SRS for Reversed Car Bay

Reversed Car Bay
Version 1.3 2014-04-09 (final version)
Prepared by Johan Angelstam, Peter Keogh, Fredrik Salin
Harbin Institute of Technology

Revision History

Date	Reasons for changes	Version
2014-04-01	Initial version	1.0
2014-04-03	Added database model and security requirements	1.1
2014-04-07	Updated class diagrams	1.2
2014-04-09	Final version	1.3

Introduction

Purpose

The purpose of this document is to give a detailed description of the requirements for the “Reverse Car Bay” software system. It will explain system constraints, technologies and external services that will be utilized in its development.

Scope

The “Reverse Car Bay” software system is a service, which allows a customer interested in buying a car to browse and customize different types of cars online. The customer can then put a price enquiry for the car he or she wants.

Car-dealers can browse the requests from the customers and bid on these. The lowest bid is collected by the system and presented to the customer. When the customer finds a suitable deal he or she pays 10% of the price to secure the deal.

The software will not:

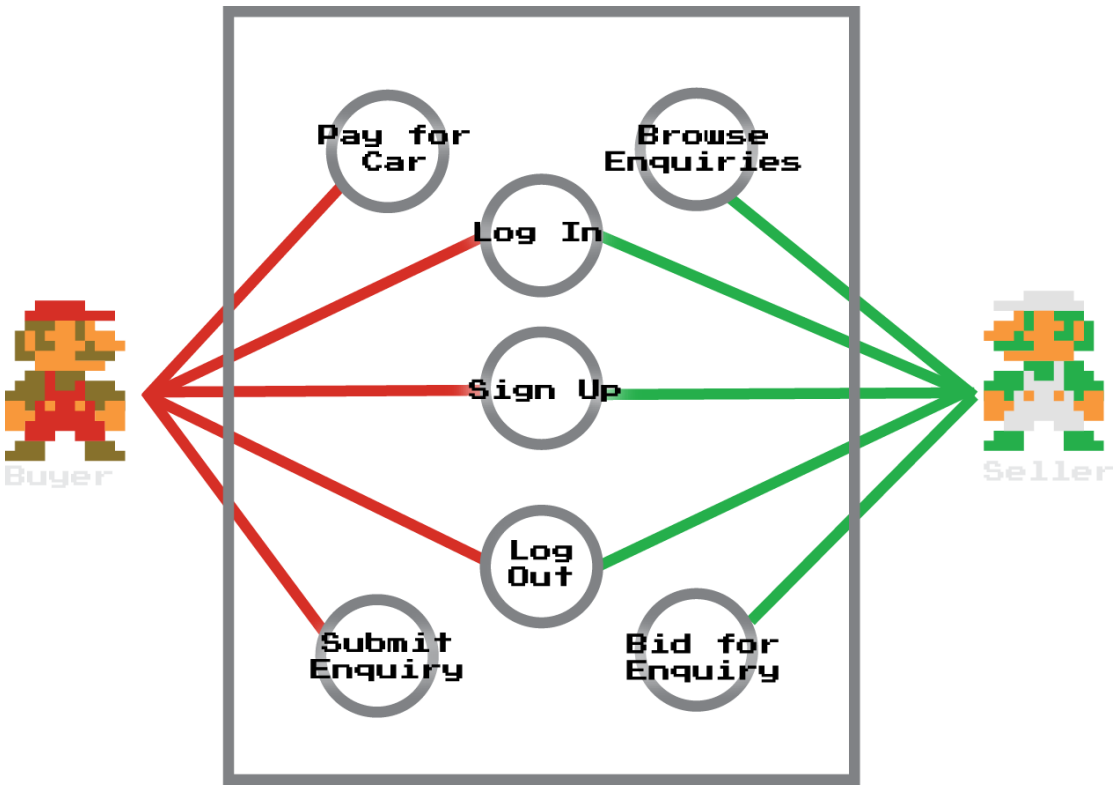
- Allow Car-dealers to offer similar cars to the customers, only bid on exact enquiries from customers.
- Allow car-dealers and customers to gain contact information to each other before the deal is secured.

Overall Description

This chapter presents the main structure and function of the system.

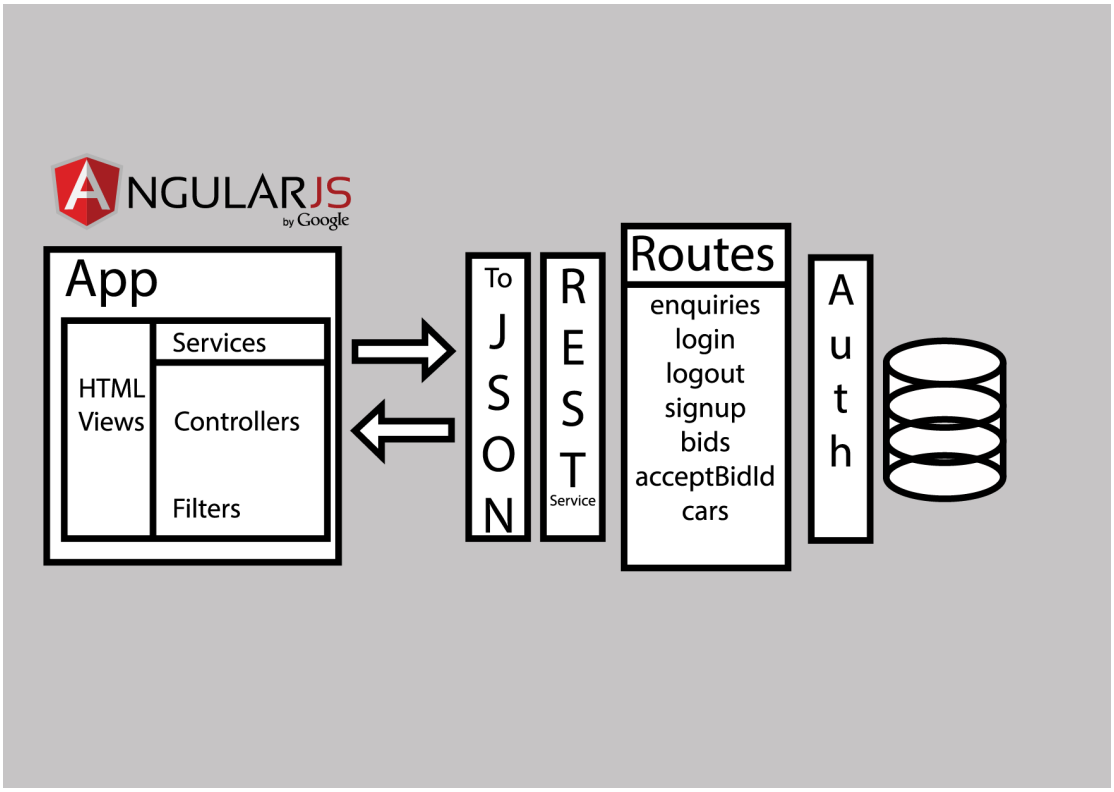
Use Case Diagram

The use case diagram below shows the functionalities and user types of the system.



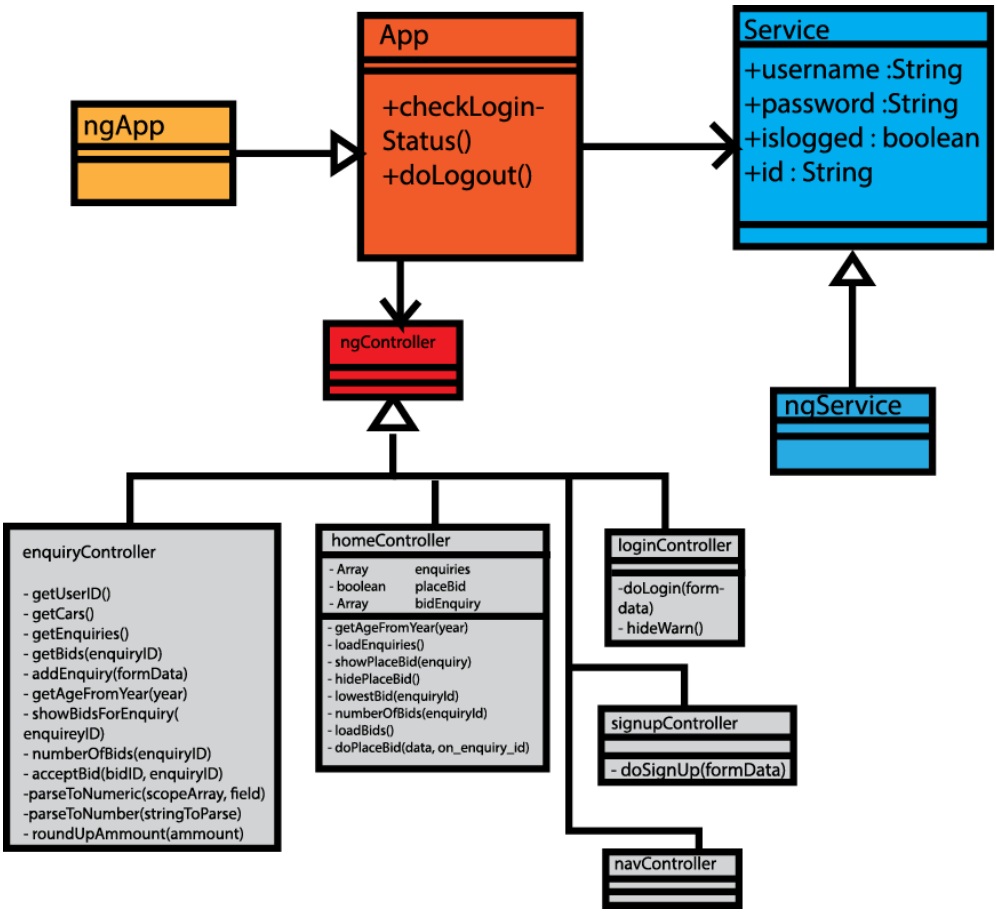
Program Architecture

The program architecture will take advantage of the model view controller design pattern. The model is implemented on the server. It simply responds with JSON formatted data to requests that are authorised. The RESTful API provided by this architecture allows the application data to be reused in other applications. The View and Controller elements of the system are implemented completely on the client side.



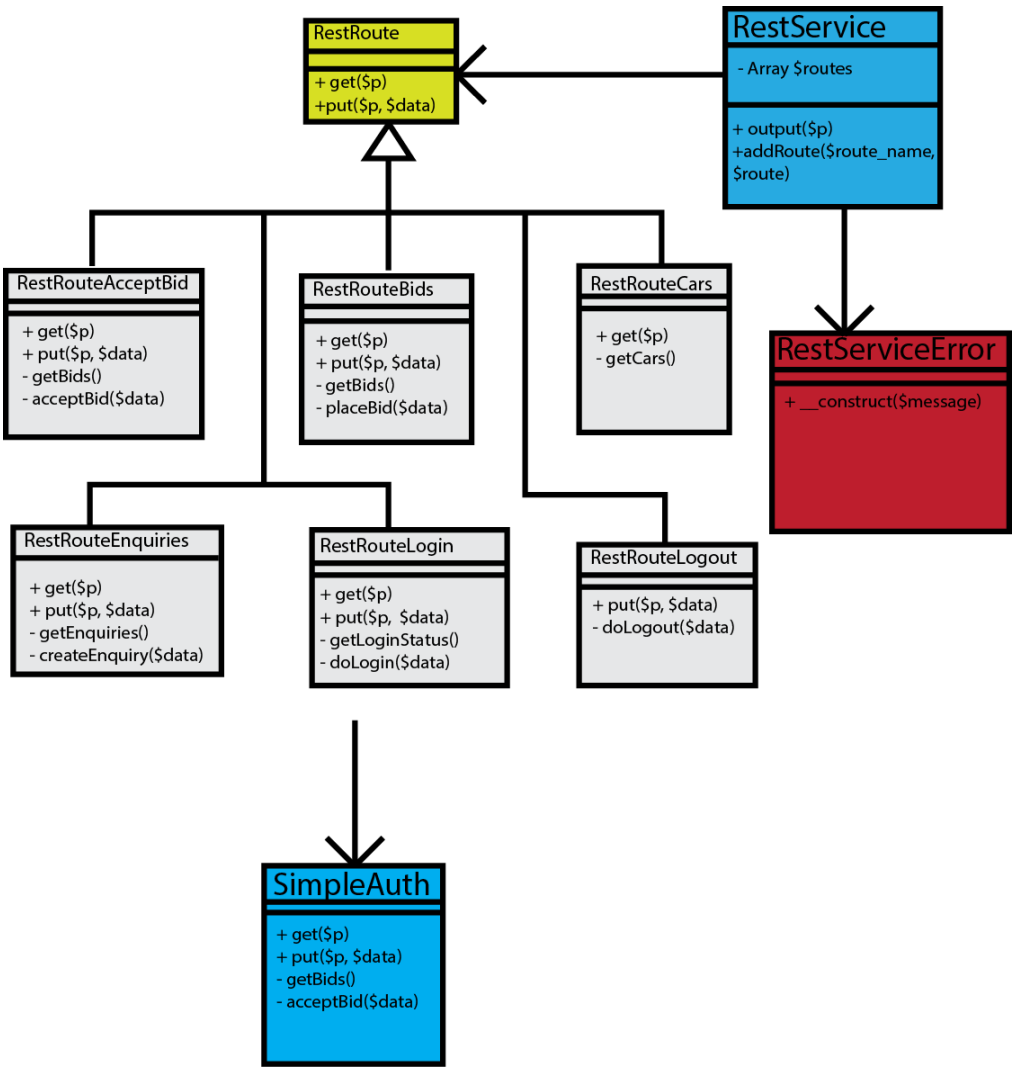
Client side architecture

The client side architecture utilizes the Singleton design pattern in the form of the App object. This object is responsible for coordinating the modules on the client side of the system. Each module on the client side has a single responsibility. The modular nature of these components allow us to reuse them in other projects, or reuse modules we already have available to us.



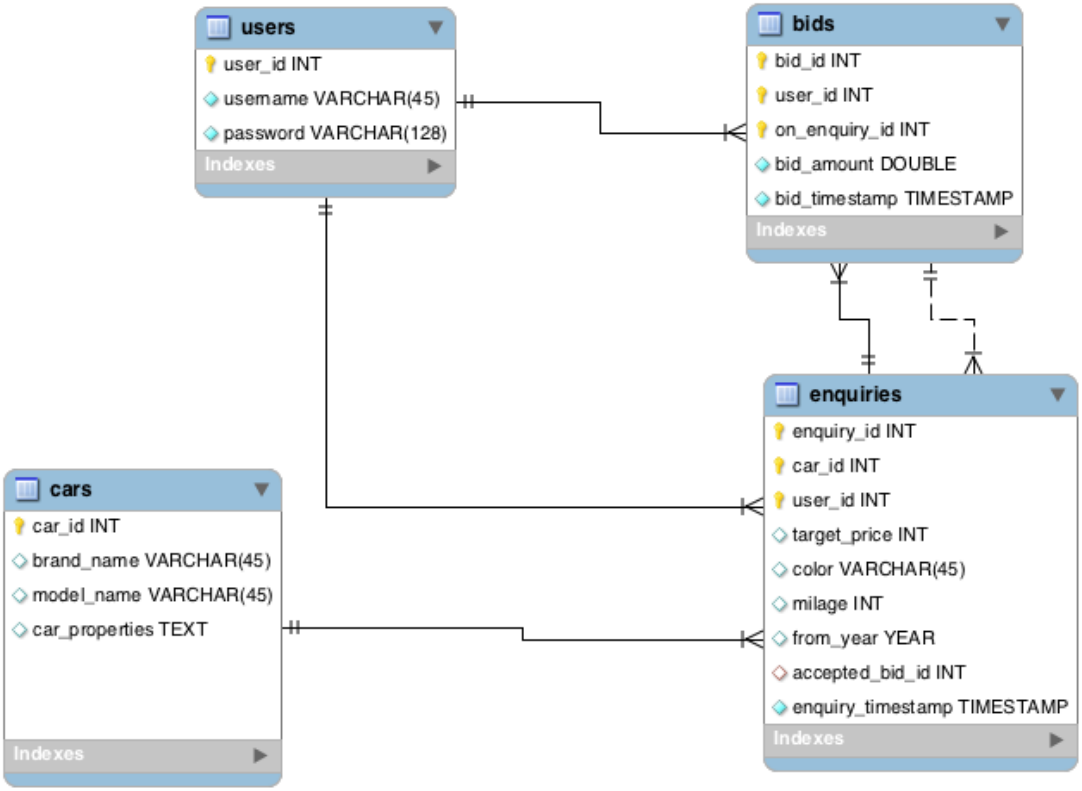
Server side architecture

The server side architecture is divided in several modules with different specific tasks. The RestRouteLogin for instance handles the login and signup function for the system, and the RestRouteEnquiries handles the creation of new enquiries as well as getting existing enquiries from the database.



Database model

This figure shows our database model with tables and relations.



Product Functions

- Allow a customer to browse and customize cars online
    - Customers should be able to choose car from predefined list
    - Customers should be able to choose to modify car with predefined alterations
    - Car buyers should be able to update a created specification
- Allow a customer to make a price enquire for his or her customized car
    - Customers should be able to set a asking price for the enquiry
- Allow car-dealers to bid on the enquiry added by customers
    - Car-dealers should be able to get the car specifications made by customers
    - Collect the cheapest deal to notify the customer (+5% broker fee)
    - Show the best offer made on the page (incl 5% broker fee)
- If the customer accept the deal, he or she should pay 10% of the price to secure the deal
    - Customers should be able to accept offers from car salesmen
    - Customers should be able to pay 10% of the price to the car-dealer

User Characteristics

Two distinct groups define our users; people who want to find the best deal on their new car and car-dealers.

People of all ages and backgrounds will use the system. It is important to design the system keeping these accessibility requirements in mind. By using the CSS Foundation framework the software can be easily parsed by screen readers and other software to help those with usability issues. These issues are also addressed by keeping the user interface design simple.

Constraints, Assumptions and Dependencies

For the development of the system it is assumed that the car dealers are only allowed to make offers on the exact details provided by the customer. The car dealer may not come with counter offers including similar or alternative offers to the customers enquiries.

The time available for the development process of the system is less than two weeks and this might be a limitation to which what features will be implemented in the system. One obvious foreseeable disadvantage of the application architecture is that it doesn't notify the user with information that is relevant to them. This could be solved in two ways. One would be to utilize emerging Web Socket technology to send push notifications to the browser. The other would be to port the client side Javascript application to Android and iOS using Phonegap. This would allow us to take advantage of those platforms push notifications. Due to the time constraint on this project these updates will be left unimplemented until a later version of the system.

Specific Requirements

External interfaces

Customer interface

Inputs:

- Car brand
- Customer id
- Car model
- Car modifications: Color, Milage, Age
- Acceptance of lowest offer from car dealers

Outputs:

- Lowest bid from dealers

Car-dealer interface

Inputs:

- Bids on a specific car specification enquiry
- Car dealer id

Outputs:

- Car specifications created by customers
- Accepted offers from car dealers

System functions

Customer side:

- Check login details/sign up new user
- Send login/register details to server
- Display error message/redirect user
- Create enquiries
- Post enquiries to server
- Accept bids from car dealer
- Retrieve list of bids for enquiries from the server
- Present best offer for a specific enquiry

Car dealer side:

- Check login details/sign up new user
- Send login/register details to server
- Display error message/redirect user
- Present a list of enquiries from customers
- Post bids to the server

Server side:

- Store posts to DB (from car dealers and customers)
- On get-request present information to the users

Non-functional requirements

The system shall be implemented using the following technologies.

- The backend will be written in PHP because all of the developers have knowledge about this language.
- The system will be using a MySQL database.
- The frontend will be using the Javascript framework AngularJS.

Security requirements

The system shall implement the following security requirements to minimize the risk of compromising the system and thereby the risk of private user data falling into the wrong hands.

- The passwords stored in the database shall be hashed using bcrypt.
- The web server shall use a SSL connection to prevent eavesdropping.