

## Ridesharing Application

Parth Kapur

Seren Yavuz

Thomas Gorney

03/10/19

<b>1.Introduction</b>	<b>3</b>
<b>2.Server Design</b>	<b>4</b>
2.1 Communications Module	5
2.2 Client Module	6
2.2.1 Passwords & Authentication	7
2.2.2 Routes	10
2.2.3 Profiles	12
<b>3.Client Design</b>	<b>14</b>
3.1 UI Module	15
3.2 Communication Module	15

## 1. Introduction

The client-server model will be utilized for our ridesharing application.

Client-server model is when the server hosts, delivers, and manages most of the resources to be used by client. Client-server model has one or more client that are connected to the central server over a network or internet connection. At Rideshare, our application is based on users' travel requests or offers. Within the application, users put their request/offer of their travel with details. Then, the server provides resources back to users again and matches them with other clients. Our server design and client design are explained in details below.



## 2. Server Design

### Server Module

#### Purpose

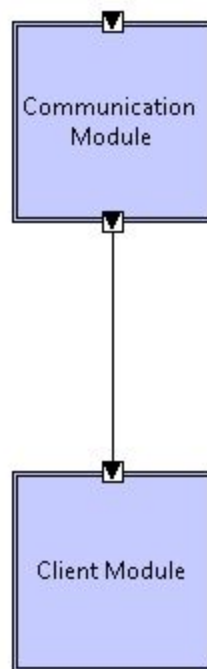
The purpose of the server module is to store, access, or change information in a centralized place.

#### Rationale

This module was created to create a centralized place to store user data, that can be retrieved and viewed. The server module will store things such as profile information, car information, trip information, and scheduling information.

#### High-Level Server Design

Considering that all users have the same functionality in terms of accessibility and privileges, the server design will mirror the high level design of the overall system.



**Provided Interface**

The provided interface of this module is the union between communication module and client module.

**Required Interface**

This module has no required interface.

## 2.1 Communications Module

**Purpose**

The communication module will provide communication services between the client module and the server. This module represents the part of the communications link that is on the server.

**Rationale**

This module was created to simplify communication between the clients and the server.

**Provided Interface**

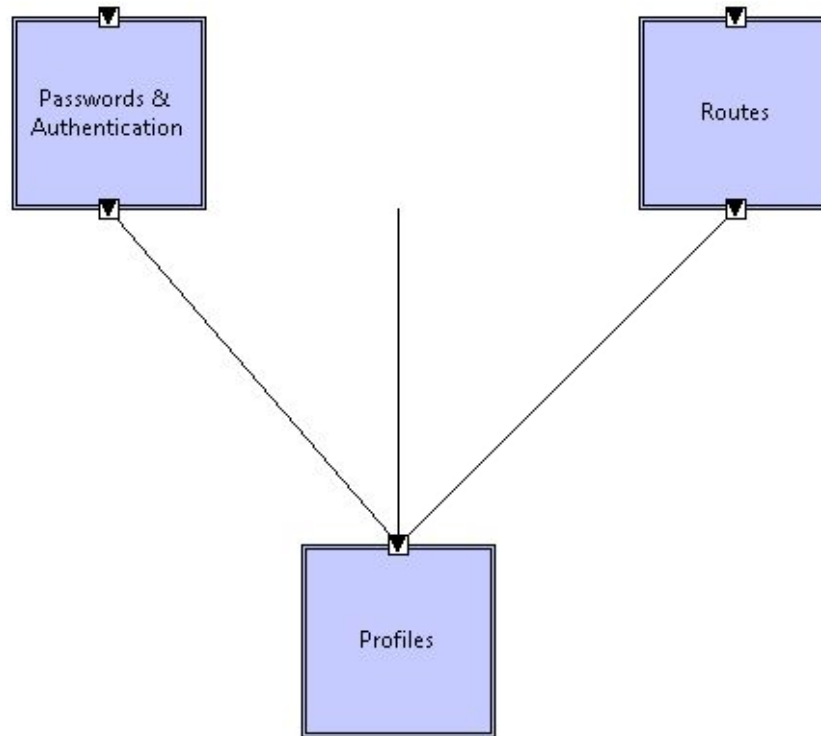
The provided interface of this module is the same as the required interface. The key difference being the interface exists for network calls, not procedure calls.

**Required Interface**

This module's required interface is the union of:

- Client Module

## 2.2 Client Module



### **Purpose**

The purpose of this module is to authenticate users, provide route information and profile details as well as management for these components.

### **Rationale**

This module is created in order to create a single space to manage users server side details.

### **High-Level Module Design**

The client module is broken down into smaller components, that can be seen as high-level modules each.

## Provided Interface

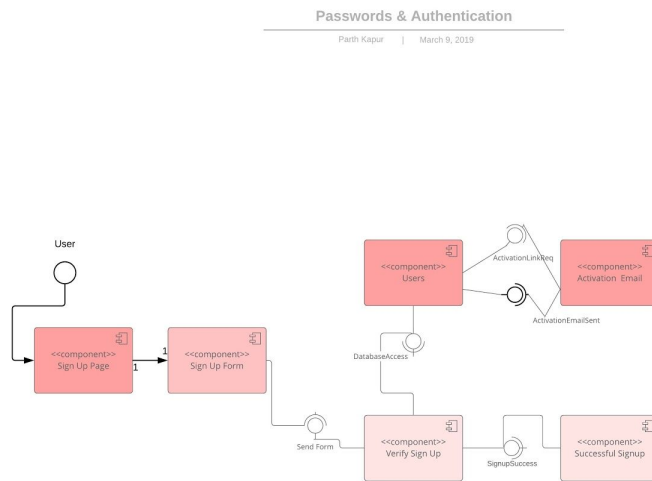
The provided interface for this module is the connections between the provided interfaces submodules.:

- passwords and authentication
- route details
- profiles submodules.

## Required Interface

This module has no required interfaces.

### 2.2.1 Passwords & Authentication



## **Purpose**

The purpose of this module is to allow the user to sign up & become a registered user, thus allowing all website features to be available to the user.

## **Rationale**

This module is created to centralize the sign in process. It will allow secure registration and data storage related to the user.

## **Required Interface**

```
Boolean userExists(UserID userID);  
Boolean emailExists(Email email);
```

## **Provided Interface**

```
Boolean fillForm(UserID userID, Password password, Email email)  
    Throws invalidFormException
```

## **Description:**

- fillForm will take information filled by the user and validate.

## **Parameters:**

- userID: User ID of the user whose password is being stored.
- Password: The password for the user
- Email: The email for that user

## **Exceptions:**

- invalidFormException: If user failed to fill in form correctly.

## **Returns:**

- Returns true if the form was filled correctly, otherwise it will return false and throw an exception.

```
Boolean activateAccount(Form formObject);
```



**Description:**

- activateAccount will take in a Form object and pass it to the database, which will then return a boolean if account activation was successful. Also will be used to send user to activation link page.

**Parameters:**

- Form: An object of Form, individual to the user who is filling form

**Returns:**

- Returns true if the account was successfully activated, otherwise it returns false.

```
Boolean logon(UserID userID, Password password);
```

**Description:**

- Validates given user/password combination without specifying which specified field is incorrect

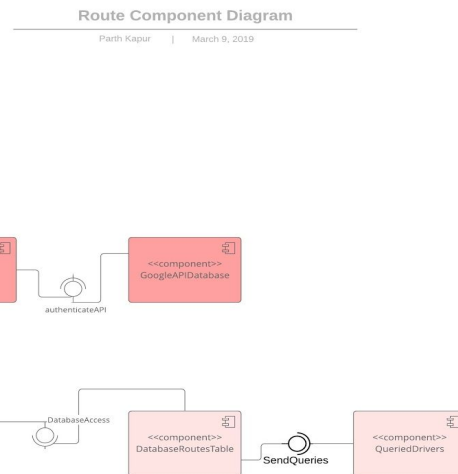
**Parameters:**

- userID: User ID of the user
- Password: password for that specific user

**Returns:**

- True if the user ID/password is correct, otherwise it will fail.

## 2.2.2 Routes



### Purpose:

The purpose of this module is to provide access to the Google Maps API embedded within the website. This API will work with both Google's API database (to ensure the API key is correct) as well as RideShare's database to query routes being offered.

### Rationale:

This module is created to centralize and encapsulate route management services.

### Required Interface:

```
Boolean apiIsValid(String APIKey);
```

### Provided Interface:

```
String[] (string startLoc, string endLoc)  
    Throws noRidesAvailableException;
```

**Description:**

This function will return an array of strings that will contain all the available rides in accordance to the user specifications. This array of strings will then be used to display all rides.

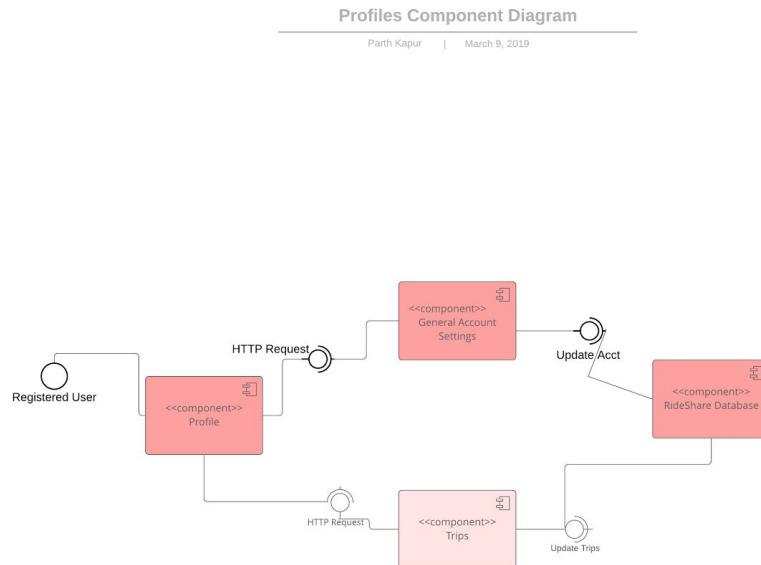
**Parameters:**

- startLoc: This will be the starting location specified by the user that will be converted to longitude and latitude coordinates within the function.
- endLoc: This will be the ending location specified by the user that will be converted to longitude and latitude coordinates within the function.

**Exceptions:**

noRidesAvailableException: Thrown if there are no rides available in accordance to the user specifications.

## 2.2.3 Profiles



### Purpose

The purpose of this module is to allow a registered user (once logged in) to look at his or her profile, add trips or change general account settings.

### Rationale

This module was created to centralize and encapsulate all user profile specific services.

### Required Interface

```
Boolean login(UserID userID, Password password);
```

### Provided Interface

```
Boolean editAccount(UserID userID, Password password, Email  
email);
```

**Description:**

This function will edit the user account settings and return a boolean once successfully (or unsuccessfully) edited.

**Parameters:**

userID: The userID of the user (if he has changed it).

Password: The password of the user (if he has changed it).

Email: The email of the user (if he has changed it).

**Returns:**

Returns a boolean if successful or not.

```
Boolean editTrips(String startLoc, String endLoc, Double price,  
Int Passengers)
```

**Description:**

This function will edit the trips for the user if he or she wishes to provide or remove a trip he or she is offering.

**Parameters:**

startLoc: Starting location of the trip

endLoc: Ending location of the trip

Price: price of the trip (per passenger)

Passenger: Number of passengers allowed

**Returns:**

Returns a boolean if editing a trip was successful or not.

### 3. Client Design

#### Client Module

##### **Purpose**

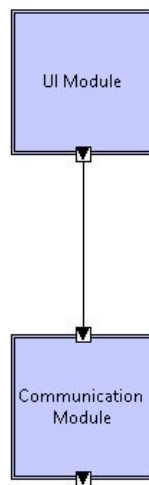
The purpose of the client module is to provide the user with an interface and functions for them to use. This is the main module that the client will interact with.

##### **Rationale**

This module was created to create a centralized place for all user functions and the user interface.

##### **High-Level Module Design**

The client module is broken down into smaller components, that can be seen as high-level modules.



##### **Provided Interface**

This module has no provided interfaces.

##### **Required Interface**

This module's required interface is the union of the UI module and the communication module.

## 3.1 UI Module

### **Purpose**

The purpose of the UI module is to provide the user with a user interface that they can interact with.

### **Rationale**

This module was created to take in user input or user data and send it to the communications module.

### **High-Level Module Design**

This UI module is broken down into high-level modules which will consist of web pages provided in the initial report.

### **Required Interface**

This module has no required interface.

## 3.2 Communication Module

### **Purpose**

The purpose of the Communication module is to provide the means for communication between the client and the server. This module is on the client side of the communications.

### **Rationale**

This module was created to create a centralized place for communications with the network between the client and the server.

### **High-Level Module Design**

This Communications module is broken down into high-level module (UI module) and will pass on input from the UI module to the server.

### **Required Interface**

This module has no required interface.