**Athlone Institute of Technology**

**School of Engineering.**

**Project Thesis – Academic Year 2017/18**

**Web application**

**In**

**Node js and HTML5**

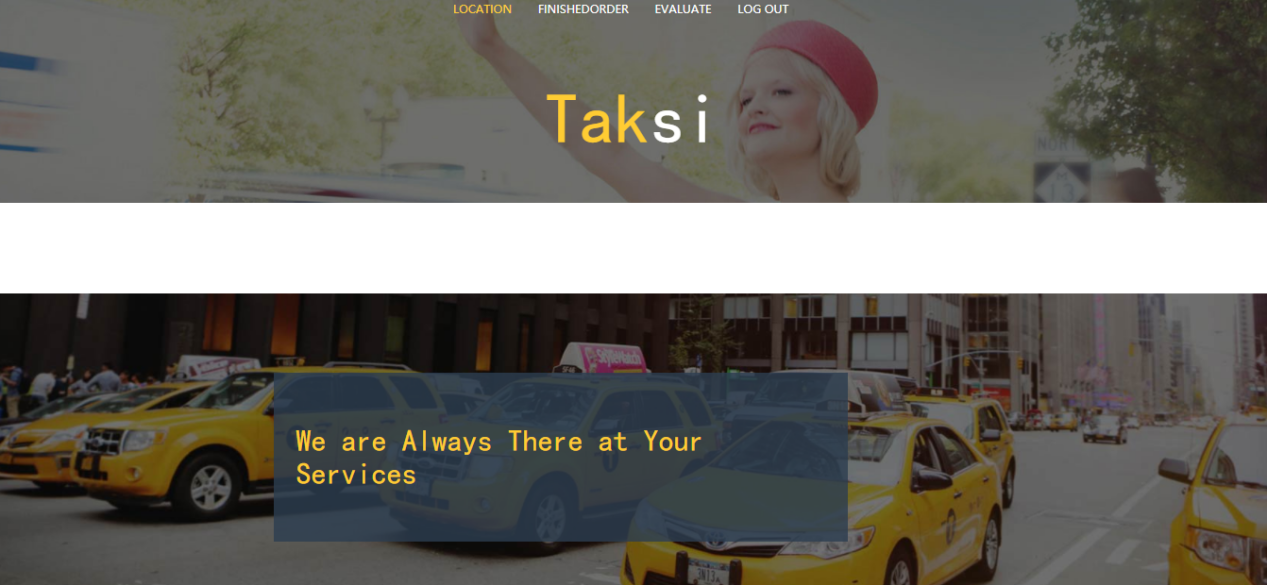
**By**

**Zheng Lyu**

**Bachelor of Engineering (Honours) in Software Engineering**

**Taksi**

(taxi-hailing application)



**Author: ZHENG LYU**

**Supervisor: Dr Declan Byrne**

# **Project Summary**

This report is to introduce the aim of developing this application, how did I come up with it and the process of designing the structure, coding, testing and complete it. And this report will outline how did I use the relevant technologies to compete this application.

This report will outline how the latest technical node.js will be use as a server side and connect to the html page and mysql. It will outline the ability to implement javascript on the server side, how Node.js embodies the use of event-driven, non-blocking I / O models to be lightweight and efficient.

目录

**[1. Introduction 5](#_Toc32720)**

**[2. Aims and Objectives 5](#_Toc10602)**

[2.1 Project aims are: 6](#_Toc10761)

[2.2 Objectives: 6](#_Toc7847)

**[3. Architecture 7](#_Toc17745)**

[3.1 Server records 8](#_Toc19444)

[3.2 Illustration( customer and driver) 8](#_Toc28562)

**[4. Scope 9](#_Toc10197)**

**[5. Build Tools (Hardware & Software) 10](#_Toc30889)**

[5.1 HTML: 10](#_Toc12425)

[5.2 Css3 11](#_Toc1092)

[5.3 Node.js: 12](#_Toc26542)

[5.4 MySQL 13](#_Toc28561)

[5.5 Netbean 14](#_Toc6072)

**[6. Design 16](#_Toc7489)**

**[7. Implementation 16](#_Toc6748)**

[7.1 Login/sign up 16](#_Toc8921)

[7.2 Customer homepage: 20](#_Toc20261)

[7.3 Driver homepage 21](#_Toc26988)

[7.4 Customer location 22](#_Toc11202)

[7.5 Driver get order 25](#_Toc9516)

[7.6 Customer confirm order 29](#_Toc23516)

[7.7 Evaluate the service 34](#_Toc25204)

[7.8 Drivers’ Evaluation and marks 38](#_Toc21610)

[7.9 Log out 40](#_Toc7225)

**[8. Database 41](#_Toc7908)**

**[9. Testing and Evaluation 44](#_Toc25960)**

[9.1 Test cases: 45](#_Toc9317)

[9.2 Evaluation: 48](#_Toc7808)

**[10. Conclusion 48](#_Toc19241)**

**[11. Reference: 51](#_Toc29619)**

# **Introduction**

Taksi is a Taxi-hailing application based on node.js and Html5. Node.js is a JavaScript run-time that was released in May 2009 and was developed by Ryan Dahl. It actually encapsulates the Google V8 engine. The V8 engine executes JavaScript very fast and its performance is very good. Node.js optimizes some special use cases and provides alternative APIs that make V8 run better in non-browser environments. I used Node.js because it is a platform built on the Chrome JavaScript run-time to easily build web applications that respond quickly and easily. Node.js uses an event-driven, non-blocking I/O model to be lightweight and efficient, making it ideal for running data-intensive real-time applications on distributed devices. Node.js works as a new front-end framework, background language, which have many attractive places like restful API and single thread, it can still handle tasks concurrently without adding extra threads and it implements concurrent operations through event loops. These features attracted me so I decided to research this new popular technical and used it as my background language.

# **Aims and Objectives**

This report will outline the development of an web application, the application works as a Taxi-hailing application. The reason I chose this as a topic is because the network services have penetrated into all aspects of the service industry and we could see many kind of Taxi-hailing application on the smart phone, so I want to come up with an web taxi-hailing application for those people are not adaptable to use smart phone.

And one of my main aim is to learn the node.js, I am interested in how this framework makes javascript easier to use on server-side script and how it get the data and send it to the database. This is a young technical very useful in Front-end development.

## 2.1 Project aims are:

Make a fully functioning web application.

Make a friendly- to-use interface for users.

Use node.js to implement background functions.

Use the Mysql to store the data for application.

## 2.2 Objectives:

Learn the new technical of node.js.

Find out how node.js makes javascript run on the server-side.

Find out more ability for node.js to use on the server-side.

Learn the process of a web application achieve data exchange from client to server

# **Architecture**

Following diagram is an overview of the system architecture, I have two servers, customer and driver and both of them have the structure of home page-server-database. I will introduce each sections for more details.

Taksi

Log In/Out

Logged in Enter WEB

Driver

Customer

Customer server

Driver server

Database

Database

## 3.1 Server records

Taksi

# 

Server

SQL

SQL

SQL

Confirm order

Select order

Driver marks

evaluation

location

SQL

SQL

SQL

SQL

## 3.2 Illustration( customer and driver)

The following diagram will outline the whole relationship in all the applications

Taksi

# 

Sign up/log in

Driver

Customer

Log out

Driver marks

Select order

Log out

evaluation

Confirm order

location

# **Scope**

After I have decided what application I am going to do, I should decided the scope of the application. I needed to come up with what is the most important function of a taxi-hailing application, it is clear that is how the customers locate themselves and send the data to make an order, then how will a driver see that order and get the order, set the price for services, then customer think the price is resonable and start the service. These are the main process of a taxi-hailing application. What’s more, I thought the customers should be able to give a feed back to evaluate about if they are satisfy with drivers’ service, and they might give some advice. To achieve this process, I considered how should this works, I thought both customers and drivers should have their own homepage because they have different duty in executing the service, so I created a login screen and a homepage for both of them. From now on, the scope of my application were clear in general.

1. **Build Tools**(Hardware & Software)

To develope the taxi-hailing application, I will use the node.js and html to achieve it. The application works as a web application, so the html is used to build it’s interface and I will use some css and javascript to fix it’s appearance as well. I choosed MySQL as a database to store the data because that is the database that I am most familiar with and it is very suitable for this application. Node.js is used as a server side to connect the database and interface and achieve the data exchange.

For the hardware, I need a computer with compiling software named NetBean. It is my first time to use NetBean, I had planned to use dreamweaver to do it but dreamweaver was a very old compiling software and had stoped update for a long time, so I thought I should try to use a new compiling software. NetBean is indeed a suitable compiling software because it supported both node.js and html, and it have the tomcat server as well.

Now here are some introduce for those techniques:

5.1 HTML:

Hypertext Markup Language (referred to as HTML) is a standard markup language for creating web pages. HTML is a basic technology and is often used together with CSS and JavaScript by many web sites to design pleasing web pages. Programs and mobile application user interfaces. Web browsers can read HTML documents and render them into visual Web pages. HTML describes the structure of a Web site with the clues, making it a markup language rather than a programming language.

HTML elements are the cornerstones for building websites. HTML allows images and objects to be embedded, and can be used to create interactive forms. It is used to structure information - such as headings, paragraphs, and lists, etc. It can also be used to describe documents to some extent. The appearance and semantics of the HTML. The language form of the HTML is an HTML element (such as <HTML>) surrounded by angle brackets. The browser uses HTML tags and scripts to interpret the content of the page but does not display it on the page.

HTML can be embedded into scripting languages such as JavaScript that affect the behavior of HTML web pages. Web browsers can also reference Cascading Style Sheets (CSS) to define the appearance and layout of text and other elements. The World Wide Web Consortium (W3C), which maintains the HTML and CSS standards, encourages people to use CSS to replace some of the HTML elements used for presentation.[[[1]](#endnote-0)]

5.2 Css3

Cascading Style Sheets (CSS), a computer language for adding styles (fonts, spacing, colors, etc.) to structured documents (such as HTML documents or XML applications), defined and maintained by the W3C.

CSS cannot be used alone and must work together with HTML or XML to decorate HTML or XML. This article focuses on CSS technology for decorating HTML web pages. Where HTML is responsible for determining what is in a web page, CSS determines what appearance (size, thickness, color, alignment, and position) these elements are presented. CSS can be used to set page layouts, set page element styles, and set global styles that apply to all web pages. CSS can be added directly to the elements of a web page to be styled, and it can also be centralized in web pages, link-introduced web pages, and import-introduced web pages. [1]

The most important goal of CSS is to separate the contents of a file from its display. Before CSS appeared, almost all HTML files contained information displayed in the file. For example, the color of the font, what the background should be, how it was arranged, edges, lines, etc. must all be listed in the HTML file, sometimes Repeatedly listed. CSS allows authors to isolate most of this information and simplify the HTML file, which is placed in an auxiliary, CSS-written file. The HTML file contains only the structure and content information. The CSS file contains only the style information.

CSS3 adds a number of features such as "border-radius," "text-shadow," "transform," and "transition." CSS3 also supports animation and preserved-3d.[[[2]](#endnote-1)]

## 5.3 Node.js:

Node.js is an [open-source](https://en.wikipedia.org/wiki/Open-source_software" \o "Open-source software), [cross-platform](https://en.wikipedia.org/wiki/Cross-platform" \o "Cross-platform) [JavaScript](https://en.wikipedia.org/wiki/JavaScript" \o "JavaScript) [run-time environment](https://en.wikipedia.org/wiki/Runtime_system" \o "Runtime system) that executes JavaScript code [server-side](https://en.wikipedia.org/wiki/Server-side" \o "Server-side). Historically, JavaScript was used primarily for [client-side scripting](https://en.wikipedia.org/wiki/Client-side_scripting" \o "Client-side scripting), in which scripts written in JavaScript are embedded in a webpage's HTML and run client-side by a JavaScript engine in the user's web browser. Node.js lets developers use JavaScript for [server-side scripting](https://en.wikipedia.org/wiki/Server-side_scripting" \o ")—running scripts server-side to produce [dynamic web page](https://en.wikipedia.org/wiki/Dynamic_web_page" \o "Dynamic web page) content *before* the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying [web application](https://en.wikipedia.org/wiki/Web_application" \o "Web application) development around a single programming language, rather than different languages for server side and client side scripts.

Though .js is the conventional [filename extension](https://en.wikipedia.org/wiki/Filename_extension" \o "Filename extension) for JavaScript code, the name "Node.js" does not refer to a particular file in this context and is merely the name of the product. Node.js has an [event-driven architecture](https://en.wikipedia.org/wiki/Event-driven_architecture" \o "Event-driven architecture)capable of [asynchronous I/O](https://en.wikipedia.org/wiki/Asynchronous_I/O" \o "Asynchronous I/O). These design choices aim to optimize [throughput](https://en.wikipedia.org/wiki/Throughput" \o "Throughput) and [scalability](https://en.wikipedia.org/wiki/Scalability" \o "Scalability) in web applications with many input/output operations, as well as for [real-time Web](https://en.wikipedia.org/wiki/Real-time_Web" \o "Real-time Web) applications (e.g., [real-time communication](https://en.wikipedia.org/wiki/Real-time_communication" \o "Real-time communication) programs and [browser games](https://en.wikipedia.org/wiki/Browser_game" \o "Browser game)).[[[3]](#endnote-2)]

5.4 MySQL

MySQL is an [open-source](https://en.wikipedia.org/wiki/Open-source" \o "Open-source) [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system" \o "Relational database management system) (RDBMS). MySQL is a central component of the [LAMP](https://en.wikipedia.org/wiki/LAMP_(software_bundle)" \o "LAMP (software bundle)) open-source web application software stack (and other "[AMP](https://en.wikipedia.org/wiki/List_of_AMP_packages" \o "List of AMP packages)" stacks). LAMP is an acronym for "[Linux](https://en.wikipedia.org/wiki/Linux" \o "Linux), [Apache](https://en.wikipedia.org/wiki/Apache_HTTP_Server" \o "Apache HTTP Server), MySQL, [Perl](https://en.wikipedia.org/wiki/Perl" \o "Perl)/[PHP](https://en.wikipedia.org/wiki/PHP" \o "PHP)/[Python](https://en.wikipedia.org/wiki/Python_(programming_language)" \o "Python (programming language))". Applications that use the MySQL database include: [TYPO3](https://en.wikipedia.org/wiki/TYPO3" \o "TYPO3), [MODx](https://en.wikipedia.org/wiki/MODx" \o "MODx), [Joomla](https://en.wikipedia.org/wiki/Joomla" \o "Joomla), [WordPress](https://en.wikipedia.org/wiki/WordPress" \o "WordPress), [Simple Machines Forum](https://en.wikipedia.org/wiki/Simple_Machines_Forum" \o "), [phpBB](https://en.wikipedia.org/wiki/PhpBB" \o "PhpBB), [MyBB](https://en.wikipedia.org/wiki/MyBB" \o "MyBB), and [Drupal](https://en.wikipedia.org/wiki/Drupal" \o "Drupal). MySQL is also used in many high-profile, large-scale [websites](https://en.wikipedia.org/wiki/Website" \o "Website), including [Google](https://en.wikipedia.org/wiki/Google" \o "Google), [Facebook](https://en.wikipedia.org/wiki/Facebook" \o "Facebook), [Twitter](https://en.wikipedia.org/wiki/Twitter" \o "Twitter), [Flickr](https://en.wikipedia.org/wiki/Flickr" \o "Flickr), and [YouTube](https://en.wikipedia.org/wiki/YouTube" \o "YouTube).

MySQL is written in [C](https://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language)) and [C++](https://en.wikipedia.org/wiki/C++" \o "C++). Its SQL parser is written in [yacc](https://en.wikipedia.org/wiki/Yacc" \o "Yacc), but it uses a home-brewed [lexical analyzer](https://en.wikipedia.org/wiki/Lexical_analysis" \o "Lexical analysis). MySQL works on many [system platforms](https://en.wikipedia.org/wiki/System_platform" \o "System platform), including [AIX](https://en.wikipedia.org/wiki/AIX_operating_system" \o "AIX operating system), [BSDi](https://en.wikipedia.org/wiki/BSD/OS" \o "BSD/OS), [FreeBSD](https://en.wikipedia.org/wiki/FreeBSD" \o "FreeBSD), [HP-UX](https://en.wikipedia.org/wiki/HP-UX" \o "HP-UX), [eComStation](https://en.wikipedia.org/wiki/EComStation" \o "EComStation), [i5/OS](https://en.wikipedia.org/wiki/IBM_i5/OS" \o "IBM i5/OS), [IRIX](https://en.wikipedia.org/wiki/IRIX" \o "IRIX), [Linux](https://en.wikipedia.org/wiki/Linux" \o "Linux), [macOS](https://en.wikipedia.org/wiki/MacOS" \o "MacOS), [Microsoft Windows](https://en.wikipedia.org/wiki/Microsoft_Windows" \o "Microsoft Windows), [NetBSD](https://en.wikipedia.org/wiki/NetBSD" \o "NetBSD), [Novell NetWare](https://en.wikipedia.org/wiki/Novell_NetWare" \o "Novell NetWare), [OpenBSD](https://en.wikipedia.org/wiki/OpenBSD" \o "OpenBSD), [OpenSolaris](https://en.wikipedia.org/wiki/OpenSolaris" \o "OpenSolaris), [OS/2](https://en.wikipedia.org/wiki/OS/2" \o "OS/2) Warp, [QNX](https://en.wikipedia.org/wiki/QNX" \o "QNX), [Oracle Solaris](https://en.wikipedia.org/wiki/Solaris_(operating_system)" \o "Solaris (operating system)), [Symbian](https://en.wikipedia.org/wiki/Symbian" \o "Symbian), [SunOS](https://en.wikipedia.org/wiki/SunOS" \o "SunOS), [SCO OpenServer](https://en.wikipedia.org/wiki/SCO_OpenServer" \o "SCO OpenServer), SCO [UnixWare](https://en.wikipedia.org/wiki/UnixWare" \o "UnixWare), Sanos and [Tru64](https://en.wikipedia.org/wiki/Tru64" \o "Tru64). A port of MySQL to [OpenVMS](https://en.wikipedia.org/wiki/OpenVMS" \o "OpenVMS) also exists.

MySQL has received positive reviews, and reviewers noticed it "performs extremely well in the average case" and that the "developer interfaces are there, and the documentation (not to mention feedback in the real world via Web sites and the like) is very, very good". It has also been tested to be a "fast, stable and true multi-user, multi-threaded sql database server".

Generally speaking, mysql has the following advantages:

1.MySQL is open source, so you don't need to pay extra.

2.MySQL supports large databases. Can handle large databases with tens of millions of records.

3.MySQL uses the standard SQL data language format.

4.Mysql can run on multiple systems and supports multiple languages. These programming languages include C, C++, Python and Java, Perl, PHP, Eiffel Tower, Ruby and Tcl.

5.Mysql has good support for PHP, PHP is the most popular web development language.

6.MySQL supports large databases, supports data warehouses with 50 million records, 32-bit system table files can support up to 4 GB, and 64-bit systems support the largest table file of 8 TB.

7.Mysql is customizable, using the GPL protocol, you can modify the source code to develop your own MySQL system.[[[4]](#endnote-3)]

5.5 Netbean

NetBeans is an open source software development integration environment. It is an open framework and extensible development platform. It can be used for the development of languages such as Java, C / C + +, and PHP. It is itself a development platform and can extend functions through extension plug-ins.

The NetBeans project is supported by an active development community. The NetBean development environment provides a wealth of product documentation and training resources as well as a large number of third-party plug-ins.

In the NetBeans platform, application software is constructed using a series of software modules (modular software components). These modules are a Java archive file, which contains a set of Java class categories and they implement all the open interfaces defined by NetBeans and a series of user-defined definition descriptors (Manifest). File). Thanks to the benefits of modularization, applications built with modules can be further expanded with the addition of new module employment.

Applications can dynamically install modules. Any application can include an update module that allows users to apply downloaded updates and add new features. In this way, upgrades and new concurrent launches do not have to force users to download the entire application each time.

The entire Netbeans platform provides services that are common to desktop applications, allowing developers to focus on the logical design of their applications only. The main features of the NetBeans platform are:

User interface management (such as menus and toolbars)

User settings management

Storage Management (Retaining and Loading Any Kind of Data)

Window Management

Wizard framework (step-by-step support dialog)[[[5]](#endnote-4)]

# **Design**

When designing my project, I decided to use HTML5 as my front-end, since it is a very good language to use for web page, and I used the MySQL as the database to store the data of application. For the server-side, I chose the Node.js to work on it because it allows me to use javascript to write code on server-side. After investigation, my project’s structure is like: I created many servers, each server has a listening port and achieve a function, those ports will listen to the operation on the front-end and execute the statement in database. Those servers works by node.js.

# **Implementation**

7.1 Login/sign up

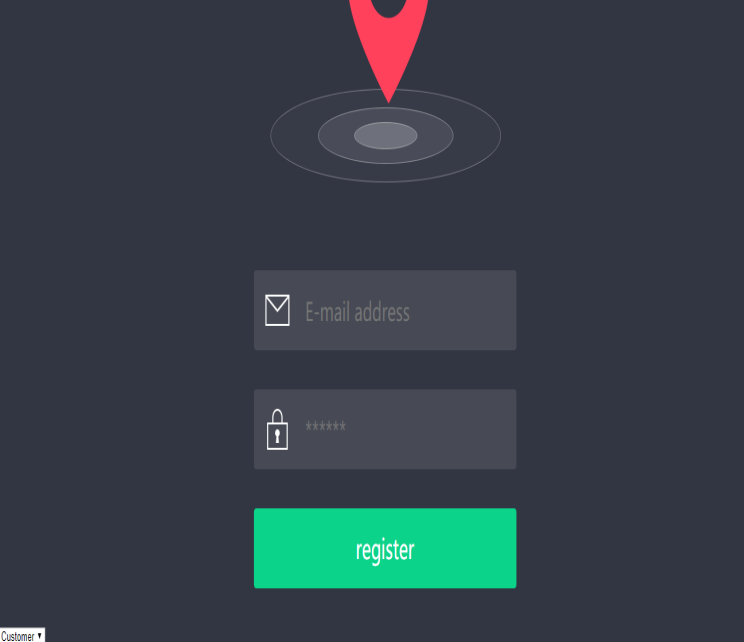


fig 7.1

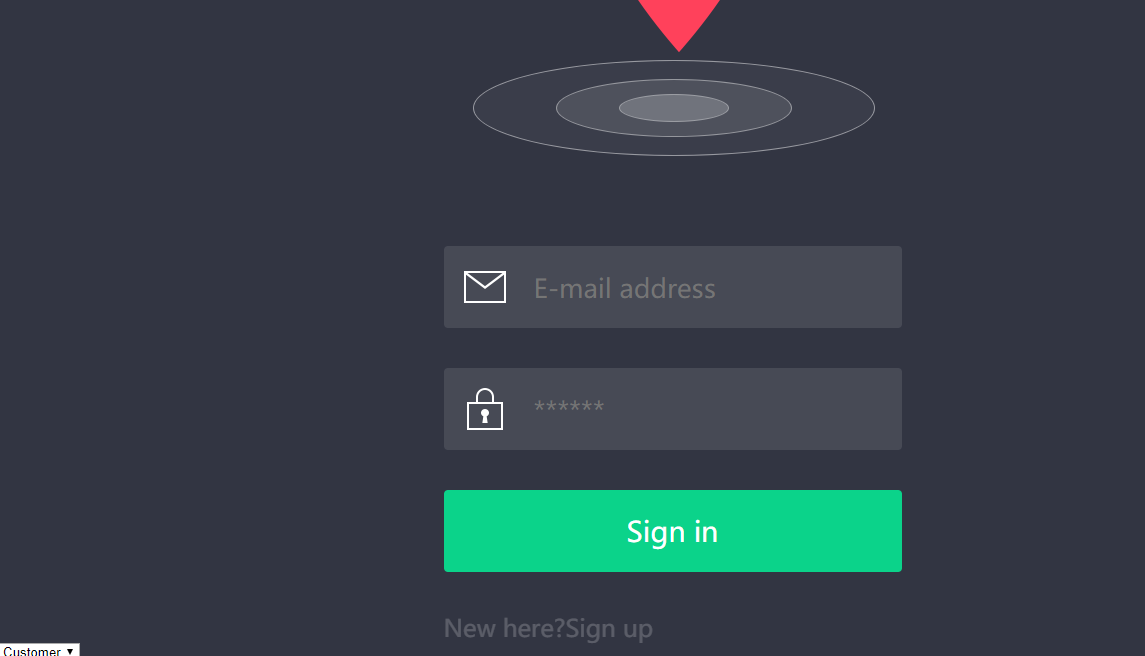


Fig 7..2

The first part I did was login and sign up. Since I have two homepages for customers and drivers, so there is a select box for user to choose whether they are customers or drivers. There are two text fields, email and password. The application varify the identity by email so users must input the correct email format. If not, it will have a message box to remind you what problem you have.

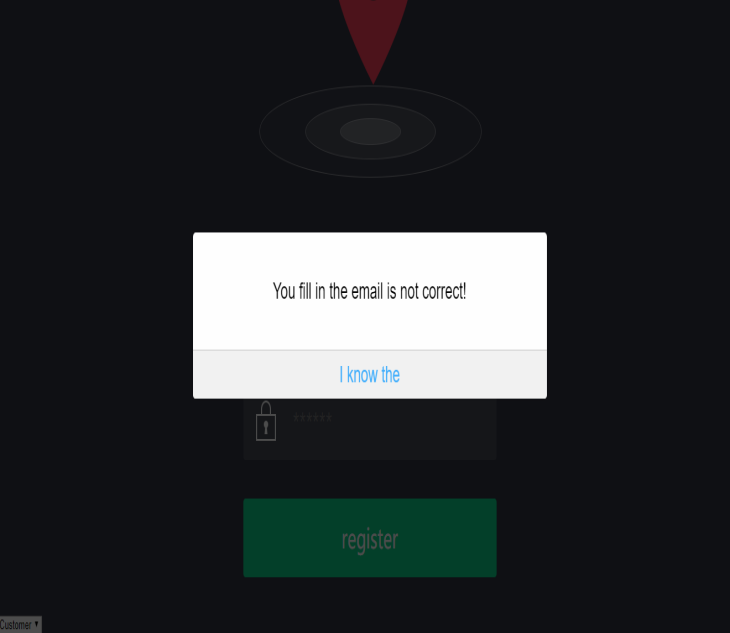


Fig 7.3

For login and sign up functions, they have their own interface(html page), but both of them use sharing the same server side. I used an ajax method to get the inputted data and send it to the server side. The ajax() method loads remote data through an HTTP request. This method is jQuery's underlying AJAX implementation. Easy-to-use high-level implementations can be found in $.get, $.post, etc. $. Ajax() returns the XMLHttpRequest object it created. In most cases you don't need to directly manipulate this function unless you need to manipulate the less common options for more flexibility. If they get the response message from server side they will execute the code in success function. After a successful sign up the page will jump to login page, and after user login successful they will be able to link their home page.



Fig 7.4

The function about operating the login and sign up is made by node.js. The above figure show a javascript file names login2.js on the server-side First the server side should connect to the database by a method connection.

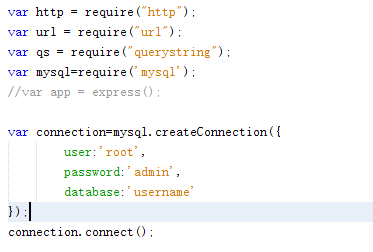


Fig 7.5

And I have already created two tables, Login\_Information for customers and Drivers\_Information for drivers.

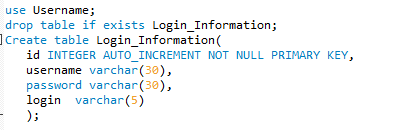


Fig 7.6

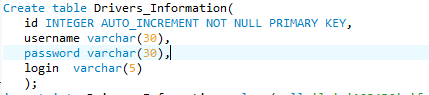


Fig 7.7

The server side will create a server and listen to the port 3010, both sign up page and login page send the data from this port. The data contains username, password and select(customer or driver). After the server side get the data, it will judge the sent data is for customers or drivers and search whether the data is in database. If the data is in database, the server side will analysis the data was sent from sign up page or login page. If the data was from sign up page, the server side will send back a message to remind user it is an existed user. If the data was sent by login page, the login will be successful and users enter the home page.

But if there is no that data searched from database, if the data was sent by login page, the server side will remind user login failed, and if the data was sent by sign up page it will record the data as a new user in database.

## 7.2 Customer homepage:



Fig 7.8

When customers succeed to login, they will enter the home page like this. There are four functions for customers: location, finished order, evaluation and log out. The customers could choose the service by click on menu. That menu works by the follow design.

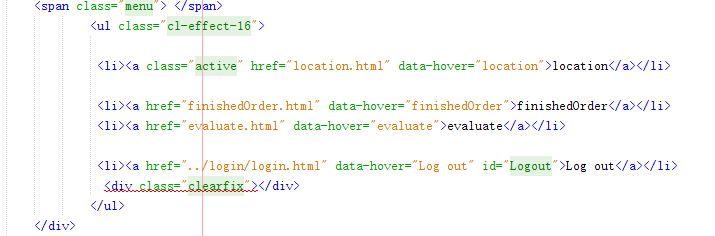


Fig 7.9

7.3 Driver homepage

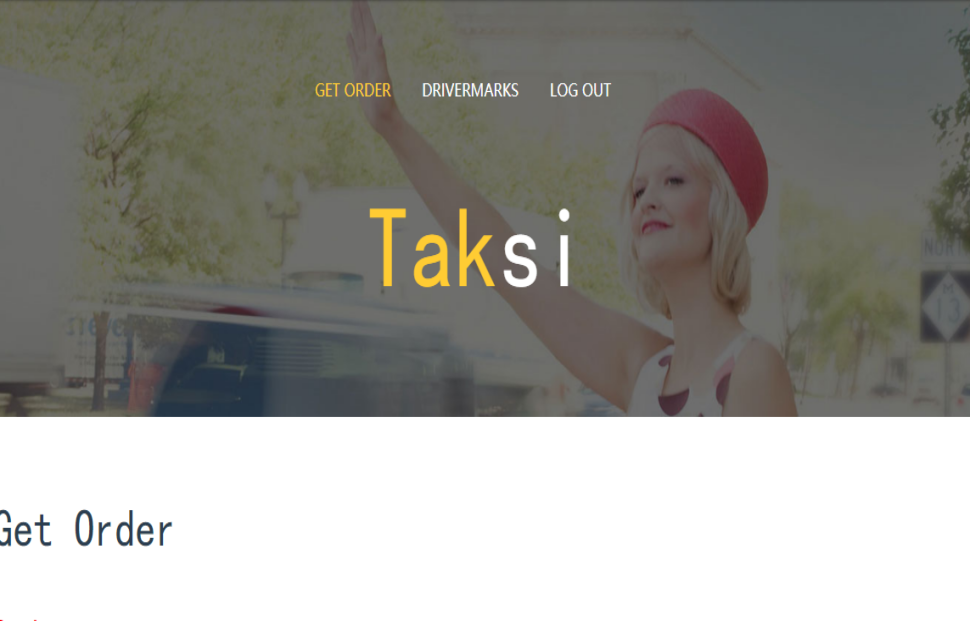


Fig 7.10

When drivers succeed to login, their home page like this. There are three functions for drivers: get order, driver marks and log out. The drivers could also choose the service by click on menu. That menu works by the follow design.

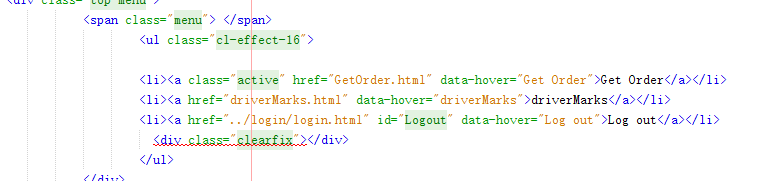


Fig 7.11

## 7.4 Customer location

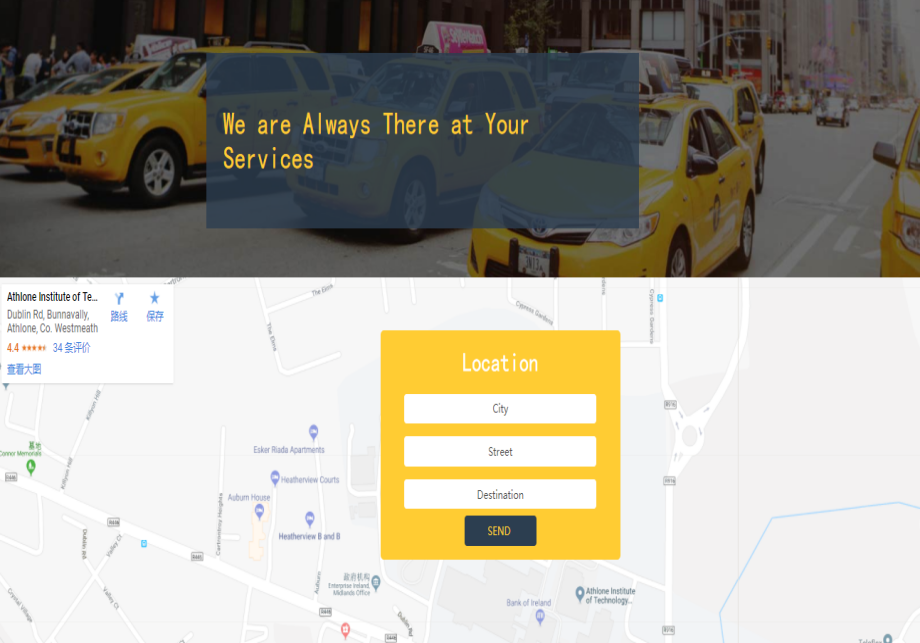


Fig 7.12

The above Screenshot is the function for customer location. In this page, customer can input where do that want to depart(the city and street), and where are their destination. At first I want to make a google map here and the map will follow the place that customers inputted, but I did not have enough time to do it so I could only make the map as a background.

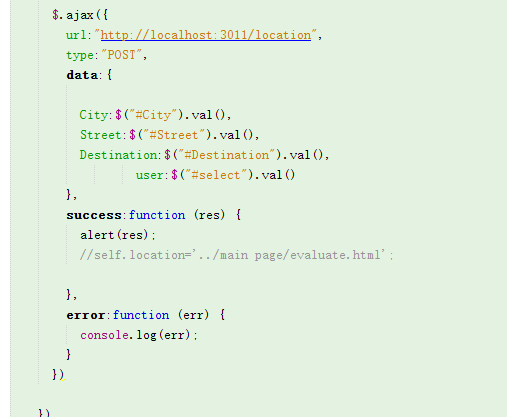


Fig 7.13

This is the code in location html page, I made an ajax method to get the data from the forms and create a port 3011.

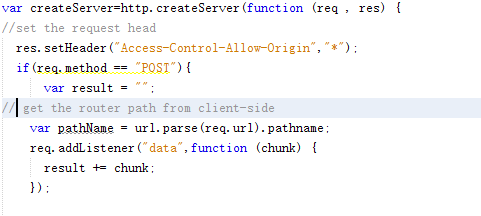


Fig 7.14

Then the above figure show a javascript file names location.js on the server-side, first I created server and got the data by defining a attribute result. And on the following figure I transferred the data to a query-string type, in this type all the data will be stored as a query format in a object, user. These steps of connection the database and get data from front-end are almost same in the whole application, so I will not introduce them in details for the next functions.

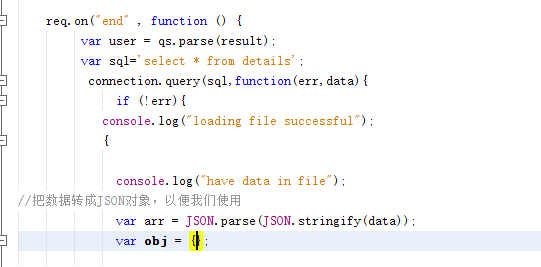


Fig 7.15

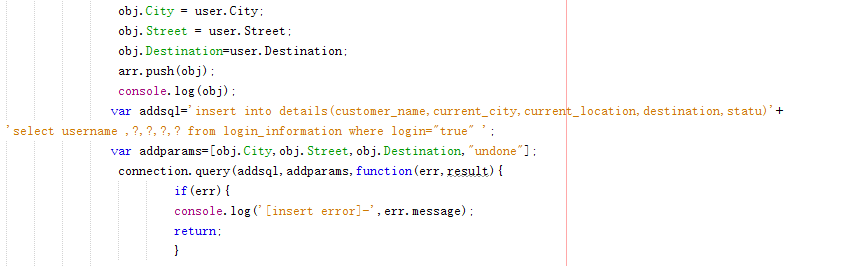


Fig 7.16

Then create a string array obj to get the data , those data will be stored in obj as the query format and you can get any data you want by tag name. Then using the SQL statement to insert the data into database, here I used the perprocessing in the SQL statement.

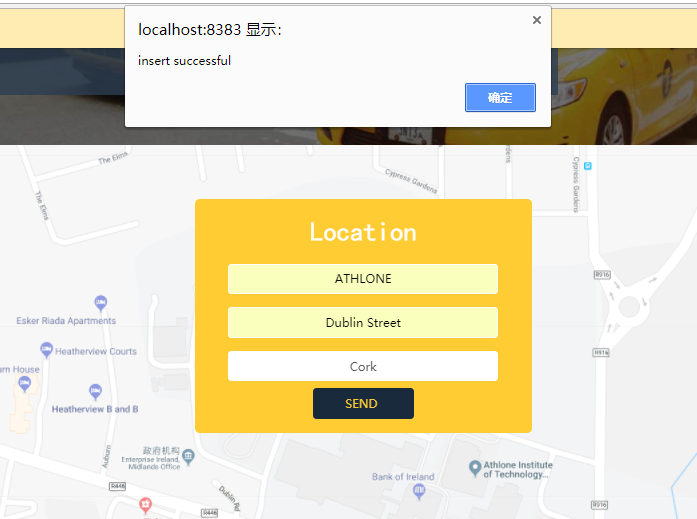


Fig 7.17

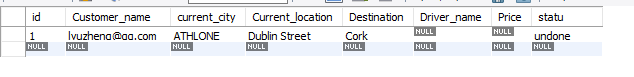


Fig 7.18

After a customer operated the location function and inputted his location, the result will like this. One customer could only take one order at the same time before their service had been finished.

## 7.5 Driver get order

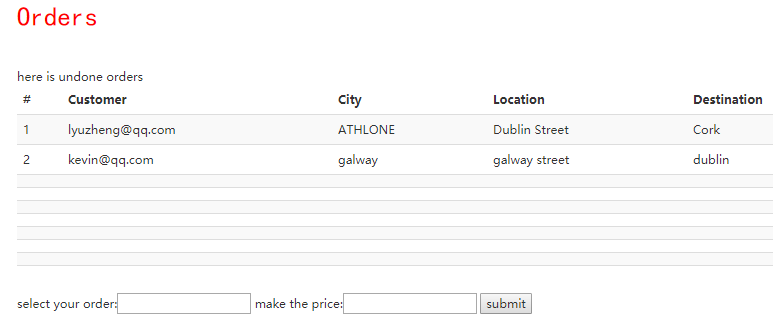


Fig 7.19

The above figure is the function for drivers to get orders. If the customers had asked for servers, their order will be shown on the drivers’ getting order page. Drivers could choose what order they want by in put the customer’s username in the “select your order” field. And if drivers selected a not existed order it will warn them to input a right customer username. After they selected the order, driver should input the price for service, the price should be only double value and have two decimal, or it will also remind the driver to input the right price format.

In this function there are two main step. First one is getting the existed and unfinished customers’ order from database, the second one is sending the drivers’ information and price to database after drivers selected the order.

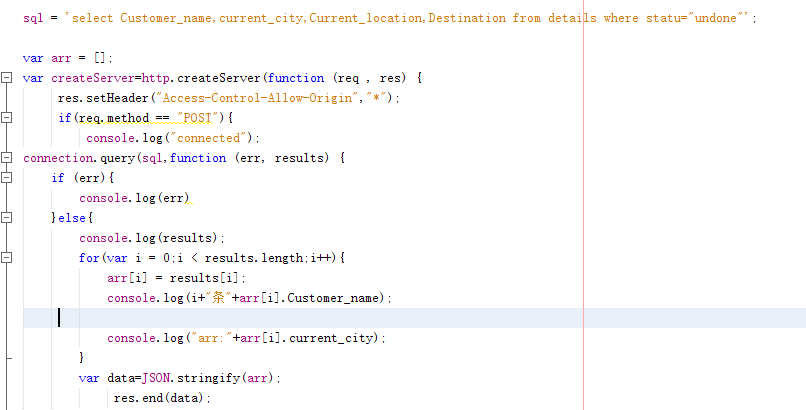


Fig 7.20

For the first step, the above figure is for a javascript file for server-side and it’s about how to get the orders from database. First it using a SQL statement to get unfinished customers’ orders, those data is in a json format and I should use an array to store it by each column. Then the data should be send to front-end as a response from server-side, but the response data must be string type should I should transfer it to string type.



Fig 7.21

Then on the front-side, the ajax() method will get the data responded from order.js. Here I get the data and transfer it to json type, then created a table and named the fields by tag name of data.

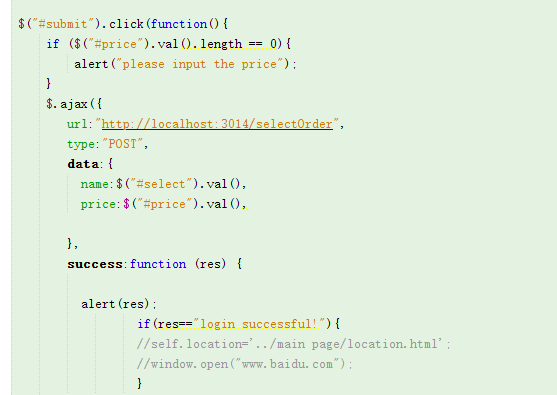


Fig 7.22

Then for the second step, I used an ajax() method to get the drivers’ input on the get order page. 

Here I made two sql statement, the first one is to update the order details in database, add the drivers’ name and price. The second one is to verify whether the inputted username has unfinished order.

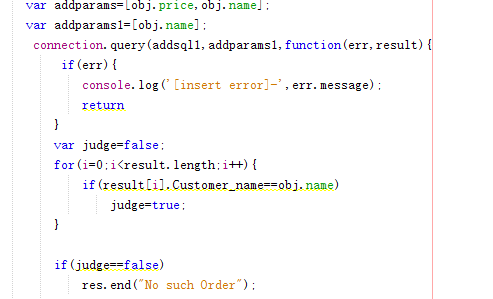


Fig 7.23

After a driver selected an order, the page will shows as follow figure. After an order been selected, after restarted the tomcat server it will not be shown on this page.

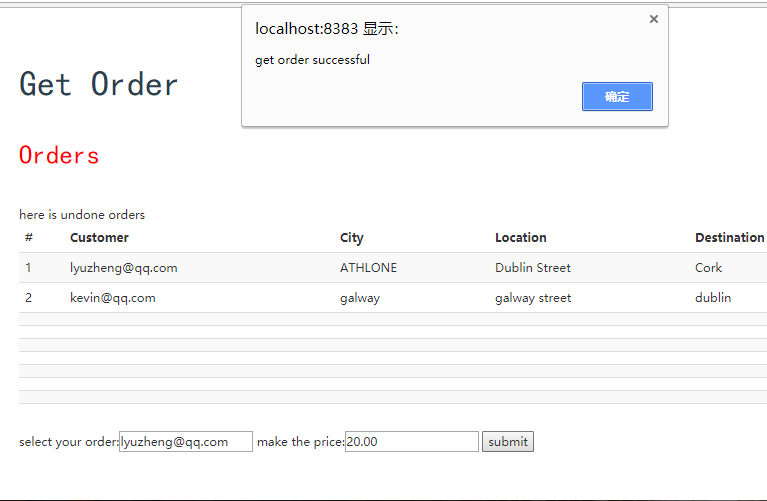


Fig 7.24

## 7.6 Customer confirm order

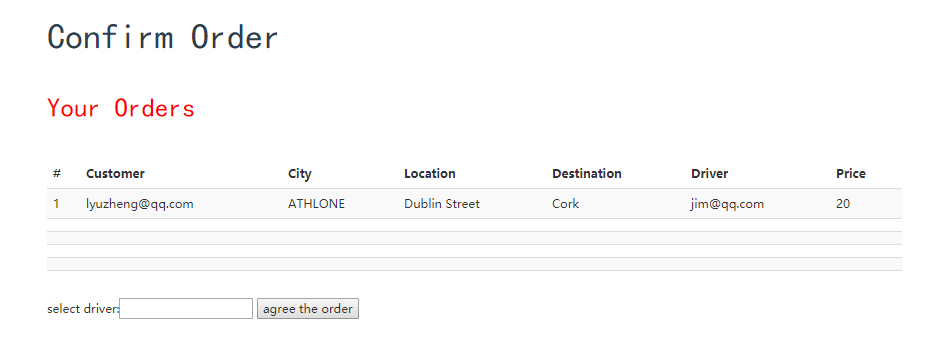


Fig 7.25

The above figure shows the page for customers to confirm the order. After a driver select the order, customers could be able to confirm if they agree with the order and want to start the service in the customers’ confirm order page. They just need to input the driver’s name in the “select driver” field and click the agree order button. When I was designing this function, I planned to make the customers be able to select from all the drivers that selected their order, but when I test it I found if there are two drivers selected the same order, the latest one will recover the earlier one, which means there will only be one order to confirm. I did not found out this problem earlier and do not have time to fix it. That was a pity for me.

In this function there are two main step. First one is getting the customers’ order information from database, the second one is making the order status to be finished after customer confirmed the order.



Fig 7.26

For the first step, the above figure is for a javascript file named finishedorder for server-side and it’s about how to get the customers’ orders from database. First it using a SQL statement to get unfinished and being selected customers’ orders, those data is in a json format and I should use an array to store it by each column. Then the data should be send to front-end as a response from server-side, but the response data must be string type should I should transfer it to string type.



Fig 7.27

Then on the front-side, the ajax() method will get the data responded from finishedorder.js. Here I get the data and transfer it to json type, then created a table and named the fields by tag name of data.



Fig 7.28

Then for the second step, I used an ajax() method to get the customers’ input on the finishedorder page. The data will only be the driver’s name selected by customer.

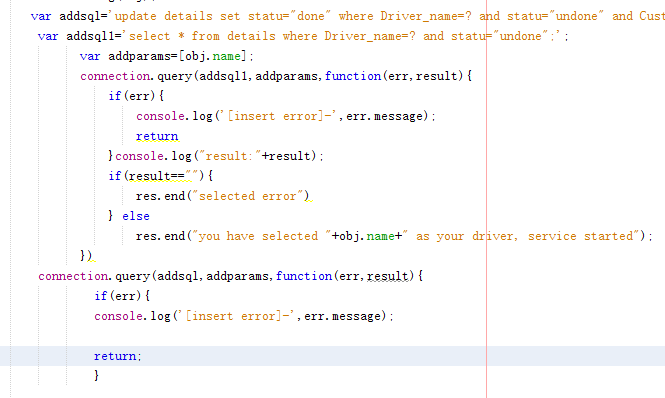


Fig 7.29

Then in the server-side, first I created two SQL statements, the first one is to update the order status from “unfinished” to “finished”, the second one is to verify the customer inputted the right driver’s username. Then execute the SQL statement, the process of select the driver and confirm the order will be finished.

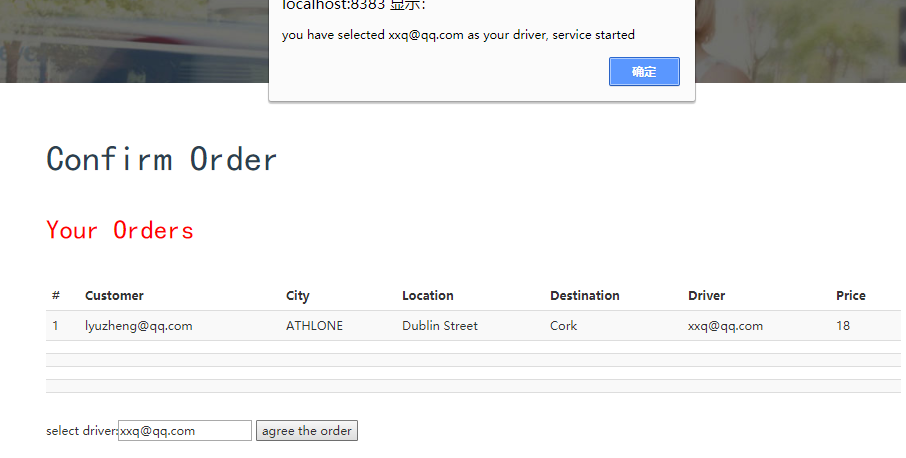


Fig 7.30

The above figure shows the success operation of confirm the order. After the process, the order will finish and server will start.

## 7.7 Evaluate the service

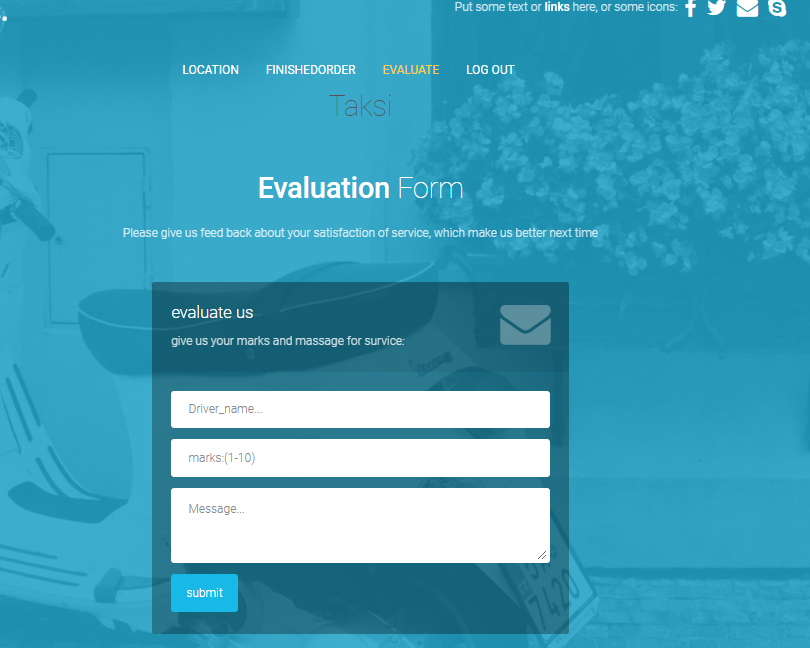


Fig 7.31

This is the function after the service has been finished. As a customer he should have the ability to give marks for services and express some advice or complain. In this function there are three text field to input: the driver’s username serviced him, the marks for the service and his advice or complain for this service, and then submit it. This evaluation will be shown on the homepage of drivers’ and will be referenced to the drivers’ Year-end assessment.

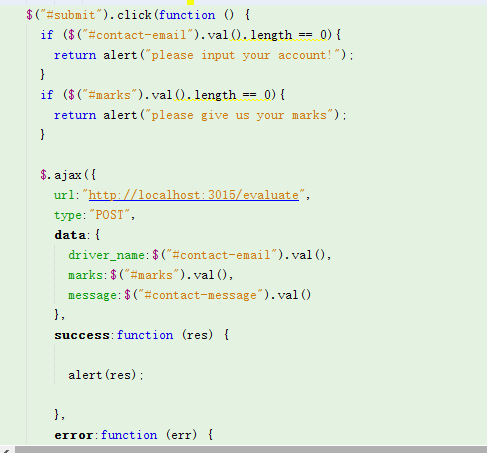


Fig 7.32

On the front-end, the inputted information will be get by the ajax() method and send to the server-side as a request, there are three attributes: driver\_name, marks and message.

Fig 7.33

I had created a table to store the information for evaluation. On the server-side, there is a javascript file named “evaluate.js”, after connected to the database and get the data from front-end , I created two SQL statements, the first one is to insert the evaluation information to database, the second one is to verify whether customers’ inputted the right drivers’ username. Then execute the statements and send the response to front-side.

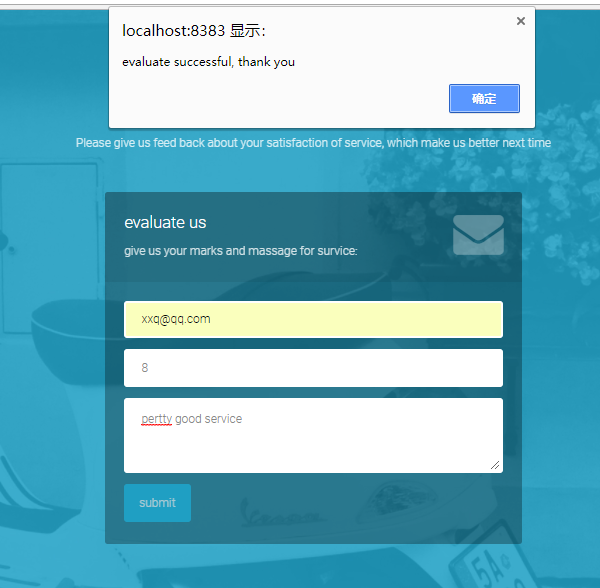


Fig 7.34

This is the result after successfully evaluated the drivers’ service.

## 7.8 Drivers’ Evaluation and marks



Fig 7.35

After the customers have been evaluate the drivers, the drivers’ should be able to see those evaluations which will help them to improve the service in the future. So all the evaluations of their past services will show on this page, there will be the evaluated customers’ username, drivers’ username, the marks and evaluations for that services.

The following code is a javascript file in server-side named “driverMarks.js”, after connected to the database there is a SQL statement to search all the evaluation for drivers who logged in, then use an array list to store those data and send them to front-end.

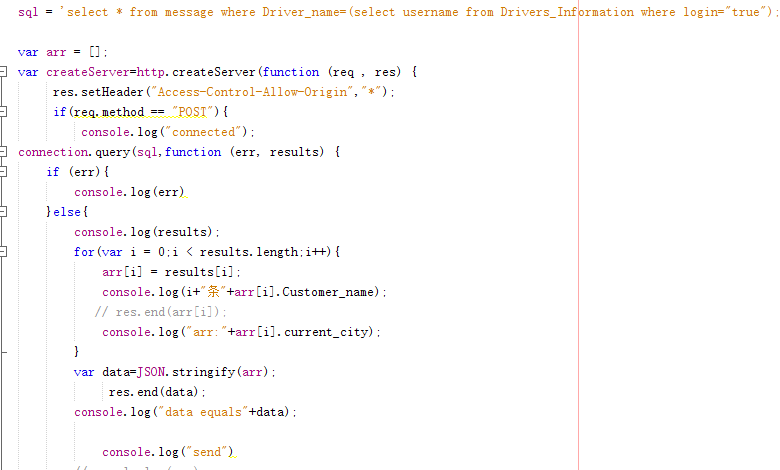


Fig 7.36

Fig 7.37

Finally on the front-end the ajax() method will get the data responded from driverMarks.js. Here I get the data and transfer it to json type, then created a table and named the fields by tag name of data.

## 7.9 Log out

On both menu there are the function log out, this function allowed the customers and drivers are able to switch their accounts. This application only allow one customer and one driver login at the same time. When customers and drivers had logged in, there is one field on customers’ and drivers’ login information named ”login” , that value will become true, so the application will know who is using the service and store the data in his column. And once they logged out that value will become false, so others would not be able to get their information any more. After logged out the page will transfer to the log in page.

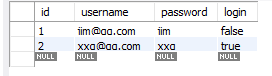


Fig 7.38

All the html pages have the following code to achieve log out. When clicking the log out link, the listening port 3012 will get the request.

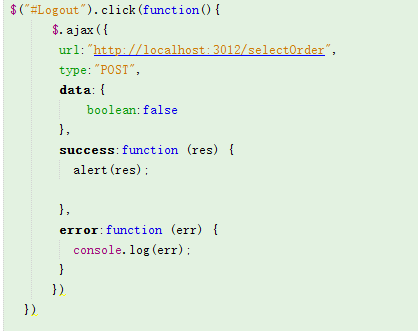
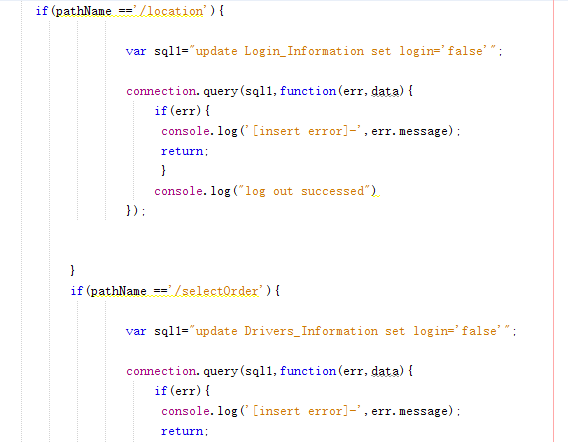


Fig 7.39

On the server-side, first it will get the path name from front-end. Then it will judge where is the log out request from, customer or driver, then execute the SQL statement to make the login value to false so that the log out process will be done.

Fig 7.40

# **Database**

In this application, there are many data to store and update between server-side and front-end, so that means we must use database to store those data. I create a database named “username”, and there are four tables in this database: “login\_information”, “Drivers\_Information”, “details” and “message”.

login\_information: This table is used to store customers’ login information, here is the code to create this table.

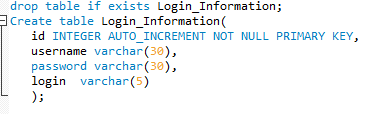


Fig 8.1

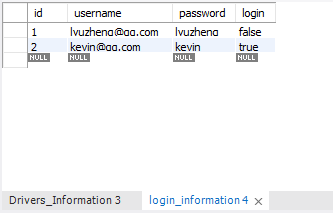


Fig 8.2

The id is the customers’ id, it grows automatically, then the username and password will be insert when customers are signing up. The login is used to judge whether a customer has been logged in.

Drivers\_Information: This table is used to store drivers’ login information, here is the code to create this table.

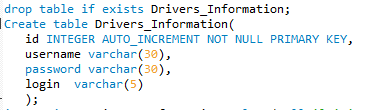


Fig 8.3

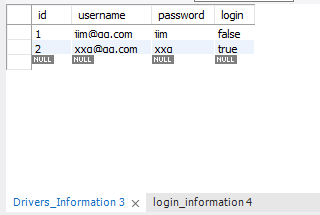


Fig 8.4

The id is the drivers’ id, it grows automatically, then the username and password will be insert when drivers are signing up. The login is used to judge whether a drivers has been logged in.

Details: This table is to store the order information, here is the code to create this table.

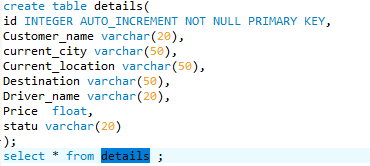


Fig 8.5

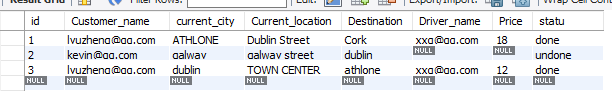


Fig 8.6

If an customer has taken a order, the information for “customer\_name”, “current\_city”, “current\_location” and “destination” will be filled and status is undone. If a driver has selected that order, the driver\_name and price will be filled and status is undone. If the customer has confirmed that order, all the information will be filled and status will become done.

Message: This table is used to store the evaluation to service, here is the code to create this table.

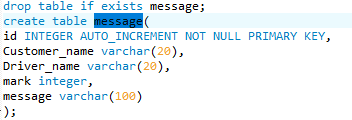


Fig 8.7

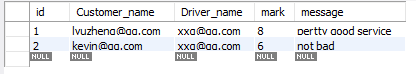


Fig 8.8

The fields include message\_id( grows automatically), customers’ name who gave the evaluation, drivers’ name who did the service, the marks for that service and the evaluation message.

In this application, users do not have to operate database directly, all the database execution is done by the operation on front-end and execution on service-side

# Testing and Evaluation

After I developed all the functional parts, I started to test and integrate. This part is hard. First of all, because I didn't perform careful functional tests after the development was completed, some parts showed errors that I didn't consider, which made me have to re-test the function. Secondly, some servers appear to have conflicting execution during the integration process. If individual servers are put together, some functions will fail. I only tried to find out the reason after testing the function for many times, but I didn't know why it would lead to server conflicts. So in the end I could only run them separately. Then, during the integration phase, there have been cases where the entire application has been failed due to a change in the code. Fortunately, I had backed up the files before making some important code changes, so I didn't spend much time on this issue. This also let me know the importance of file backup. On the whole, although the integration and testing part did not go very smoothly, it was still completed without any risk.

## **9.1 Test cases:**

Login:

Test Identifier: 001

Test Objective: To log into homepage using username and password.

Input(s): username + password

Expected Output: Fail (no such user exist sign up)

Test Identifier: 002

Test Objective: To sign up an account using username and password.

Input(s): username + password

Expected Output: Pass(successfully sign up)

Test Identifier: 003

Test Objective: To sign up an account using username and password.

Input(s): username + password

Expected Output: Fail (existed account)

Test Identifier: 004

Test Objective: To log into homepage using username and password.

Input(s): username + password

Expected Output: Pass(successfully login)

Customer location

Test Identifier: 005

Test Objective: To generate an order and require for service

Input(s): City,Street, Destination

Expected Output: Pass

Driver select order

Test Identifier: 006

Test Objective: To select an order

Input(s): Customer, price

Expected Output: Fail(No such order)

Test Identifier: 007

Test Objective: To select an order

Input(s): Customer, price

Expected Output: Fail(Invalid price)

Test Identifier: 008

Test Objective: To select an order

Input(s): Customer, price

Expected Output: Pass

Customer confirm order

Test Identifier: 009

Test Objective: To confirm an order

Input(s): driver

Expected Output: Failed(No such driver)

Test Identifier: 010

Test Objective: To confirm an order

Input(s): driver

Expected Output: Pass

Customer evaluate service

Test Identifier: 010

Test Objective: To evaluate an service

Input(s): driver, marks, message

Expected Output: Fail(No such driver)

Test Identifier: 011

Test Objective: To evaluate an service

Input(s): driver, marks, message

Expected Output: Pass

## 9.2 Evaluation:

Testing is a very important item in software development. This is a lesson I learned in this project development. In the beginning, after each part of the project was completed, I only ran its function roughly, but did not perform a complete test. This made me suffer a big loss when I integrated the project. In the final stage of the project, the failure of the operation often occurs, but it is difficult for me to find out which part has a problem. When I started the final test, I often found that there were some areas that I had previously considered, and some parts that I could modify, but some parts I could not modify, because that will be modified from the implementation principle and I do not have so much time, For example, on the customer's "Confirm Order" page, if more than one driver chooses an order, the selected driver information will overwrite the previously selected driver information. The reason for this is because of the poor design of the design on the one hand, and the inability to test this possibility in time when it was completed early. And sometimes there is a problem with the compatibility of multiple projects. For example, when I open the server of "select Driver" and "select Order" at the same time, I will find that only one of them can display the content normally and the other is always blank. That's why I had to execute some servers separately.

# **Conclusion**

At the beginning of the project, I was planning to make a web application because almost everybody in the world can't live without the Internet nowadays, and almost everybody browses a lot of websites every day. Most of the applications are based on web pages, so I think this is a very hot technology, and I don't know enough about the structure of the front-end-backend server, so I hope to get a better understanding of the web development process through this project.

In the beginning, I didn't come up with the topic. My supervisor gave me a suggestion to do a taxi-hailing application. I investigated it and found out that taxi-hailing applications are more popular nowadays. However, almost all taxi-hailing applications are mobile applications and none of them works on the web page. Nowadays, smart phones have developed rapidly. However, some people may not be accustomed to using smart phones, but prefer to use web applications. This made me decide to develop such an application on web pages to help those people.

After my investigation I found that to implement a taxi-hailing application on the web is a bit like a train ticketing system. After the client chose the start and destination stations, he chose the time and clicked on the purchase. A transaction was completed. The same is true for taxi-hailing applications, but the difference is that the taxi-hailing applications is a two-way option. It requires not only the client but also the driver. After the customer enters the start and destination, it only produces the order and the driver also needs to accept the order and give the price. The customer also has the right to measure whether the price given by the driver is reasonable, and then decide whether or not to start the service. And after the completion of the service, the customer has the right to evaluate the quality of the service and give marks and suggestions. This is an evaluation system. This is the approximate running process and function of this taxi application.

After deciding what the function of this software is, the next step is how to implement it and what technologies should be used. On the feature homepage I decided to use HTML5 because this is a mainstream web design language, and I am also more familiar with how to design web pages using HTML5, CSS3 and JavaScript. The database I used MySQL because I have used to use this database, and it is very easy to use as an relational database. The key is how to choose the technology for the server-side. One classmate recommended to me node.js, which is a new technology that I have not contacted, and this technology is very young but has developed very fast in recent years. It makes JavaScript available to use on the server, which helping me get started soon with this technology. And I haven't studied this technology before, I'm happy to learn this new technology and it can help me get a higher project evaluation.

After started this project, I also encountered a lot of problems, such as when the user logged in at the beginning, how to make the user stay logged in and get the corresponding information has become a problem, I tried to use the cookie to store login information but it did not work. Finally, the supervisor suggested me to add a login attribute to the database to distinguish whether users had log in. This indeed made it easier to manipulate the data. There was also a case when I did not know how to get the data line by line and output it on the user's home page, but finally I solved each of these problems one by one in the constant search on the Internet. When I finished the project I was relieved and full of confidence because I did learn a lot, not only node.js, but also JavaScript and some complex queries in SQL.

Admittedly, this project does have some shortcomings. First of all, because of using the "login" attribute in the database to distinguish whether the user is logged in, this application cannot implement the simultaneous login of two people, either the client or the driver. If two people log in at the same time, duplicate orders will be generated when the order is accepted. Secondly, because of my negligence on the test, if two drivers simultaneously take orders from a customer, the driver’s information of the latest one to take orders will overwrite the driver’s information who earlier to take orders. This makes it impossible for the customer to be fair when confirming the orders. It is a pity for not solving these problems, and this reminds me that I need to be more careful and considerate the next time I develop applications.

In this project development, I think I should use two points to measure success. One is whether the project has developed the expected results, and the second is whether I have learned something in myself. I think the first point is not perfect to achieve the desired results, and the program has some shortcoming. But the second point I think I can say with certainty is that this development is a good experience for me. This is the first time I spent a long time developing a program with a newly learned technology and completed it. I feel very happy about it. This will certainly provide great help in my future software development career.

1. **Reference:**

Zh.wikipedia.org. (2018). *HTML*. [online] Available at: https://zh.wikipedia.org/wiki/HTML [Accessed 12 May 2018].

Zh.wikipedia.org. (2018). *层叠样式表*. [online] Available at: https://zh.wikipedia.org/wiki/%E5%B1%82%E5%8F%A0%E6%A0%B7%E5%BC%8F%E8%A1%A8 [Accessed 13 May 2018].

En.wikipedia.org. (2018). *Node.js*. [online] Available at: https://en.wikipedia.org/wiki/Node.js [Accessed 12 May 2018].

En.wikipedia.org. (2018). *MySQL*. [online] Available at: https://en.wikipedia.org/wiki/MySQL [Accessed 13 May 2018].

Baike.baidu.com. (2018). *netbeans\_百度百科*. [online] Available at: https://baike.baidu.com/item/netbeans/9854491?fr=aladdin [Accessed 13 May 2018].

1. [] [↑](#endnote-ref-0)
2. [] [↑](#endnote-ref-1)
3. [] [↑](#endnote-ref-2)
4. [] [↑](#endnote-ref-3)
5. [] [↑](#endnote-ref-4)