

**CS673F14 Software Engineering**  
**Group Project - UPark**  
**Software Design Document**

Your project Logo  
here if any

<u>Team Member</u>	<u>Role(s)</u>	<u>Signature</u>	<u>Date</u>
Fanghui Zhang	Team Leader		<u>09/12/14</u>
Xin Shan	Configuration Leader		<u>09/12/14</u>
Lu Zhang	Environment and Integration Leader		<u>09/12/14</u>
Rui Li	Requirement Leader		<u>09/12/14</u>
Xi Tang	Design Leader		<u>09/12/14</u>
Yilun Xie	Implementation Leader		<u>09/12/14</u>
Feiyu Shi	QA Leader		<u>09/12/14</u>
Mingli Li	Architect	<i>Mingli Li</i>	<u>09/12/14</u>

**Revision history**

<u>Version</u>	<u>Author</u>	<u>Date</u>	<u>Change</u>
<b><u>1.0</u></b>	<b>Fanghui Zhang</b>	<b><u>11/13/2014</u></b>	

[Introduction](#)[Software Architecture](#)[Design Patterns](#)[Key Algorithms](#)[Classes and Methods](#)[REST API](#)[test](#)[Glossary](#)

## 1. Introduction

This software design documentation describe the detailed structure of the components of the software and the implementation to satisfy the requirement as specified in the SPPP. It assumed that the reader has read the SPPP.

The design description defined in this document serves purposes below:

- To describe the functional structure, data and algorithms to be implemented
- To help to produce test cases
- To be used to verify compliance with requirement
- To help for further developing (If needed)

As is stressed in the SPPP, the goal of our software system is to build a online parking marketplace. We connect renters with drivers through our platform and provide them services they need to finish the rent activity.

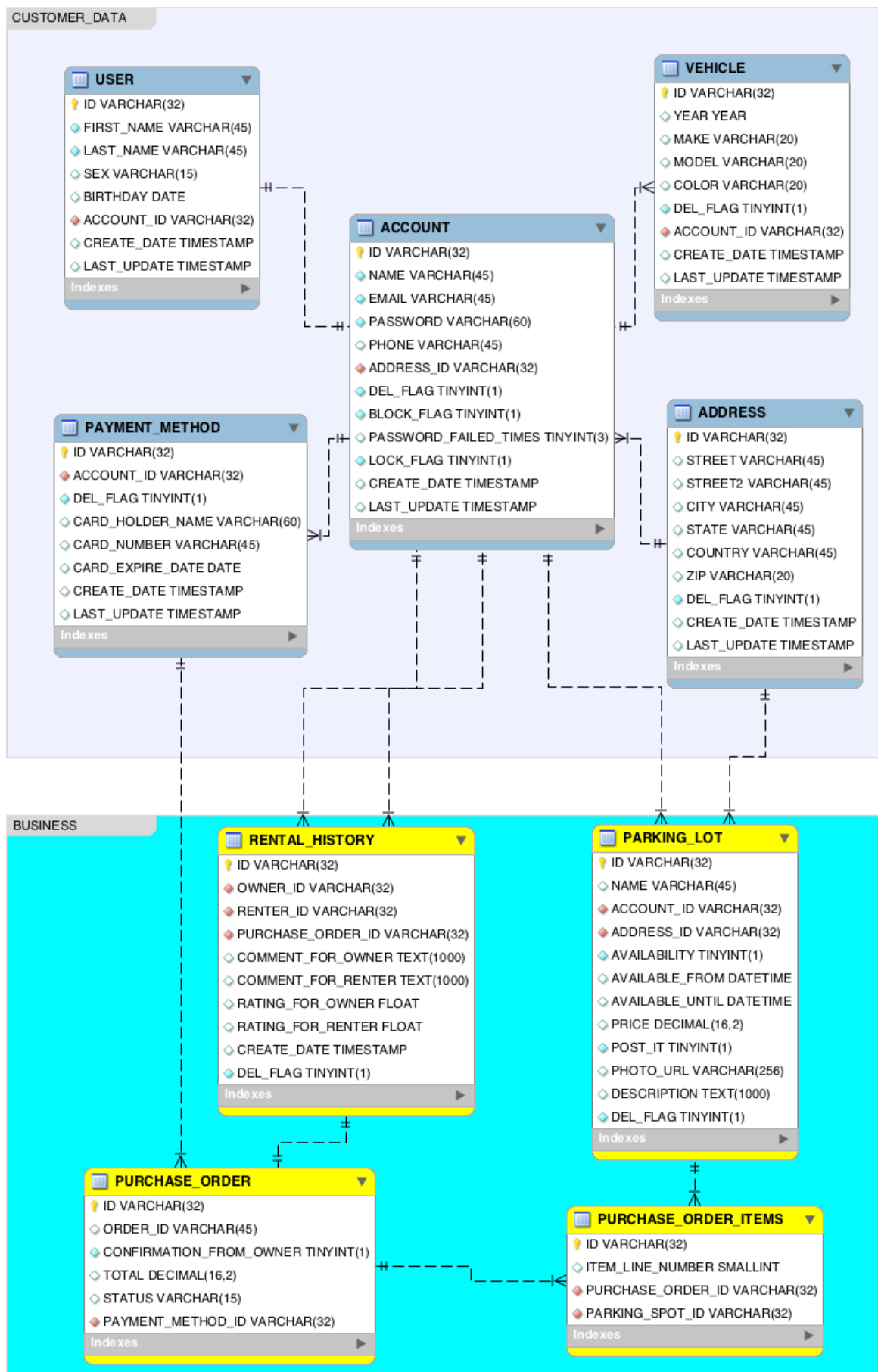
## 2. Software Architecture

In this section, you will describe the decomposition of your software system, which include each component (which may be in terms of package or folder) and the relationship between components. You shall have a diagram to show the whole architecture, and class diagram for each component. The interface of each component and dependency between components should also be described. If any framework is used, it shall be defined here too. Database design should also be described if used.

### 2.1 Database Design

#### 2.1.1 Database Schema

MySQL is the chosen database for this project, for it is open source and capable of dealing with large amount of transaction data of the website. Below is the database schema for the project. It not only describes the entity structures but also shows the relationships between them. This schema only depicts the client-side of the system; administration-side system may be designed later as the system extends. The content of the schema might also be changed due to possible requirement adjustment.



### 2.1.2 Details of Tables

Tables are roughly divided into two groups: customer data and business data.

Customer data includes user and account information. Payment, address, and vehicle are also stored. Business data contains parking lot information, rental history, orders and order items. These are essential in transactions.

All tables have an 'id' as its primary key. It's a 32 bit varchar, storing a 32 bit UUID (universally unique identifier). This ensures the uniqueness of every entry in the database.

Some of tables have an 'del\_flag', which is a boolean type data. It is true when the entry is "deleted" by the user or administrator. The entry is not actually removed from the table; this prevents problems when the data is referenced by another table but is deleted in the current table.

We will outline each table below:

User table:

Primary information of the user, including names, sex, birthday, etc. It also contains the id of the accounts belonging to the user. Statistics like creation and last update date are also recorded.

Account table:

An active image of the user on the website. It includes primary security information, for example, email, password, phone, and some flags for reset account. User will use these information to login the website and involve in transactions with other users.

Vehicle table:

Basic vehicle information associated with the account. Information are like year, make, model, color, etc.

Payment method table:

Credit card information associated with the account. This type of payment method may not be used as it might be up-front cash transaction.

Address table:

Address information associated with the account or parking lot.

Parking lot table:

Information for available parking lots for rent. Information like available period, price, description are included. 'post it' flag is for search display use.

Rental history table:

It records the history of every transactions between users. It also includes ratings and comments for the owner of parking lots and renters. Each history entry contains one order id linked to an order.

Purchase order table:

This contains some information for the order, for example, subtotal, status, payment method, etc. It contains several order items (id).

Purchase order item table:

This entry includes a parking lot id, and its associated order id. It is 'one item' in an order.

### 2.1.3 Controllers

Controllers provide access to the application behavior that you typically define through a service interface. Controllers interpret user input and transform it into a model that is represented to the user by the view. Spring implements a controller in a very abstract way, which enables you to create a wide variety of controllers.

Additionally, the `@Controller` annotation indicates that a particular class serves the role of a *controller*.

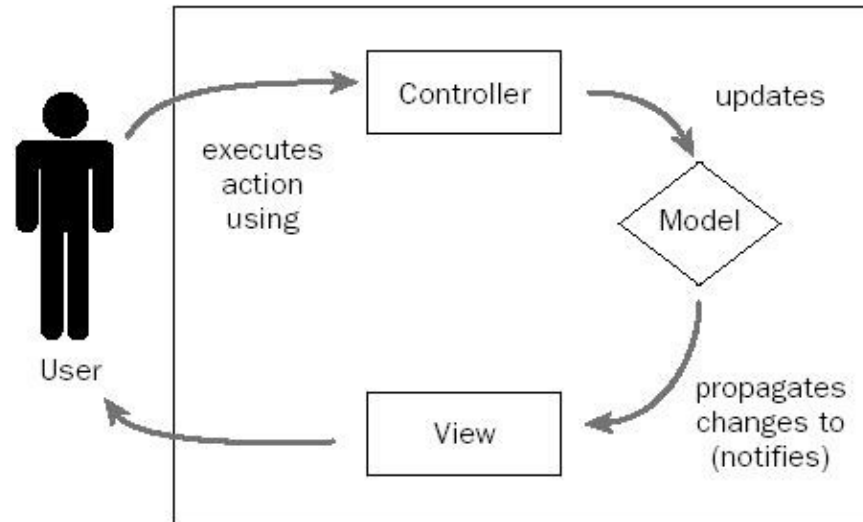
```
LoginController
    ---LoginService()
RegisterController
    ---RegisterService()
LogoutController
    ---LogoutService()
SearchController
    ---SearchServices()
```

## 3. Design Patterns

- MVC

We use model-view-controller(MVC) to isolate business logic from the user interface. Using MVC, the model represents the information(data) of the application and the business rules used to manipulate the data, the view corresponds to elements the user interface as we will show on the demo, and the controller manages details involving the communication between the model and view. The controller handles user actions and pipes them into the model or view as required.

Additionally, we use spring web MVC framework, the request processing workflow of the Spring Web MVC is illustrated in the following diagram.



#### 4. Key Algorithms

In this section, you shall describe any key algorithms used in your software system, either in terms of pseudocode or flowchart.

In this iteration, we are going to implement the basic search algorithm, we call this “stupid algorithm”, we will test the performance to see if we can improve.

Figure 1 shows how search function goes in every step. Figure 2 shows the diagram of the “stupid algorithm”, basically, the parameter will be decided by mathematical calculation. That is, the conversion for kilometers and latitude is about  $111\text{km}/1^\circ$ , and for kilometers and longitude is about  $111\cos\alpha\text{ km}/1^\circ$ .

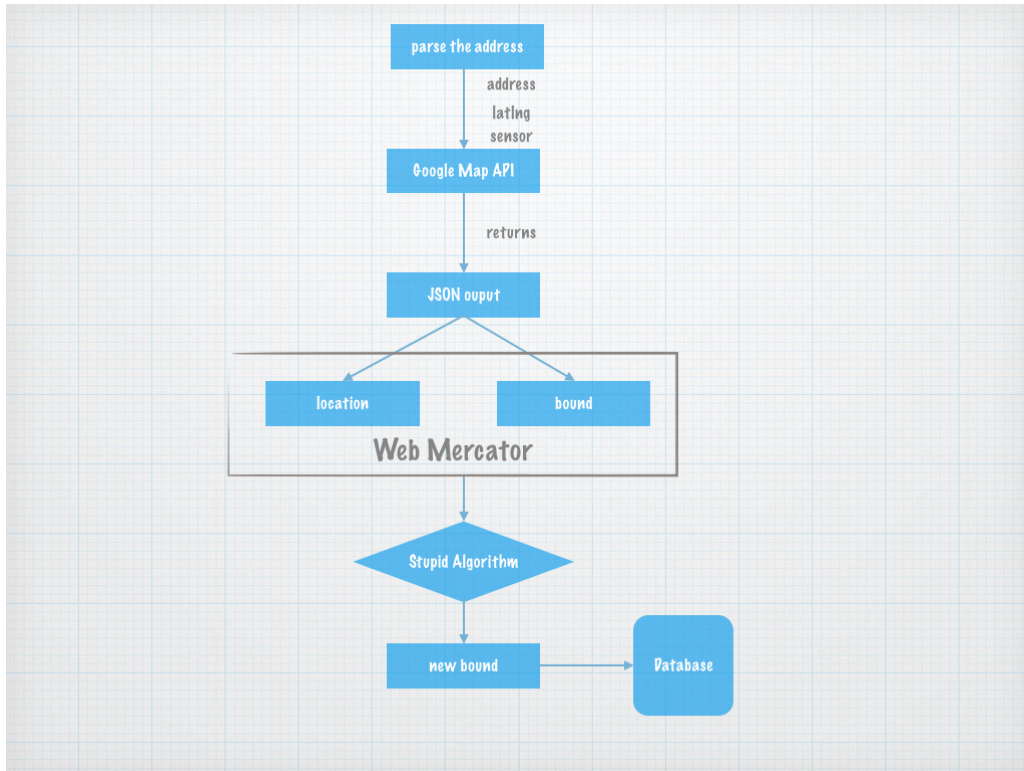


Figure 1

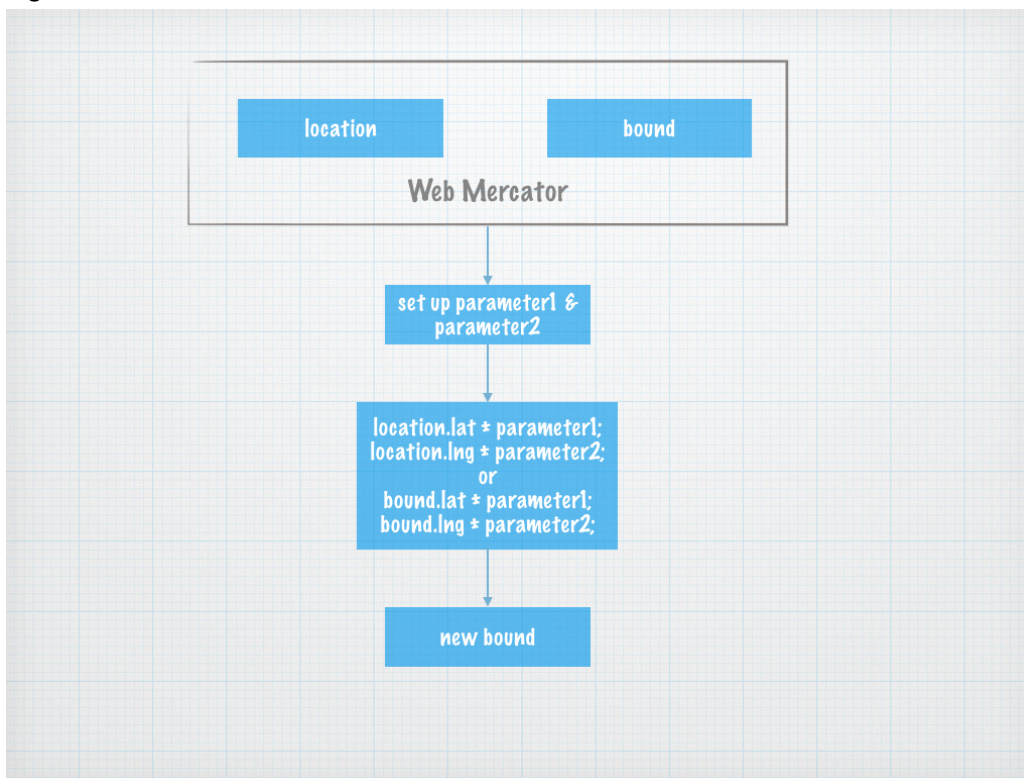


Figure 2 "Stupid Algorithm" for searching

## 5. Classes and Methods

This part can be a reference to automatic generated document for all classes and methods.

## 6. RESTful API

GET /user

Description:
This method will return information about all users in the database.
Input:
None
Returns:
<pre> &lt;data&gt;   &lt;batch_program_data&gt;     &lt;batch_program batch_name="HealthCheck"       platform_type="VNX"       program_name="Rockies"&gt;       &lt;identities&gt;         &lt;identity name="batch_program_id"           value="1" /&gt;         &lt;identity name="batch_info_id"           value="1" /&gt;       &lt;/identities&gt;     &lt;/batch_program&gt;     &lt;batch_program batch_name="Upgrade"       platform_type="VNX"       program_name="Rockies"&gt;       &lt;identities&gt;         &lt;identity name="batch_program_id"           value="1" /&gt;         &lt;identity name="batch_info_id"           value="2" /&gt;       &lt;/identities&gt;     &lt;/batch_program&gt;     &lt;batch_program batch_name="HealthCheck" </pre>



```

        platform_type="VNXe"
        program_name="KittyHawk">
    <identities>
        <identity name="batch_program_id"
            value="3" />
        <identity name="batch_info_id"
            value="3" />
    </identities>
</batch_program>
<batch_program batch_name="Upgrade"
    platform_type="VNXe"
    program_name="KittyHawk">
    <identities>
        <identity name="batch_program_id"
            value="3" />
        <identity name="batch_info_id"
            value="4" />
    </identities>
</batch_program>
</batch_program_data>
</data>

```

## 7. References

[1] Web MVC framework

<http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html>

[2] Design Patterns

[http://sourcemaking.com/design\\_patterns](http://sourcemaking.com/design_patterns)

[3] Example of Software Design Document

[http://portal.unimap.edu.my:7778/portal/page/portal30/Lecturer%20Notes/KEJURUTE RAAN\\_KOMPUTER/Semester%202%20Sidang%20Akademik%2020112012/EKT420 %20Software%20Engineering/Example%20of%20Software%20Design%20Document\(SDD\)/EDDISS.pdf](http://portal.unimap.edu.my:7778/portal/page/portal30/Lecturer%20Notes/KEJURUTE%20RAAN_KOMPUTER/Semester%202%20Sidang%20Akademik%2020112012/EKT420%20Software%20Engineering/Example%20of%20Software%20Design%20Document(SDD)/EDDISS.pdf)

## 8. Glossary

