**CS490**

**Blitz**

**Design Document**

Tianyang Yu, Jing Chang, Zhicheng Zhou, Yuying Wang, Hanchen Li, Dingzhe Li, Arthur Chen

## 

[1. Purpose](#h.8mc67jfwhoi)

[2. Design Outline](#h.v50mhvr9xe8r)

[(a) Components of the system:](#h.eysft0impg40)

[Server](#h.jrwevysa5w1s)

[Client](#h.10s35z4r9fee)

[Database](#h.2dusyt84rsr0)

[(b) Interactions Between Individual System Components:](#h.ek2r2u1pjd9v)

[Client and Server](#h.ljb2i2rnjgiy)

[Server and Database](#h.ltlcq9n3wtfc)

[(c) High-level Structure Diagram:](#h.k5ack6b6bx7w)

[3. Design Issues](#h.wv39p54l7md2)

[(a) Functional Issues:](#h.6hzhgh7480w1)

[Issue 1: Category Selection](#h.543xk61f77yl)

[Issue 2: User Role Separation](#h.3zk0v4yrjjpk)

[Issue 3: Creating request/service](#h.hk44rnethse5)

[Issue 4: Sorting](#h.twi44dbzw6yq)

[(b) Non-Functional Issues:](#h.vtd4tf5d91j2)

[Issue 1: Database Selection](#h.1amlega0m4ef)

[Issue 2: Language for Server](#h.9triyheizy1y)

[Issue 3: Language for ios Client](#h.leqt2zftt0tm)

[4. Design Details](#h.gf42cscwmog6)

[(a) class diagrams](#h.ly2jnwqmwqu)

[(b) classes and interactions between the classes:](#h.lknpdaqw4oeu)

[(c) sequence diagrams](#h.q0whx4iu95oj)

[(1) Get posts from Server](#h.ss8lqje4v4jc)

[(2) Watch a specific post](#h.a22qesgoxd72)

[(3) Post a new request/service](#h.vnsswfsnygke)

[(d) Activity Diagrams](#h.33puciu0gyex)

[(1) Login](#h.fwvnhvaso0dh)

[(2) Sign-up](#h.egbog9s9bb8)

[(3) Find service](#h.h9lf542uk63w)

# 

# 

# 

# 

# 

# 

# **1.** Purpose

In real life, people will encounter all kinds of small problems they cannot handle and need help. A platform (app/website) will provide an efficient way for them to find someone who is willing to help.

# **2. Design Outline**

We use client-server model.

## (a) Components of the system:

### Server

* + **Account Management Component**: Used to handle requests regarding to user account, such as register, login, change password etc.
  + **Query Component:** This component is used to fetch data of a service/request from the database.
  + **Post Component:** Used to handle the request regarding to creation of a new service/request.
  + **Matching Component:** Used to match the requests by conditions with similar service and return the results to client.
  + **Chatting Component:** Component can enable user to chat with each other.

### Client

Enable users to utilize all the functionalities of the app with friendly UI.

### Database

Stores all the data of user information and service information.

## (b) Interactions Between Individual System Components:

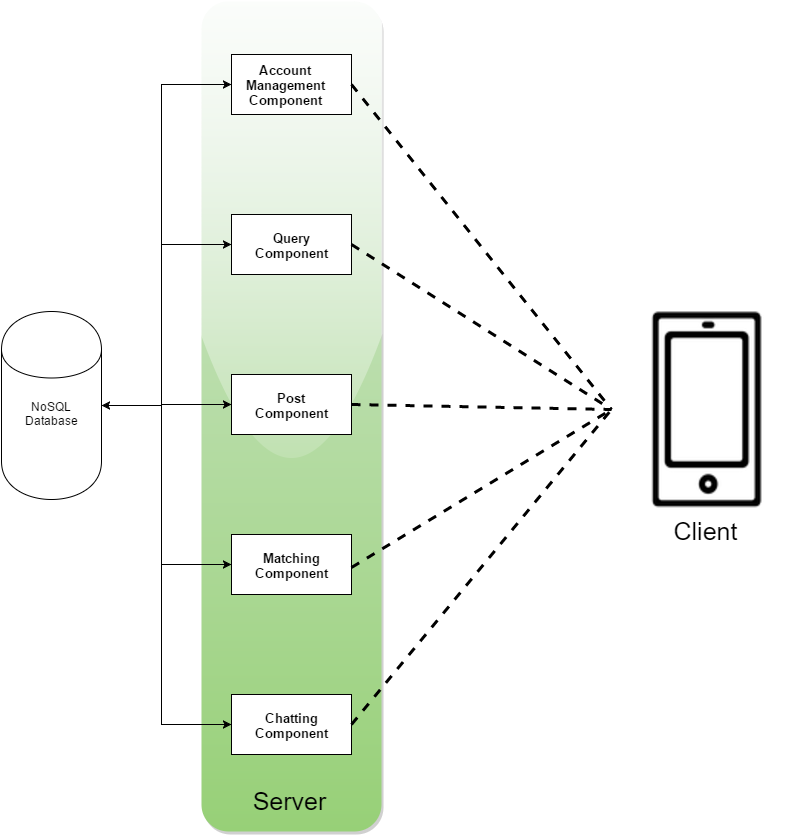
### Client and Server

* + Clients send some requests to server, server handle those events and store the data.
  + Server return the data that it handled to clients, clients transfer it into local data.

### Server and Database

* + When server receive an event that request data, it will retrieve data from database.
  + When server receive an event that update data, it will update the corresponding data in the database.

## (c) High-level Structure Diagram:



# **3. Design Issues**

## (a) Functional Issues:

### **Issue 1: Category Selection**

Opinion 1.1: User must select the category of the service first, then generate a list of existing provided/requested service.

Opinion 1.2: Users can firstly enter to the list of services including all the categories, then filter the category later.

Decision: Opinion 1.2. We think users may want to look for multiple categories at the same time, so we decide not to restrict user by enforcing them to select a category first.

### **Issue 2: User Role Separation**

Opinion 2.1: Dividing users into service provider and consumer, they will have different kinds of account.

Opinion 2.2: Do not divide users to provider and consumer, which means users are able to take both roles.

Decision: Opinion 2.2. Since user are able to be both provider and consumer in our app, so we decide to use the mixed mode.

### **Issue 3: Creating request/service**

Opinion 3.1:All category of service should be two way, both “requester” and “provider” can post the request.

Opinion 3.2: By different category requirement, we decide some of them are “two way” while others are “one way”, which means only “requester” or “provider” can post.

Decision: Opinion 3.2. According to our common sense. For example, for the taxi feature, only passengers can post the request message while driver are not allowed to post the service message.

### **Issue 4: Sorting**

Opinion 4.1: In all category, requests or offers are shown in the order of time ascending.

Opinion 4.2: requests or offers will be listed based on post time and those with long distance will be filtered out.

Decision: Opinion 4.2, because it makes more sense. For example, an indiana user would not like to receive request from NYC.

## (b) Non-Functional Issues:

### **Issue 1: Database Selection**

Option 1.1: Using MongoDB as database for our app.

Option 1.2: Using Mysql as database for our app.

Decision: Option 1.1. Since it is faster with millions of queries.

### **Issue 2: Language for Server**

Opinion 2.1: Use Java to implement the server.

Opinion 2.2: Use Node.js to implement the server.

Decision: Opinion 2.2. Since our product is cross

### **Issue 3: Language for ios Client**

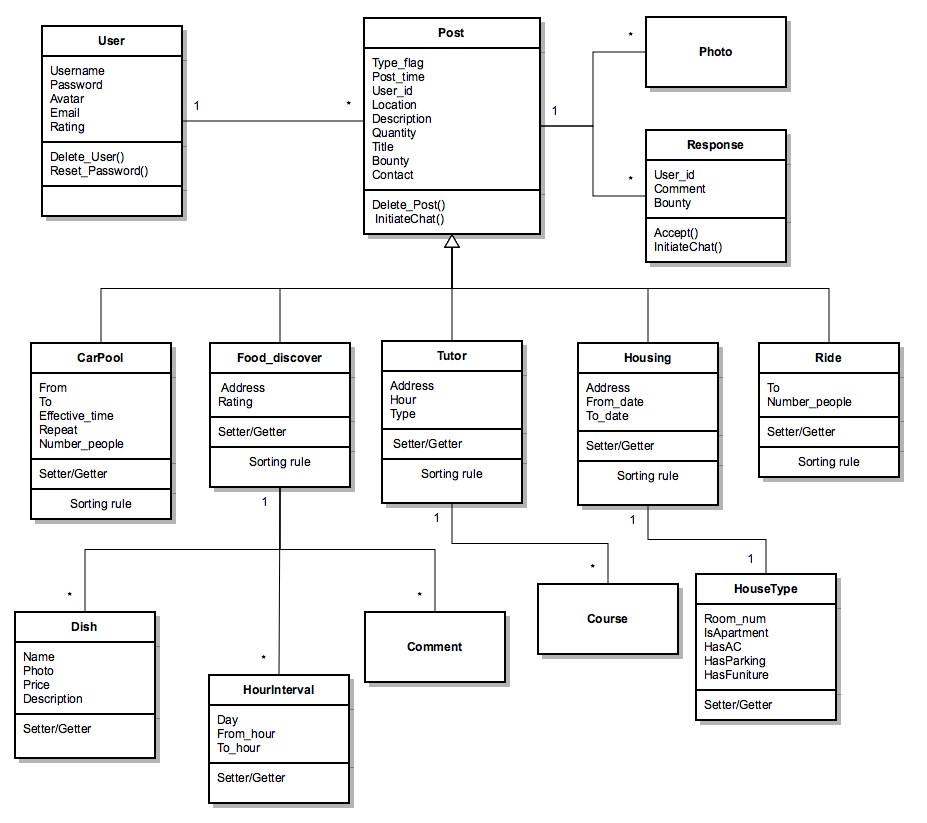
Opinion 3.1: Use Objective-C to implement the Client

Opinion 3.2: Use Swift to implement the client

Decision: Opinion 3.2. We choose to use Swift since the syntax is much better than Objective-C, which can reduce the time cost for learning the language.

# 4. Design Details

## (a) class diagrams



## (b) classes and interactions between the classes:

Classes:

* User is the class that stores user information, such as username, password, email, and scores that other users have rated.
* Post is a class that represents a request or offer that user post.
* Photo is a class to save photo
* Response is a reply sent by user.
* CarPool is a class that represents carpool category. It will save information related to requesting or offering a carpool, such as from and to location, when the service will be needed, how many people needs service.
* food\_discover is a class represents a category that helps user to find food. It will save information such as where people needs food to be delivered.
* Tutor is a class that represents a category that helps user to find tutor. It will store additional information, such as price and type of tutor.
* Housing is a class that represents a category that helps user to find house.It will have information about where people needs house and when people needs house.
* Ride is class that represents a category that helps user to find a taxi. It will store destination and how many people needs a taxi.
* Dish is a class that represents a dish. It will have name, photo, price, description about the dish.
* HourInteval is a class represents operating time of the restaurant or personal food providers.
* Comment is a class to store comments from users about the restaurant.
* Course is a class that represents course that tutor can teach or course that students need tutor.
* HouseType is a class that represents type of accommodation. It will have information about how many rooms it has, whether it is an apartment or house, and amenities information ,

Interactions:

* User writes Post
* Post is one of the type of CarPool, Food Discover, Tutor, Housing, or Ride.
* A post can contain 0 or more photos and can have 0 or more responses from other users.
* A Food discover class can have several dishes, opening time intervals, and comments.
* A tutor can choose to teach on several courses.

## (c) sequence diagrams

### (1) Get posts from Server

### 屏幕快照 2015-09-10 下午8.26.40.png

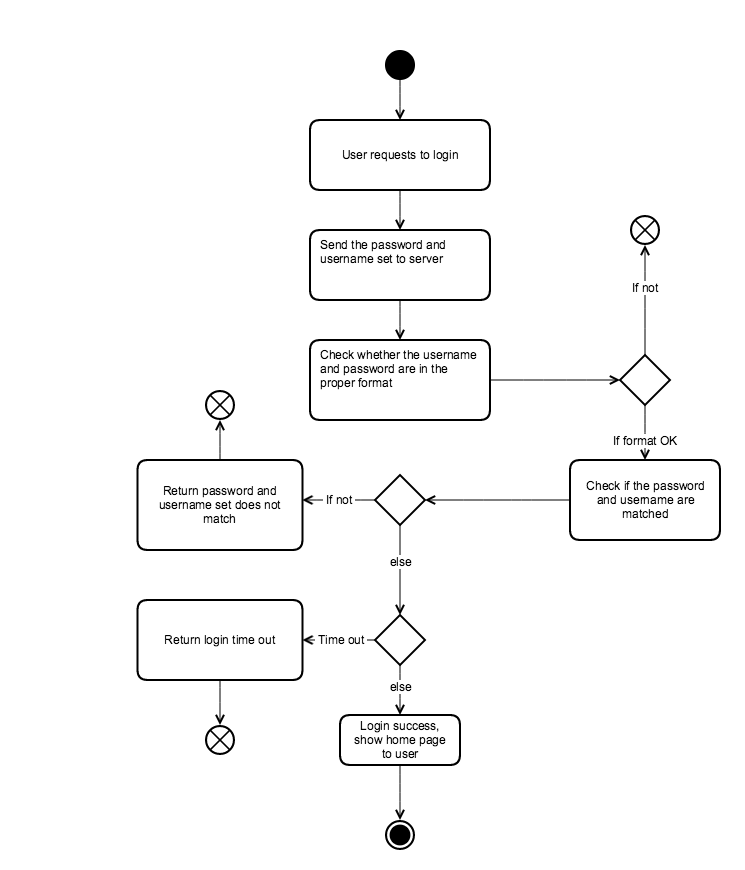
### 

### (2) Watch a specific postget_post.png

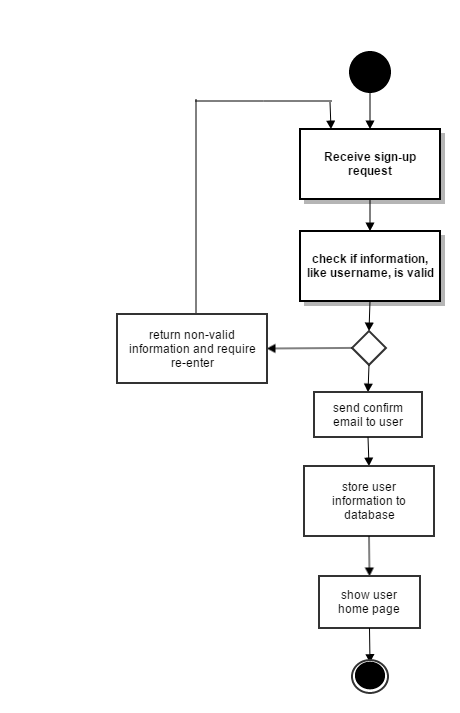
### (3) Post a new request/servicepost.png

## (d) Activity Diagrams

### (1) Login



### (2) Sign-up



### (3) Find service

