**Project 3 Web App**

Daniel Arrick

Advanced Programming Techniques

**Introduction**

Clients can login or create an account from the home page. Ideally employees would be able to login and modify service requests assigned to them. When a client logs in, they have access to their profile page which shows basic information, active or complete service requests and the ability to make new requests. Ideally, employees would login to modify those requests. Data is stored and pulled from a remote SQL database.

**Challenges**

Unforeseen problems took plenty of time to solve and limited my ability to implement other functions. Setting up the remote sql database and incorporating javascript buttons and popups were some of those time consuming issues. The employee class, although planned for, is not fully implemented. I also would have liked to polish up organization and separated more code into methods and classes in certain circumstances and develop a better UI. I also would have liked to have implemented more control over user entered data.

**Disclosure**

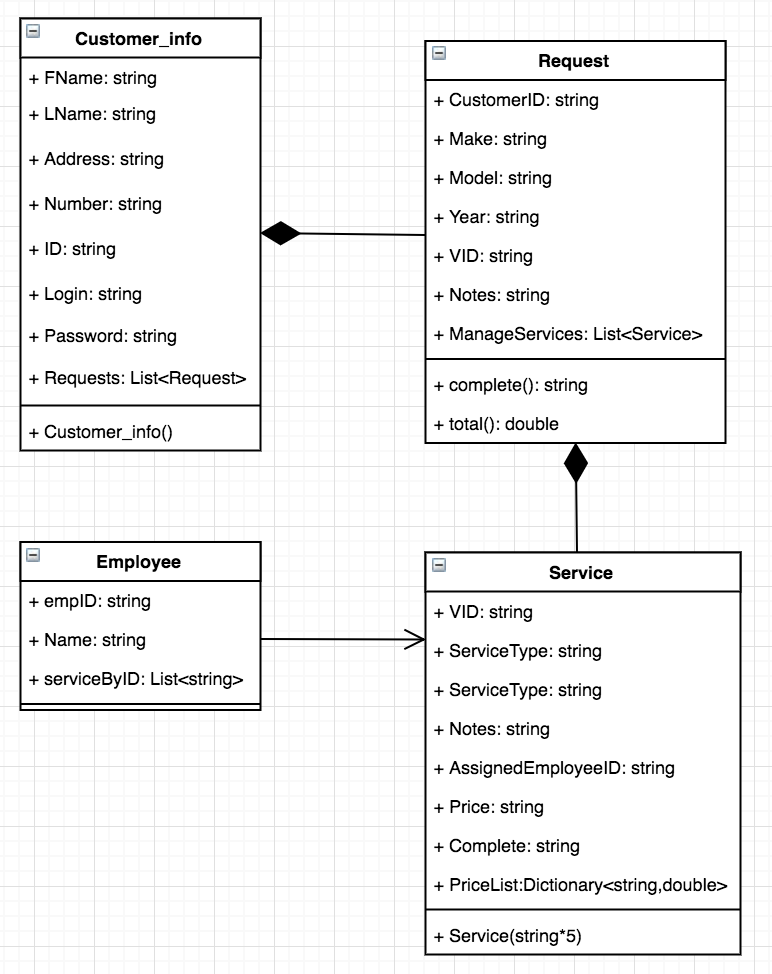
Code from stack overflow was used to establish the database connection with slight modification. Additionally css style code was used and slightly modified from w3schools for Request popup windows.

**Database**

Login at: http://www.phpmyadmin.co/index.php

Host: sql9.freesqldatabase.com  
Database name: sql9235287  
Database user: sql9235287  
Database password: kL4qWu22zx  
Port number: 3306

**UML of Entities**



**Controllers**

*Home*

The home controller simply returns the home page view

*Login*

The login controller will generate a new customer to be logged in based on valid username and password provided. The controller will make method calls to CustomerLoginQuerys model in order to pull information from the database.

*Registration*

The registration controller creates a new customer to be registered and pushes that information into the database along with a unique ID; info received from RegisterForm view. Ideally the queries would be moved to their own class (in the beta to come).

*Request*

The request controller receives service information from a form in the LoggedIn view (user profile view). Bool and string data from a form is processed to determine what a client has requested and those services requested are added to a Request object and pushed to the database using queries in model RequestQuerys.

**Views**

*Index*

The index view hosts the login page, client registration link and employee login link (non functional at the moment). Login requests are sent to the controller.

*RegisterForm*

The register form provides mandatory fields for client registration. Form data is sent to the Registration controller.

*LoggedIn*

This view displays a clients profile page once logged in. Basic information is displayed as well as active and completed requests. Requests can be clicked into to see an itemized list of services corresponding to a particular vehicle request, the mechanic assigned, and other helpful information. A client can also make a new request here by completing a form which is sent to the Request controller.

*EmployeeLoggedIn*

To host employee profile with active services

**Models**

*Customer\_info*

This is the customer model which contains basic information and a List in which clients’ requests are maintained.

*Employee*

The employee model contains some basic information about the employee and a List of services assigned be vehicle ID.

*Request*

The request model stores information on a particular vehicle and a List of services added to the request. The Request contains a completion check for all services within it.

*Service*

The service model stores information about the particular job a mechanic will perform including type of service, price, assigned mechanic and completion check. The Service model also contains a dictionary of corresponding prices to particular service.