PetrolHead

Presenters,

Ganesh Arumugam

Manikandan Nagarajan

Saravana

Project Description

- The project focuses on automobile industry. The primary task is to display the car models and their specifications.

- Will create a social community wherein, users can login and provide ratings for any car.

- Feedback about the car can also be given in the form of comments.

- Users can comment or rate the car models even though they are not direct owners, but could have driven that car anytime.

Problem Solution

- The solution to this problem is done by retrieving the data provided by Edmunds API.

- The complete specification of each of the car is provided by the Edmunds API.

- The social community is created, by maintaining the activities of each user in the database.

- The profile of each of the user has the recent searches, the comments or ratings given by the user, along with his favorite cars.

Edmund Api

- All the api data are picked from the below URL,

- http://developer.edmunds.com/api-documentation/vehicle/

- The team maintained a single key throughout the application for making all the api calls.

Technology Stack

1. Presentation Tier (jsp, angular.js, jquery, html, css)

2. Middle Tier (JPA, JWS, JSON Parsers)

3. Database (MySql – Tables, triggers, normalised tables, different table strategy)

Class Diagram
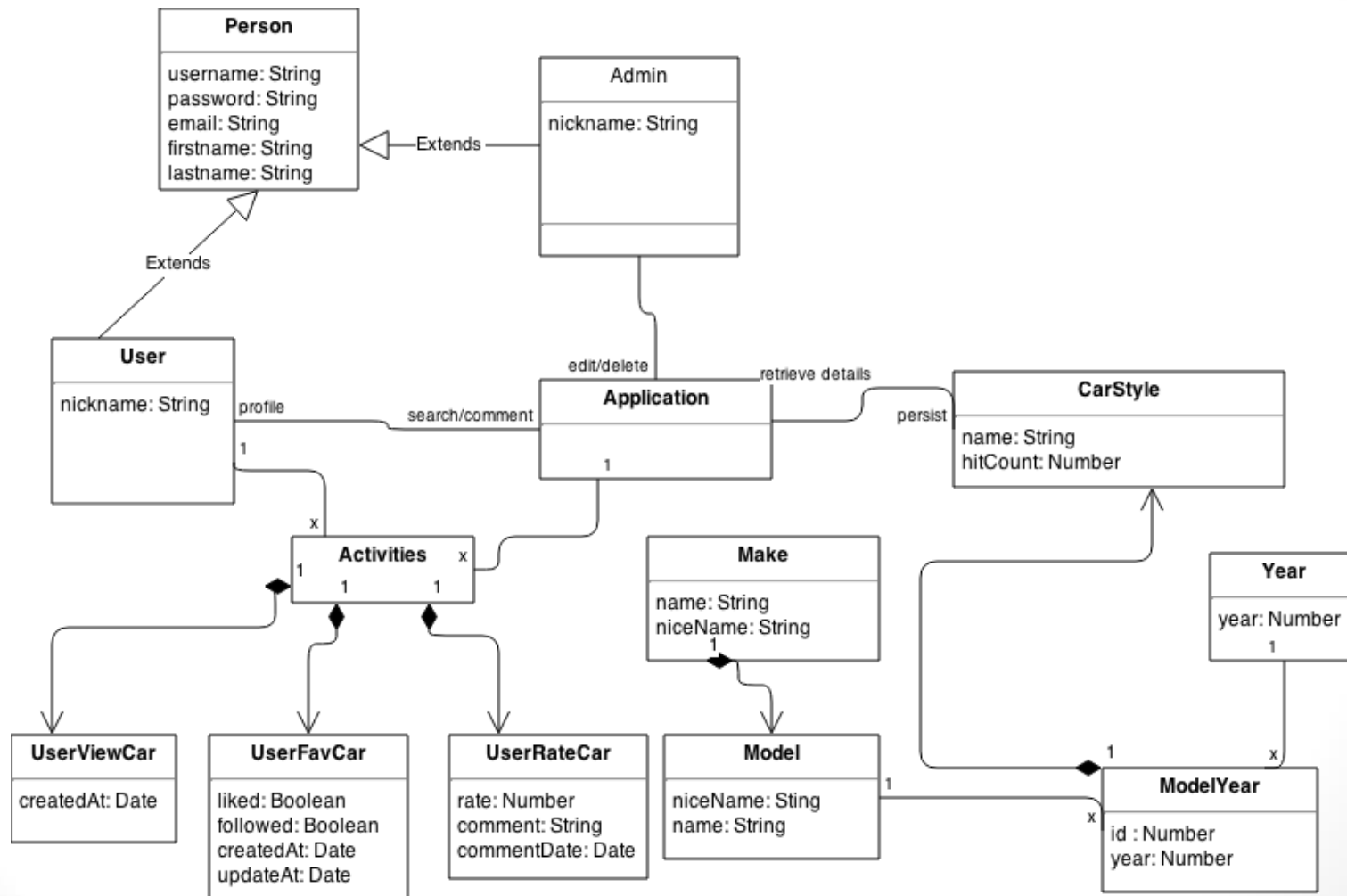
Class Diagram

- Generalization
  - The classes like user and admin are generalized from the Person class. We have used single table strategy in JPA.
- Class vs Attributes
  - Certain activities like review a car is used as a attribute instead of keeping it as a separate class.
- Composition
  - Admin can edit/delete a user and his comments. So cascade delete is required for all the user activities.
- Reification
  - Many to Many relationship that exists along the classes like model and year, user and application are broken down into association tables.

Schema

- Mapping tables

- Association tables

JPA

- We have used JPA for the representing the entity classes.

- We have used the techniques discussed in the class, like single and multiple table strategy.

- We have used a separate package to represent all the dao classes.

- Difficulty faced: Many to many table mappings along with new fields.

JWS

- We have used JWS to expose all our database services in form of GET, POST, PULL and DELETE requests.

- Our project web services produces and consumes JSON for transferring data to and from the application.

- Our web pages creates a AJAX requests to consume the web service and parse the JSON object and display it on the page.

- We insert the Edmunds Api call data as a web service into corresponding tables.
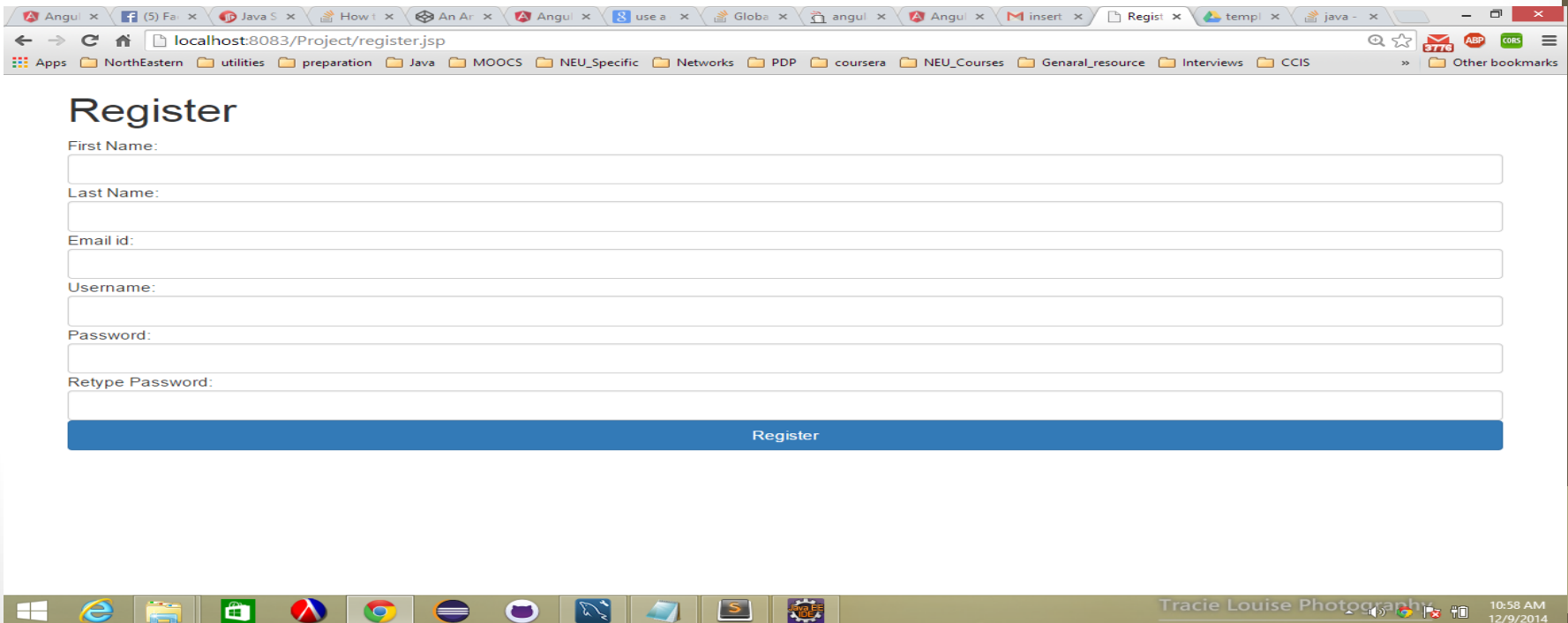
Login Page

- We have used Servlets and JSP to validate the user of the system. Our system allows only authorized users.

- Screenshot
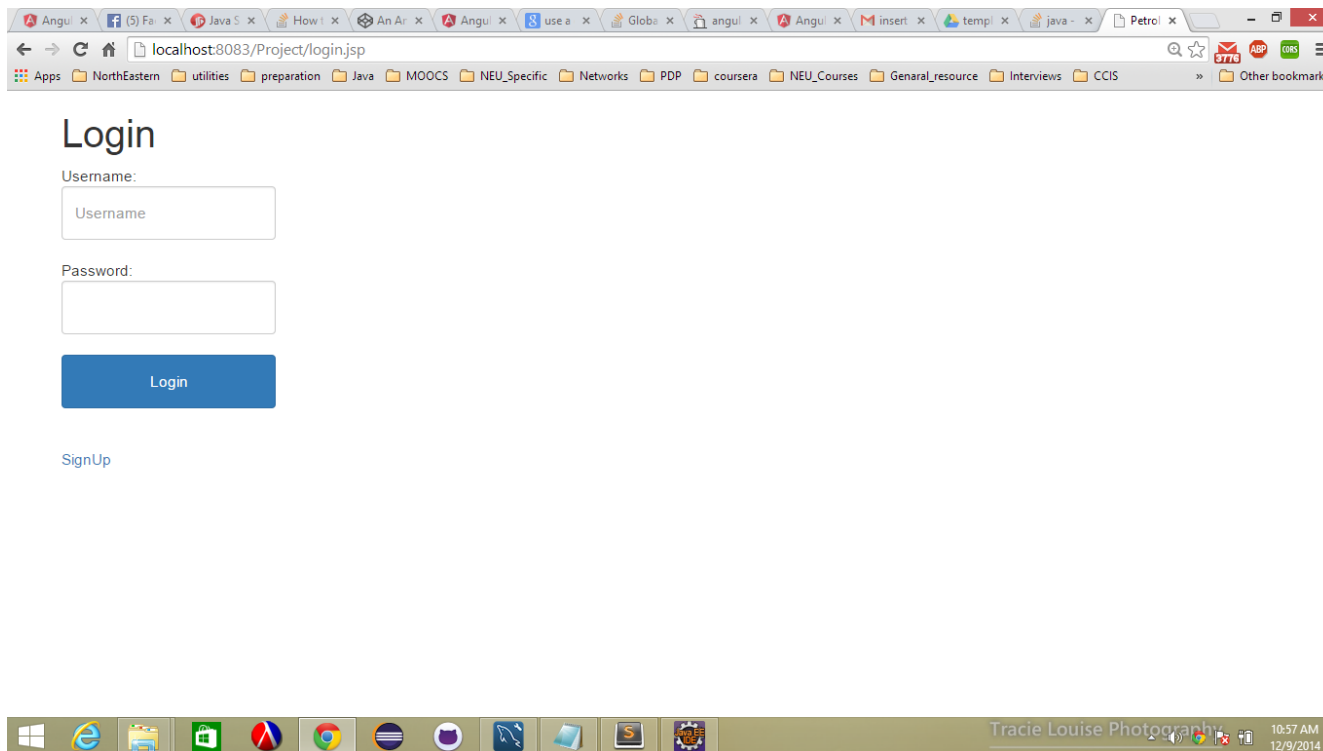
Register Page

- Makes a new user for the system and uses the same technology like login page.
- Screenshot

Home Page

- The index page of the application. It displays the top three trending cars of the application.

- Trending cars is based on the hit count stored internally based on the users activities.

- The main tab page consists of Profile tab, Search tab, About tab.

Home Page

Profile Page

- Every logged in user can view four basic functionalities like,

- Recent Search

- Favorite Cars

- Followed Cars

Profile Page
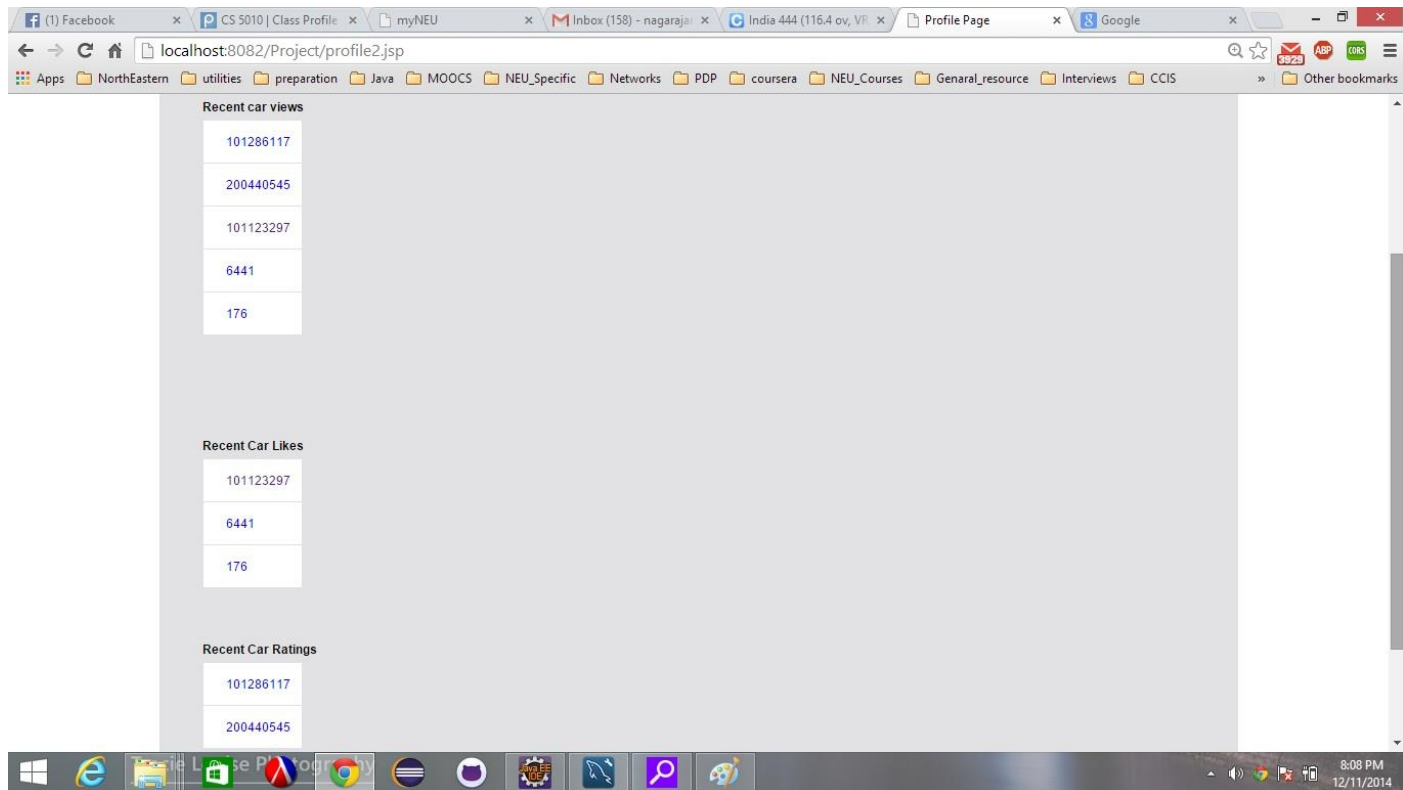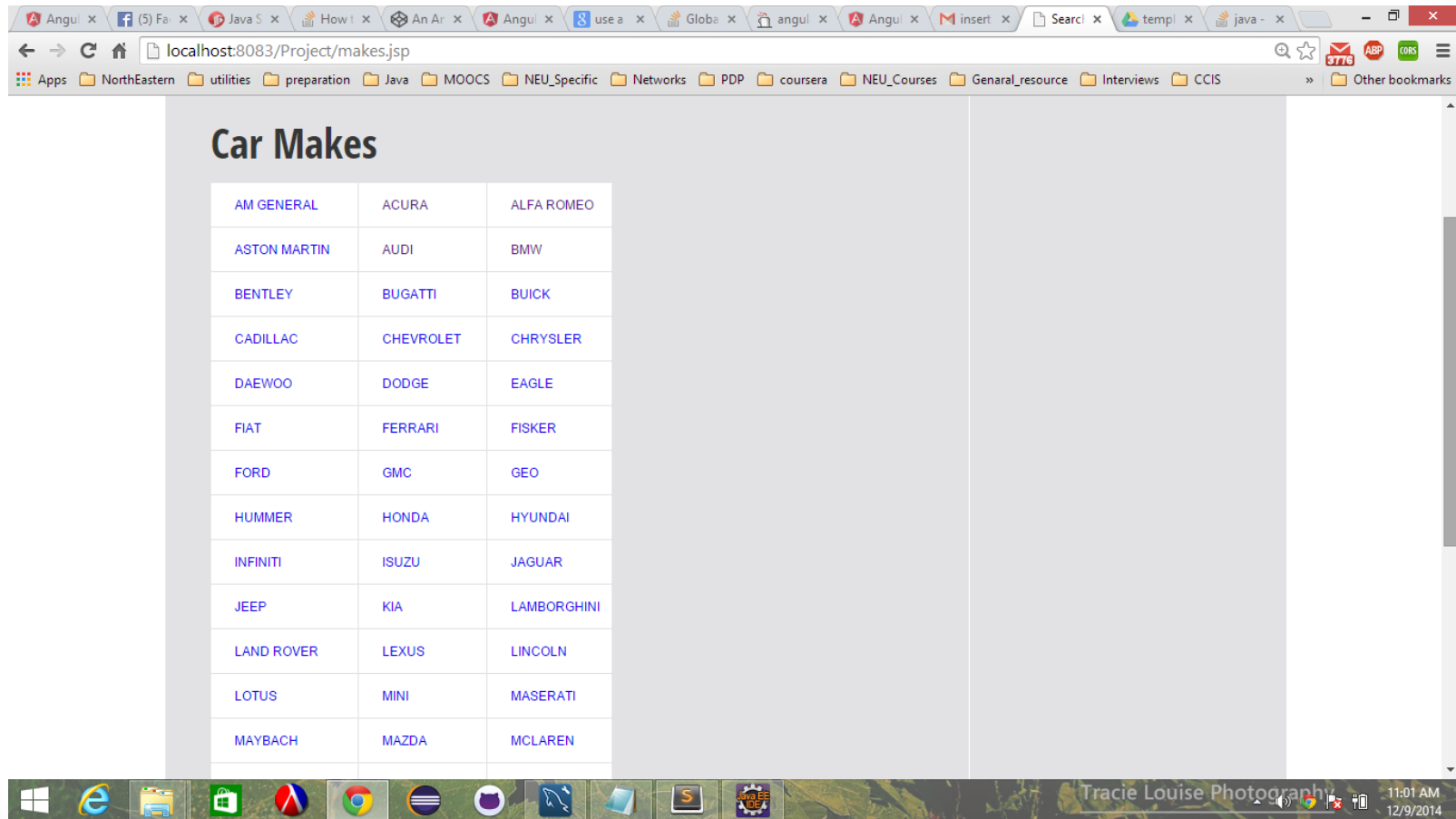
Search Page

- We provide the search functionality based on the Car makes.

- The user can select anyone of the Car make and then can select the model and the year.

- Based on the year, we display the details of the car on the details page.

Search Page - Make

Search Page - Model

Details Page - Year

Car Details

- The finer details of the car are presented in this page.

- We provide a whole list of functionalities to the user like,

- Like and follow a car.

- Review and post comments.

- All these user based activities are stored in the database by making use of JWS service calls.

Details Page

Like Car

- We maintain all the records of a user liking the car from the application.

- We also maintain the other side of this relationship as well, like how each car is liked by various users.

- We maintain this relationship in the form of 'Like' button.

- The user can 'update' his action by unliking the car as well.

Like Car

Unlike Car

Follow Car

- It is implemented on the same lines of 'Like' functionality.

- Only difference is that, we need to update the user about any activity performed on that particular car.

- These updates are in the form of comments given by other users for the same car.

- User can update his action by 'unfollow' button feature.

Follow Car

Unfollow Car

Review and Rating

- Every authorized can provide a valid rating and review for the car.

- The user need not follow or like that particular car to comment on it.

- User can make use of the search functionality to access all the cars present in the application.

Review Car

Edit Comment

Delete Comment
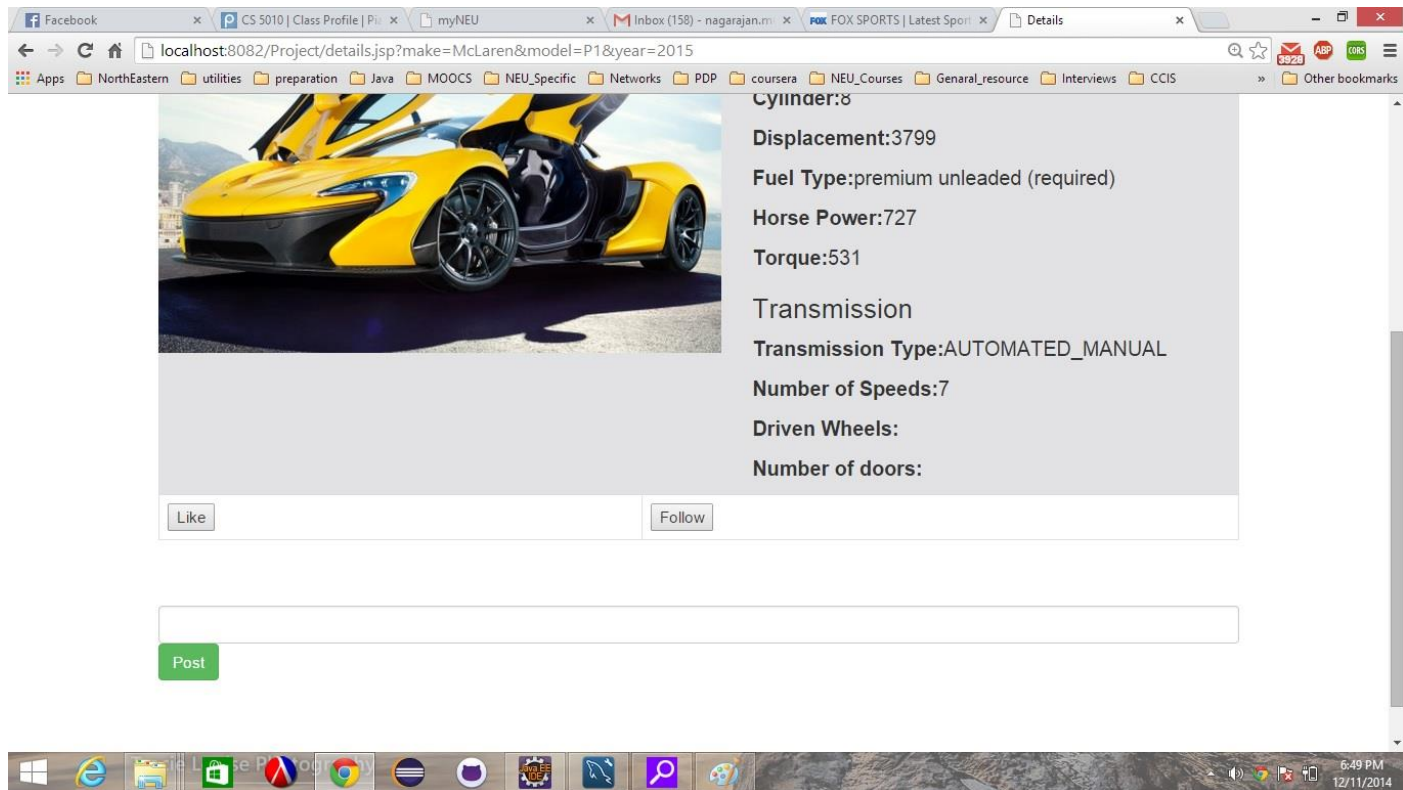
Admin Page

- Admin users are created through the back door.

- Main functionality is to provide a filter mechanism for weeding out the dummy users.
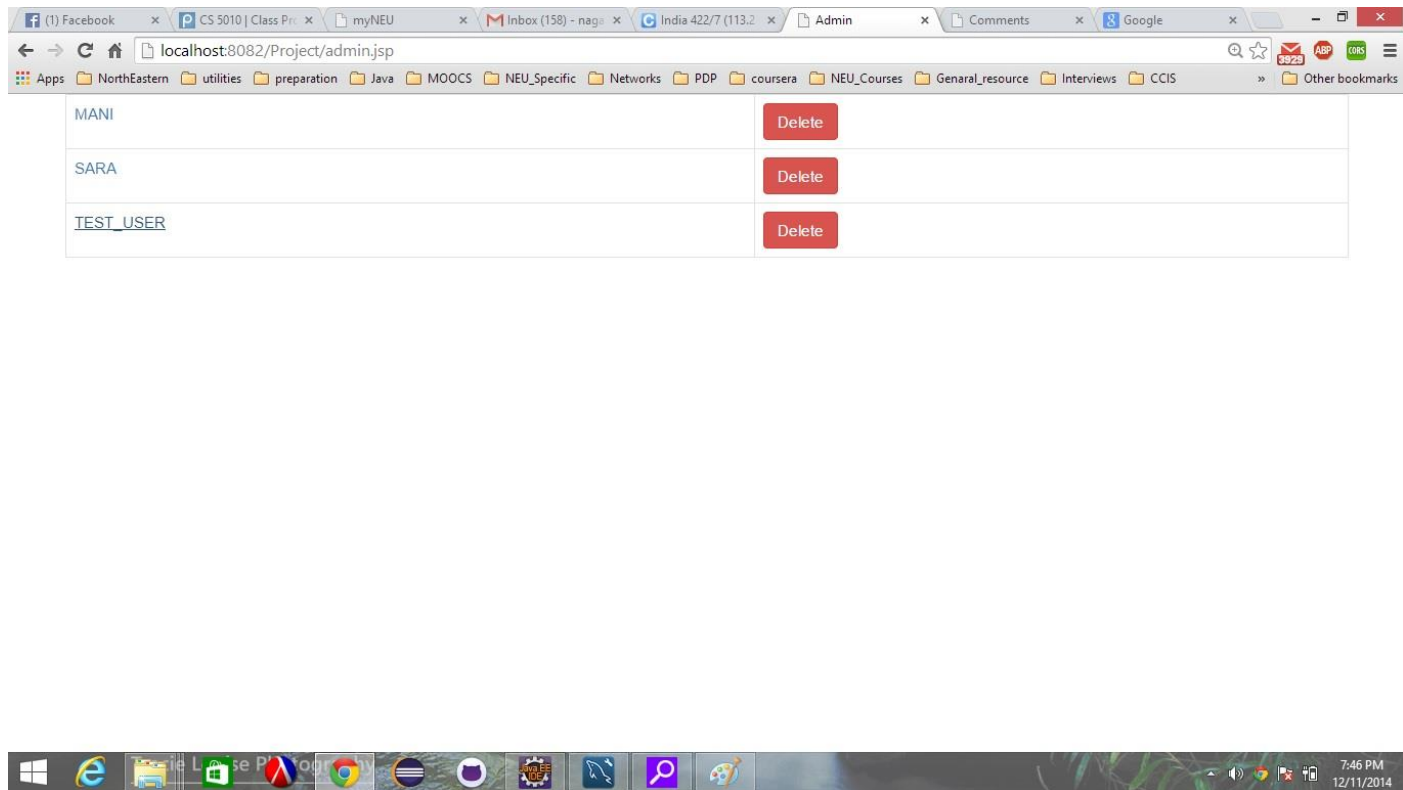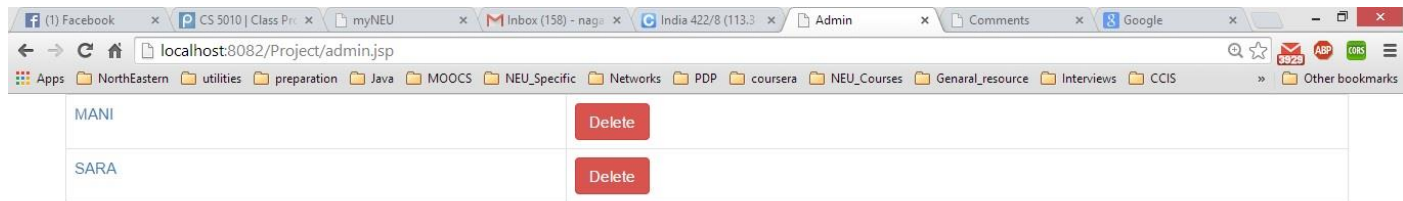
- Some inappropriate comments need to be removed as well with the help of super user like admin.

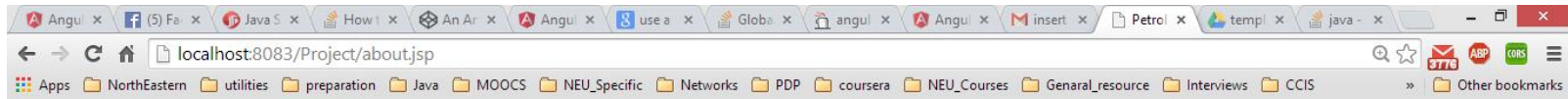- We have created a trigger to store the admin log into a new table.

Admin Page

Delete User

Delete Comments

About Us

TODO

- We need to complete the hit list count in the home page. We have developed the back end JWS services.

- A nice feature to have is a community based on the price tag/range. Edmund api provide price information.

References

- Piazza Resources

- MySql resources

- http://developer.edmunds.com/api-documentation/vehicle/#sec-5

-