



탄소 배출량 계산 웹 서비스 개발

Team 9



목차



01

GOALS



Web Service Development

02

OVERVIEW



Development Tools
System Architecture

03

UI/UX



UI/UX Scenarios

04

TEST &
CHALLENGE



Result of stress test
& challenge

05

GREEN
PATTERN



How to Find Green Pattern

01

CALCULATE

Java 코드의 탄소 배출량을 계산하는
웹 서비스 개발
기존 탄소 배출량 계산 사이트에 없는
코드 컴파일 기능 구현

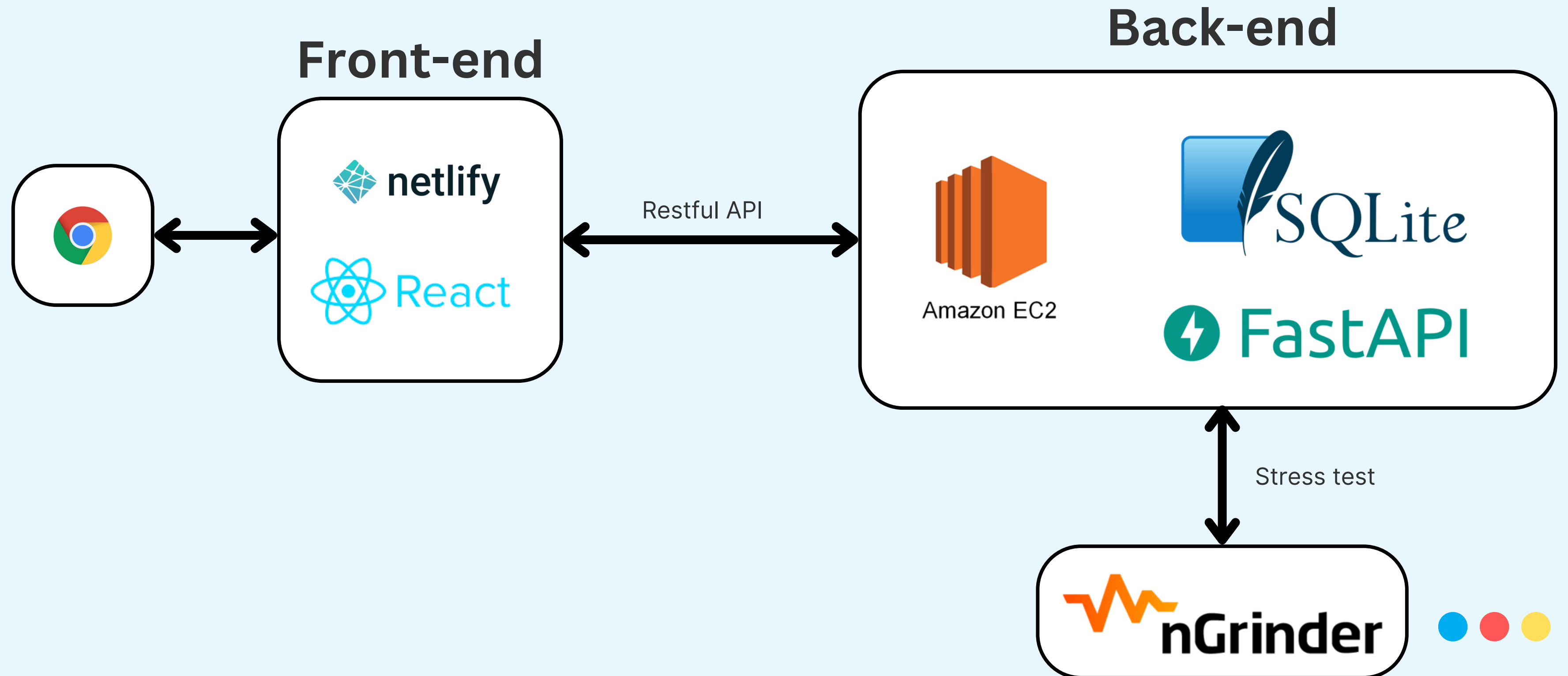
Goals

02

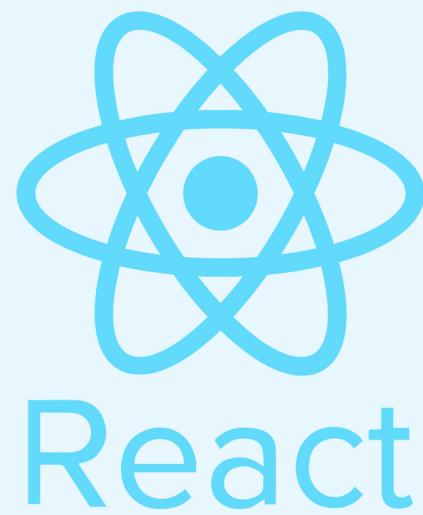
COMPAIR

사용자가 입력한 코드를 기록하여
탄소 배출량을 비교할 수 있는 기능 구현
최대 5개의 코드 비교 가능

Overview

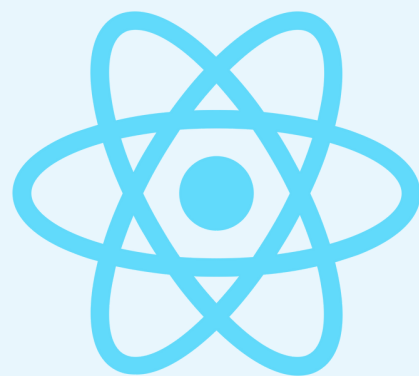


Used Tools



Used Tools

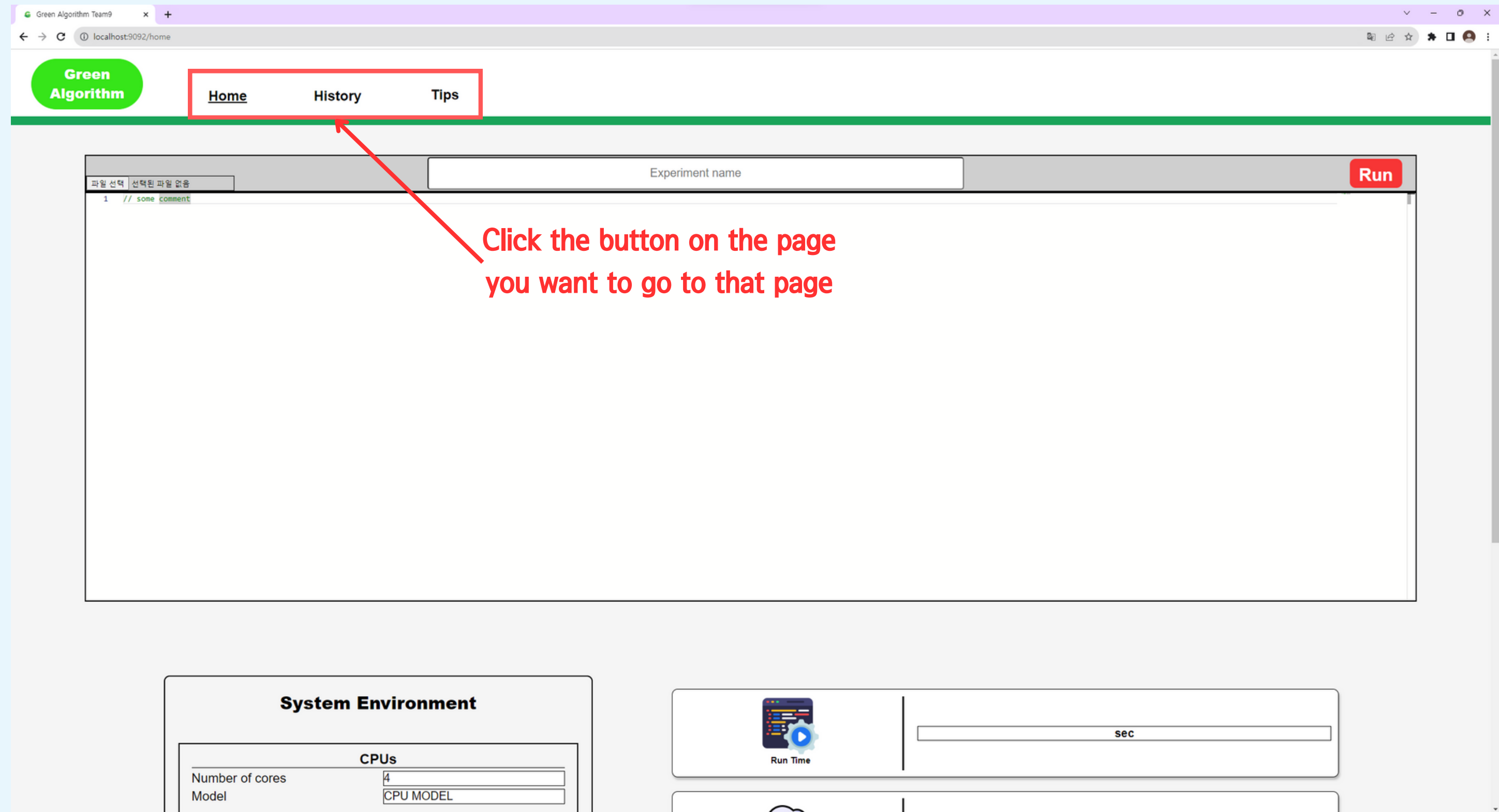
Open sources



React



UI/UX Scenarios



HOME PAGE

UI/UX Scenarios

Carbon Emissions Calculation System Environmental Information

Comparison of Carbon Emissions

System Environment

CPU

Number of cores

1

Model

Intel Xeon E5-2686 v4

Memory

Memory available

1GB

Cloud computing


Cloud computing

Amazon Web Services

Location


Nation

Korea




Run Time

sec




Carbon Footprint

g CO2e




Carbon sequestration

tree-month



in a passenger car

km

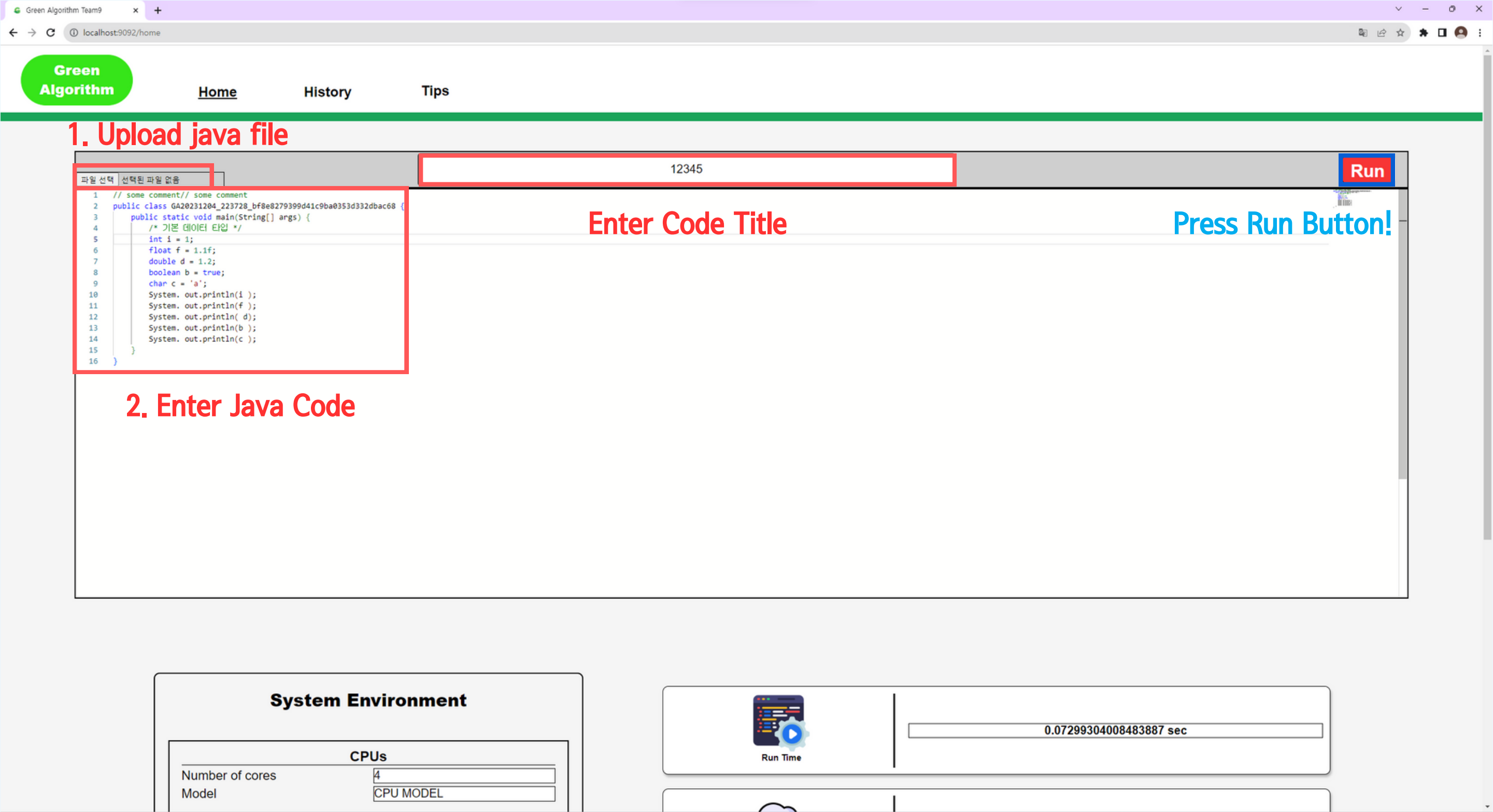


of a flight Korea-Japan

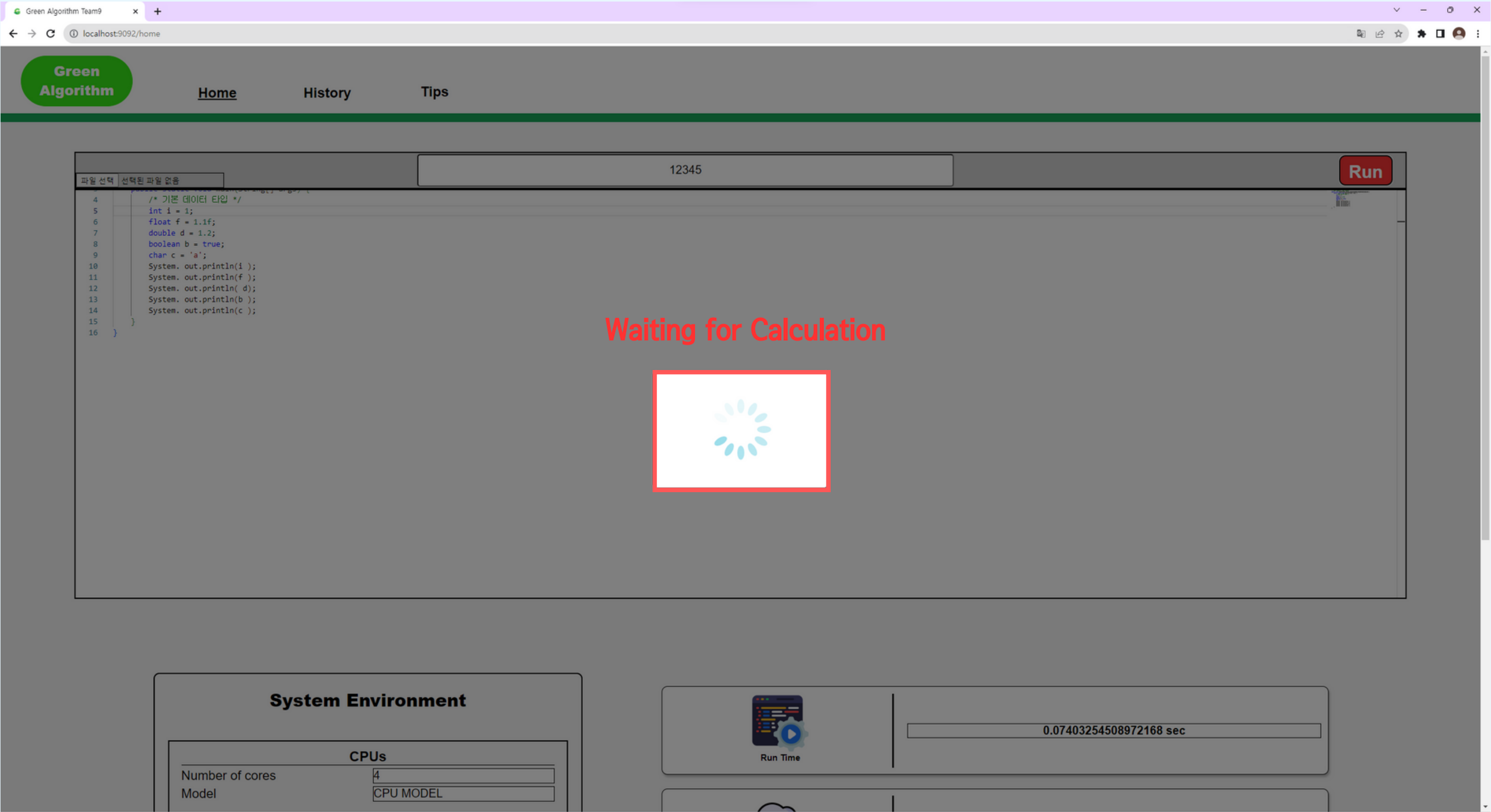
%

HOME PAGE

UI/UX Scenarios



UI/UX Scenarios



HOME PAGE

UI/UX Scenarios

System Environment

CPU

Number of cores

1

Model

Intel Xeon E5-2686 v4

Memory

Memory available

1GB

Cloud computing

Cloud computing


Amazon Web Services

Location

Nation


Korea

Value changed according to calculated code




Run Time

0.07472586631774902 sec




Carbon Footprint

1.69090984555807 g CO2e




Carbon sequestration

0.0018446289224269869 tree-month



in a passenger car

0.009662341974617551 km

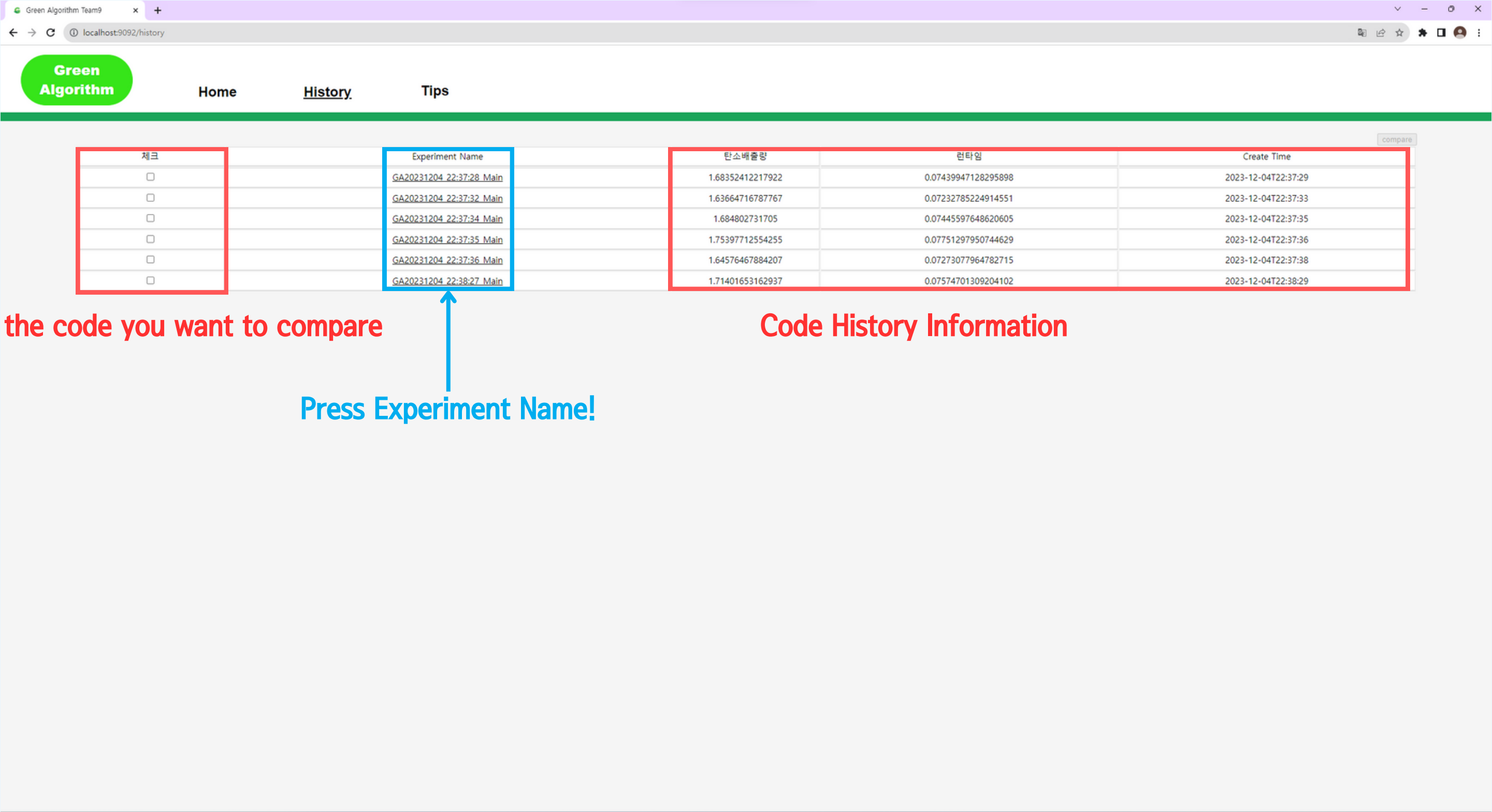


of a flight Korea-Japan

0.0000029665085009790725 %

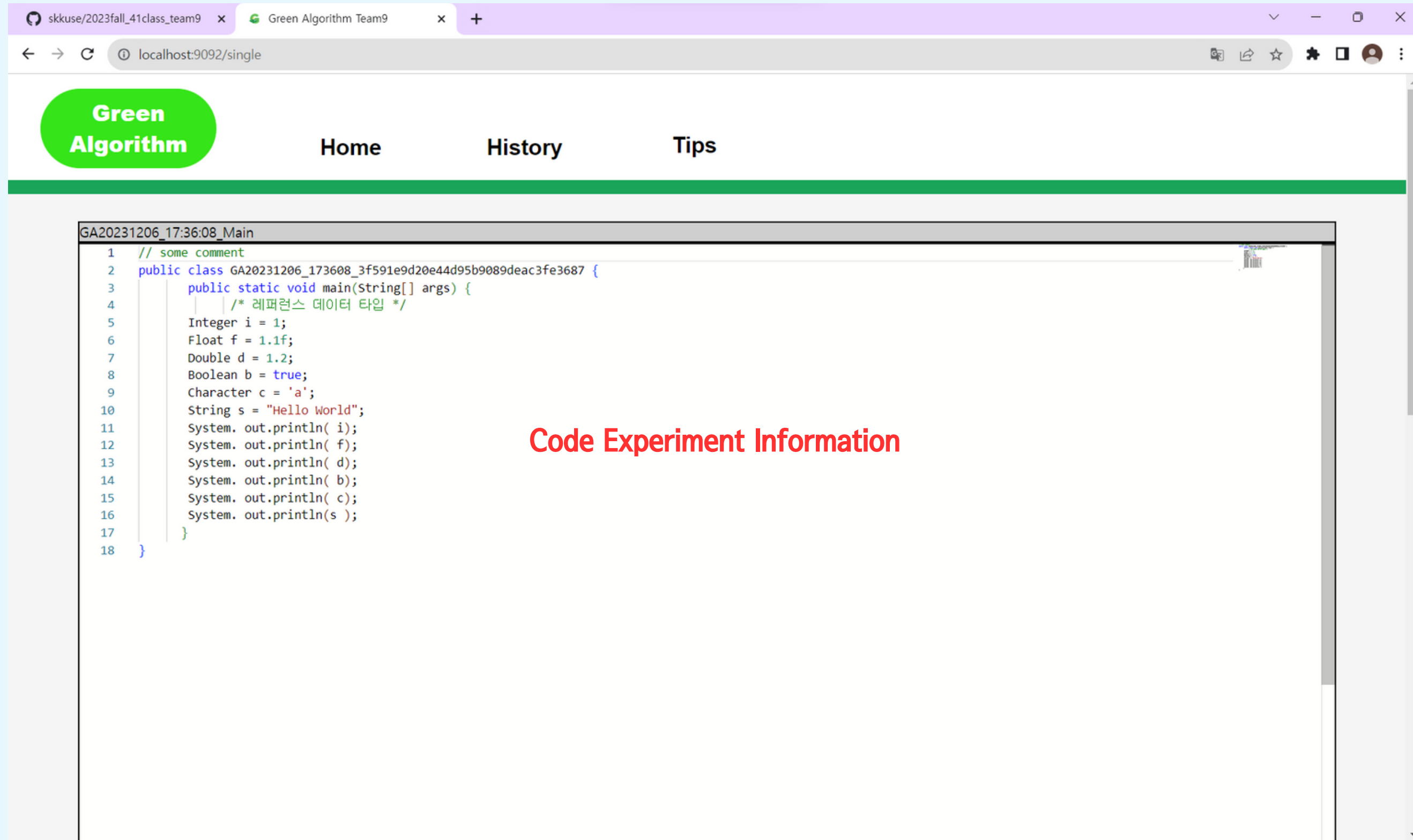
HOME PAGE

UI/UX Scenarios



HISTORY PAGE

UI/UX Scenarios



HISTORY PAGE

UI/UX Scenarios

System Environment

CPU

Number of cores

1

Model

Intel Xeon E5-2686 v4

Memory

Memory available

1GB

Cloud computing

Cloud computing


Amazon Web Services

Location

Nation


Korea

Value changed according to calculated code




Run Time

0.07472586631774902 sec




Carbon Footprint

1.69090984555807 g CO2e




Carbon sequestration

0.0018446289224269869 tree-month



in a passenger car

0.009662341974617551 km



of a flight Korea-Japan

0.0000029665085009790725 %

HISTORY PAGE

UI/UX Scenarios



The screenshot shows a web browser window with the address bar at 'localhost:9092/history'. The application has a green header with a 'Green Algorithm' logo and navigation links for 'Home', 'History', and 'Tips'. The 'History' page displays a table of experiment records. A red box highlights the '체크' (Check) column, which contains checkboxes for selecting records. A blue box highlights the 'compare' button in the top right corner of the table, with a blue arrow pointing to it.

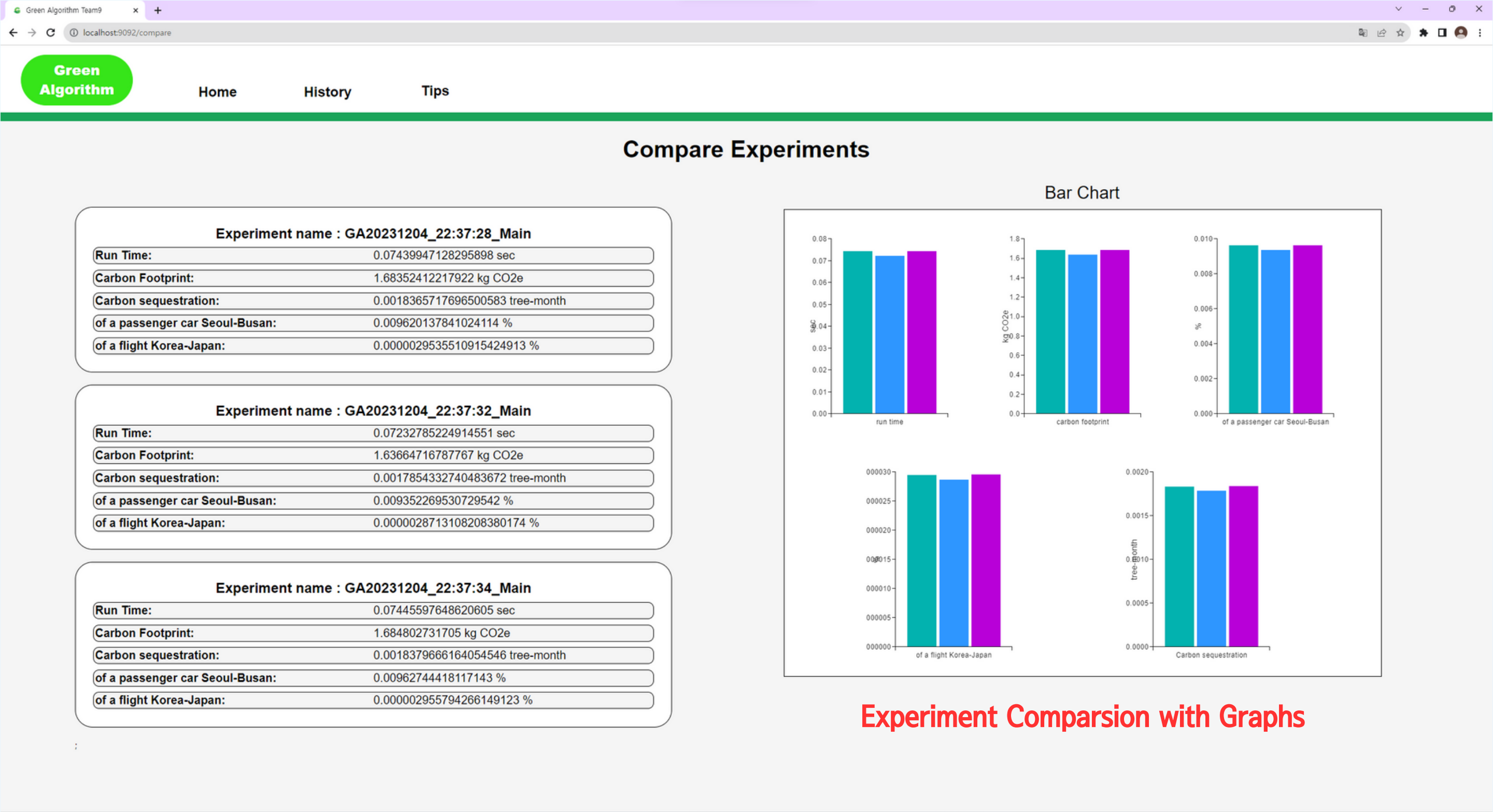
체크	Experiment Name	탄소배출량	런타임	Create Time
<input type="checkbox"/>	GA20231204 22:37:28 Main	1.68352412217922	0.07439947128295898	2023-12-04T22:37:29
<input type="checkbox"/>	GA20231204 22:37:32 Main	1.63664716787767	0.07232785224914551	2023-12-04T22:37:33
<input type="checkbox"/>	GA20231204 22:37:34 Main	1.684802731705	0.07445597648620605	2023-12-04T22:37:35
<input type="checkbox"/>	GA20231204 22:37:35 Main	1.75397712554255	0.07751297950744629	2023-12-04T22:37:36
<input checked="" type="checkbox"/>	GA20231204 22:37:36 Main	1.64576467884207	0.07273077964782715	2023-12-04T22:37:38
<input checked="" type="checkbox"/>	GA20231204 22:38:27 Main	1.71401653162937	0.07574701309204102	2023-12-04T22:38:29
<input checked="" type="checkbox"/>	1234	1.76119560463743	0.07783198356628418	2023-12-05T00:56:58

Maximum 5
Select the code you want to compare

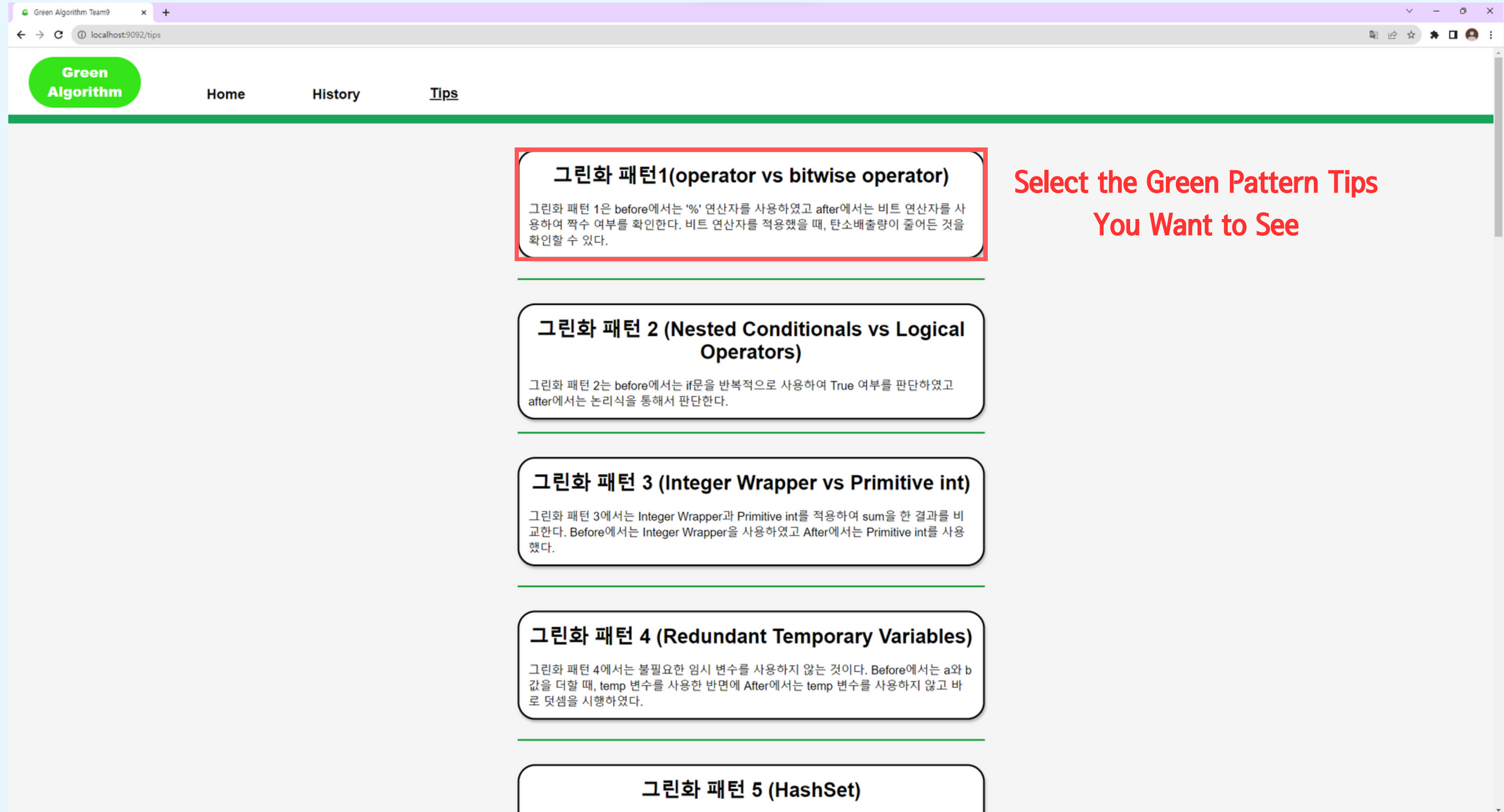
Press Compare Button!

HISTORY PAGE

UI/UX Scenarios



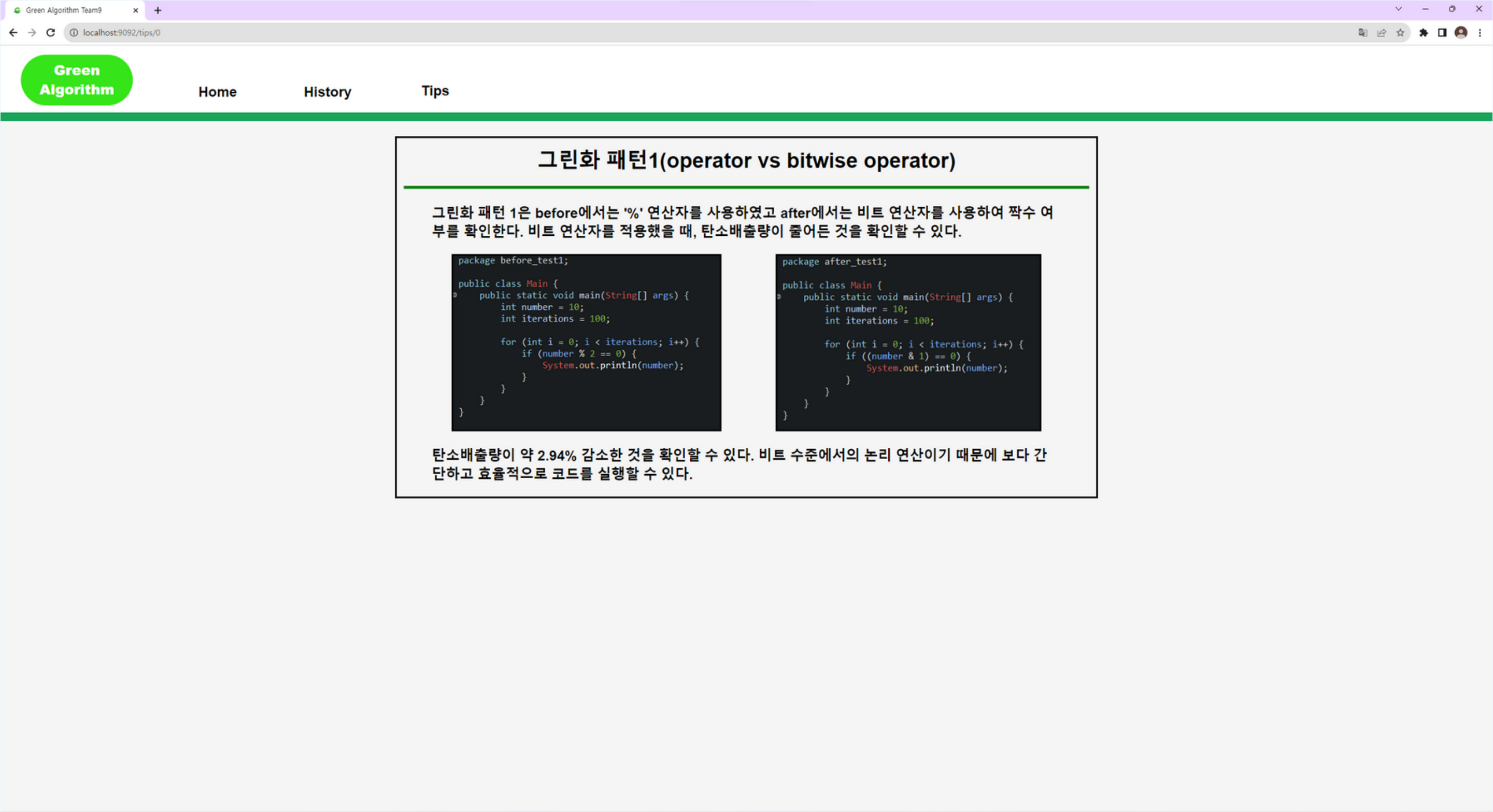
UI/UX Scenarios



Select the Green Pattern Tips
You Want to See

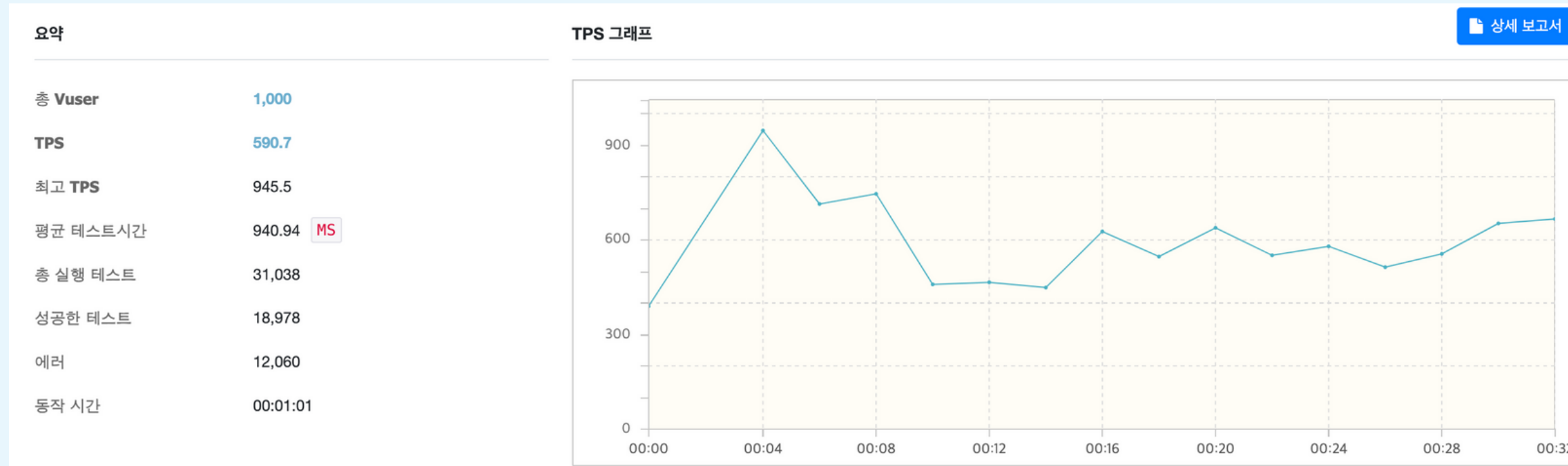
TIPS PAGE

UI/UX Scenarios

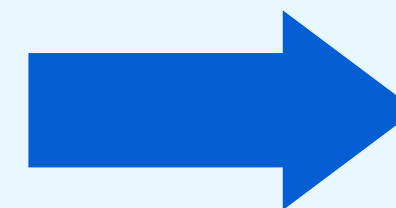


Testing & Challenge

nGrinder Test 결과



	일반 요청	실험 생성
가상 유저수	1000명	10명
최고 TPS	945	5
평균 TPS	590	1



Scale out

- Load balancing
- Docker container

Scale up

- Better CPU, More memory

Green Patterns 수집 방법

Methods of Collecting Green Patterns

BOOKS



Java Performance Tuning
Java Optimization

THESIS



How Green Are Java
Best Coding Practices

SEARCH



Java Optimization
Java Performace
Java Memory
Java Runtime

Green Patterns 적용 전과 후의 탄소배출량에서 중요한 비교요소

RUNTIME



Green Patterns 수집 방법

Methods of Collecting Green Patterns

$$\text{Carbon footprint} = \text{Runtime} \times (\text{power draw for cores} \times \text{usage} + \text{power draw for memory}) \times \text{PUE} \times \text{PSF} \times \text{Carbon intensity}$$

테스트 환경

Location: South Korea

TDP: 15(W)

Memory available: 11(GB)



Green Patterns Test

그린화 패턴 테스트 과정

1. 그린화 패턴이 적용되기 전후의 **jar 파일 로드**
 - a. before_test#.jar
 - b. after_test#.jar
2. 각 파일의 **런타임 평균과 총합 계산**
 - a. 테스트의 객관화를 위해 **100번 측정**
3. 평균 런타임을 통해 **탄소배출량 계산**
4. 탄소배출량 **감소 비율 계산**

```
package run_time;

public class Main {
    public static void main(String[] args) {
        String beforejarPath = "C:\\Users\\User\\Desktop\\jar file\\before_test20.jar";
        String afterjarPath = "C:\\Users\\User\\Desktop\\jar file\\after_test20.jar";

        int numExecutions = 100;
        long beforetotalRuntime = 0;
        long aftertotalRuntime = 0;

        for (int i = 0; i < numExecutions; i++) {
            long beforestartTime = System.currentTimeMillis();

            try {
                ProcessBuilder beforeprocessBuilder = new ProcessBuilder("java", "-jar", beforejarPath);
                Process beforeprocess = beforeprocessBuilder.start();
                beforeprocess.waitFor();

                long beforeendTime = System.currentTimeMillis();
                long beforeexecutionTime = beforeendTime - beforestartTime;
                beforetotalRuntime += beforeexecutionTime;
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        double beforeaverageRuntime = (double) beforetotalRuntime / numExecutions;
        System.out.println("Before 평균 실행 시간: " + beforeaverageRuntime + "ms");

        for (int i = 0; i < numExecutions; i++) {
            long afterstartTime = System.currentTimeMillis();

            try {
                ProcessBuilder afterprocessBuilder = new ProcessBuilder("java", "-jar", afterjarPath);
                Process afterprocess = afterprocessBuilder.start();
                afterprocess.waitFor();

                long afterendTime = System.currentTimeMillis();
                long afterexecutionTime = afterendTime - afterstartTime;
                aftertotalRuntime += afterexecutionTime;
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        double afteraverageRuntime = (double) aftertotalRuntime / numExecutions;
        System.out.println("After 평균 실행 시간: " + afteraverageRuntime + "ms");

        System.out.println((beforeaverageRuntime/3600000)* (15*1+4.0975)*1*100*500*0.001);
        System.out.println((afteraverageRuntime/3600000)* (15*1+4.0975)*1*100*500*0.001);
    }
}
```

Team Works & Schedule

Frontend

이진혁 이한준

Backend

박태현 한상안

Green & Doc

김한얼 한채윤



감사합니다

Web Service Development

Calculation of carbon emissions

TEAM 9

