**Source Codes & Documentation**

**Team 9**

201835419 김민준

201835430 김재원

201835483 이동섭

201935149 허윤영

**1. Introduction**

**1.1 Motivation**

As students moved from the IT Bldg to the AI Engineering Bldg, students frequently came to the AI Bldg by kickboard from Gachon University Station. There were several types of shared kickboards around the school, so it was difficult to choose a kickboard. This project was carried out to tell you which company's kickboard is the cheapest or the most expensive. In addition, if you have to use a kickboard not only around the school but also elsewhere, I wanted to let you know which company's kickboard is the cheapest or most expensive so that users can save on the kickboard using fee.

**1.2 Scope of the system**

The system allows users to know the distance from origin to destination, the expected route, the cost of each keyboard company, the estimated travel time, and the lowest-cost kickboard company when compared overall.

**1.3 Objectives and success criteria of the project**

The objective of our project was to let users know which kickboard was the cheapest among many kickboard companies.

So, it was a topic to measure the expected travel distance with the origin and destination information, and these problems were successfully solved by bringing up a T-map API.

**1.4 Technical Skill**

T-map api was brought in, responsive web design was introduced, and the logic of each kickboard company's rate policy was implemented to compare rates. Spring server was deployed using AWS EC2 Instance server. AWS LoadBalancer, Certificate manager, Route 53 functionality was used for HTTPS implementation. IP address connected to dongsseop2.com domain successfully.

**2.Proposed System**

**2.1 Overview**

User wanted to find out which kickboard company's price is the cheapest and most expensive for the same route because of different plans for each shared kickboard company. The kickboard start position and end position were input from the called T map api, and the expected distance (km), expected time, and expected path were output. By identifying the fee for each kickboard company, we try to inform the user of the estimated rate at a glance by using the estimated distance (km).

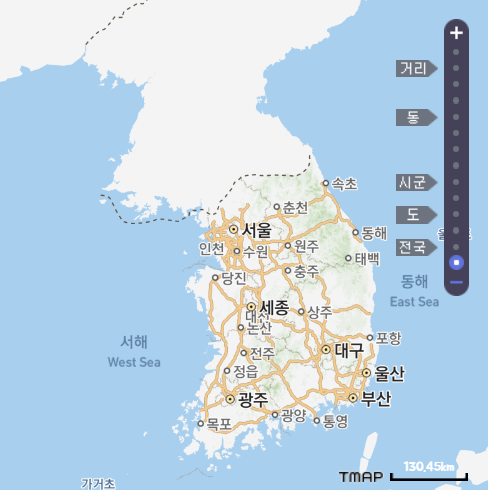
**2.2 Requirements analysis**

**Derived requirements**

1. T map api call
2. Start point or end point marker setting
3. Zoom in or out of the map
4. Web server 24 hours a day
5. HTTPS application for gps function
6. Kickboard company fee
7. Location search function
8. Expected rate accuracy
9. Responsive web
10. Result value within 5 seconds

**Requirements Critical Order & Implementable Scope**

1. T map api call & South Korea Map only



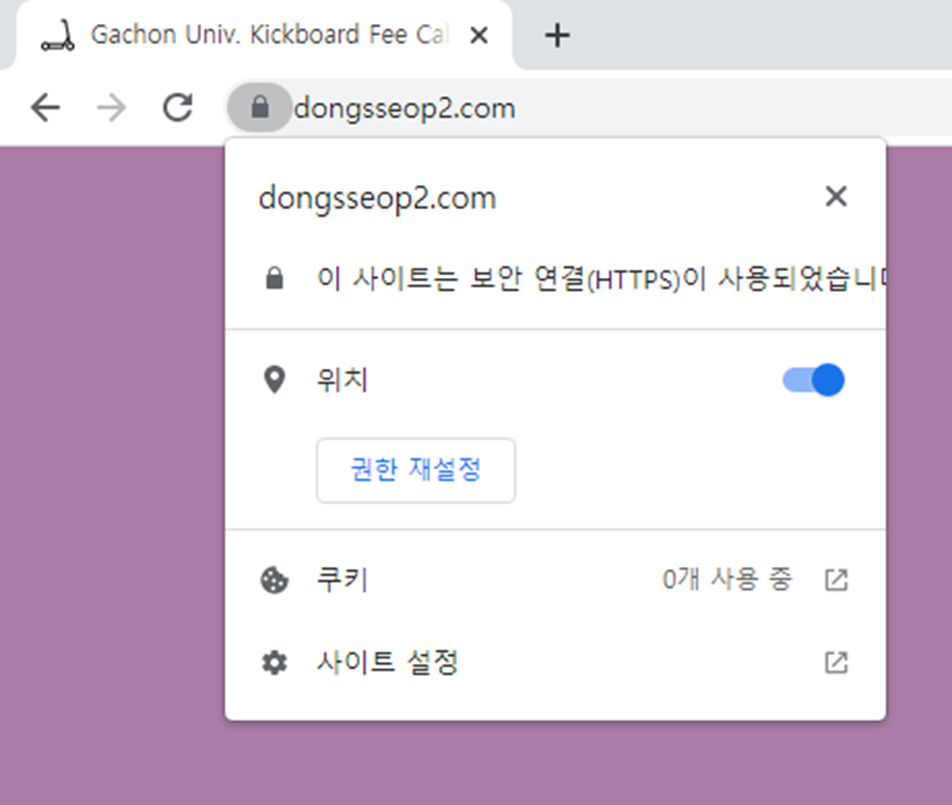
1. Start point or end point marker setting & South Korea Map only



1. Kickboard company fee & Only companies that have released kickboard plans



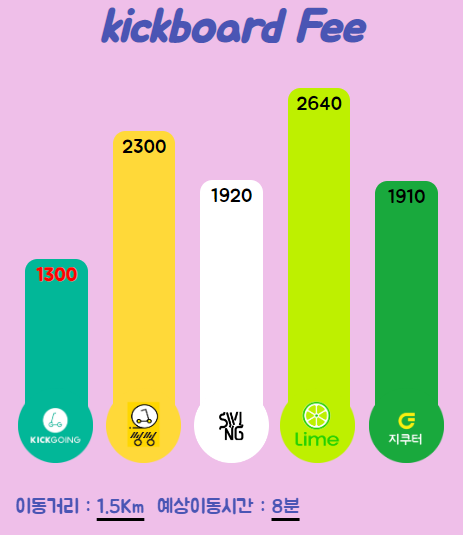
1. HTTPS application for gps function & https implementation for gps functionality (Chrome, MS Edge, Naver Whale)



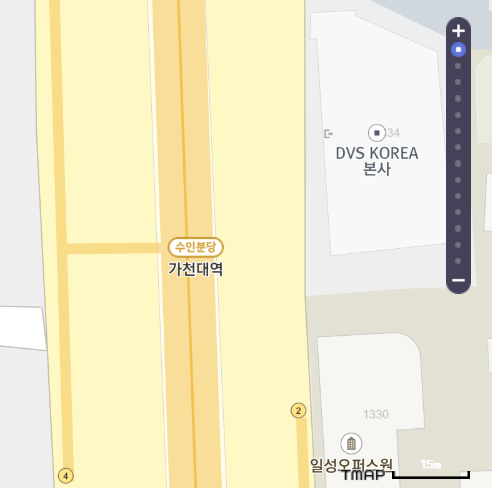
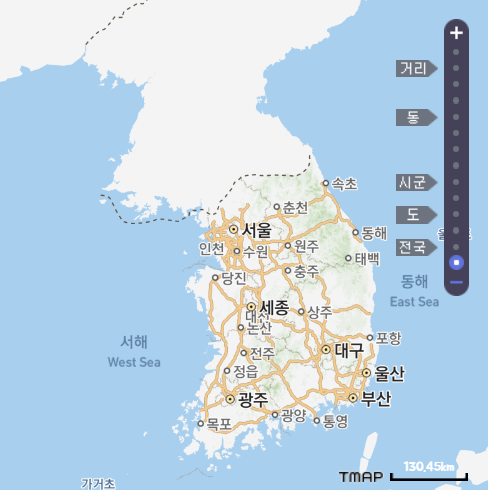
1. Location search function & South Korea only



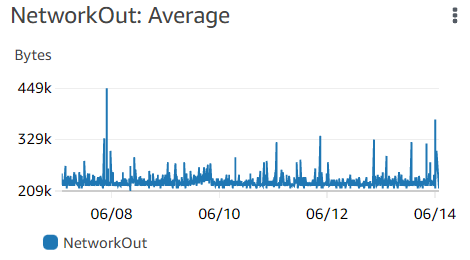
1. Expected rate accuracy & error range ± 300 won (Expected rate with average kickboard speed 12km/h)



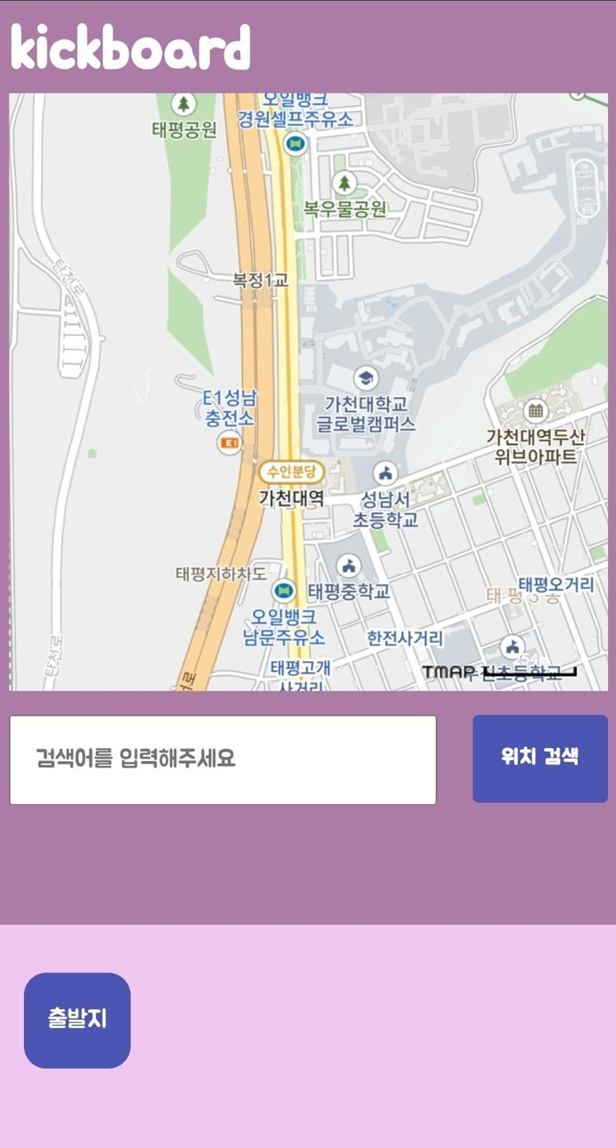
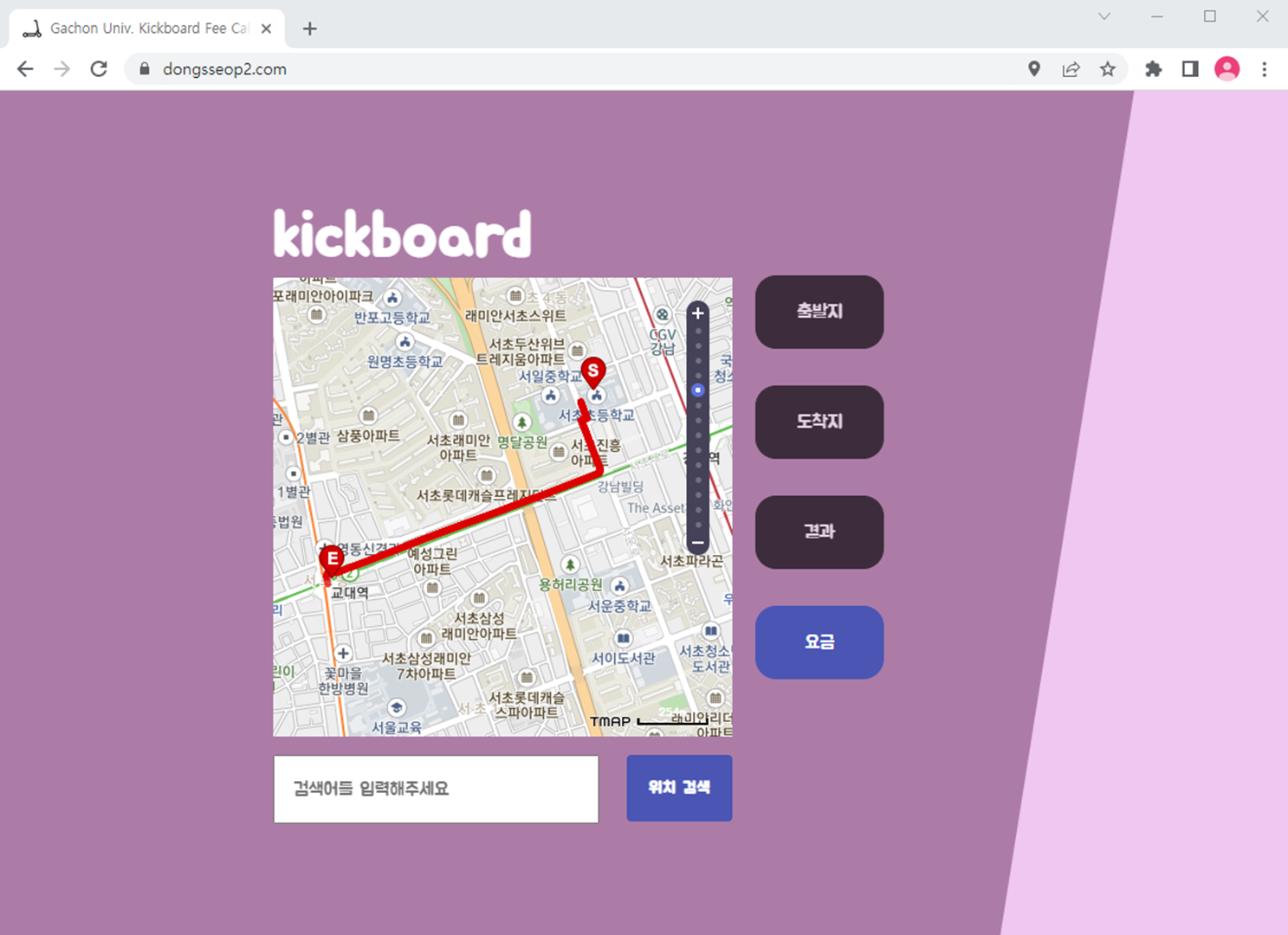
1. Zoom in or out of the map & From the map of Korea (maximum zoon-out and zoom-in)



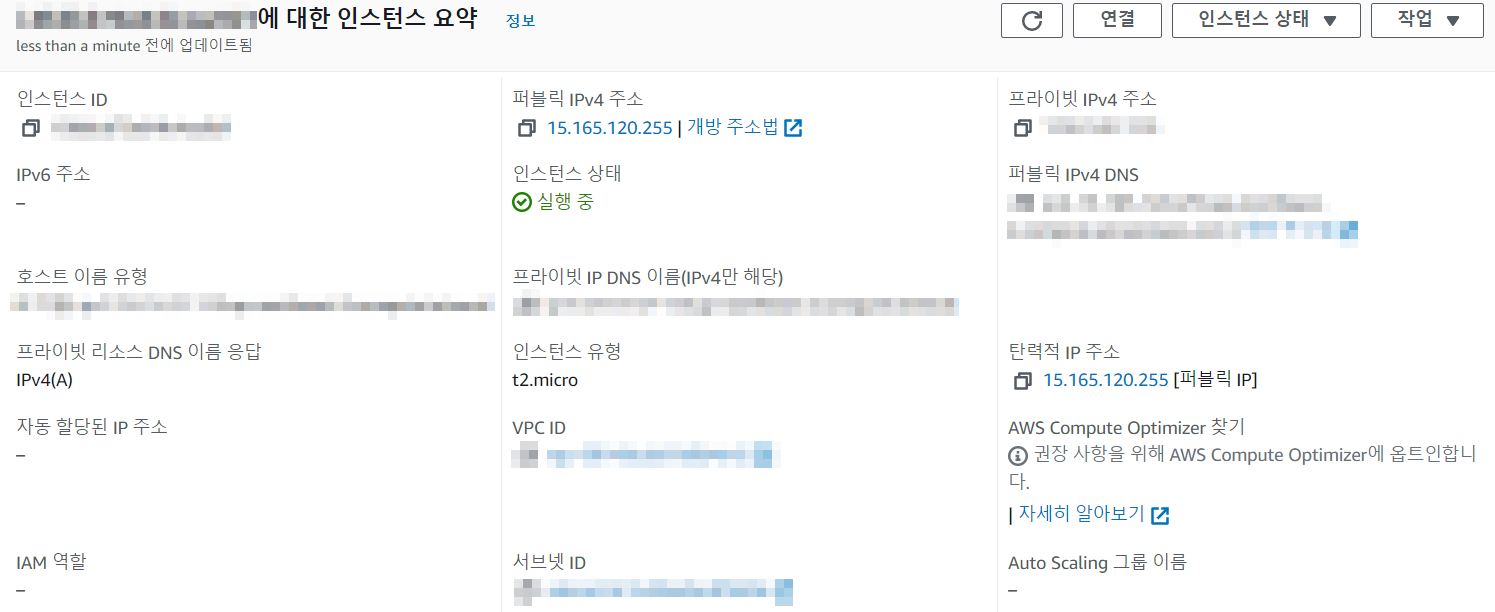
1. Web server 24 hours a day & Server 24 hours an hour with AWS EC2 Instance



1. Responsive web & The size of the screen changes fluidly on mobile and desktop



1. Result value within 5 seconds & Determined by the AWS EC2 Instance server type (t2.micro)



**Functional requirements**

1. T map api call

2. Start point or end point marker setting

3. Zoom in or out of the map

4. Web server 24 hours a day

5. Responsive web

6. Location search function

7. Kickboard company fee

**Non-Functional requirements**

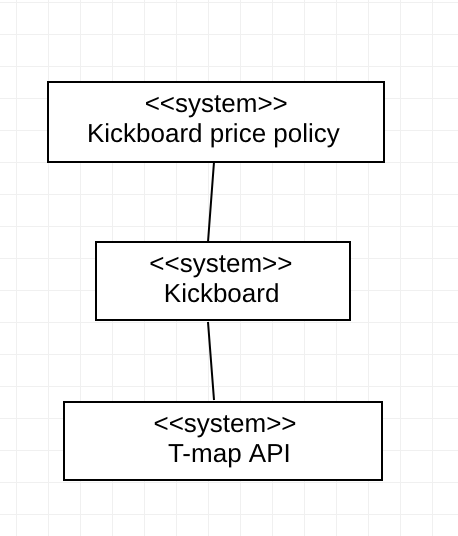
1. HTTPS application for gps function (Security)

2. Result value within 5 seconds (Quality)

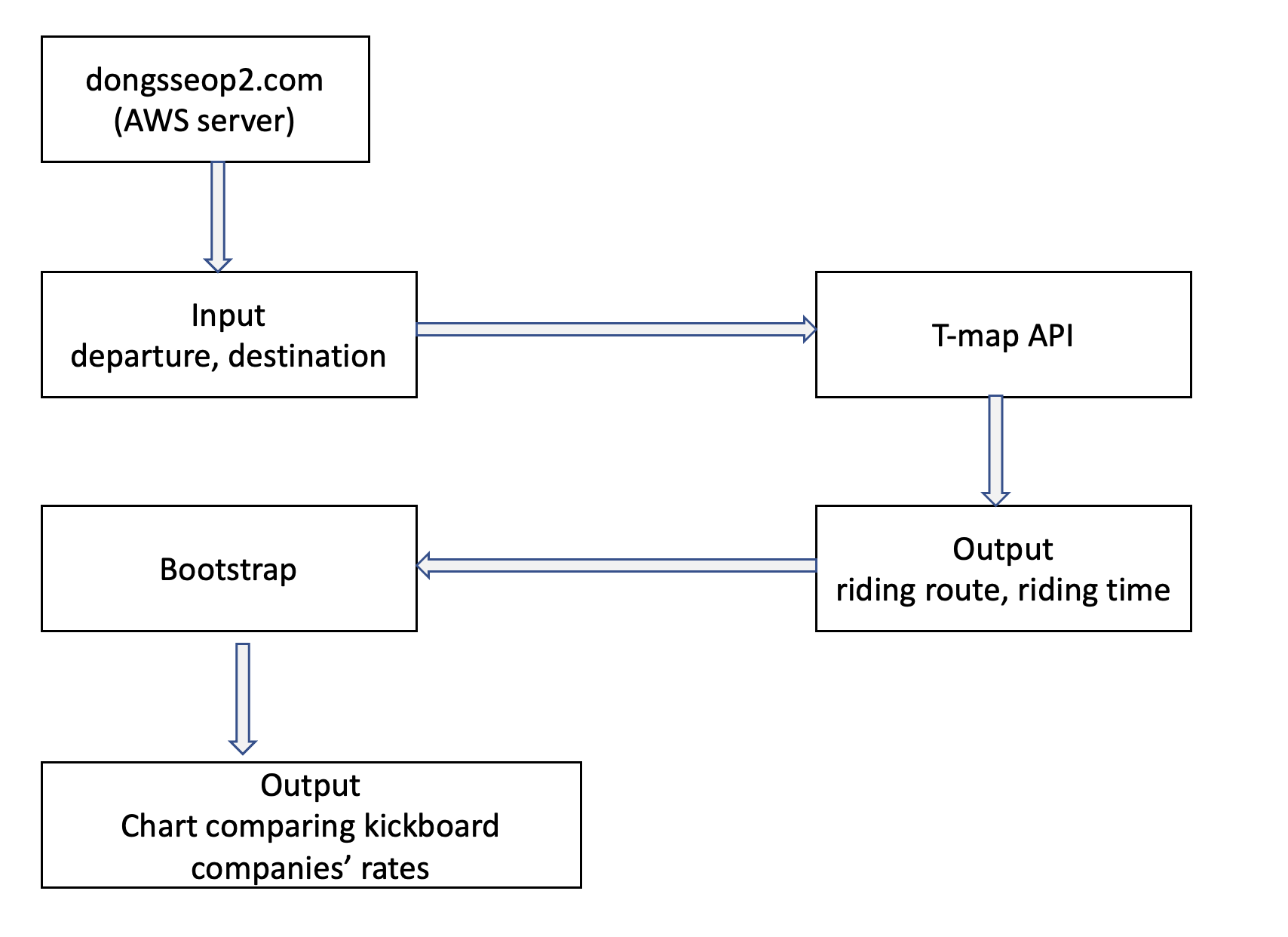
3. Expected rate accuracy (Performance)

**2.3 Sysyem modeling**

**Context modeling**

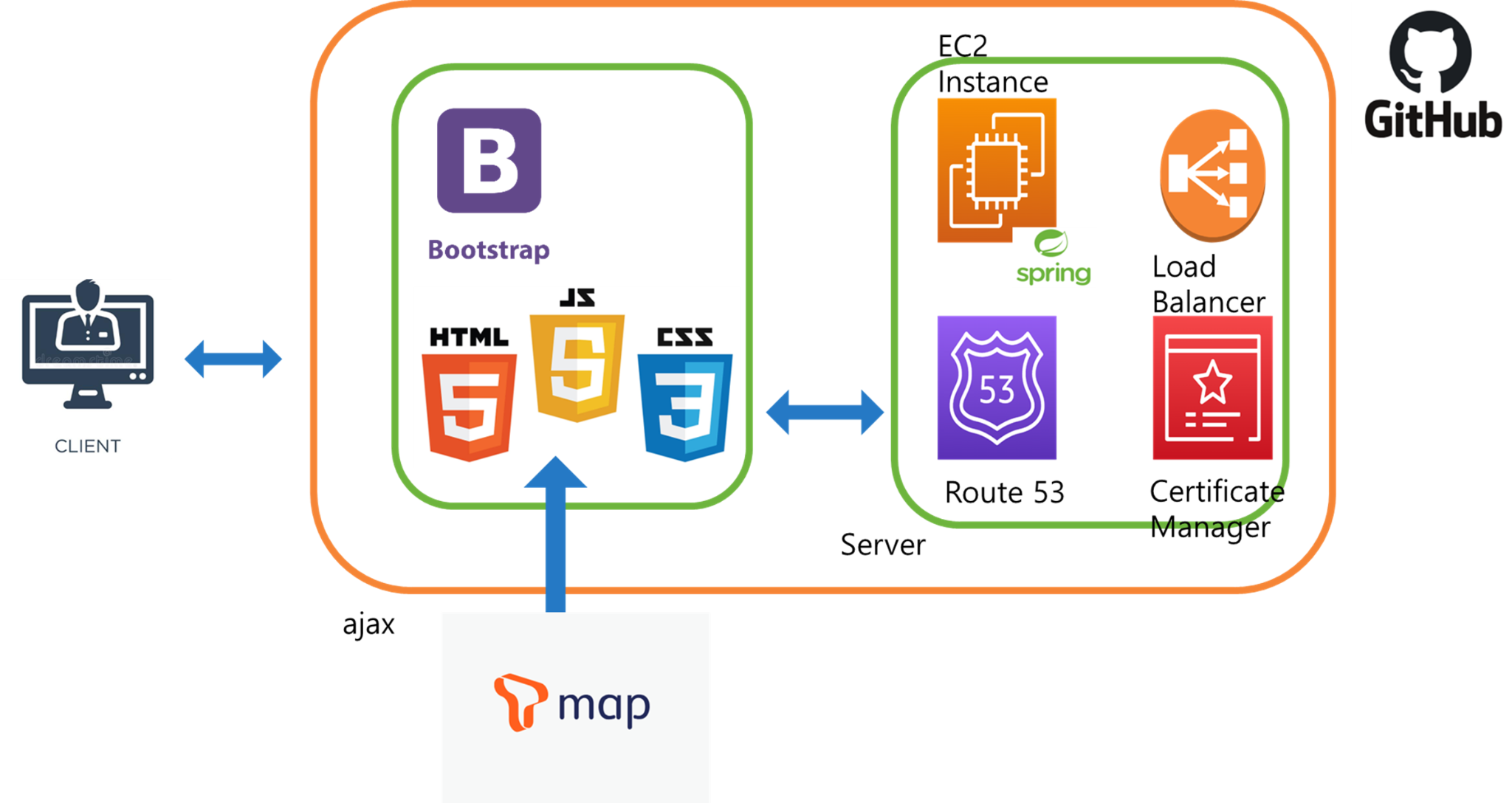


**Behavioral modeling**

****

**2.4 Architectural design**

The frontend used bootstrap, html, JavaScript, and css. Spring was used as the backend server and the spring server was deployed on aws ec2 instances. Additionally, the Load Balancer, Certificate Manager, and Route 53 were used to use https for gps usage. As a collaboration tool, team members collaborated using GitHub.



**3.Validation**

**User Validation**

Before using the shared kickboard, we used our program to verify that the actual fee, distance andusage time match our program expectations.

**Unit test**

By using the Junit5 library, We performed unit tests on 16 methods such as kickboard charge logic for each company, late-night time judgment logic, and late-night additional charge logic.

**4.Glossary**

**AWS EC2 Instance:** Amazon Elastic Compute Cloud (EC2) is a part of Amazon.com's cloud-computing platform, Amazon Web Services (AWS), that allows users to rent virtual computers on which to run their own computer applications.[[1]](#endnote-1)

**Load Balancer:** load balancing refers to the process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient.[[2]](#endnote-2)

**Route 53:** Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. You can use Route 53 to perform three main functions in any combination: domain registration, DNS routing, and health checking.

Responsive Web.[[3]](#endnote-3)

**Junit5:** JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.[[4]](#endnote-4)

**Certificate Manager:** A web service for provisioning, managing, and deploying Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services.

Behavioral Model.[[5]](#endnote-5)

**Context Model:** A context model (or context modeling) defines how context data are structured and maintained (It plays a key role in supporting efficient context management).[[6]](#endnote-6)

**Functional Requirements:** Functional requirement defines a function of a system or its component, where a function is described as a specification of behavior between inputs and outputs.[[7]](#endnote-7)

**Non-Functional Requirements:** Non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours.[[8]](#endnote-8)

**HTTPS:** Hypertext Transfer Protocol Secure (HTTPS) is an extension of the Hypertext Transfer Protocol (HTTP). It is used for secure communication over a computer network, and is widely used on the Internet.[[9]](#endnote-9)

**5.Review**

The kickboard fee website was created to solve the inconvenience we feel. It is not just a project deliverables, but we made it so that our team members and people can actually use it. So We deployed on the AWS EC2 Instance server so that people can access the website. When creating the program, a lot of consideration was given from the point of view of the user who actually uses it. It has been made into a responsive web that can be accessed regardless of web or mobile. There is also a function that shows a map where the user is located using gps. Because we use personal location information, enhanced security by using the https protocol. Through the techniques learned in software engineering, We were able to develop the program and make it more complete.

**6. Member’s participation rate**

김민준 25%

김재원 25%

이동섭 25%

허윤영 25%

**Source Codes**

Please visit <https://github.com/dongseoplee/Kickboard-Fee-Calculator>

**References**

1. https://en.wikipedia.org/wiki/Amazon\_Elastic\_Compute\_Cloud [↑](#endnote-ref-1)
2. https://en.wikipedia.org/wiki/Load\_balancing\_(computing) [↑](#endnote-ref-2)
3. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/Welcome.html [↑](#endnote-ref-3)
4. https://en.wikipedia.org/wiki/JUnit [↑](#endnote-ref-4)
5. https://docs.aws.amazon.com/general/latest/gr/glos-chap.html#acm [↑](#endnote-ref-5)
6. https://en.wikipedia.org/wiki/Context\_model [↑](#endnote-ref-6)
7. https://en.wikipedia.org/wiki/Functional\_requirement [↑](#endnote-ref-7)
8. https://en.wikipedia.org/wiki/Non-functional\_requirement [↑](#endnote-ref-8)
9. https://en.wikipedia.org/wiki/HTTPS [↑](#endnote-ref-9)