UTILIZATION AWARE TRIP ADVISOR IN BIKE SHARING SYSTEM BASED ON USER BEHAVIOUR ANALYSIS

N.SRINIVASA RAO [1],THATAVARTHI L V KANAKA DURGA[2].

[1] Assistant Professor**, DEPT OF MCA, SKBR PG COLLEGE , AMALAPURAM, Andhra Pradesh**

Email:- naagaasrinu@gmail.com

[2]PG Student of MCA, SKBR PG COLLEGE , AMALAPURAM, Andhra Pradesh

Email:- kanakadurga88036@gmail.com.

ABSTRACT

Over the past ten years, the fast advancement of bike-sharing programmes has greatly benefited individuals. High levels of transit flexibility, however, create issues for both users and operators. Users frequently encounter stations where the check-in or check-out service is not accessible due to dynamic distribution of shared bikes brought on by unequal user demand. Unbalanced bike usage results in more broken bikes and rising maintenance costs for operators. By guiding customers to stations with a greater success rate of rental and return, we aim to improve user experiences and rebalance bicycle utilisation in this study. For the first time, we create a travel planner that suggests bike check-in and check-out locations while taking service quality and bicycle usage into account together. To guarantee service excellence, To determine the likelihood of rental and return in the future, we first forecast the user demand for each station. Experiments show that our approach's precision is as high as 0.826, up 25.9% from the accuracy of the historical average method. From historical data, we can determine that biassed bicycle use is a result of constrained bicycle circulation among a small number of active stations. As a result, by encouraging users to switch bikes between very active and inactive stations, we maximise the bike circulation with defined station activeness.

1 INTRODUCTION

Sustainable development is now widely accepted throughout the world due to the increasing severity of environmental degradation and devastation brought on by human activity in recent years as the economy has grown. Due to its low pollution, low energy consumption, and great flexibility, bike-sharing systems (BSS) are developed in this situation as a solution for short car journeys. Public bike-sharing programmes not only expand the reach of transit and walking journeys, giving individuals a healthy transportation option, but they can also spur increased interest in cycling and boost cycling ridership. This reduces the demand for personal vehicle trips in addition to other benefits. By the end of 2016, more than 1,100 cities will have deployed an estimated 2,000,000 public automated bike-sharing systems. bicycles everywhere. Using a smart card to hire a bike from a local station and return it to another station is simple with bike-sharing systems. But the benefits cannot erase the problems, which are becoming more and more obvious. The user demand for stations is unstable and imbalanced, which frequently results in the check-in or check-out service being unavailable at some stations and has a detrimental effect on user experience. The uneven distribution of bike usage frequency presents a challenge for both users and system administrators.

On the one hand, the system generally results in an unequal allocation of bikes throughout the different stations (due to the unregulated, uneven demand), thus rendering the check ineffective because of the great degree of flexibility of the bikesharing system. The check out service is not available at some stations with either full or empty bicycle docks. User demand characteristics vary amongst stations in certain areas during peak times. For instance, return demand increases in commercial districts whereas rental demand typically increases near residential regions in the morning hours of the workday. Currently, operators redistribute bikes based on user complaints and monitor footage. But this approach has revealed a significant latency. Operators frequently begin to issue scheduling instructions when an occurrence causes service to be unavailable. It can be challenging to satisfy user requests during rush hour because by the time the vehicle arrives, service-unavailable events may have already passed.

Studies have been carried out to raise service availability and improve user experience. to enhance these bike redistribution plans based on forecasts and models for bicycle mobility. The majority of earlier research is on forecasting rental volume and bike usage patterns for each station without taking into account online data. Demand forecasting for each cluster from the perspective of bike flow mobility patterns, which may not be appropriate for recommending stations to users, has received less attention.In conclusion, creating a precise prediction model that takes into account a number of variables has proven to be challenging and has mostly gone unexplored. Bicycle traffic is not only extremely dynamic and connected in both the temporal and spatial domains, but is also further influenced by complicated factors like timing and meteorology. This is the fundamental technical problem. In order to address the issue of uneven demand,

we establish Using the random forest technique, a fine-grained demand forecasting model can forecast check-in and check-out demand on a station-by-station basis with sub-hour resolution. To represent the cyclical patterns of customer demand, offline elements like time and weather are chosen in our model. The station's real-time availability is reflected online, which is useful for unusual traffic. The use of some bikes, however, is far greater than that of others. Overuse of bikes makes them more prone, which raises repair costs and could result in service denial. According to reports, the first bicycle from Hangzhou BSS was rented more than 6,000 times and travelled more than 20,000 kilometres in just three years. Likewise, the The 2016 bike with the longest rental history has been rented 5,616 times, or more than 15 times every day on average. The average lifespan of the bicycles owned by the Hangzhou public bike-sharing company is less than 4 years due to their frequent use under heavy loads and lack of timely maintenance and replacement. Contrarily, the lifespan of a private bicycle is 10 years or longer. The cost of labour and repairs, meanwhile, makes up a sizable amount of overall operating costs. The cost of fixing Hangzhou's bike-sharing system in 2012 was close to 6 million yuan. In 2012, the annual maintenance fees for bicycles in Washington, D.C., ranged from $200 to $300.

2. LITERATURE SURVEY AND RELATED WORK

The objectives for JDBC are significant. They will help you understand why particular classes and functions act in the manner that they do. The eight JDBC design objectives are as follows:

API at SQL Level

The creators believed that creating a Java SQL interface was their primary objective. It is not the lowest database interface level that can be achieved, but it is low enough to allow for the development of higher-level tools and APIs. On the other hand, it is competent enough for application programmers to use. By achieving this objective, tool providers in the future will be able to "generate" JDBC code and hide many of JDBC's complexity from end users.

SQL Compatibility

As you switch databases, the syntax of SQL changes. vendor to vendor database. Any query statement may be provided through JDBC to the underlying database driver in an effort to support a wide range of vendors. By doing so, the connectivity module is able to manage non-standard functionality in a way that is appropriate for its customers.

Implementing JDBC on top of standard database interfaces is required. Other widespread SQL level APIs must "sit" underneath the JDBC SQL API. By using a software interface, this objective enables JDBC to make advantage of already-in-use ODBC level drivers. This interface would convert ODBC calls to JDBC calls and the other way around.

Ensure that the Java interface you provide is in line with the rest of the Java system. Due to Java's current user community acceptability, the Designers believe that they shouldn't deviate from the core Java system's current design.

The goal of keeping things simple is probably listed in all lists of software design goals. JDBC is no different. Sun believed that JDBC's design should be very straightforward, with only one way for any mechanism to complete a task. Duplicate functionality is only intended to confuse API users.

whenever possible, type with strength and static

Strong typing enables more thorough error checking during compilation and reduces the number of runtime errors.the typical cases simple

These queries should be easy to execute with JDBC because the typical SQL calls used by programmers are simple SELECT, INSERT, DELETE, and UPDATE statements. Even so, more Additionally, complicated SQL statements ought to be possible.

programming language used in Java

All of the following catchphrases fit the high-level language that is the Java programming language:

Simple, neutral in terms of architecture, object-oriented, portable, distributed, and high-performing

◎ Interpreted ◎ Multithreaded ◎ Robust ◎ Dynamic ◎ Secure

To run a programme on your computer, you typically need to either compile it or interpret it, depending on the programming language. Given that a programme can be both compiled and interpreted, the Java programming language is uncommon. With the compiler, you first translate a programme into the platform-independent Java byte codes, which are then interpreted by the Java platform's interpreter. Each Java byte code instruction is parsed and executed by the interpreter on the computer. There is just one compilation and multiple interpretations. when the programme is run. The operation of this is shown in the following figure.

Java byte codes can be thought of as the Java Virtual Machine's (Java VM) machine code instructions. Every Java interpreter, including those found in development tools and web browsers that support applets, is a particular Java Virtual Machine (VM) implementation. The phrase "write once, run anywhere" is made feasible because to Java bytes. On any platform with a Java compiler, you can convert your programme into byte codes. The byte codes can then be executed on any Java VM implementation. This means that the same Java programme can operate on Windows 2000, a Solaris workstation, or on any other machine as long as it has a Java Virtual Machine (VM) an iMAC.

3 PROPOSED WORK AND ALGORITHM

Architecture Flow:

The architecture diagram below primarily depicts the request flow from users to databases via servers. The presentation layer, business layer, and data link layer are the three layers that make up the entire system in this situation. Three-tier architecture was used in the creation of this project. In order to address the shortcomings of the two-tier design, the three-tier software architecture (also known as a three-layer architecture) was developed in the 1990s. Between the user interface (client) and data management (server) components is the third tier, sometimes known as the middle tier server. By offering services like queuing, application execution, and database staging, this middle tier provides process management where business logic and rules are put into action and can support hundreds of users (as opposed to only 100 users with the two tier architecture).

Advantages of Three-Tier:

- Distinguishes presentation from functionality.
- Sharp distinction - improved comprehension.
- Limited modifications to clearly defined components.
- May function on the WWW.
- Successful network operation.

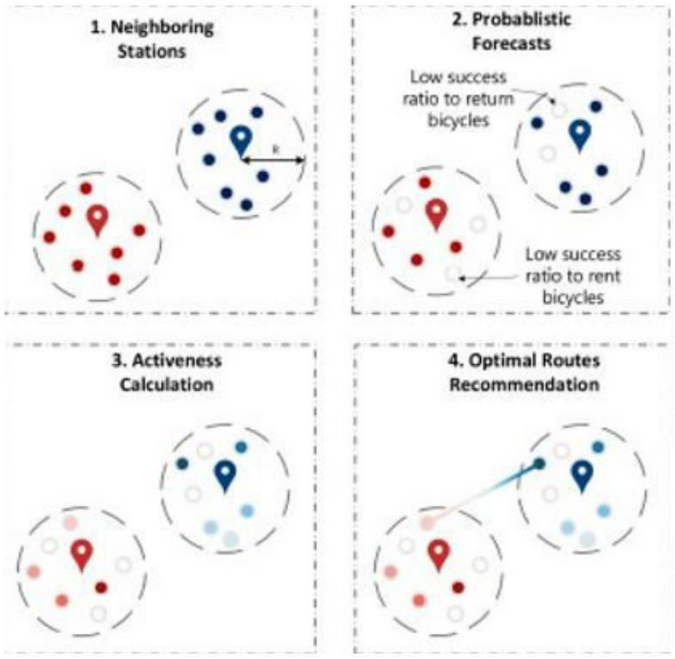


Fig 1: System architecture

4 METHODOLOGIES

The architecture diagram below primarily depicts the request flow from users to databases via servers. The presentation layer, business layer, and data link layer are the three layers that make up the entire system in this situation. Three-tier architecture was used in the creation of this project. In order to address the shortcomings of the two-tier design, the three-tier software architecture (also known as a three-layer architecture) was developed in the 1990s. Between the user interface (client) and data management (server) components is the third tier, sometimes known as the middle tier server. By offering services like queuing, application execution, and database staging, this middle tier provides process management where business logic and rules are put into action and can support hundreds of users (as

opposed to only 100 users with the two tier architecture). a huge collection of ready-made software elements, such as widgets for graphical user interfaces (GUI), that provide a wide range of useful capabilities. Packages, which are assemblages of related classes and interfaces, are how the Java API is organised. The next section is What Can Java Technology Do? showcases some of the Java API's package offerings' features.

Mobile phones and pagers are among the devices that MIDP is designed for. The MIDP, like KJava, is based on CLDC and offers a common run-time environment that enables the dynamic deployment of new applications and services on end user devices. MIDP is a widespread, open-source mobile device profile that is independent of any one vendor. It serves as a robust and well-supported framework for creating mobile applications. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

- java.lang
- java.io
- java.util
- javax.microedition.io
- javax.microedition.lcdui
- javax.microedition.midlet
- javax.microedition.rms

JDBC

In an effort to provide an independent database standard API for Java, Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC provides a standard interface to several RDBMSs via a generic SQL database access technique. This standard interface is made possible by the use of "plug-in" database connectivity modules or drivers. To have JDBC support, a database vendor must provide the driver for each platform that Java and the database work on.

Servlet technology, which runs on the server and generates dynamic web pages, is used to construct web applications.

Servlet technology is dependable and scalable because of Java. Before Servlet, the CGI (Common Gateway Interface) scripting language was a popular server-side programming language.

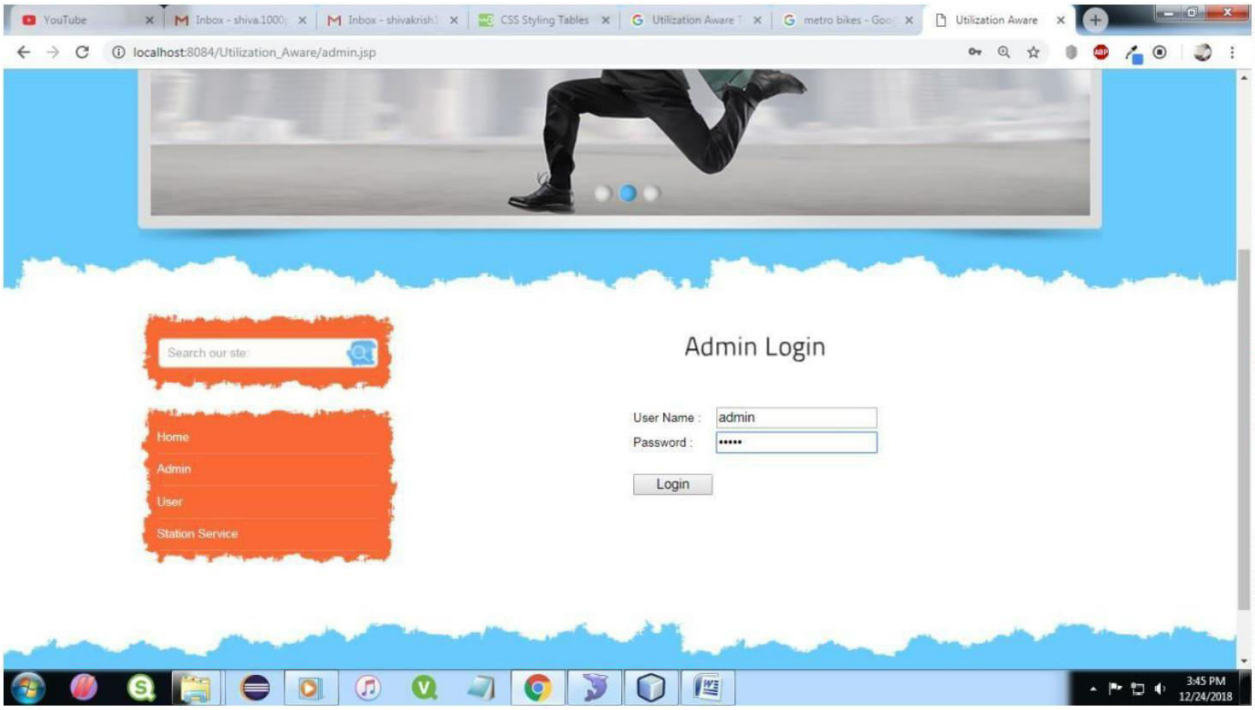
JSP technology is comparable to Servlet technology. is employed to build web apps. It can be seen as an extension of servlet because it provides more functionality than servlet, such as expression language, JSTL, etc.

A JSP page is made up of both HTML and JSP tags. JSP pages are easier to handle than Servlet pages since designing and code may be separated. Among other extra features, it provides Expression Language and Custom Tags.

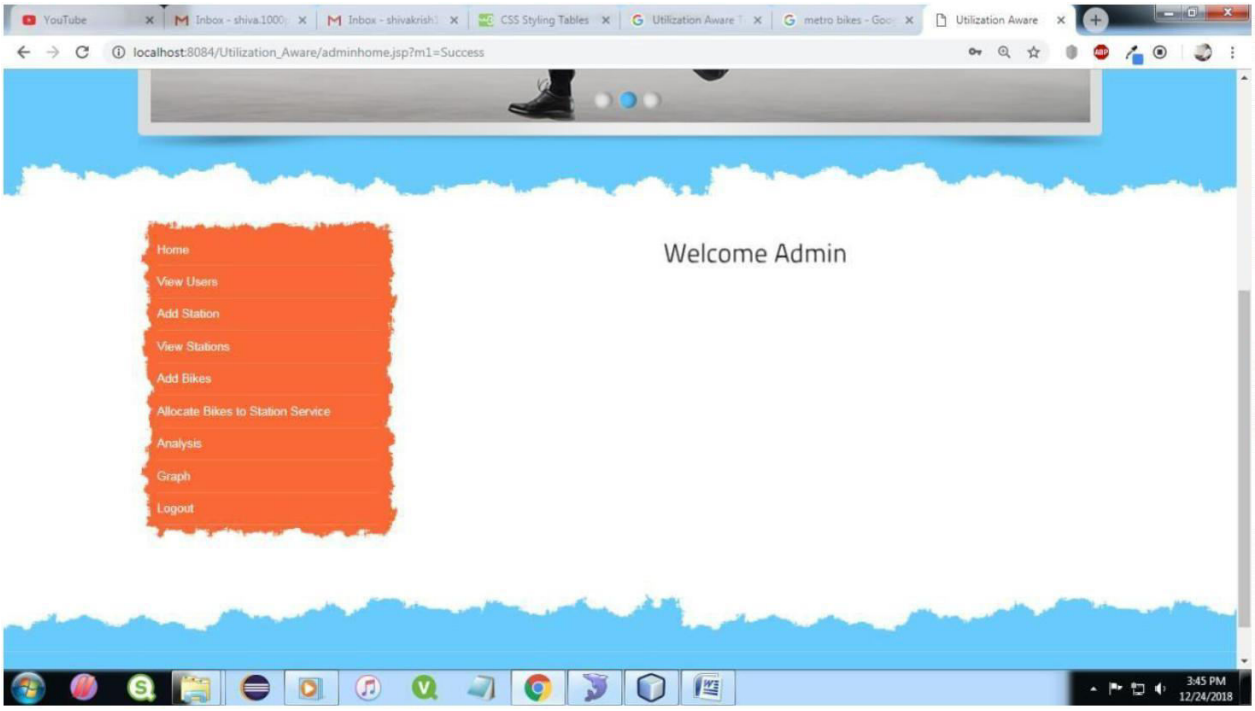
5.RESULTS AND DISCUSSION SCREENSHOTS

Home:

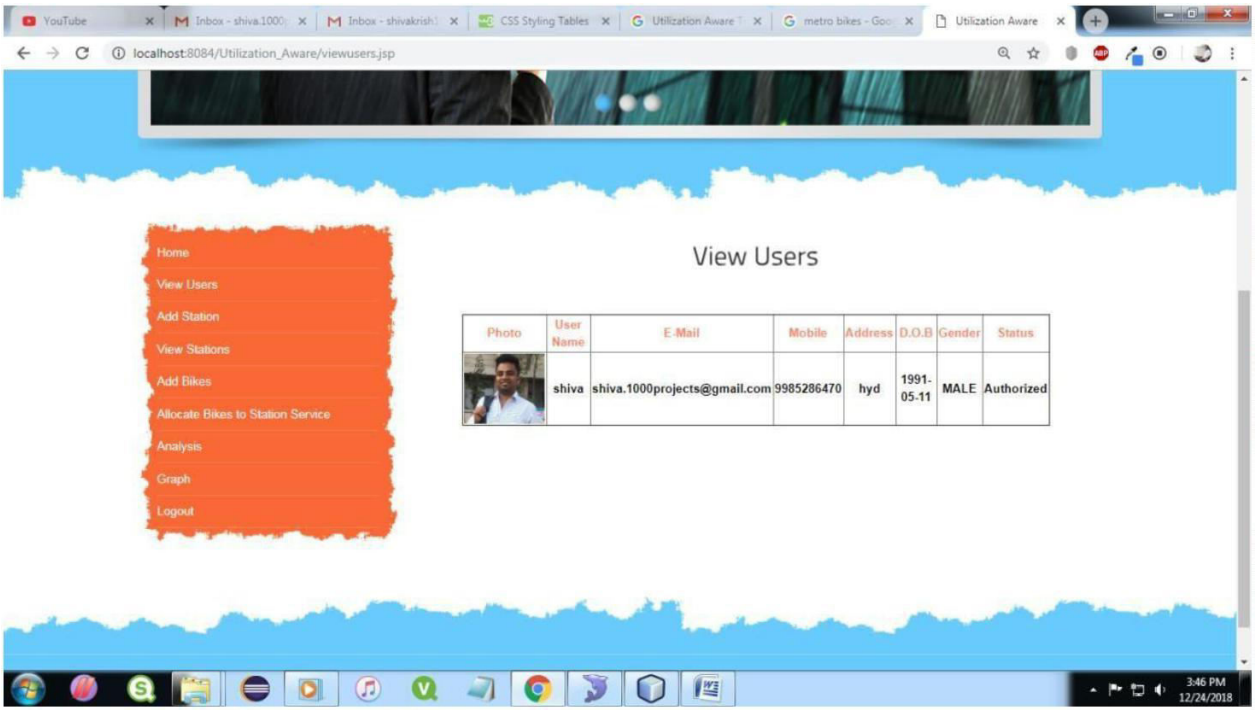
Admin login:



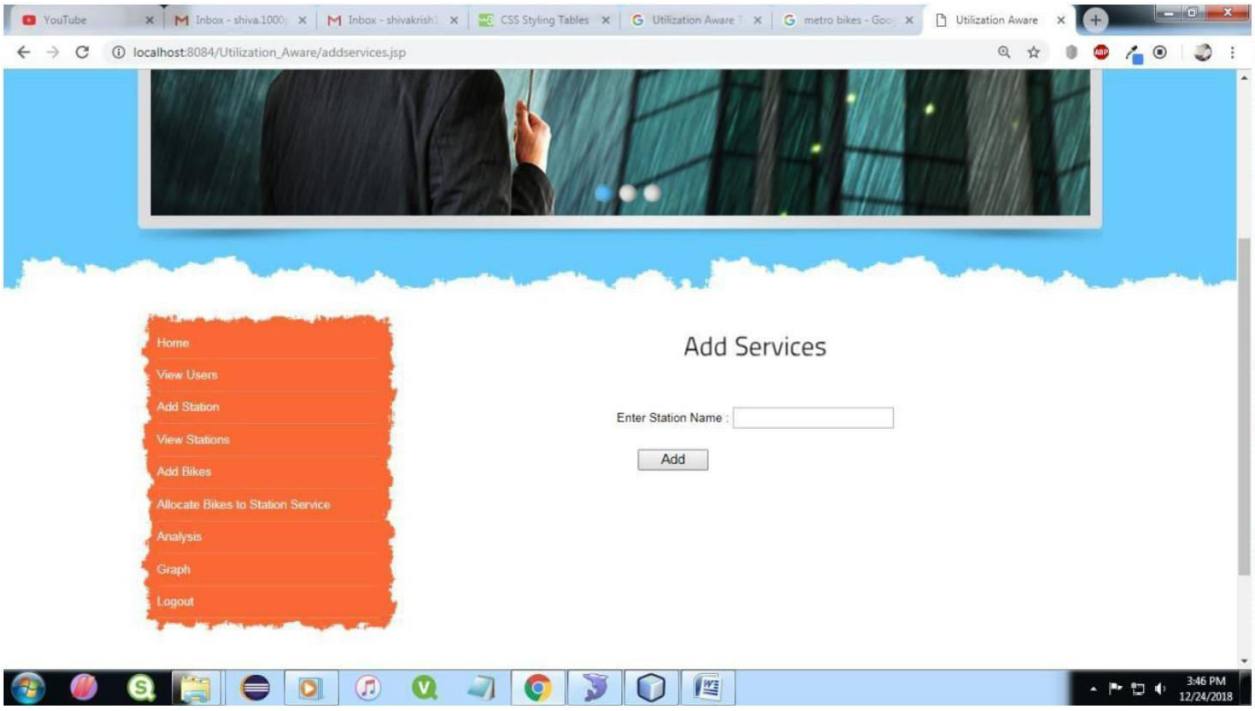
Admin home:



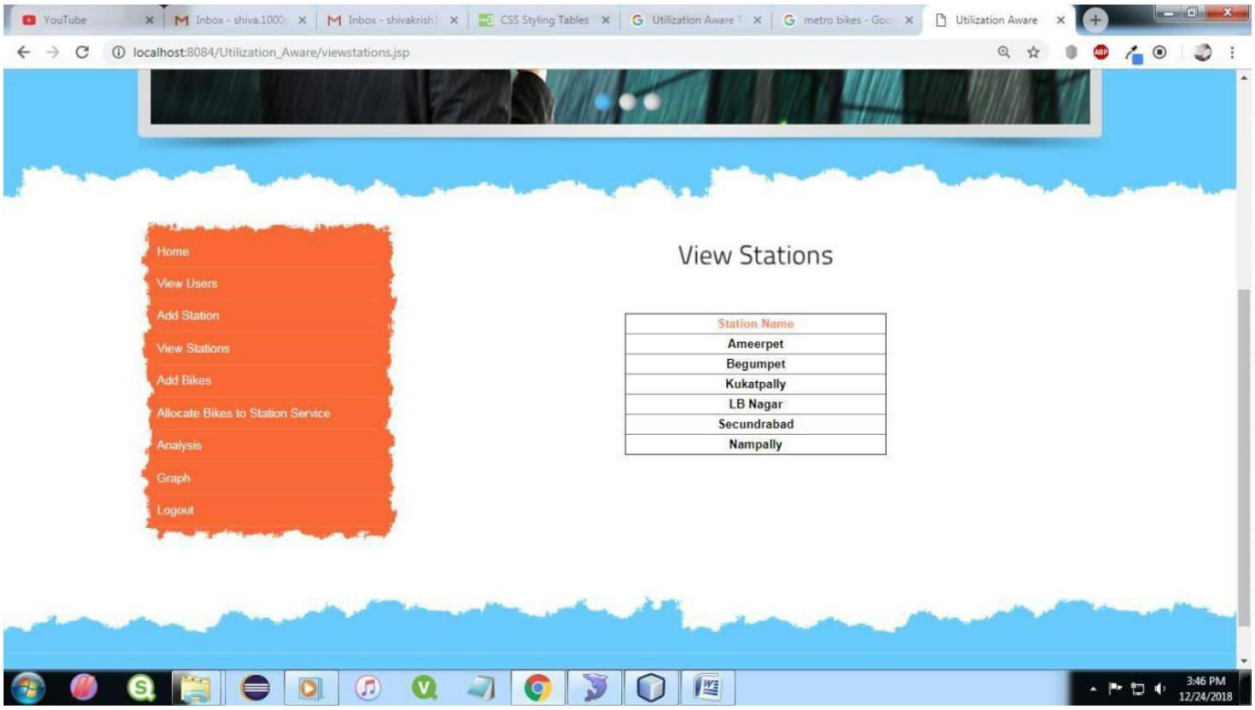
View users:



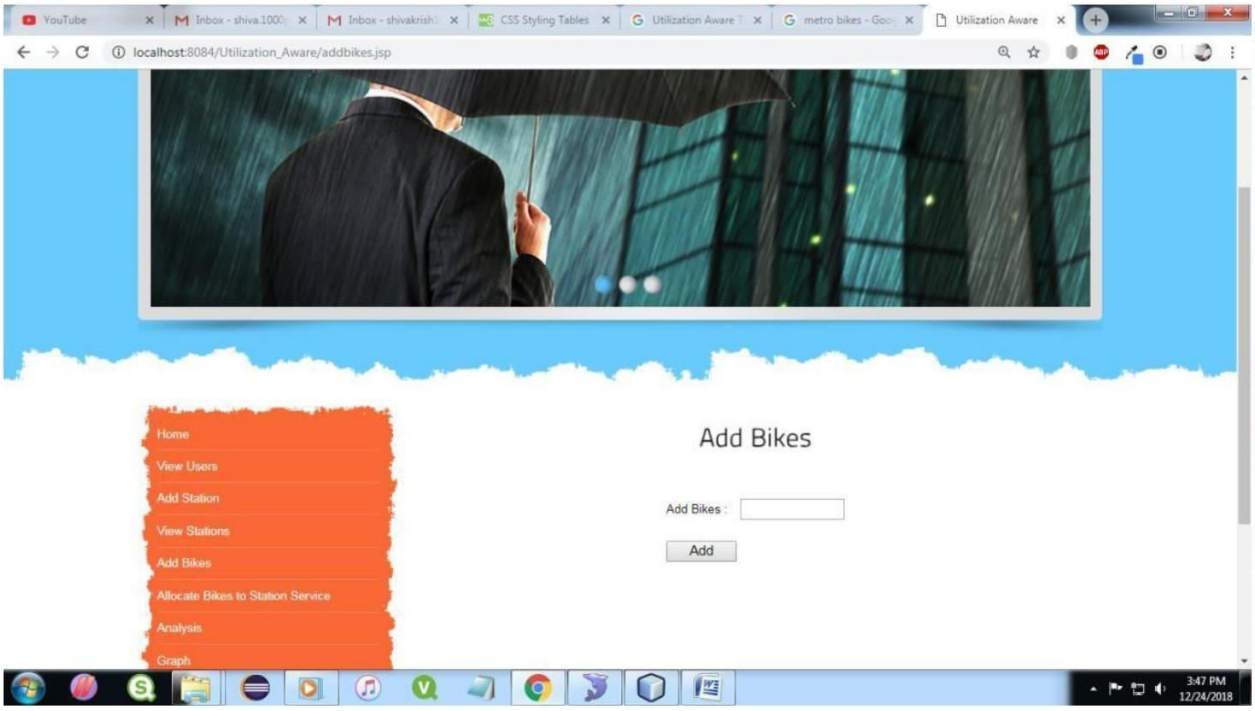
Add stations:



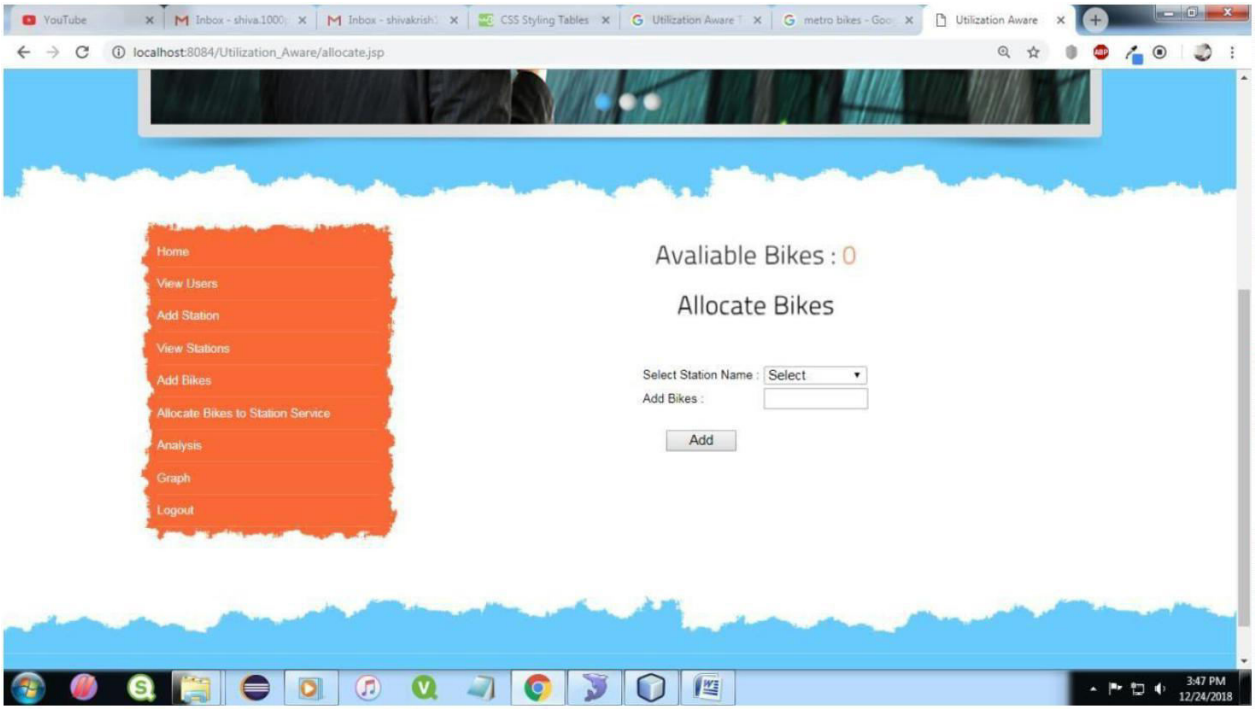
View Stations:



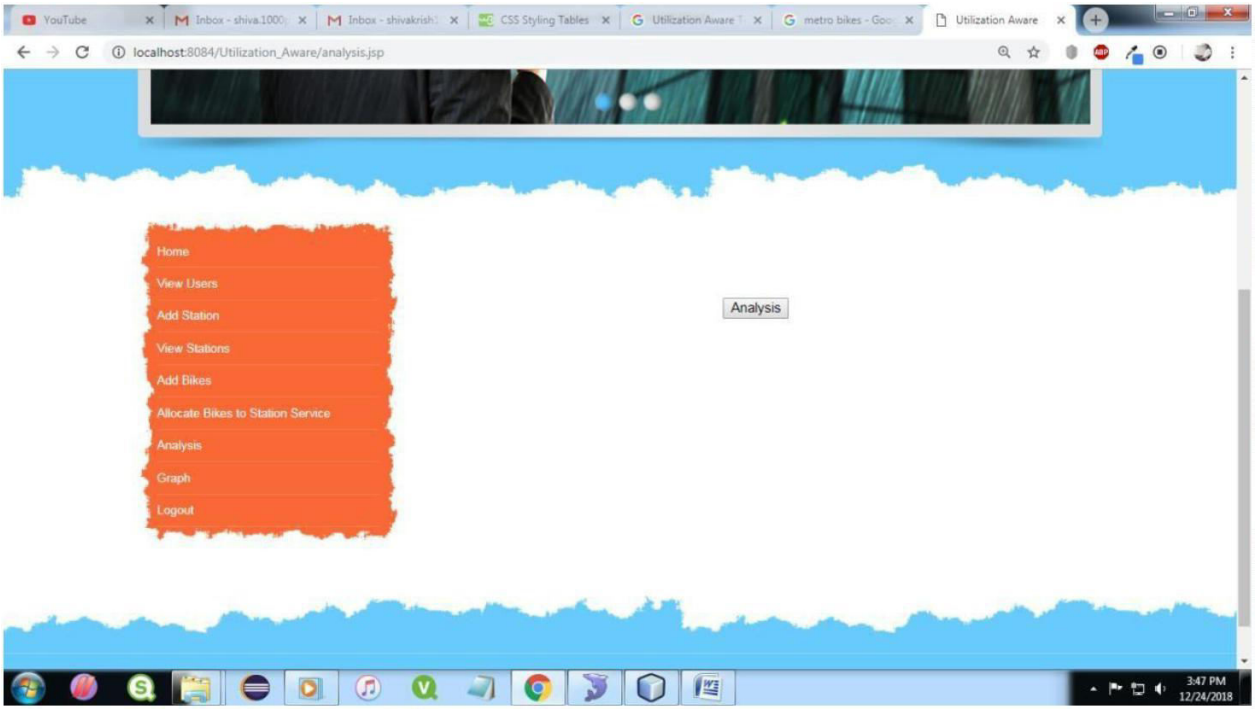
Add Bikes:

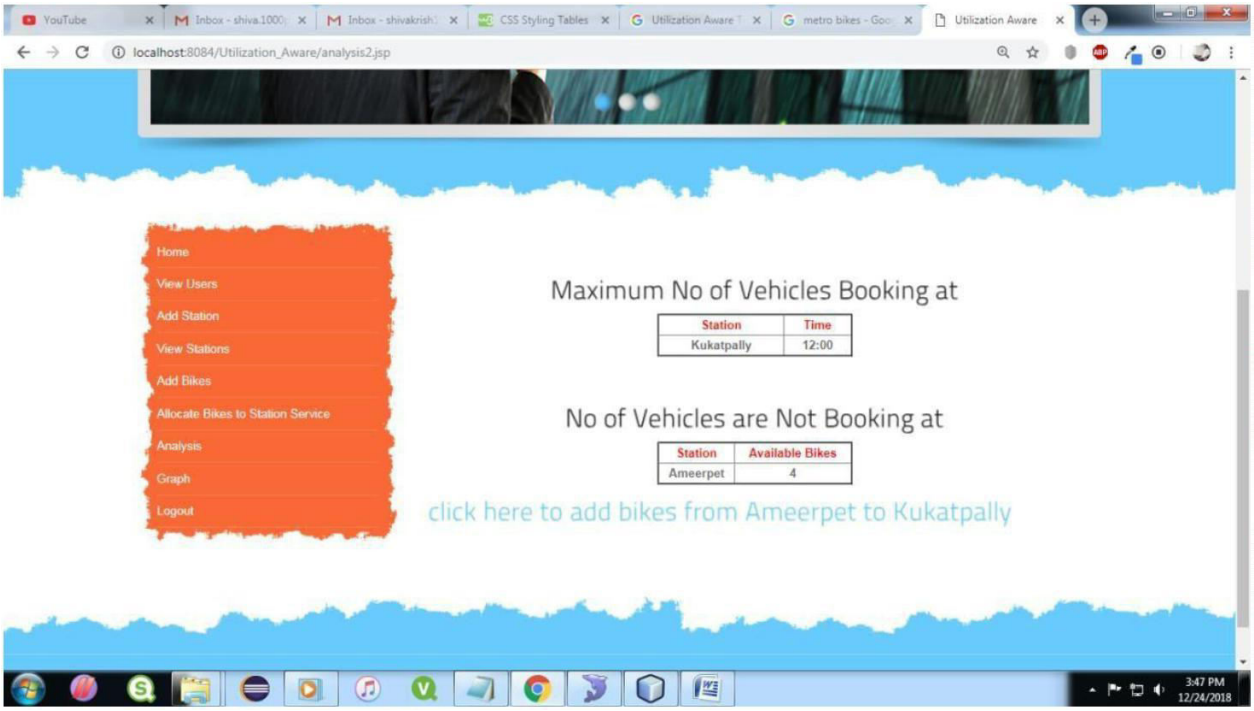


Allocate Bikes:

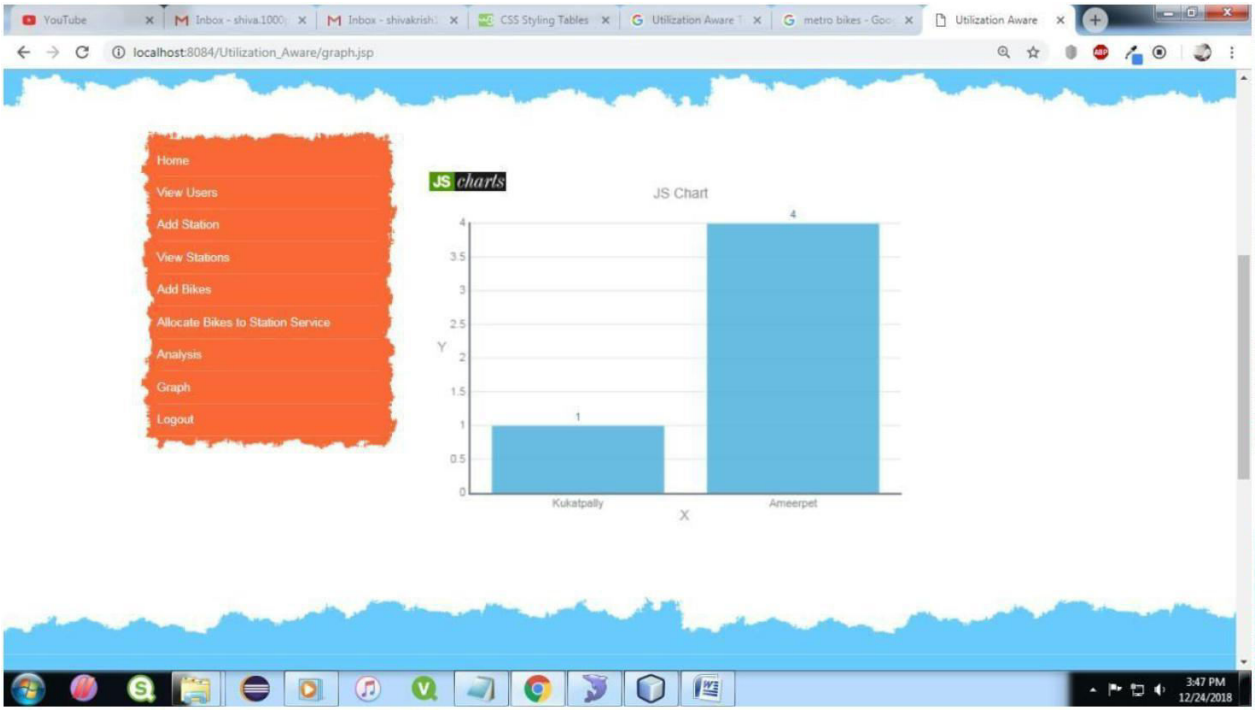


Analysis :

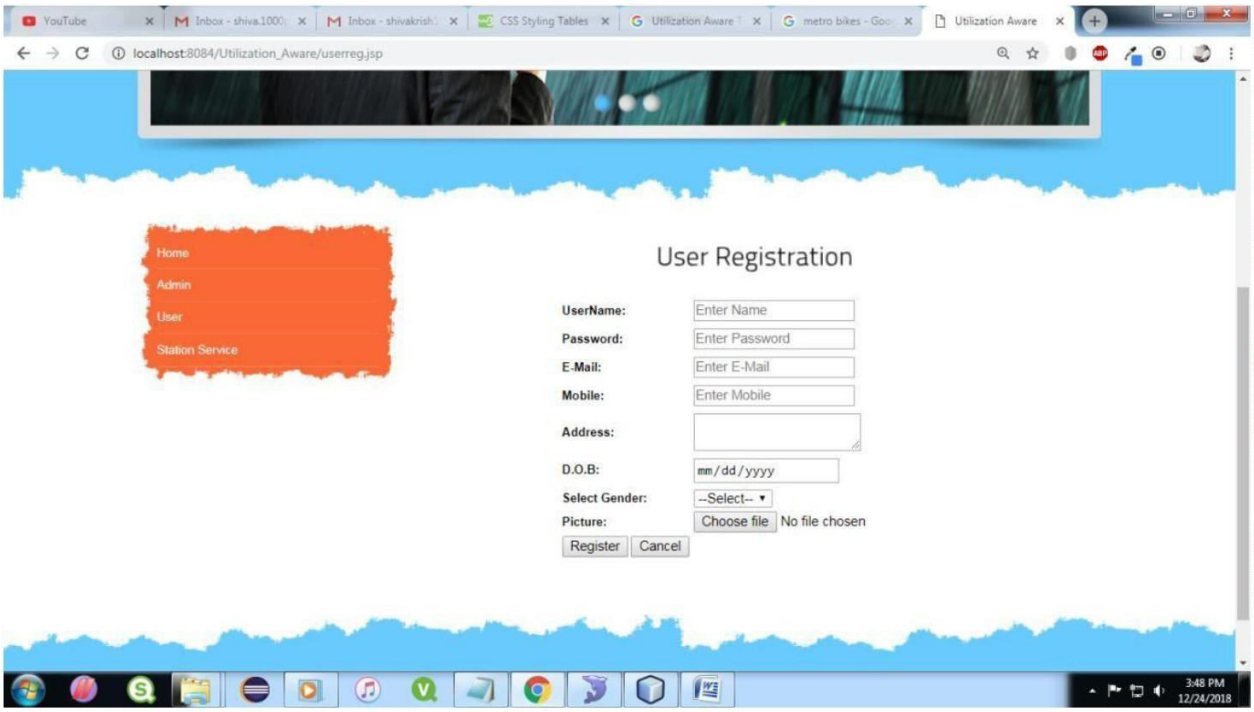

YouTube
Inbox - shiva.1000
Inbox - shivakrish
CSS Styling Tables
Utilization Aware
metro bikes - Goo
Utilization Aware
localhost:8084/Utilization_Aware/analysis.jsp
Home
View Users
Add Station
View Stations
Add Bikes
Allocate Bikes to Station Service
Analysis
Graph
Logout
Analysis
3:47 PM
12/24/2018


Home
View Users
Add Station
View Stations
Add Bikes
Allocate Bikes to Station Service
Analysis
Graph
Logout
Maximum No of Vehicles Booking at
Station
Time
Kukatpally
12:00
No of Vehicles are Not Booking at
Station
Available Bikes
Ameerpet
4
click here to add bikes from Ameerpet to Kukatpally

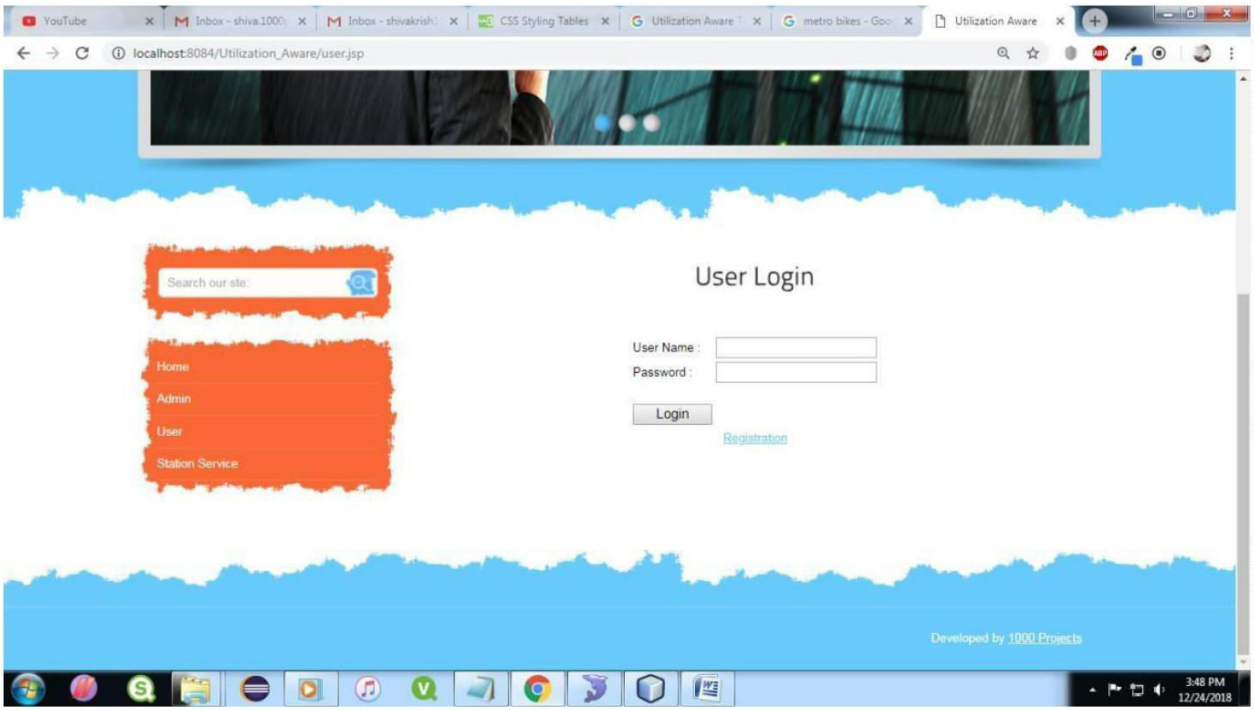
Graph:



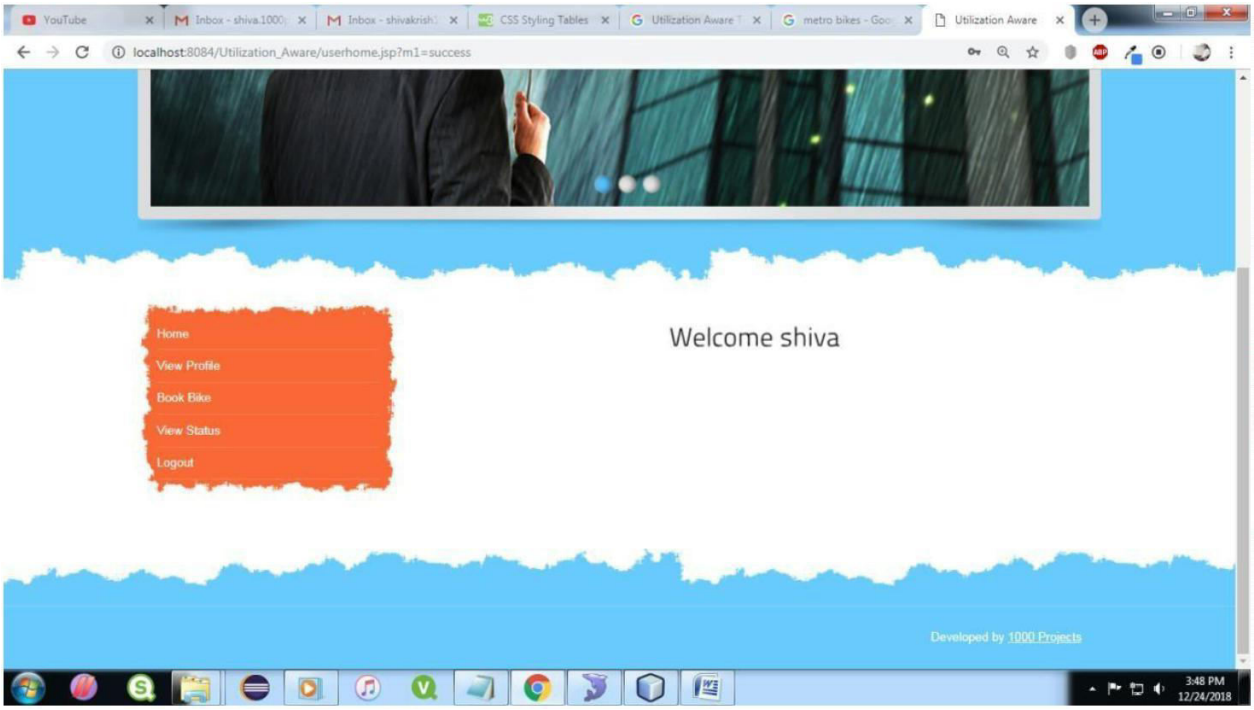
User Registration:



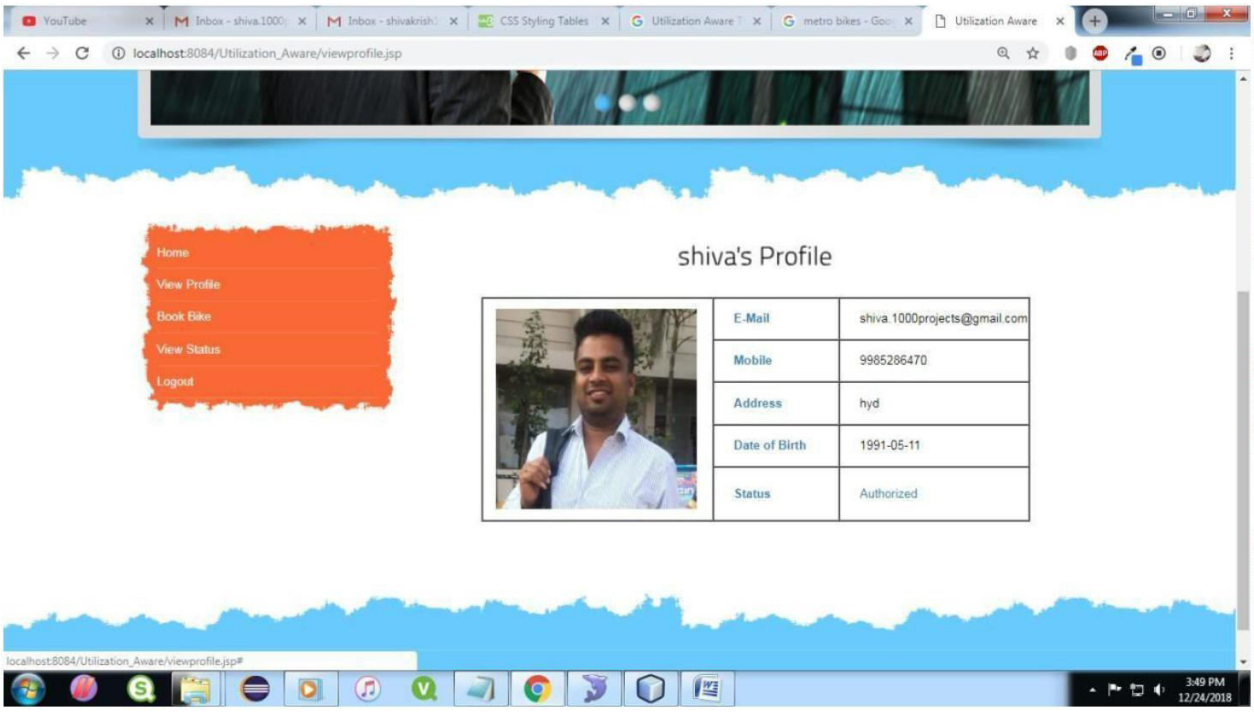
User Login:



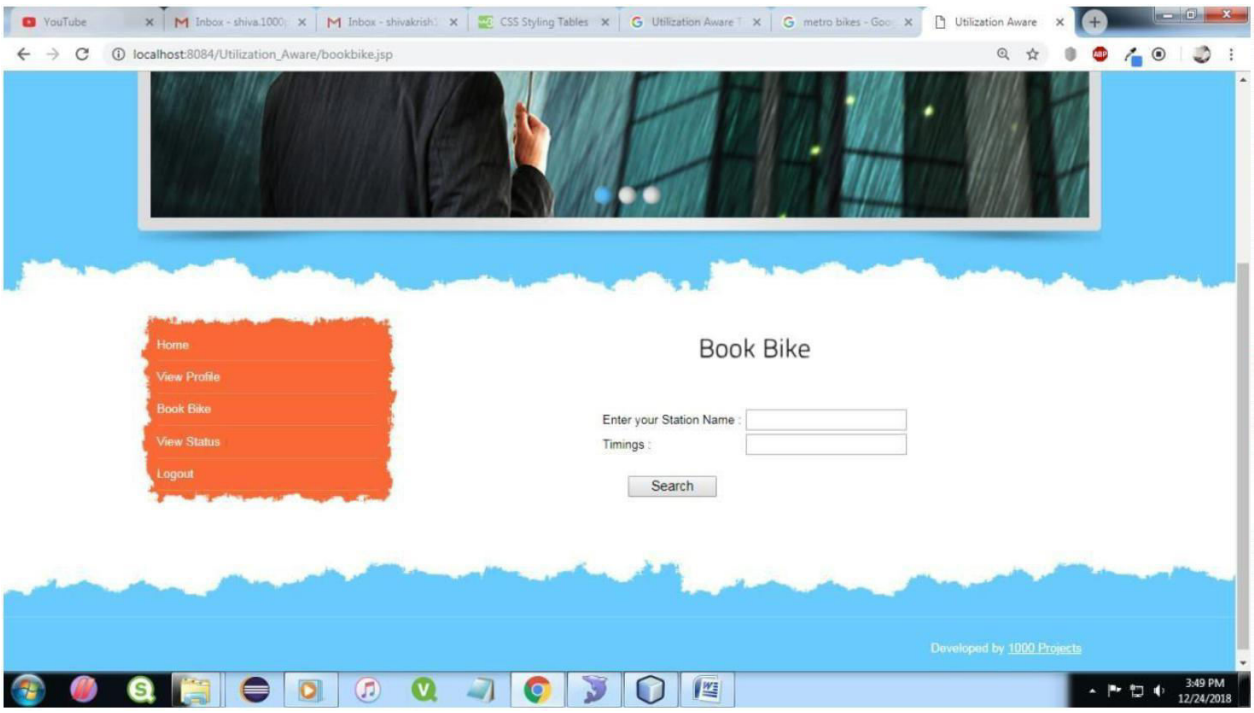
User Home:



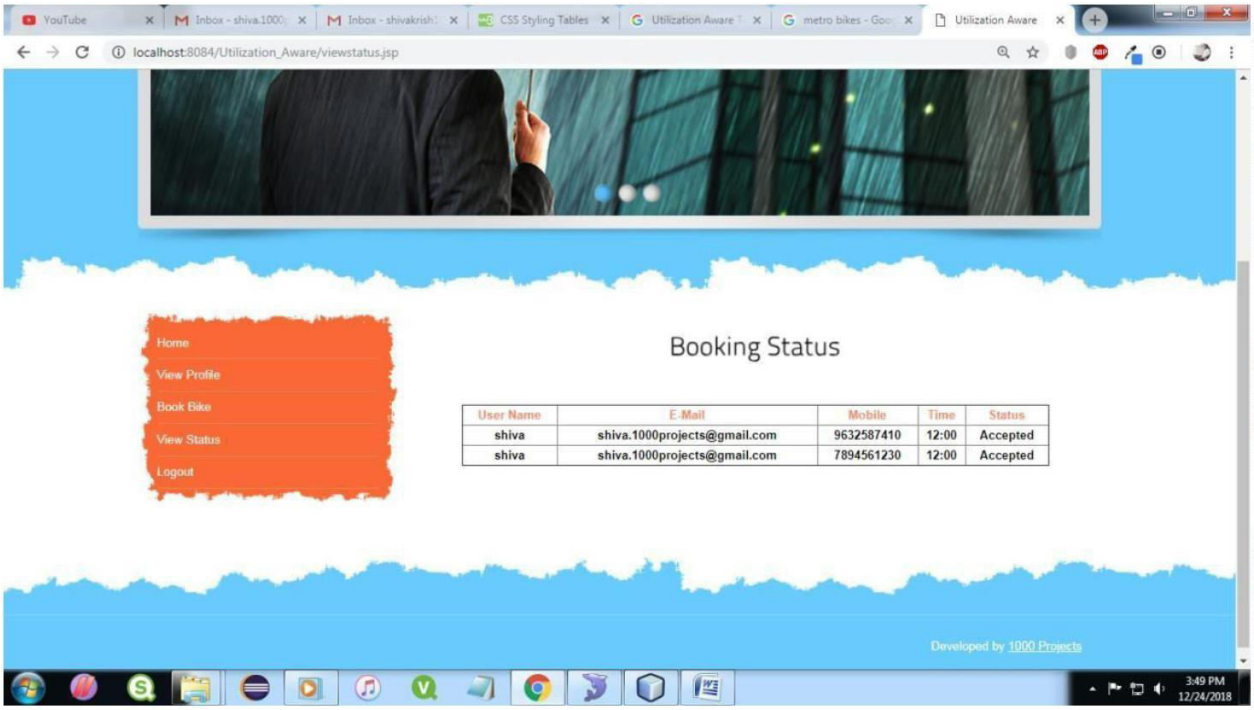
View Profile:



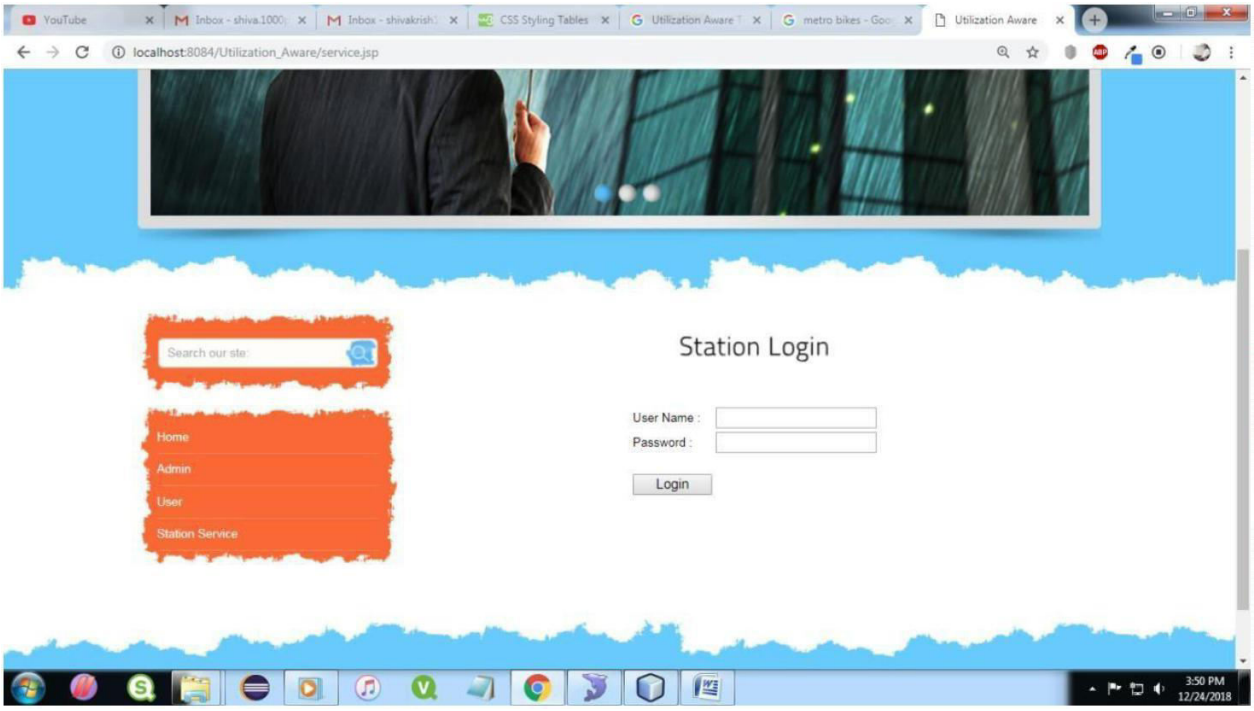
Book bike:



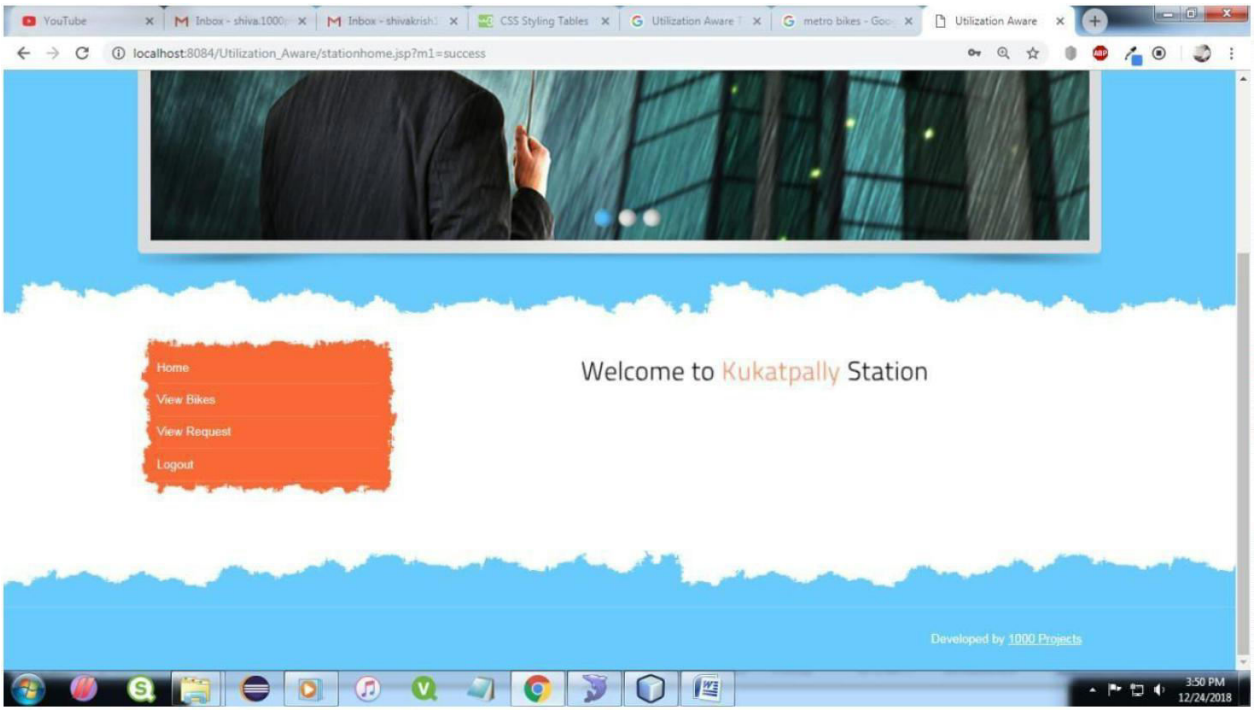
View Status:



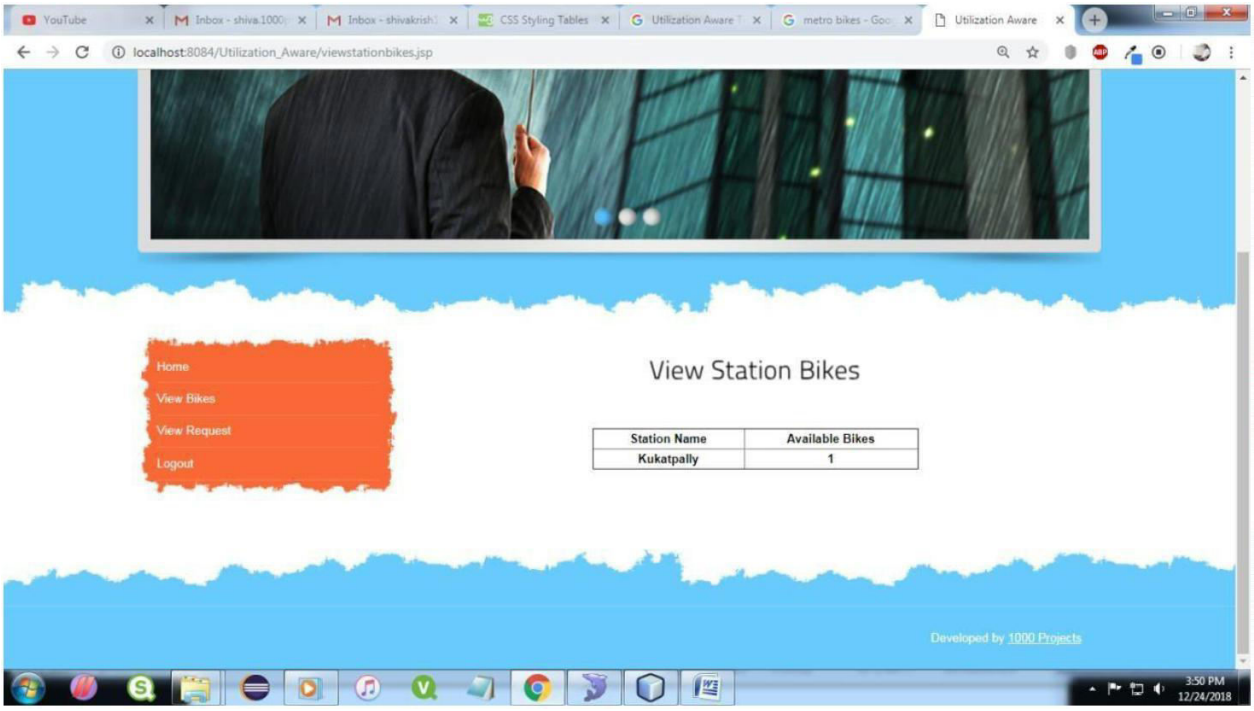
Station Service Login:



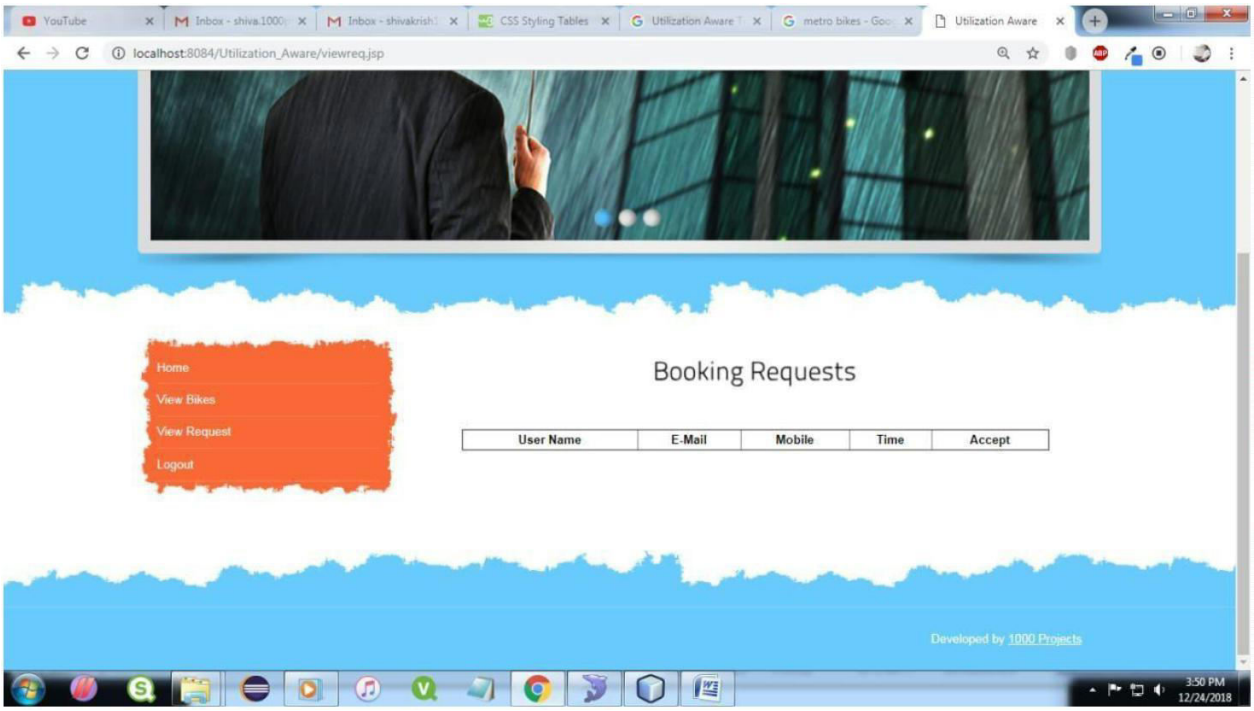
Station Service Home:


localhost:8084/Utilization_Aware/stationhome.jsp?m1=success
Home
View Bikes
View Request
Logout
Welcome to Kukatpally Station
Developed by 1000 Projects

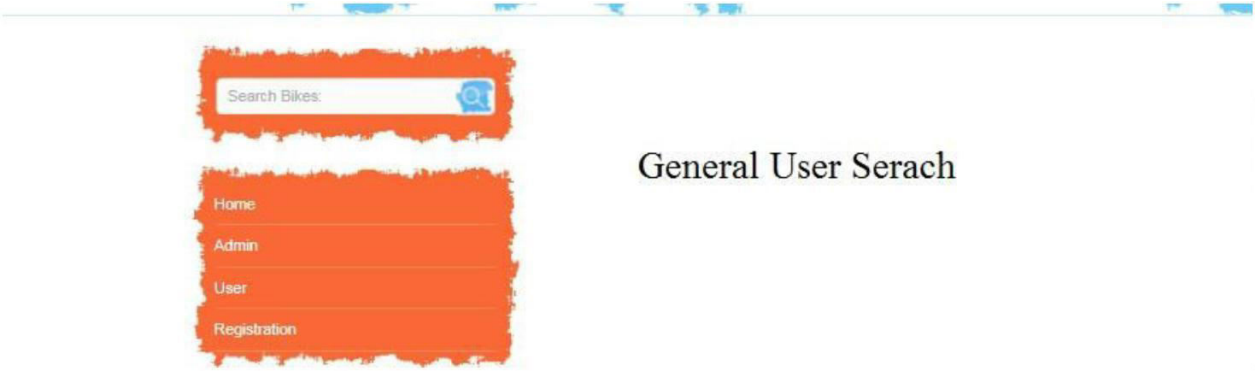
View Bikes:

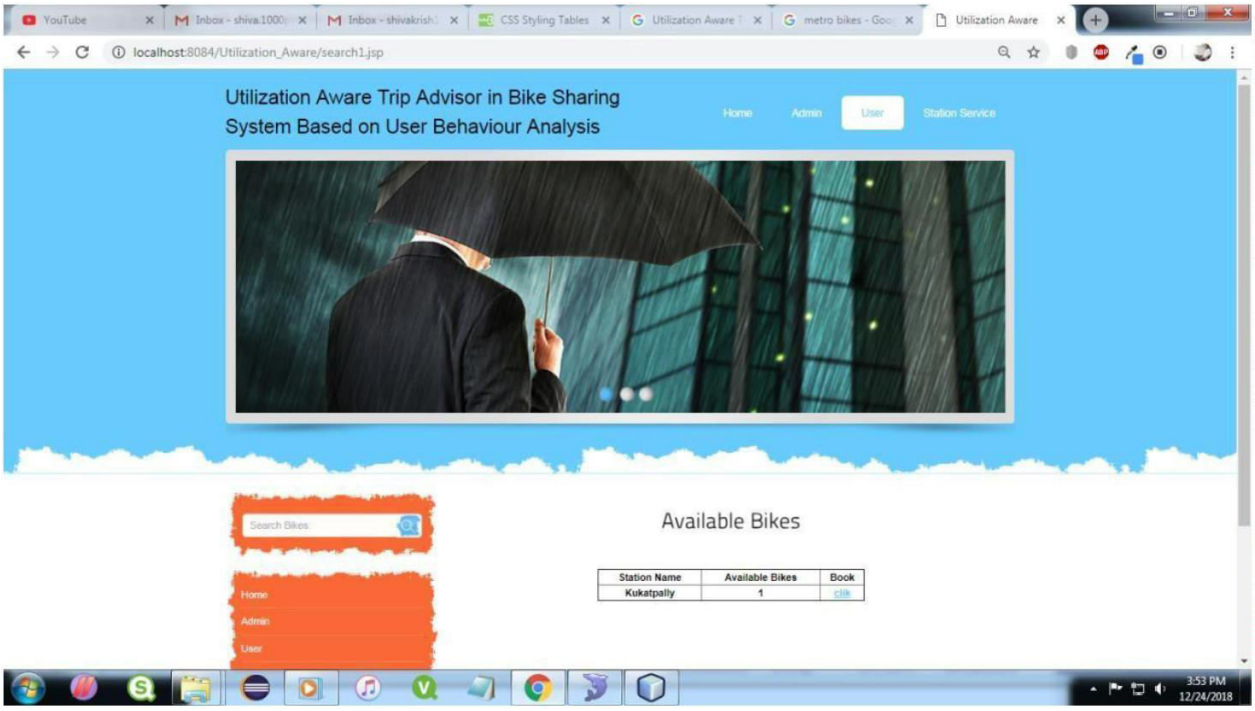

localhost:8084/Utilization_Aware/viewstationbikes.jsp
Home
View Bikes
View Request
Logout
View Station Bikes
Station Name
Available Bikes
Kukatpally
1
Developed by 1000 Projects

View Request:



General User Search:




localhost:8084/Utilization_Aware/search1.jsp
Utilization Aware Trip Advisor in Bike Sharing System Based on User Behaviour Analysis
Home
Admin
User
Station Service
Search Bikes
Home
Admin
User
Available Bikes
Station Name
Available Bikes
Book
Kukatpally
1
click
3:53 PM
12/24/2018

6.CONCLUSION

In this research, we present a novel architecture for a utilization-aware trip adviser that encourages users to balance bike usage and extend bike maintenance intervals based on an analysis of general characteristics, spatial temporal patterns, and bike utilisation in bikesharing. Beginning with assuring users' rental and return success rates, the advisor is made to dynamically suggest the best stations depending on their current level of bike utilisation.We confirmed the efficiency of our framework by thoroughly simulating the suggested system and comparing the results to historical data from the largest bike-sharing system in the world.

7.REFERENCES

[1] P. DeMaio, "Bike-sharing: History, Impacts, Models of Provision, and Future," Journal of Public Transportation,vol. 12, no. DeMaio 2004, pp. 41–56, 2009. [Online].Available: http://www.transitinformatics.org/test/nctr/wp-content/uploads/2010/03/JPT12-4DeMaio.pdf

[2]

[3] P. Midgley, "Bicycle-sharing schemes: enhancing sustainable mobility in urban areas," United Nations, Department of Economic and Social Affairs, pp. 1–12, 2011.

[4]

[5] L. MetroBike, "2016 Year-end wrap-up will appear at the end of January," http://bike-sharing.blogspot.com/2017/01/2016-year-end-wrap-up-will- appear-at.html.

[6]

[7] J. Froehlich, J. Neumann, and N. Oliver, "Sensing and Predicting the Pulse of the City through Shared Bicycling," in IJCAI, 2009.

[8]

[9] Kaltenbrunner, R. Meza, J. Grivolla, J. Codina, and R. Banchs, "Urban Cycles and Mobility Patterns: Exploring and Predicting Trends in a

[10] Bicycle-based Public Transport System,"Pervasive and Mobile Computing, vol. 6, no. 4, pp. 455–466, 2010.