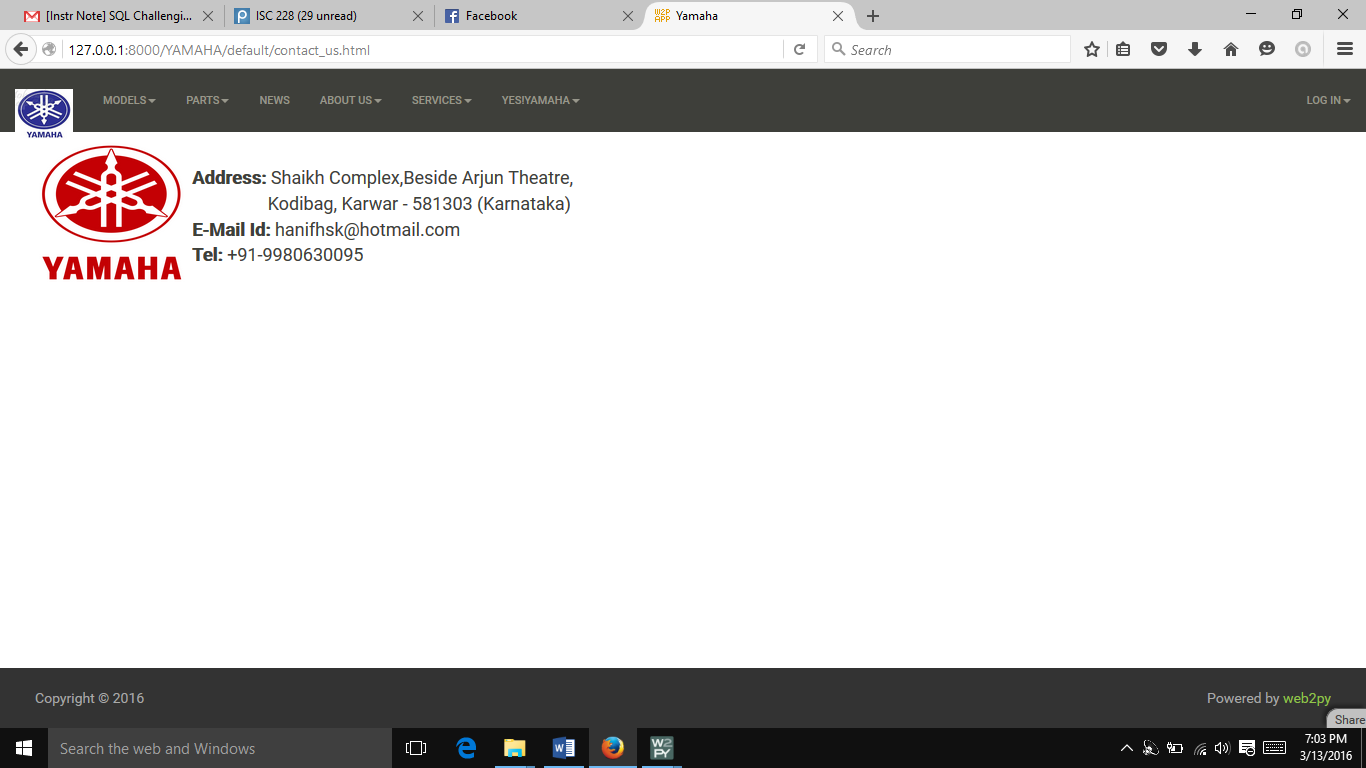
The above relations can be normalized only in one way. Therefore there are no multiple answers. The above normalization is the optimized solution.

**Question 3:** UI design

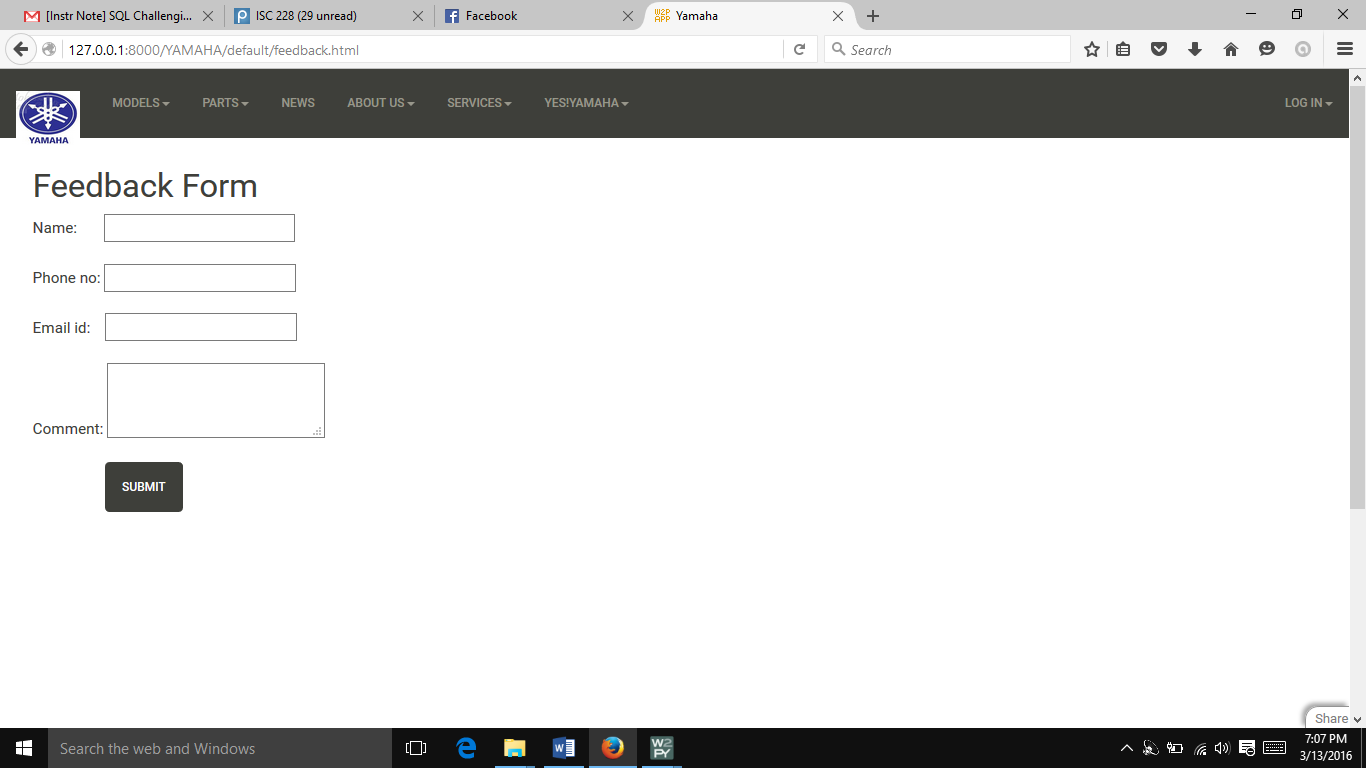
**Home page**



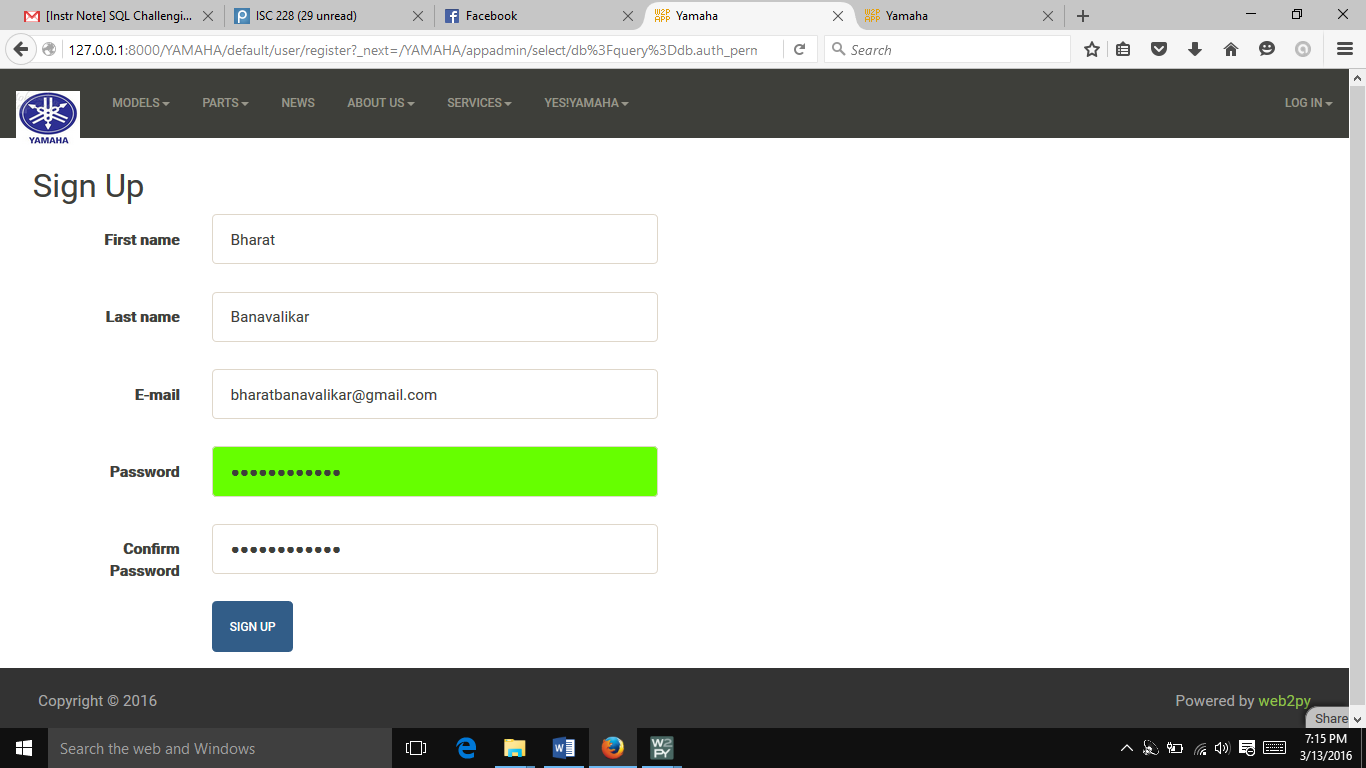
**Contact Us in About**



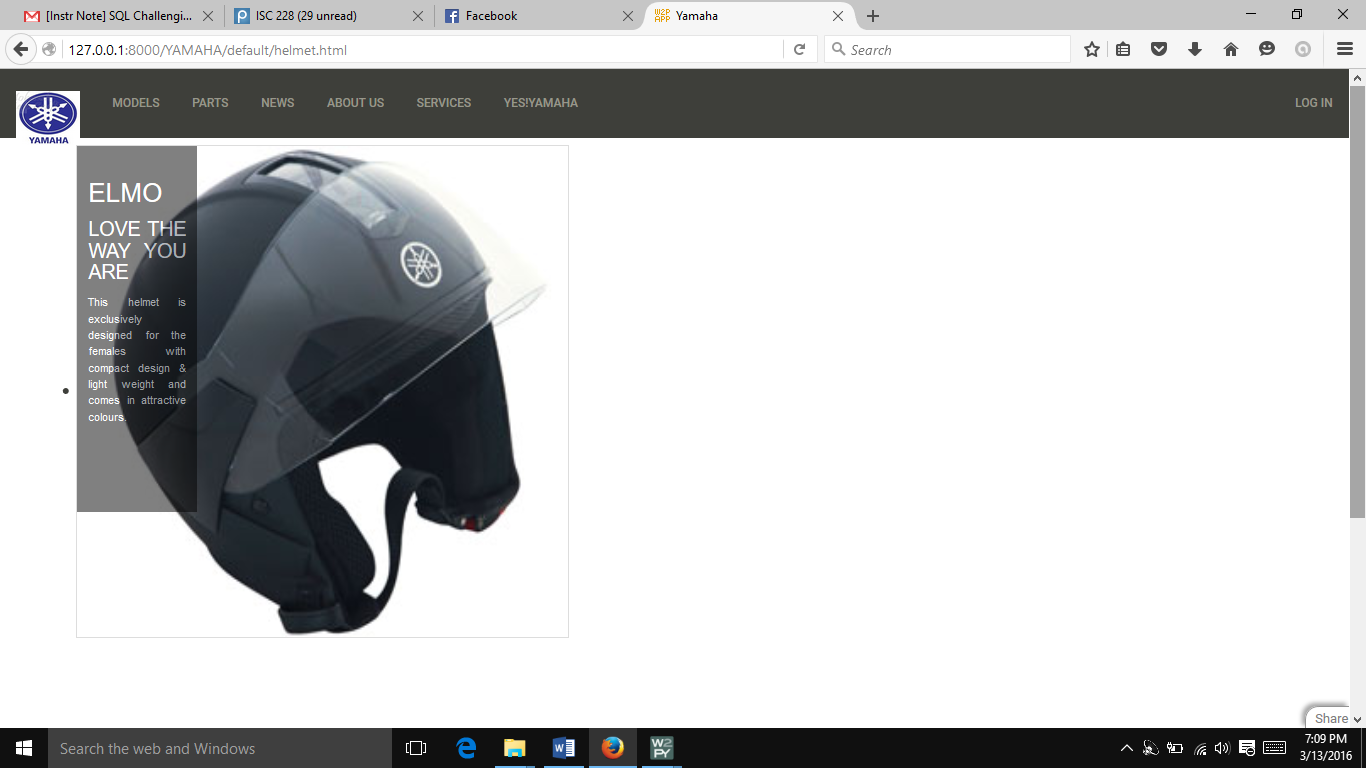
**Feedback form for Customers**



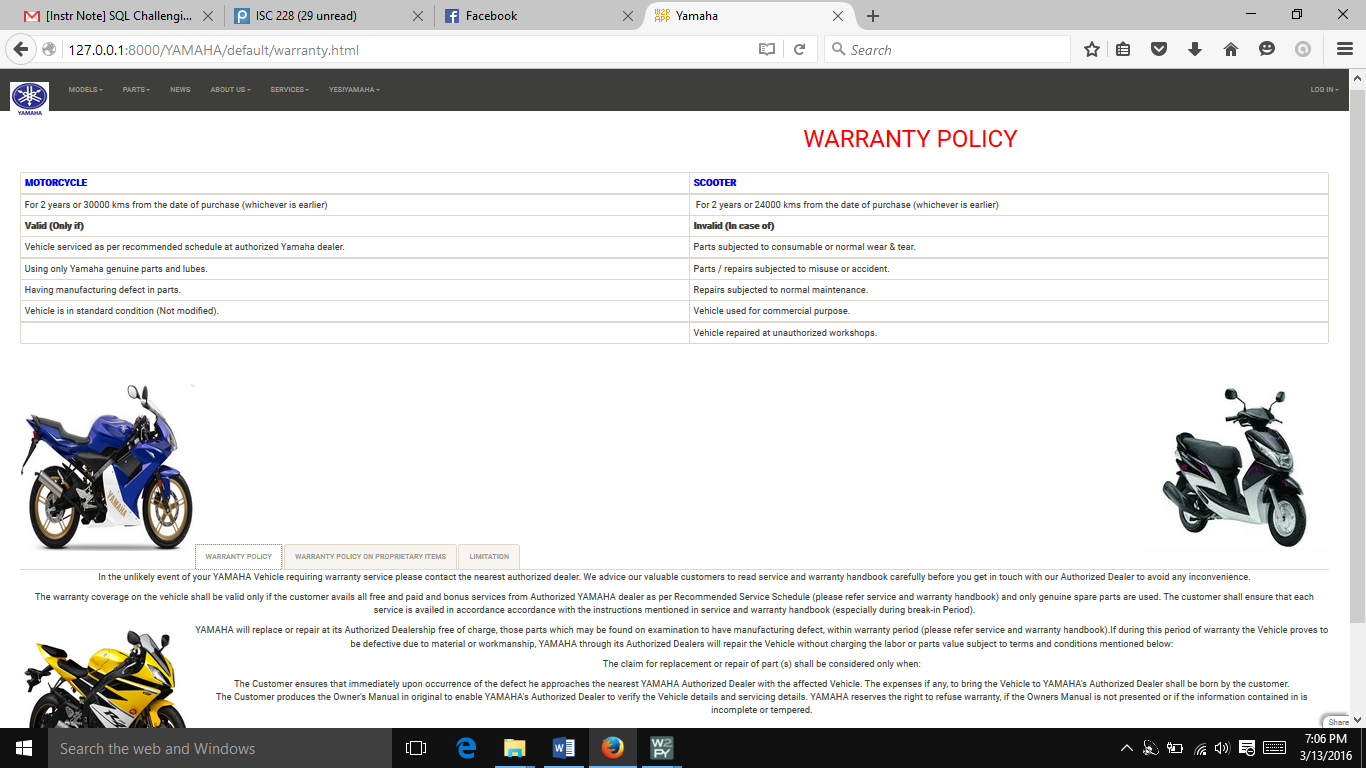
**Sign Up form**



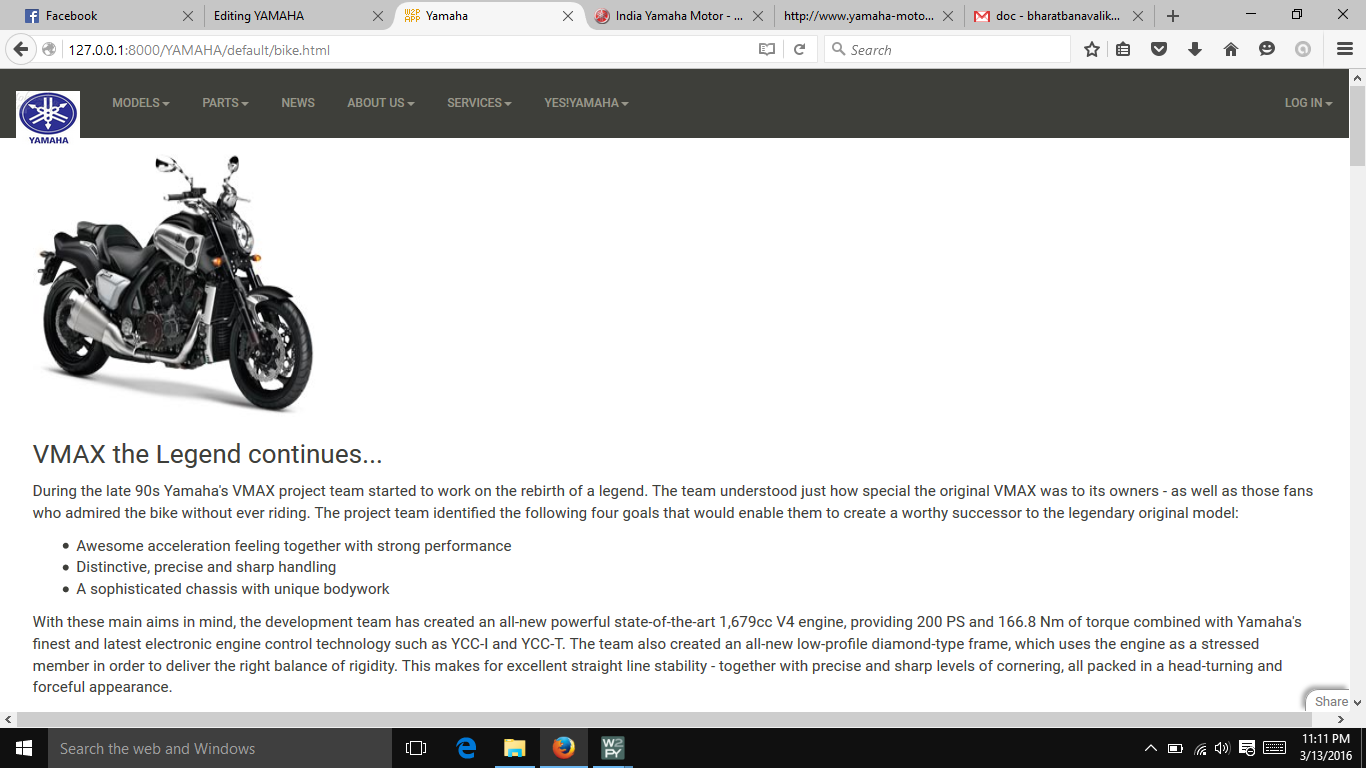
**Parts**



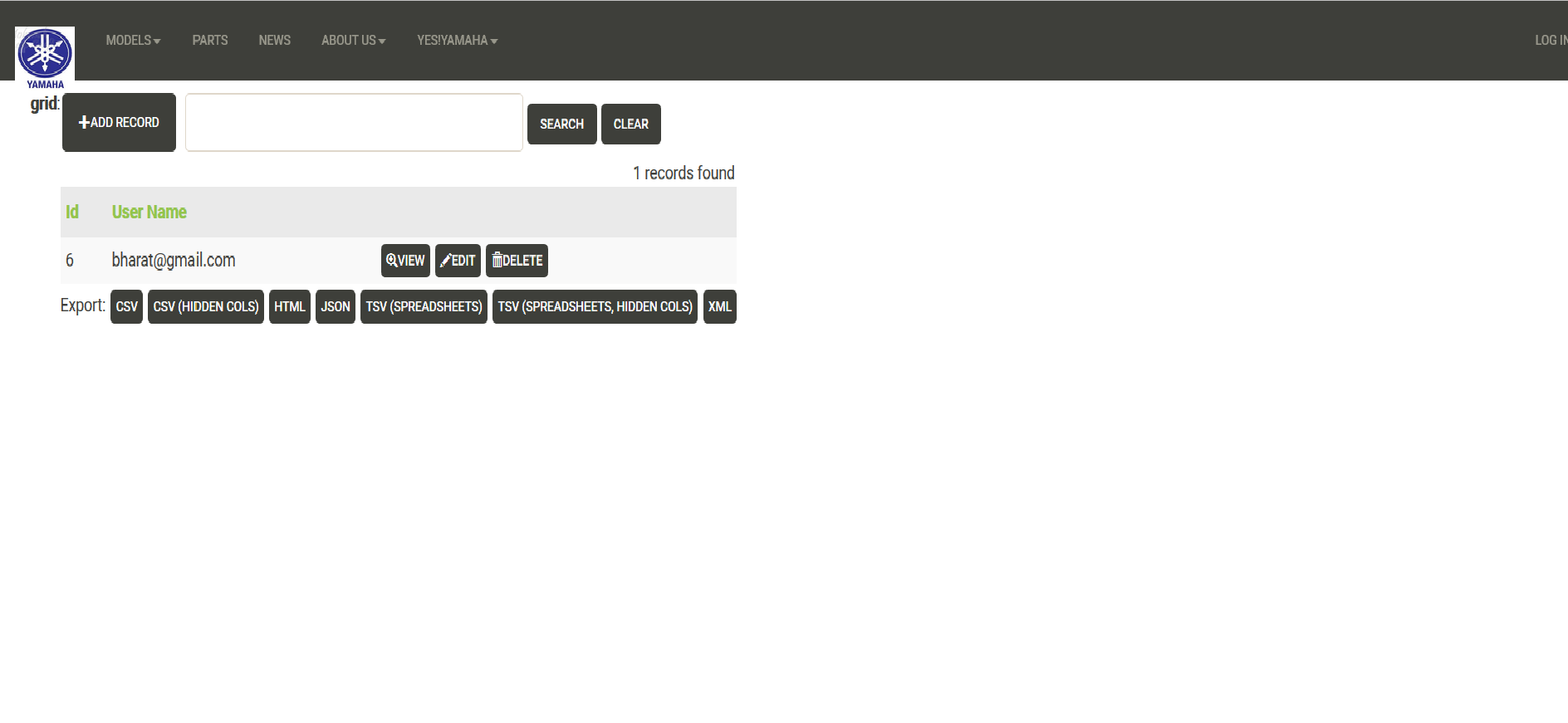
**Warranty**



**Bikes data**

****

**Data manipulation**

**Submission Date: 13-March-2016**

**Implementation Phase Report**

**Title: Yamaha Showroom Database Application**

**List of Team Members: Bharat Banavalikar, Dileep Hegde, Sandeep Dm, Rishabh**

**Team Leader: Bharat Banavalikar**

**Responsibilities:**

Bharat has done the data creation.

Sandeep did tables and vehicles parts.

Dileep did admin and employee pages and edited all pages.

Rishabh added bikes and scooter specifications page.

**Design updating:** [Describe the changes (if any) that need to be made to the Entity-Relationship Diagram you have given in Design Phase before it can be implemented as tables in SQL. If you

No changes made.

**Implementation Phase Questions to be answered**

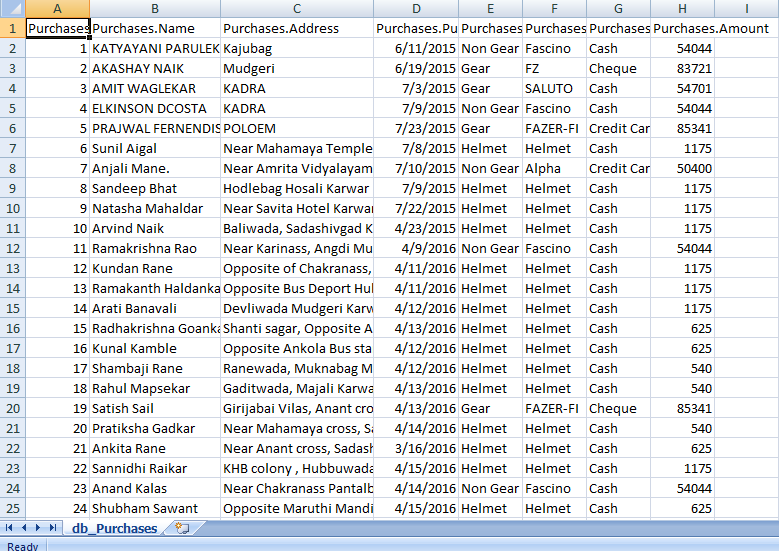
**Normalization:** [Any change in normal form as compared to design phase do include]

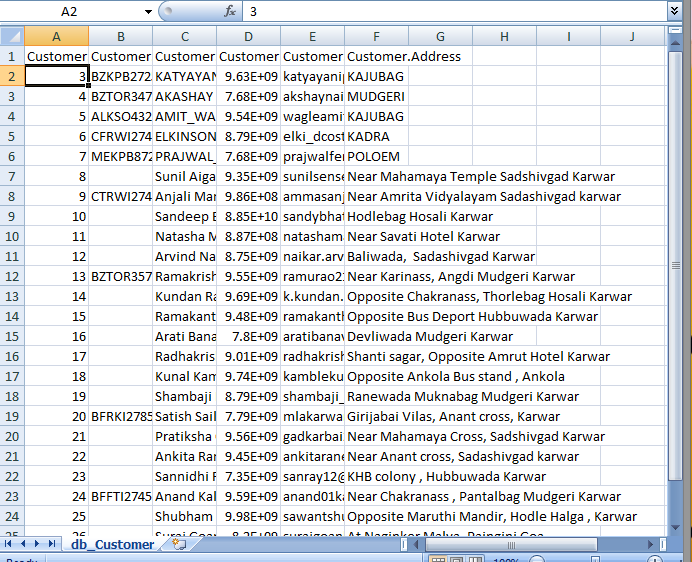
No changes made.

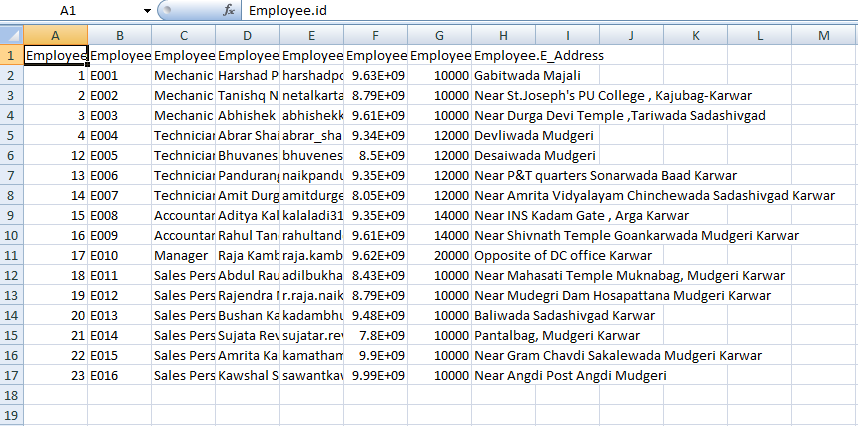
**Question1:** Give the SQL statement(s) used to create the Oracle/MySQL database tables needed to implement the normalized relational schema.

db=DAL(‘sqlite://storage.sqlite’)

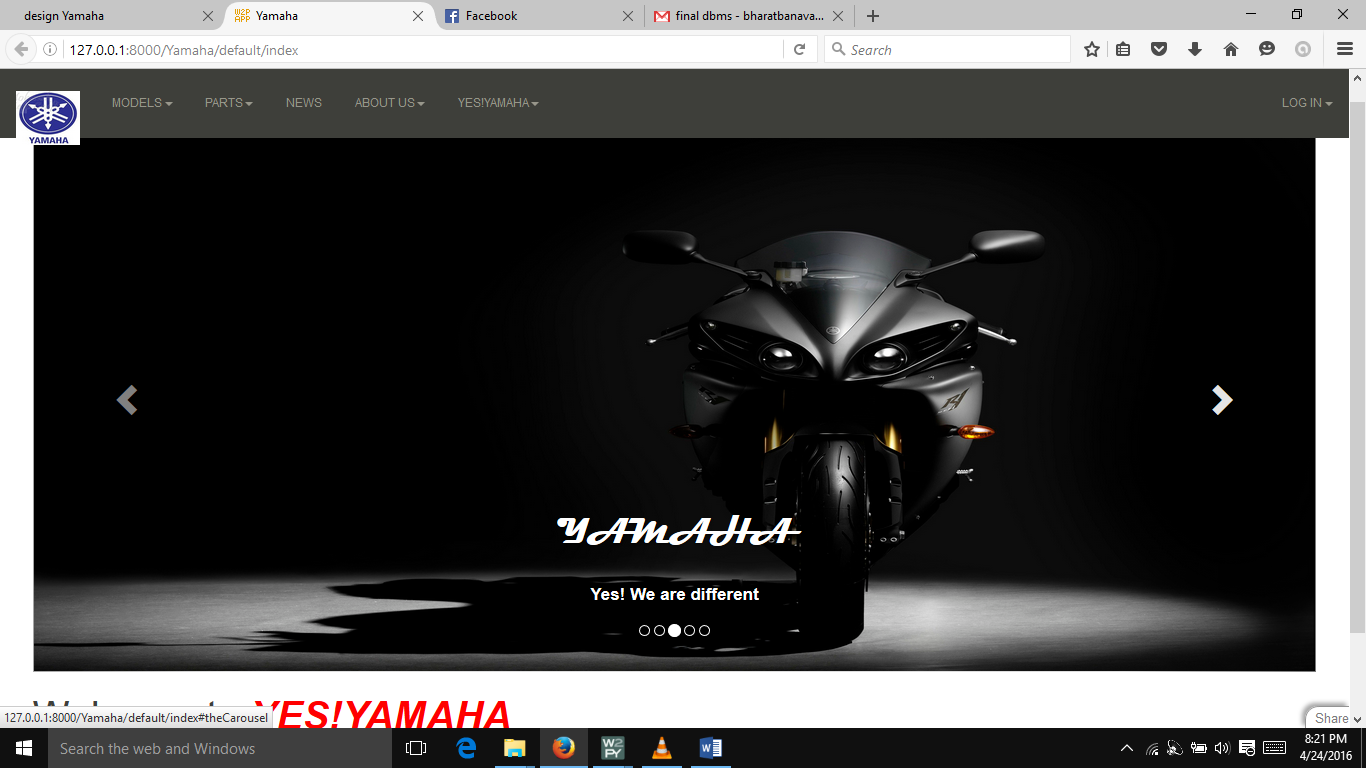
**Question2:** Give the actual data stored in each table of the database. (samples)

****



****

**Question3:** Give the snapshots, description and SQL queries for each of the user interface forms for your application. (Create the front end using python and hook it up to the SQL.)



Default view of the application which is visible to everyone without login in. Here user can get details about bikes/scooters, parts, warranty policy, client details and can give their feedbacks

**Quries**

def index():

if auth.has\_membership('Employee'):

redirect(URL('YAMAHA','manage','index1'))

elif auth.has\_membership('Admin'):

redirect(URL('YAMAHA','manage','index'))

return dict()

**The queries that define the main frame view(default view)**

def feedback():

form=SQLFORM(db.feedback,\_style='width:500px;height:40px',separator=': \*')

if form.accepts(request,session):

response.flash = 'Thanks! The form has been submitted.'

elif form.errors:

response.flash = 'Please correct the error(s).'

else:

response.flash = '(\*)Try again - no fields can be empty.'

return dict(form=form)

**The above Queries are the once that define feedback forms**

db.define\_table('feedback',

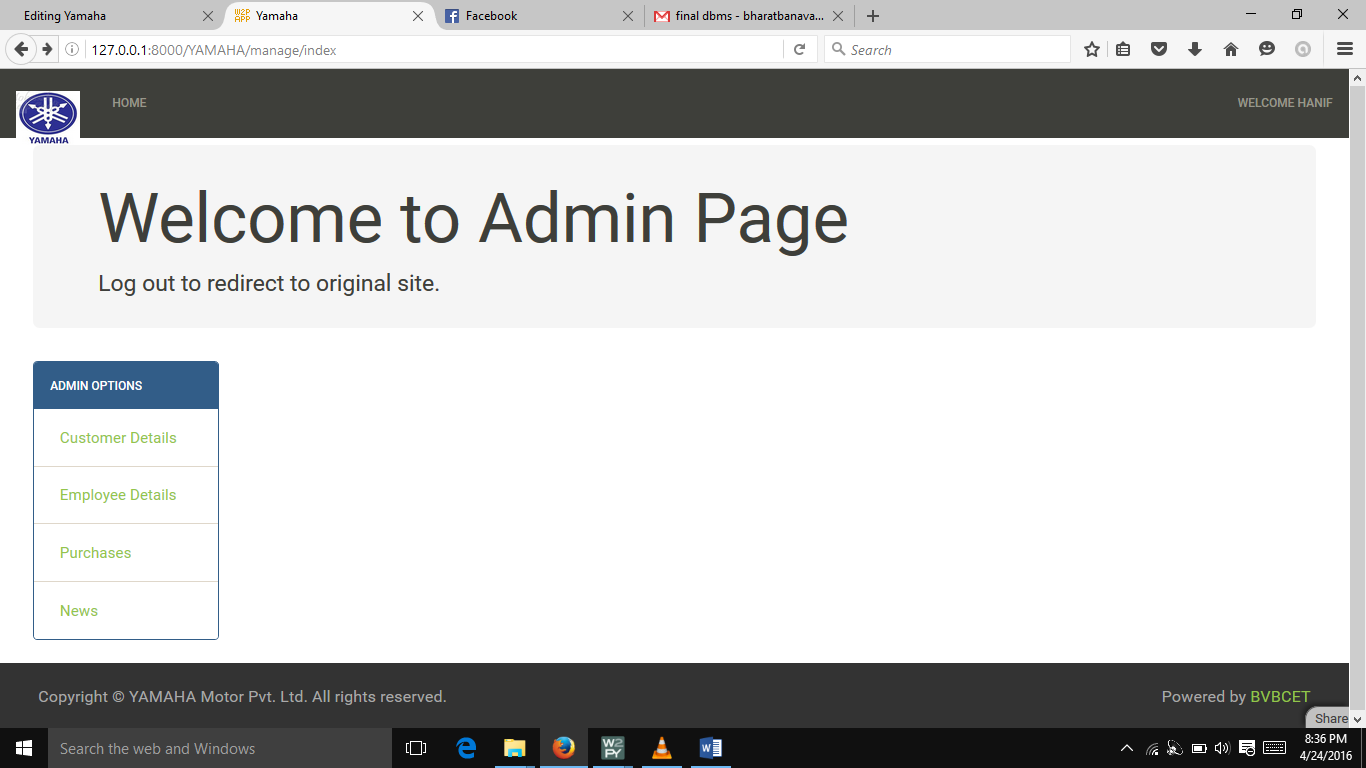
Field('Name',requires=IS\_NOT\_EMPTY()),

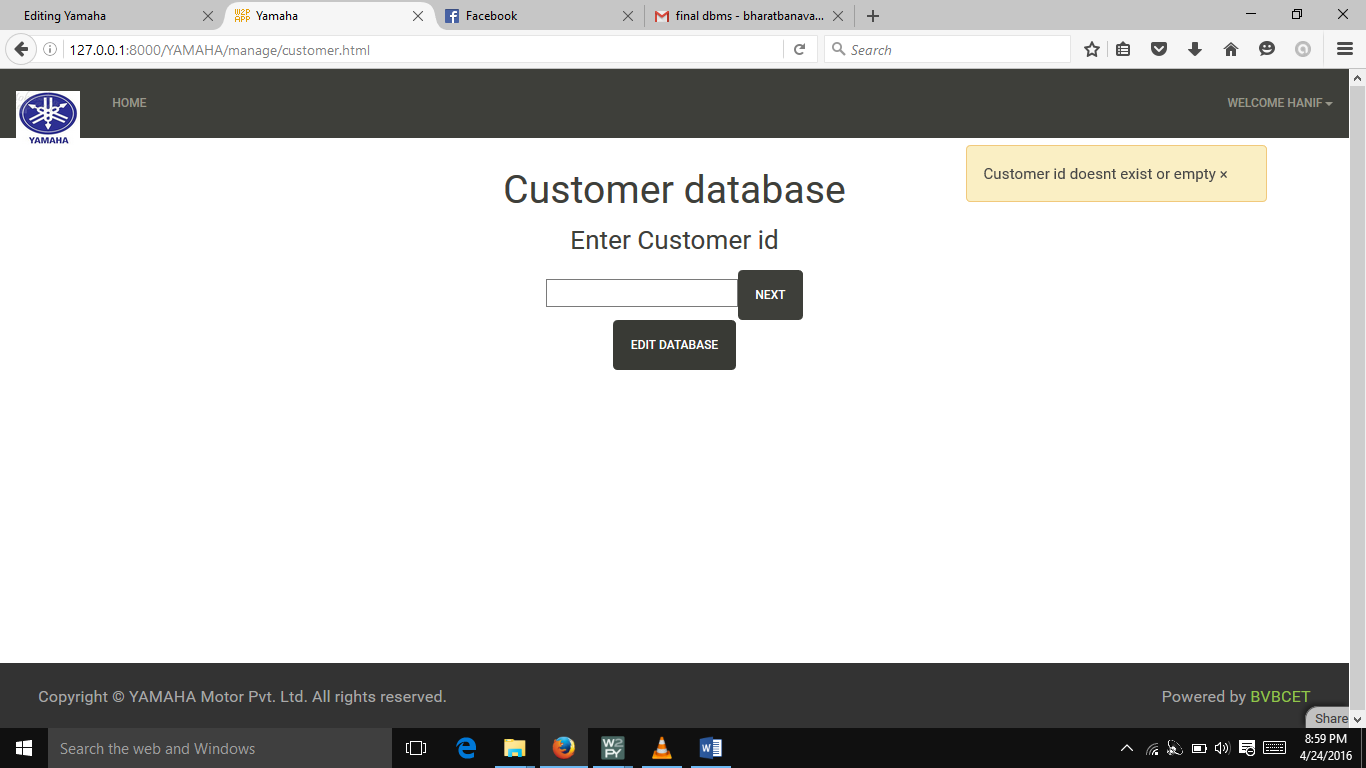
Field('Phone'),

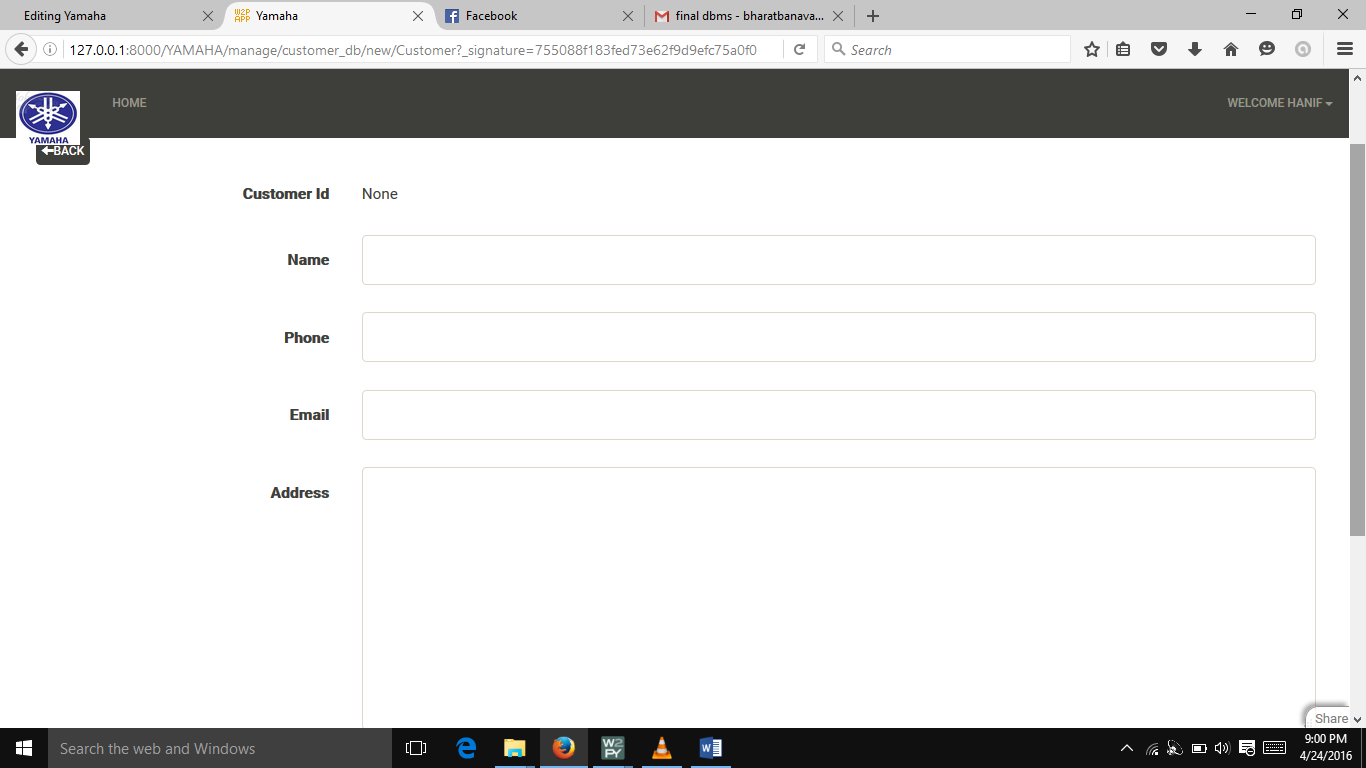
Field('Comments',type='text')

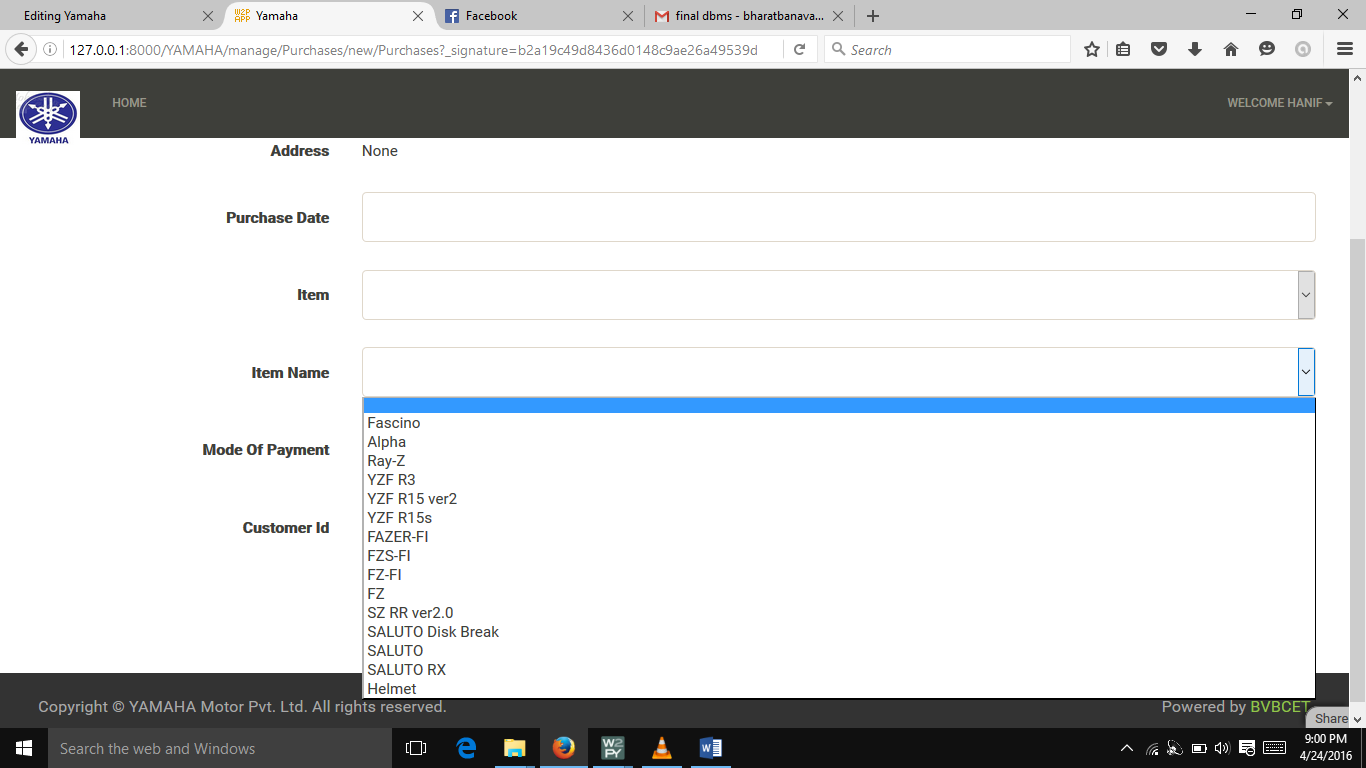
)

**The above Query is for feedback table**









Admin view, where in admin can add/edit employee details, customer details, purchase details,news etc..

**Query for customer table**

db.define\_table('Customer',

Field('Customer\_ID',requires=IS\_NOT\_EMPTY(),writable=False ),

Field('Name',requires=IS\_NOT\_EMPTY() ),

Field('Phone',requires=IS\_NOT\_EMPTY() ),

Field('Email',requires=IS\_EMAIL() ),

Field('Address',type='text'),

Field('E\_ID')

)

**Query for Purchases table**

db.define\_table('Purchases',

Field('Name',requires=IS\_NOT\_EMPTY(),writable=False),

Field('Address',requires=IS\_NOT\_EMPTY(),writable=False),

Field('Purchase\_Date',type='date', requires=IS\_DATE()),

Field('Item',requires=IS\_IN\_SET(['Bikes','Scoter','Helmet'])),

Field('Item\_Name',requires=IS\_IN\_SET(['Fascino','Alpha','Ray-Z','YZF R3','YZF R15 ver2','YZF R15s','FAZER-FI','FZS-FI','FZ-FI','FZ','SZ RR ver2.0','SALUTO Disk Break','SALUTO','SALUTO RX','Helmet']) ),

Field('Mode\_of\_Payment',requires=IS\_IN\_SET(['CreditCard','Cash','Cheque','Bank Loan','Internet Banking'])),

Field('Customer\_ID')

)

**Query for news table**

db.define\_table('News',

Field('Details',type='text')

)

**Query for employee table**

db.define\_table('Employee',

Field('E\_Id',requires=IS\_NOT\_EMPTY(),writable=False),

Field('E\_Designation',requires=IS\_NOT\_EMPTY() ),

Field('E\_Name',requires=IS\_NOT\_EMPTY() ),

Field('E\_Email',requires=IS\_EMAIL() ),

Field('E\_Phone',requires=IS\_NOT\_EMPTY() ),

Field('E\_Address',type='text')

)

**Queries to search customer and to authenticate admin/employee view to edit/add customer**

@auth.requires\_login()

def customer\_db():

grid=SQLFORM.grid(db.Customer,

editable=(auth.has\_membership('Admin') or auth.has\_membership('Employee')),

deletable=auth.has\_membership('Admin'))

return dict(grid=grid)

@auth.requires\_login()

def customer():

form = FORM('',INPUT(\_name='SSN'),INPUT(\_type='submit',\_value='Next'))

if form.process().accepted:

session.x=form.vars.SSN

redirect(URL('YAMAHA','manage','cust'))

elif form.errors:

response.flash = 'form has errors'

else:

response.flash = 'Customer id doesnt exist or empty'

return locals()

@auth.requires\_login()

def cust():

y=session.x

rows=db(db.Customer.SSN==(y)).select(db.Customer.ALL)

if not rows:

redirect(URL('YAMAHA','manage','customer'))

return locals()

**Queries to search purchases details and to authenticate admin/employee view to edit/add customer**

@auth.requires\_login()

def Purchases():

grid=SQLFORM.grid(db.Purchases,

editable=(auth.has\_membership('Admin') or auth.has\_membership('Employee')),

deletable=auth.has\_membership('Admin'))

return dict(grid=grid)

@auth.requires\_login()

def c\_view():

y=session.x

rows=db(db.Customer.SSN==(y)).select(db.Customer.ALL)

if not rows:

redirect(URL('YAMAHA','manage','customer'))

return locals()

**Queries to authenticate admin/employee view for news section**

@auth.requires\_login()

def News():

grid=SQLFORM.grid(db.News,

editable=(auth.has\_membership('Admin') or auth.has\_membership('Employee')),

deletable=auth.has\_membership('Admin'))

return dict(grid=grid)

Queries for admin view to edit/add employee details

@auth.requires\_login()

def employee\_db():

grid=SQLFORM.grid(db.Employee,

editable=(auth.has\_membership('Admin') or auth.has\_membership('Employee')),

deletable=auth.has\_membership('Admin'))

return dict(grid=grid)

@auth.requires\_login()

def employee():

form = FORM('',INPUT(\_name='E\_Id'),INPUT(\_type='submit',\_value='Next'))

if form.process().accepted:

session.a=form.vars.E\_Id

redirect(URL('YAMAHA','manage','emp'))

elif form.errors:

response.flash = 'form has errors'

else:

response.flash = 'Employee id doesn''t exist or empty'

return locals()

@auth.requires\_login()

def emp():

b=session.a

rows=db(db.Employee.E\_Id==(b)).select(db.Employee.ALL)

if not rows:

redirect(URL('YAMAHA','manage','employee'))

return locals()

@auth.requires\_login()

def e\_view():

b=session.a

rows=db(db.Employee.E\_Id==(b)).select(db.Employee.ALL)

if not rows:

redirect(URL('YAMAHA','manage','employee'))

return locals()

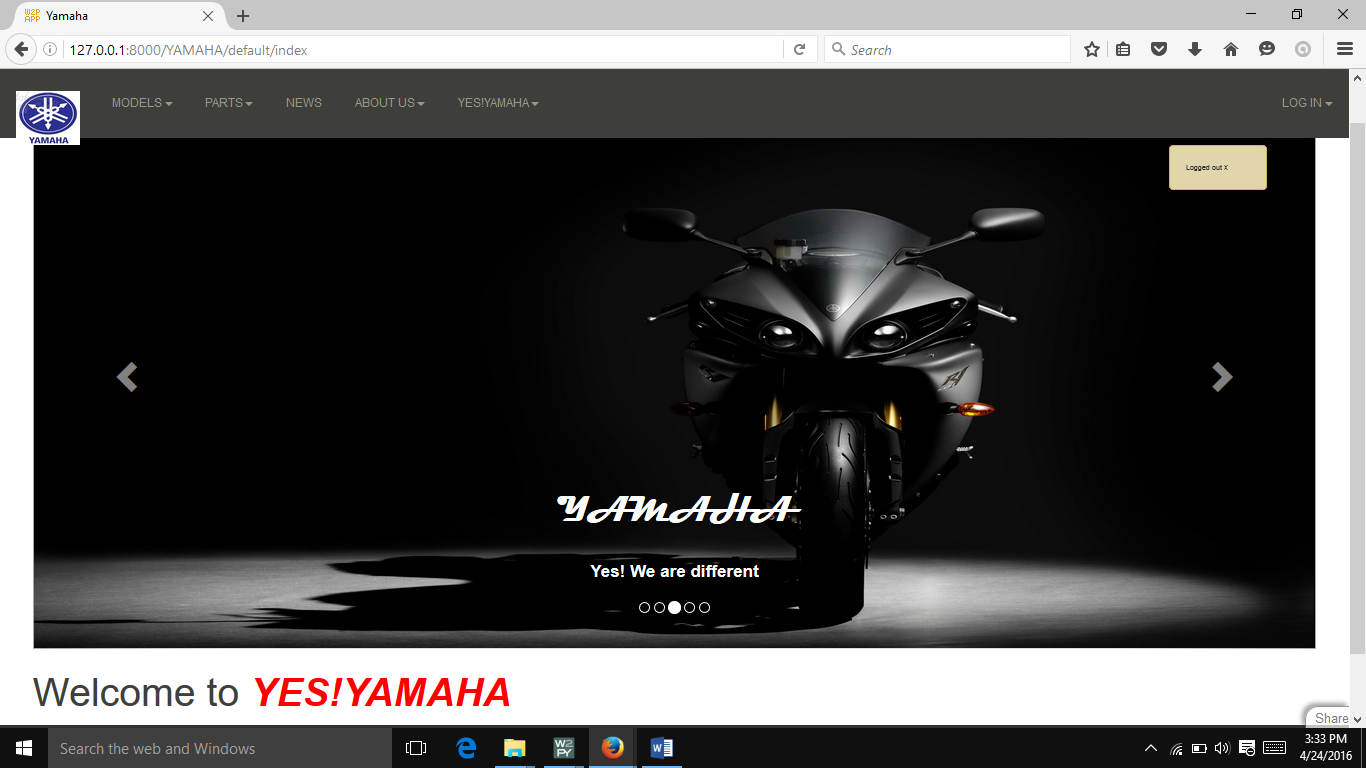
**Question4:** Give all possible final reports and graphs obtained by your application.

**No graphs obtained.**

**Submission Date:25-04-16**

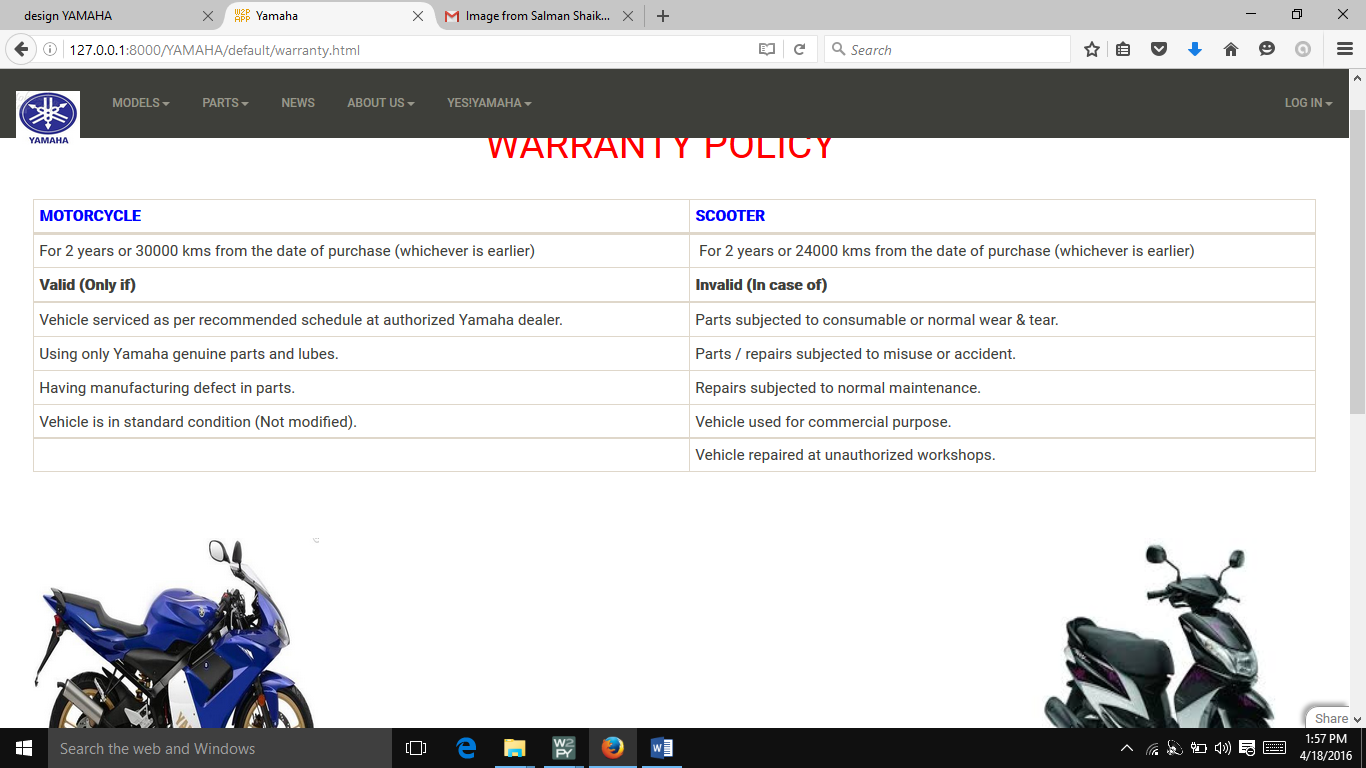
**Readme**

Step 1:Go to url : bharatbanavalikar/pythonanywhere.com/YAMAHA

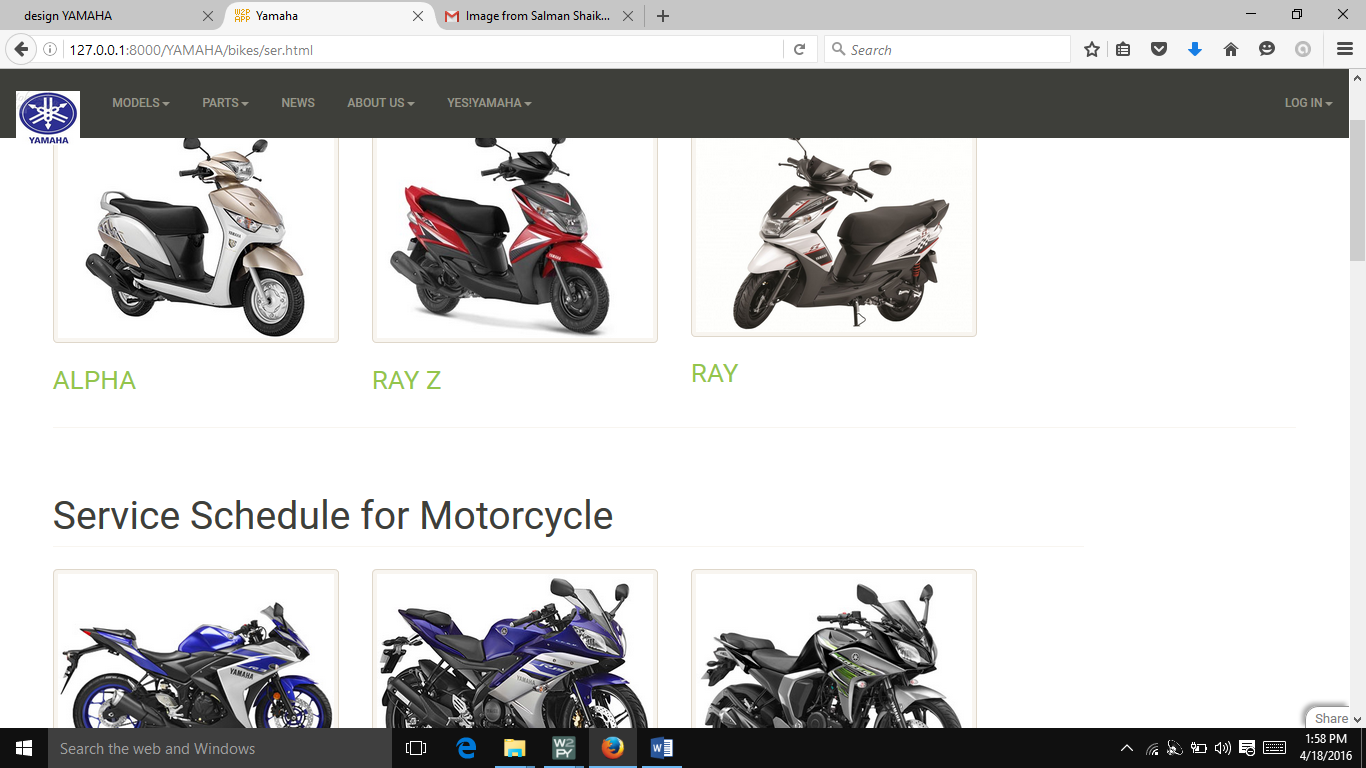


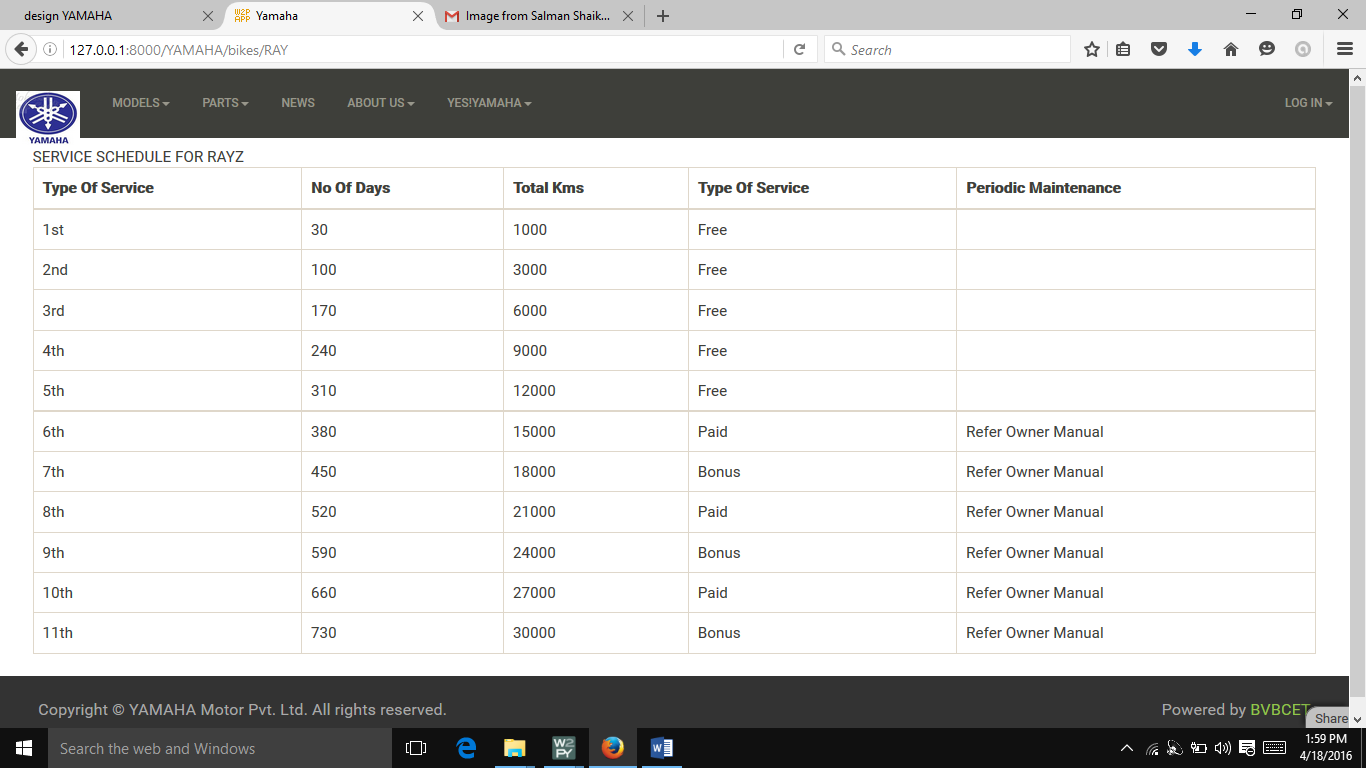
This view is designed for customers. Wherein they can refer details about Bikes/Scooters, parts, News, Contact details of our client, feedback forms, warranty policy, services schedules etc.

Warranty Policy

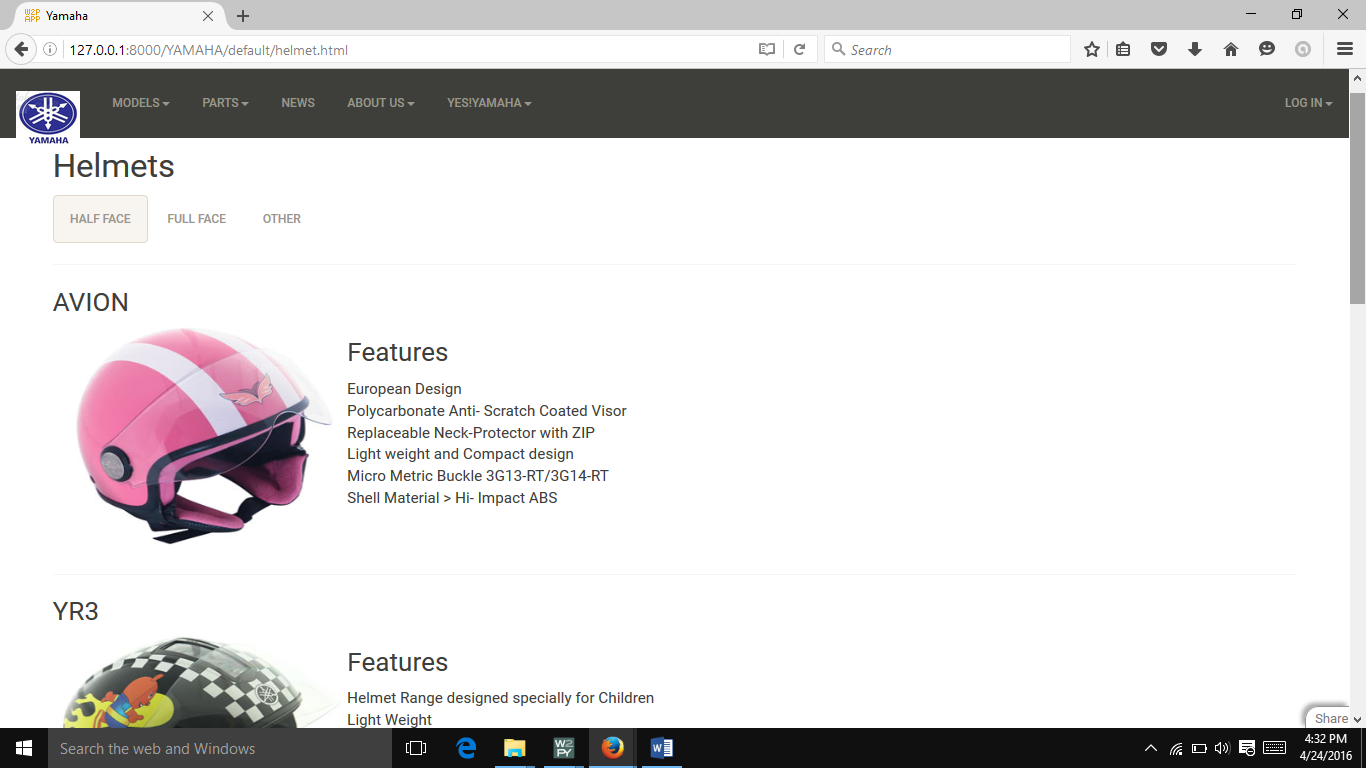


Service Schedules

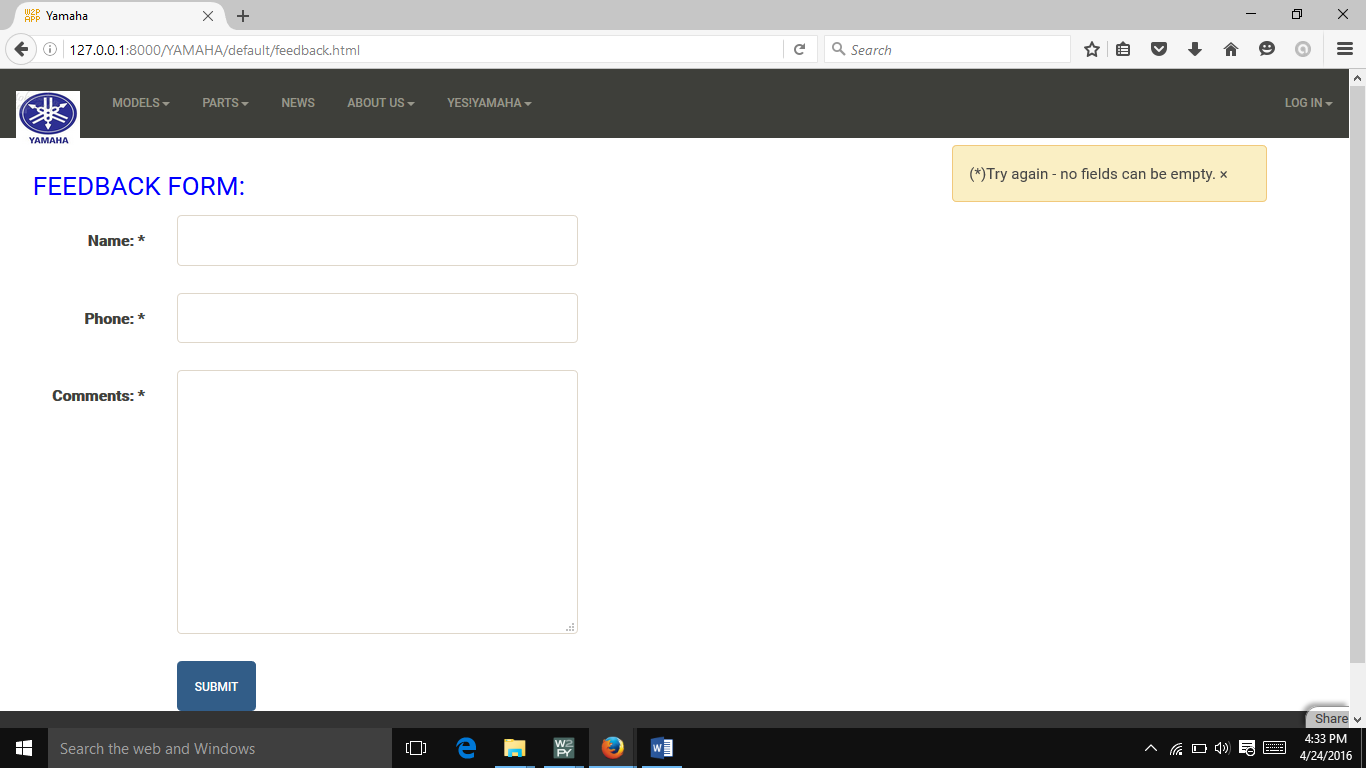




Helmets

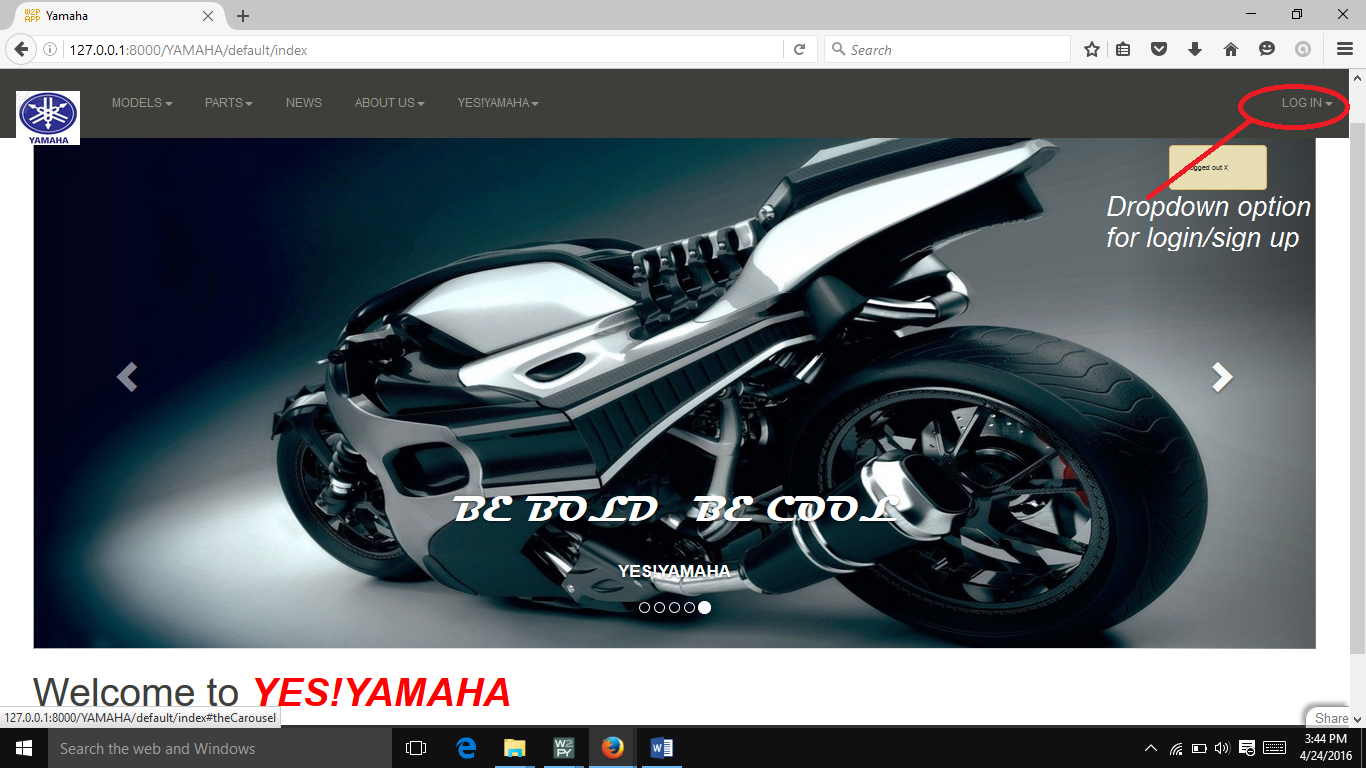


Feedback



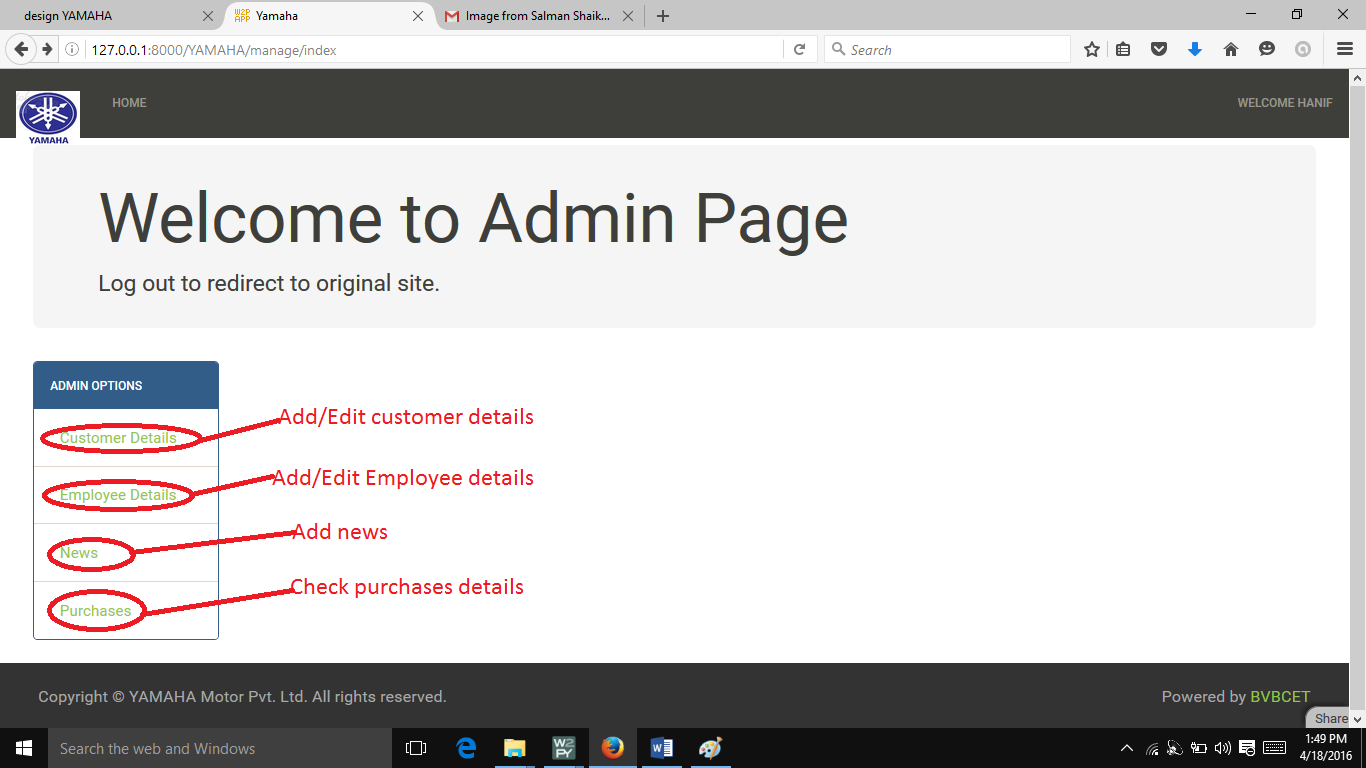
Step 2:

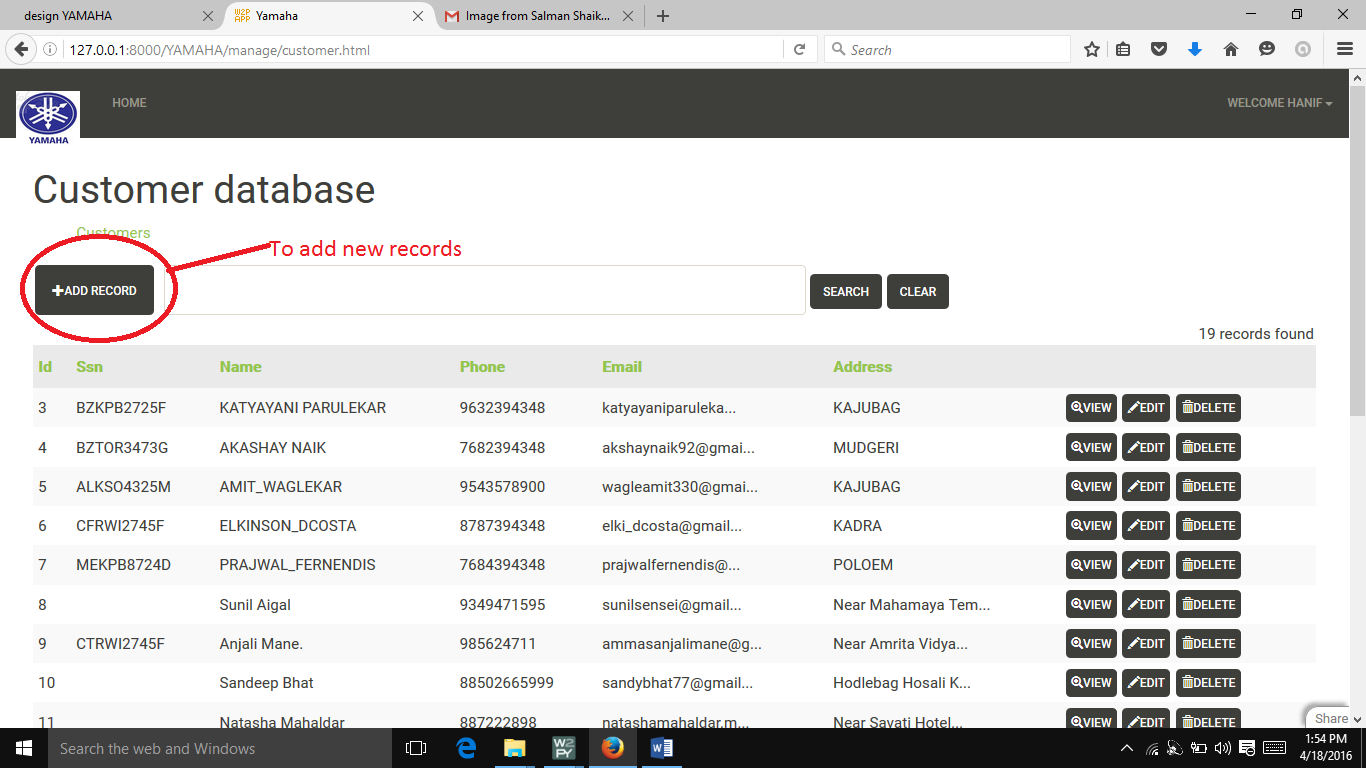
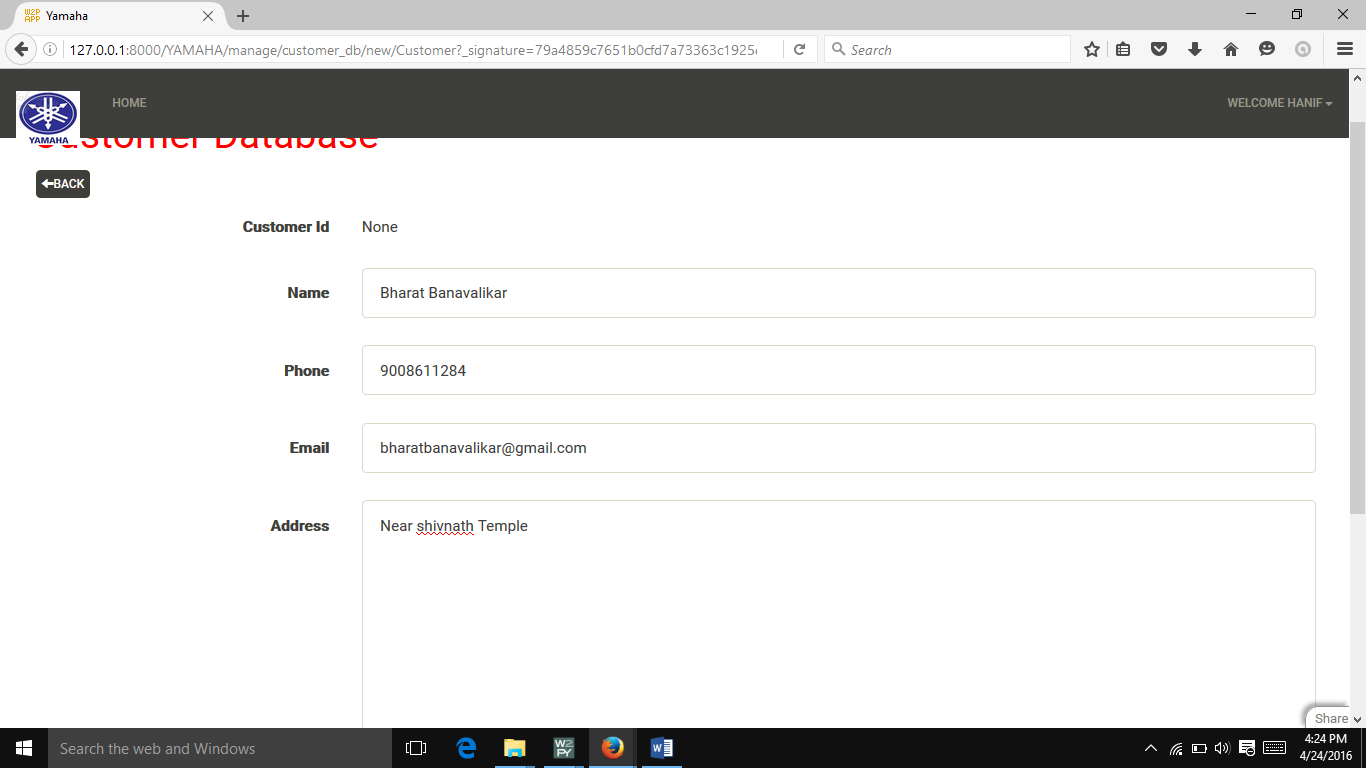
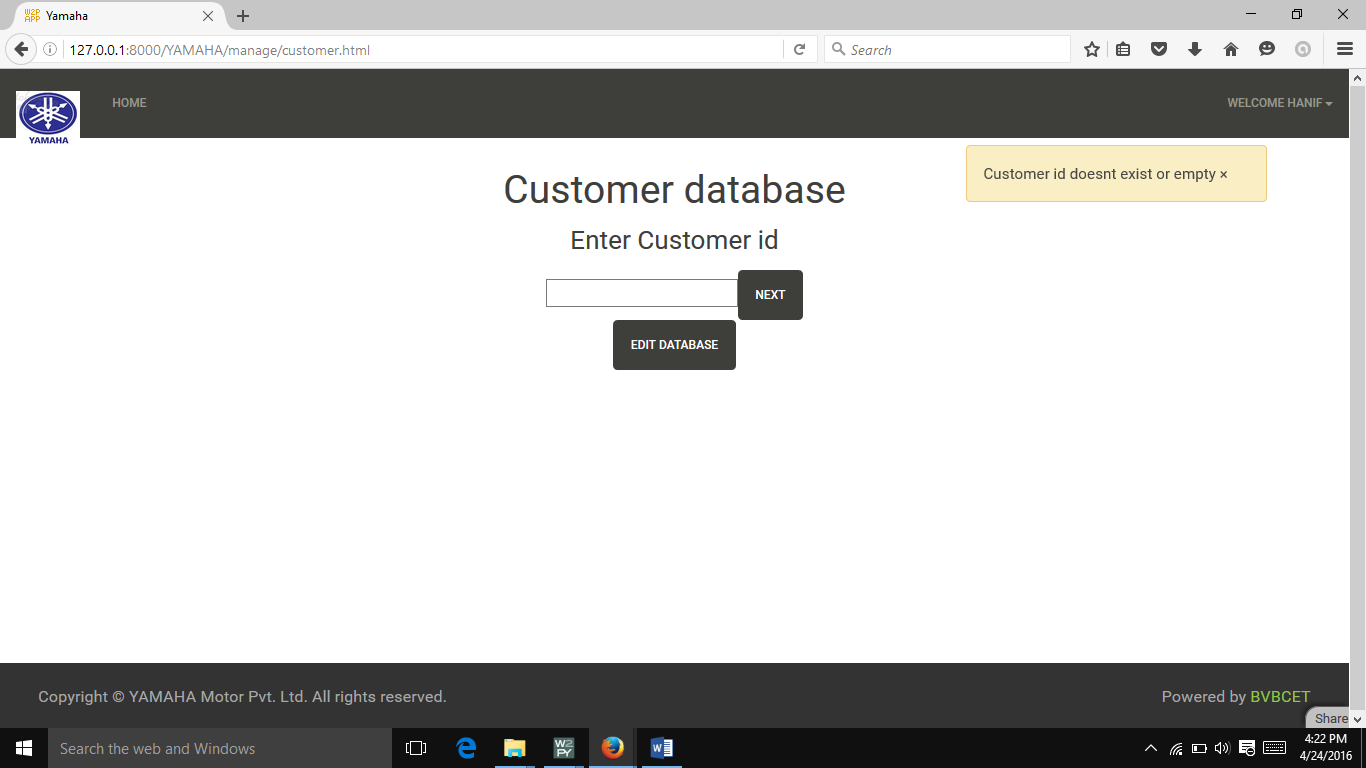
Login in as admin or employee with the given Id and password or create new id with signup option.



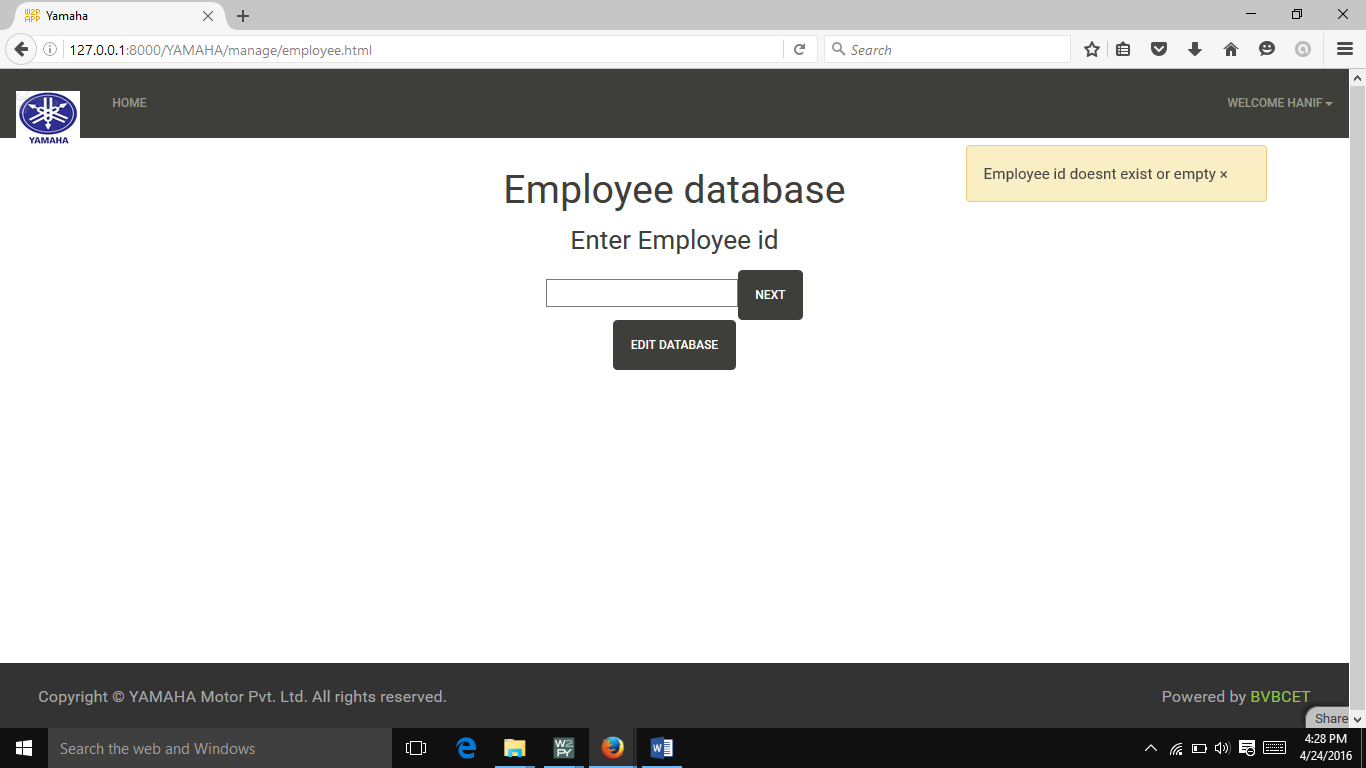
Step 3:

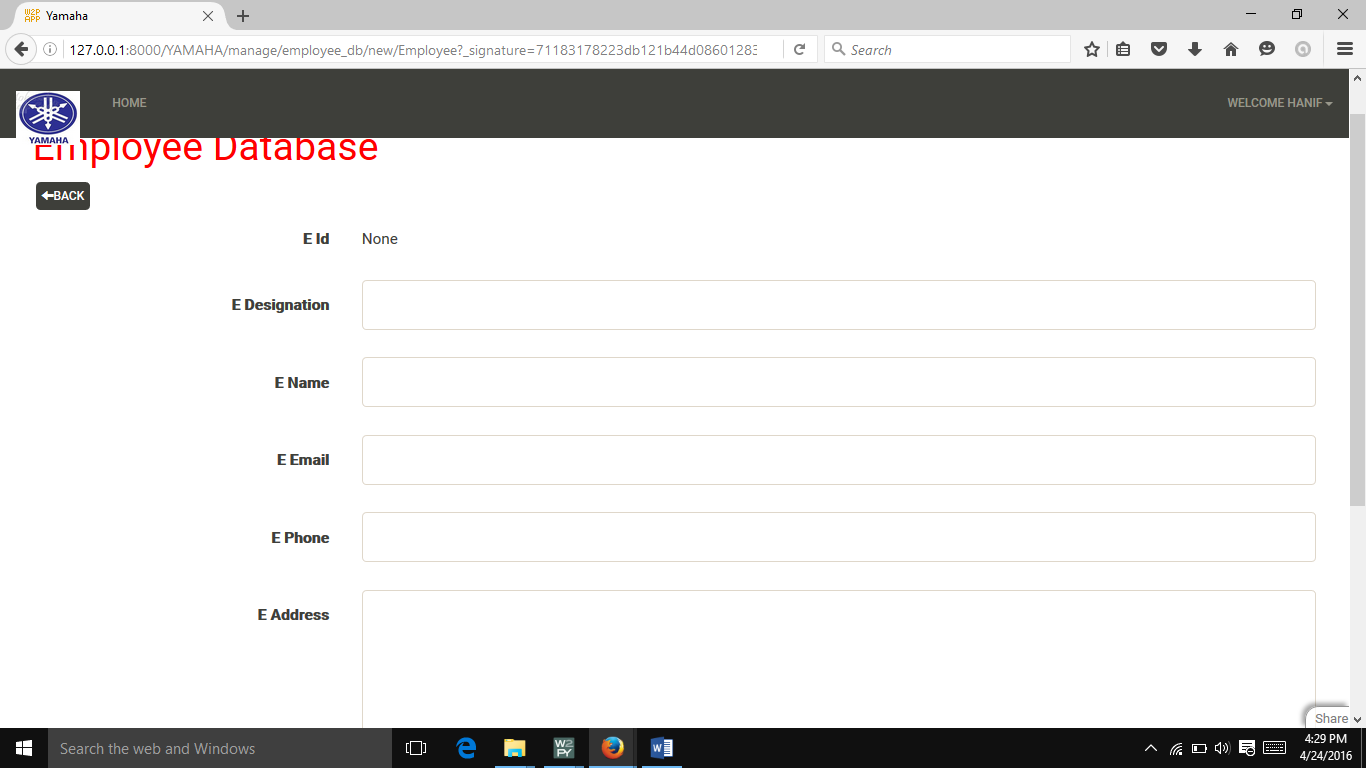
Management Page, click on any of the options to edit or add



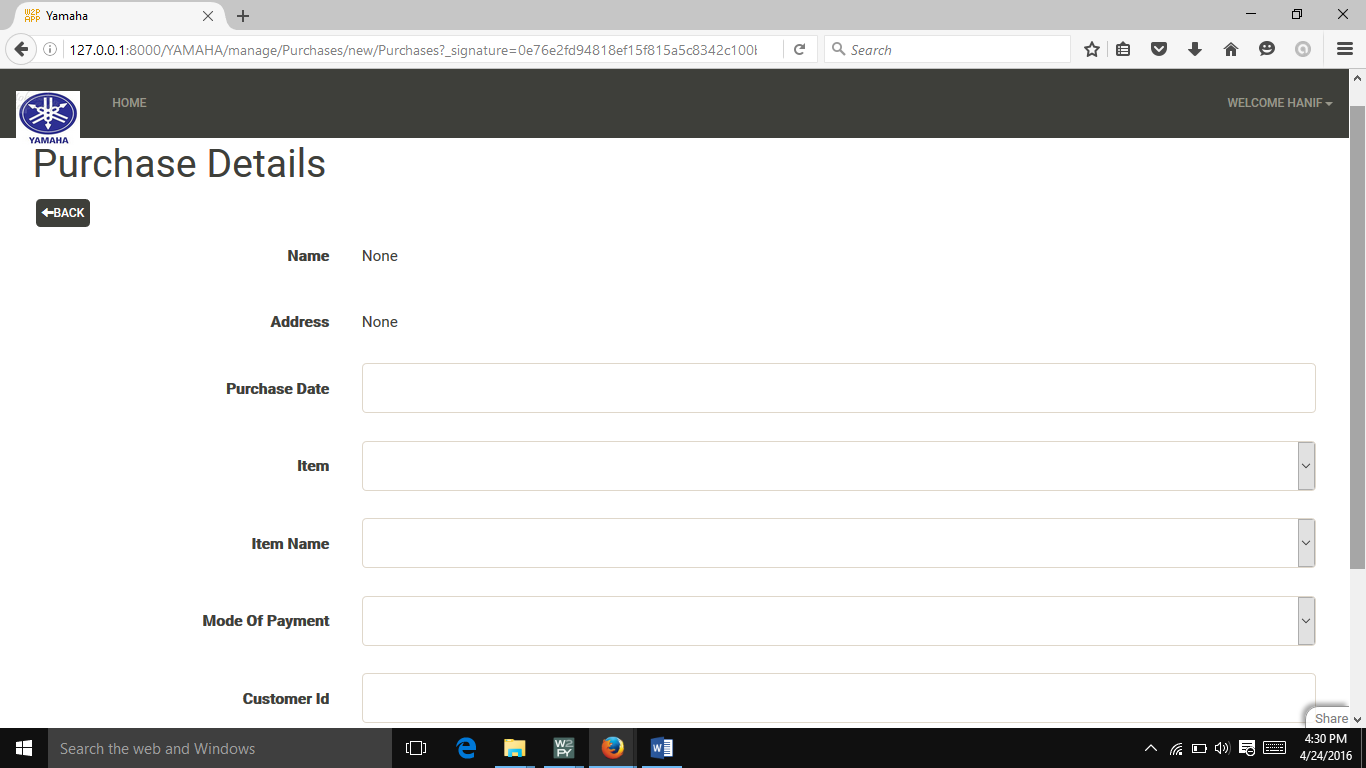
Step 4:Customer Data insertion or searching..

Step 5: Add/Edit/search Employee details (only available to Admin)

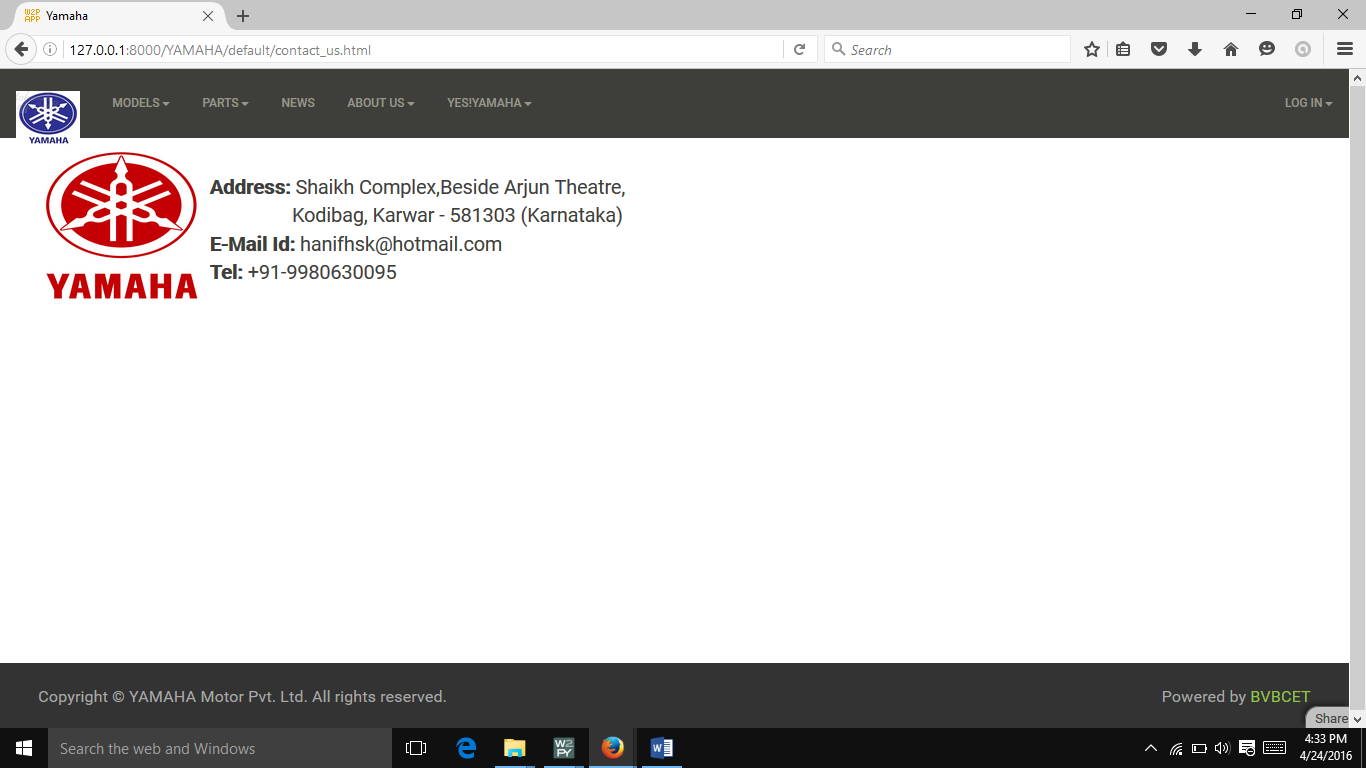




Step 6: Edit/add details for Purchases



Contact Details

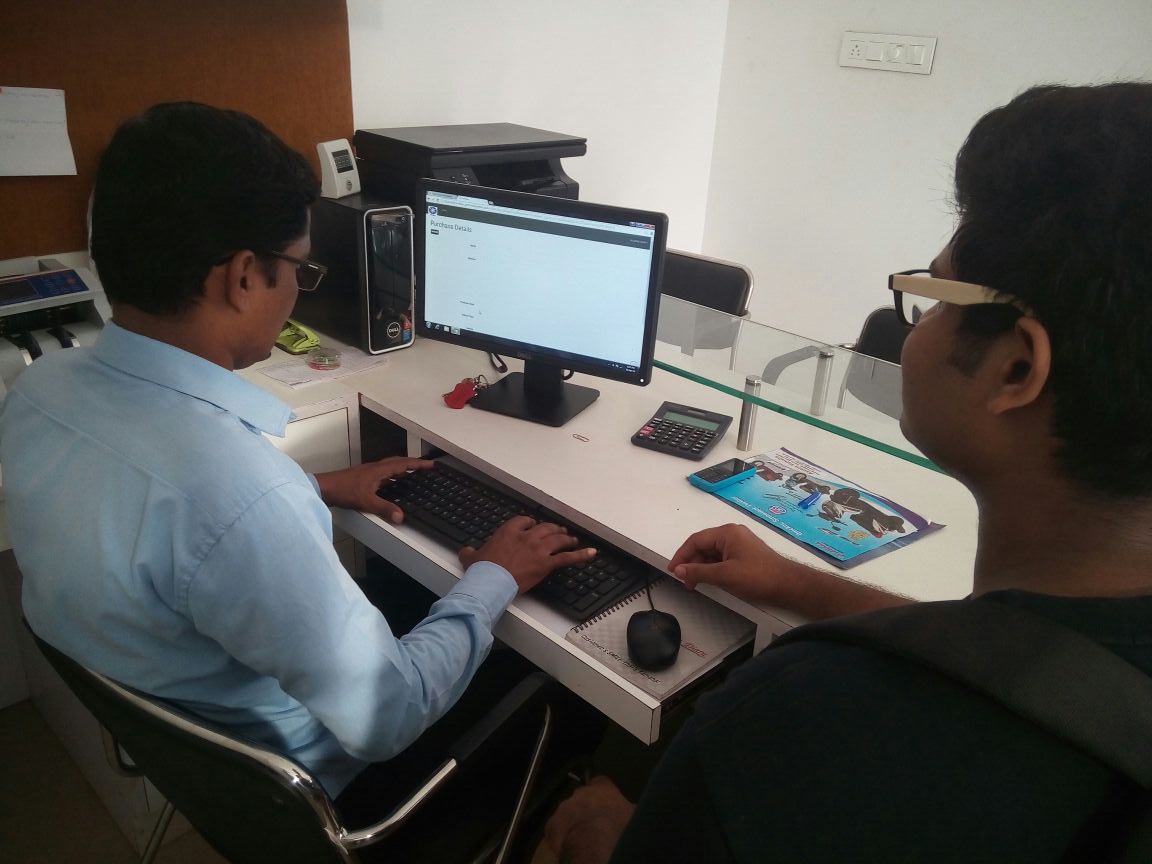


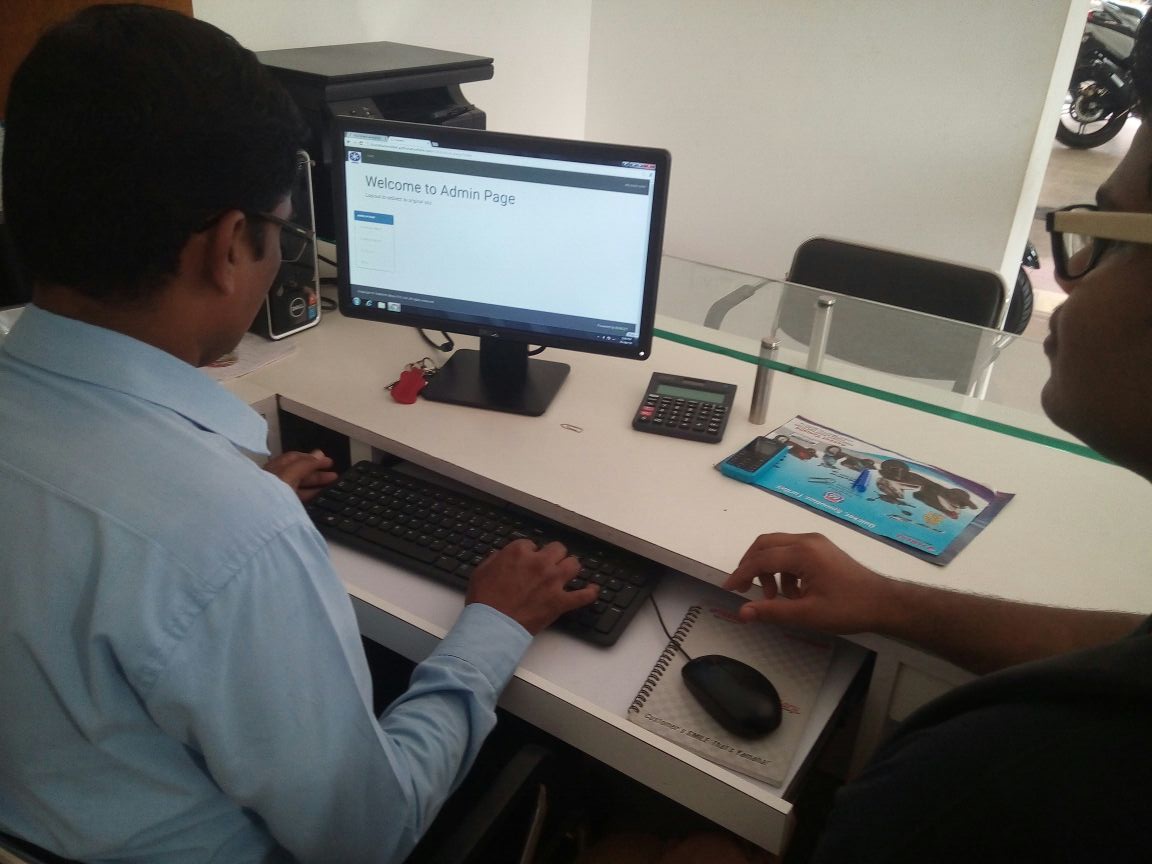
SNAPSHOTS

Outside view of showroom.



Inside view of the showroom

Entering data into purchases page

Logged in to Admin page.

INITIAL LETTER



FINAL LETTER



FEEDBACK

