**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No.** | **Experiments** | **Page**  **No** | **Date** |
| **1** | Generate regression model and interpret the result for a given data set. |  |  |
| **2** | Design and develop chat messaging app which is a location-based. |  |  |
| **3** | Demonstrate use of OpenGTS (Open Source GPS Tracking System) |  |  |
| **4** | Develop application demonstrating Human Computer Interaction. (Using Key Listener) |  |  |
| **5** | Develop application demonstrating Human Computer Interaction. (Using Mouse Listener) |  |  |
| **6** | Write a Java Card applet |  |  |

**Practical No.1**

**Aim: Design and develop location based messaging app.**

**Theory:**

**Source Code:**

**activity\_main.xml**

**activity\_main.xml**

## *<?*xml version="1.0" encoding="utf-8"*?>*

<**RelativeLayout xmlns:android="**[**http://schemas.android.com/apk/res/android**](http://schemas.android.com/apk/res/android)**" xmlns:tools="**[**http://schemas.android.com/tools**](http://schemas.android.com/tools)**" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:paddingBottom="@dimen/activity\_vertical\_margin" android:paddingLeft="@dimen/activity\_horizontal\_margin" android:paddingRight="@dimen/activity\_horizontal\_margin" android:paddingTop="@dimen/activity\_vertical\_margin" tools:context="com.example.dell.lbmsg.MainActivity"**>

## <TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

**android:textAppearance="?android:attr/textAppearanceLarge" android:text="Large Text"**

**android:id="@+id/textView" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true"** />

## <TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

**android:textAppearance="?android:attr/textAppearanceLarge" android:text="Large Text"**

**android:id="@+id/textView3" android:layout\_below="@+id/textView" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:layout\_marginTop="96dp"** />

## <Button android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Get Longitude and Latitude" android:id="@+id/button" android:layout\_centerVertical="true" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" />

## <TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

**android:textAppearance="?android:attr/textAppearanceLarge" android:text="Large Text"**

**android:id="@+id/textView2" android:layout\_alignParentBottom="true" android:layout\_alignRight="@+id/button" android:layout\_alignEnd="@+id/button"** />

## </RelativeLayout>

**MainActivity.java**

**package** com.example.dell.lbmsg;

**import** android.Manifest;

**import** android.content.Context;

**import** android.content.pm.PackageManager;

**import** android.location.Location;

**import** android.location.LocationListener; **import** android.location.LocationManager; **import** android.net.Uri;

**import** android.os.Bundle;

**import** android.support.v7.app.AppCompatActivity; **import** android.support.v4.app.ActivityCompat; **import** android.util.Log;

**import** android.view.View; **import** android.widget.Button; **import** android.widget.TextView;

**import** com.google.android.gms.appindexing.Action;

**import** com.google.android.gms.appindexing.AppIndex;

**import** com.google.android.gms.common.api.GoogleApiClient;

## public class MainActivity extends AppCompatActivity implements LocationListener {

TextView **t1**, **t2**, **t3**; Button **b1**;

**protected** LocationManager **locationManager**; **protected** LocationListener **locationListener**; **double lat**, **longg**;

**private** GoogleApiClient **client**; @Override

**protected void** onCreate(Bundle savedInstanceState) { **super**.onCreate(savedInstanceState); setContentView(R.layout.***activity\_main***);

**t1** = (TextView) findViewById(R.id.***textView***); **t2** = (TextView) findViewById(R.id.***textView3***); **t3** = (TextView) findViewById(R.id.***textView2***); **b1** = (Button) findViewById(R.id.***button***);

**b1**.setOnClickListener(**new** View.OnClickListener() { @Override

**public void** onClick(View view) { **t1**.setText(**"Latitude = "** + **lat**); **t2**.setText(**"Longitude = "** + **longg**);

**if** (**lat**<38 &&**lat**>36 &&**longg**<122 &&**longg**>118) {

**t3**.setText(**"In-Side The Area"**);

} **else** {

**t3**.setText(**"Out-Side The Area"**);

}

}

});

**locationManager**= (LocationManager) getSystemService(Context.***LOCATION\_SERVICE***);

**if** (ActivityCompat.*checkSelfPermission*(**this**, Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***&&ActivityCompat.*checkSelfPermission*(**this**, Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {

## return;

}

**locationManager**.requestLocationUpdates(LocationManager.***GPS\_PROVIDER***, 0, 0,

**this**);

**client** = **new** GoogleApiClient.Builder(**this**).addApi(AppIndex.***API***).build();

}

@Override

**public void** onLocationChanged(Location location) {

*// txtLat = (TextView) findViewById(R.id.textview1);*

**lat**= location.getLatitude(); **longg**= location.getLongitude(); Log.*d*(**""** + **lat**, **""** + **lat**);

Log.*d*(**""** + **longg**, **""** + **longg**);

**if** (**lat**== 38 &&**longg**== 118) {

**t3**.setText(**"You Are at Perfect Place !!!!"**);

} **else** {

## t3.setText("You are not at Perfect Place !!!!

}

@Override

**public void** onStatusChanged(String provider, **int** status, Bundle extras)

{ Log.*d*(**"Latitude"**, **"status"**);

}

@Override

**public void** onProviderDisabled(String provider) { Log.*d*(**"Latitude"**, **"disable"**);

}

@Override

**public void** onProviderEnabled(String provider)

{ Log.*d*(**"Latitude"**, **"enable"**);

}

@Override

**public void** onStart()

{ **super**.onStart();

// *ATTENTION: This was auto-generated to implement the App Indexing API.*

// *See https://g.co/AppIndexing/AndroidStudio for more information.*

**client**.connect();

Action viewAction = Action.*newAction*( Action.***TYPE\_VIEW***,

*//* ***TODO: choose an action type.***

**"Main Page"**, *//* ***TODO: Define a title for the content shown.***

// ***TODO: If you have web page content that matches this app activity's content,***

// *make sure this auto-generated web page URL is correct.*

// *Otherwise, set the URL to null.*

Uri.*parse*(**"**[**http://host/path"**](http://host/path)),

*//* ***TODO: Make sure this auto-generated app URL is correct.***

## Uri.*parse*("android-app://com.example.dell.lbmsg/http/host/path")

);

AppIndex.***AppIndexApi***.start(**client**, viewAction);

}

@Override

**public void** onStop()

{

**super**.onStop();

// *ATTENTION: This was auto-generated to implement the App Indexing API.*

// *See https://g.co/AppIndexing/AndroidStudio for more information.*

Action viewAction = Action.*newAction*( Action.***TYPE\_VIEW***, *//* ***TODO: choose an action type.***

**"Main Page"**, *//* ***TODO: Define a title for the content shown.***

// ***TODO: If you have web page content that matches this app activity's content,***

// *make sure this auto-generated web page URL is correct.*

// *Otherwise, set the URL to null.*

Uri.*parse*(**"**[**http://host/path"**](http://host/path)),

*//* ***TODO: Make sure this auto-generated app URL is correct.***

## Uri.*parse*("android-app://com.example.dell.lbmsg/http/host/path")

);

AppIndex.***AppIndexApi***.end(**client**, viewAction);

**client**.disconnect();}

}

**AndroidManifest.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*

<**manifest xmlns:android=**[**"http:/**](http://schemas.android.com/apk/res/android)**/**[**schemas.android.com/apk/res/android**](http://schemas.android.com/apk/res/android)**" package="com.example.dell.lbmsg"**>

## <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

## <uses-permission android:name="android.permission.INTERNET"/>

## <application android:allowBackup="true" android:icon="@mipmap/ic\_launcher" android:label="@string/app\_name" android:supportsRtl="true" android:theme="@style/AppTheme">

<**activity android:name=".MainActivity"**>

## <intent-filter>

<**action android:name="android.intent.action.MAIN"** />

## <category android:name="android.intent.category.LAUNCHER"/>

</**intent-filter**>

## </activity>

## <meta- data

## android:name="com.google.android.gms.version" android:value="@integer/google\_play\_services\_version"/>

</**application**>

</**manifest**>

## Output:

**Practical No.2**

**Aim: Design and develop chat messaging app which is a location-based**

**Theory:**

## Source Code :

**activity\_main.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*

<RelativeLayout xmlns:android=["htt](http://schemas.android.com/apk/res/android)p[://schemas.android.com/apk/res/android"](http://schemas.android.com/apk/res/android) xmlns:tools=["htt](http://schemas.android.com/tools)p[://schemas.android.com/tools](http://schemas.android.com/tools)" android:layout\_width="match\_parent" android:layout\_height="match\_parent" android:paddingBottom="@dimen/activity\_vertical\_margin" android:paddingLeft="@dimen/activity\_horizontal\_margin" android:paddingRight="@dimen/activity\_horizontal\_margin" android:paddingTop="@dimen/activity\_vertical\_margin" tools:context="com.example.dell.location\_based.MainActivity">

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge" android:text="Large Text"

android:id="@+id/textView" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" />

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge" android:text="Large Text"

android:id="@+id/textView3" android:layout\_below="@+id/textView" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" android:layout\_marginTop="96dp" />

<Button

android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:text="Get Longitude and Latitude" android:id="@+id/button" android:layout\_centerVertical="true" android:layout\_alignParentLeft="true" android:layout\_alignParentStart="true" />

<TextView android:layout\_width="wrap\_content" android:layout\_height="wrap\_content"

android:textAppearance="?android:attr/textAppearanceLarge" android:text="Large Text"

android:id="@+id/textView2" android:layout\_alignParentBottom="true" android:layout\_alignRight="@+id/button" android:layout\_alignEnd="@+id/button" />

</RelativeLayout>

**MainActivity.java**

**package** com.example.dell.location\_based;

**import** android.Manifest;

**import** android.content.Context;

**import** android.content.pm.PackageManager;

**import** android.location.Location;

**import** android.location.LocationListener; **import** android.location.LocationManager; **import** android.support.v4.app.ActivityCompat; **import** android.util.Log;

**import** android.view.View; **import** android.widget.Button; **import** android.widget.TextView;

**import** android.support.v7.app.AppCompatActivity;

**import** android.os.Bundle;

## public class MainActivity extends AppCompatActivity implements LocationListener { TextView t1, t2, t3;

Button **b1**;

**protected** LocationManager **locationManager**; **protected** LocationListener **locationListener**; **double lat**, **longg**;

@Override

**protected void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.***activity\_main***); *// setContentView(R.layout.activity\_main);*

**t1** = (TextView) findViewById(R.id.***textView***); **t2** = (TextView) findViewById(R.id.***textView3***); **t3** = (TextView) findViewById(R.id.***textView2***); **b1** = (Button) findViewById(R.id.***button***);

**b1**.setOnClickListener(**new** View.OnClickListener()

{ @Override

**public void** onClick(View view) { **t1**.setText(**"Latitude = "** + **lat**); **t2**.setText(**"Longitude = "**+ **longg**);

**if**(**lat**<38 &&**lat**>36 &&**longg**<122 &&**longg**>118)

{

**t3**.setText(**"In-Side The Area"**);

}

## else {

**t3**.setText(**"Out-Side The Area"**);

} }

});

**locationManager**= (LocationManager) getSystemService(Context.***LOCATION\_SERVICE***);

**if** (ActivityCompat.*checkSelfPermission*(**this**, Manifest.permission.***ACCESS\_FINE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***&&ActivityCompat.*checkSelfPermission*(**this**, Manifest.permission.***ACCESS\_COARSE\_LOCATION***) != PackageManager.***PERMISSION\_GRANTED***) {

## return;

}

**locationManager**.requestLocationUpdates(LocationManager.***GPS\_PROVIDER***, 0,

0, **this**);

}

@Override

**public void** onLocationChanged(Location location)

{ **lat**=location.getLatitude(); **longg**= location.getLongitude(); Log.*d*(**""**+**lat**,**""**+**lat**);

Log.*d*(**""**+**longg**,**""**+**longg**);

**if**(**lat**<38 &&**lat**>36 &&**longg**<122 &&**longg**>118) {

**t3**.setText(**"In-Side The Area"**);

}

## else {

**t3**.setText(**"Out-Side The Area"**);

}

}

@Override

**public void** onProviderDisabled(String provider) { Log.*d*(**"Latitude"**,**"disable"**);

}

@Override

**public void** onProviderEnabled(String provider)

{ Log.*d*(**"Latitude"**,**"enable"**);

}

@Override

**public void** onStatusChanged(String provider, **int** status, Bundle extras)

{ Log.*d*(**"Latitude"**,**"status"**);

}

}

**AndroidManifest.xml**

*<?***xml version="1.0" encoding="utf-8"***?>*

<**manifest xmlns:android=**[**"http:/**](http://schemas.android.com/apk/res/android)**/**[**schemas.android.com/apk/res/android**](http://schemas.android.com/apk/res/android)**" package="com.example.dell.location\_based"**>

<**application android:allowBackup="true" android:icon="@mipmap/ic\_launcher" android:label="@string/app\_name" android:supportsRtl="true" android:theme="@style/AppTh me"**>

<**activity android:name=".MainActivity"**>

<**intent-filter**>

## <action android:name="android.intent.action.MAIN" />

<**category android:name="android.intent.category.LAUNCHER"**/>

</**intent-filter**>

</**activity**>

</**applicatio**>

## <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION"/>

<**uses-permission android:name="android.permission.INTERNET"** />

</**manifest**>

## Output :

**Practical No.3**

**Aim: Demonstrate use of OpenGTS (Open Source GPS Tracking System)..**

**Theory:** OpenGTS, which stands for "Open GPS Tracking System," is an open-source GPS tracking platform designed for monitoring and managing a fleet of vehicles or mobile assets. It provides a set of tools and features for tracking, mapping, and managing location data from GPS-enabled devices, such as vehicles, smartphones, and other assets equipped with GPS receivers.

Key features and components of OpenGTS include:

**GPS Device Integration:** OpenGTS supports a wide range of GPS tracking devices and protocols, making it compatible with various GPS hardware.

**Real-Time Tracking:** The platform allows real-time tracking of vehicles or assets on a map, providing information about their current locations, speed, and status.

**Historical Tracking**: OpenGTS stores historical tracking data, enabling users to review past routes and generate reports on vehicle or asset movements.

**Geofencing:** Users can set up geofences, which are virtual boundaries on the map. OpenGTS can trigger alerts when a tracked object enters or exits a geofenced area.

**Event Notifications:** It supports event-driven notifications, such as speeding alerts, ignition on/off, and device tampering alerts, which can be sent via email or SMS.

**Web-Based Interface**: OpenGTS provides a web-based user interface for easy access and management from any device with internet connectivity.

**Mobile Access:** Some versions of OpenGTS offer mobile apps or mobile-friendly interfaces for tracking on smartphones and tablets.

**Settings:-**

**Required Software:-**

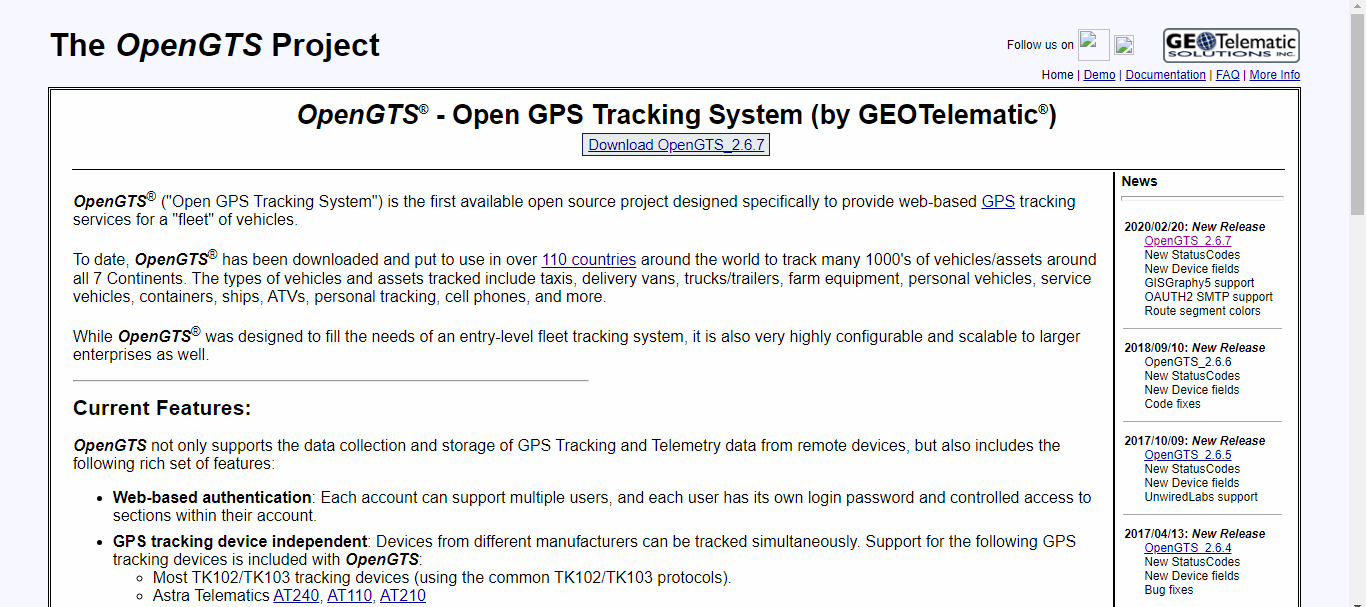
1) JDK 1.6

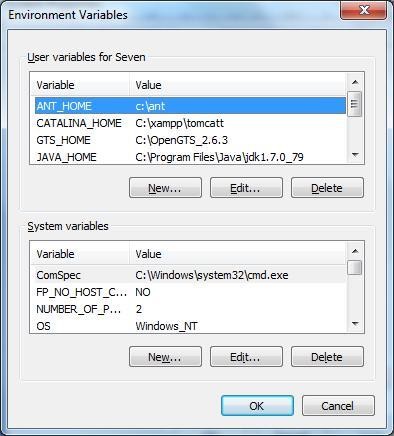
2) XAMPP Server

3) Mysql-java connector

4) OpenGTS application**.**

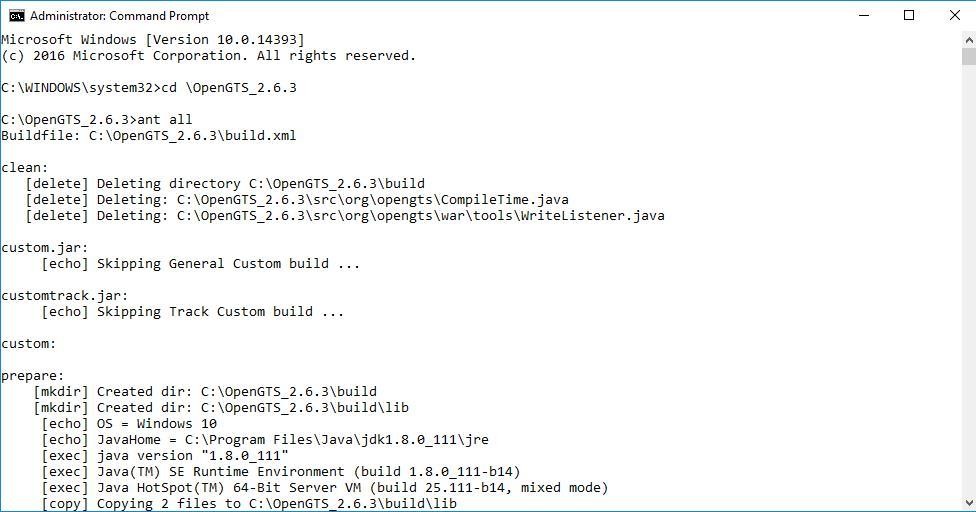
<http://www.opengts.org/>



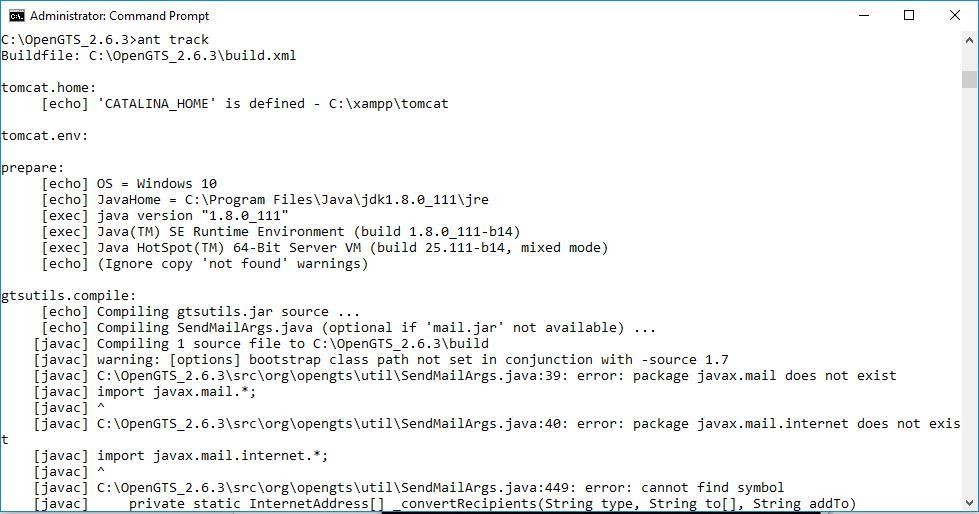


## ADMIN CMD--

1. Open command Prompt and go to D:\OpenGTS\_2.6.2
2. Type command ant all



1. Type command ant track

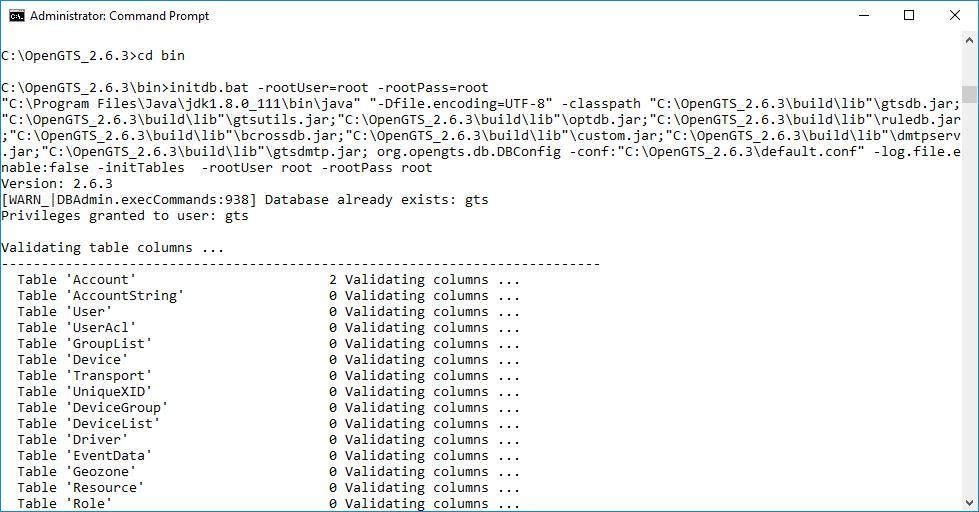


1. Type command ant track.deploy



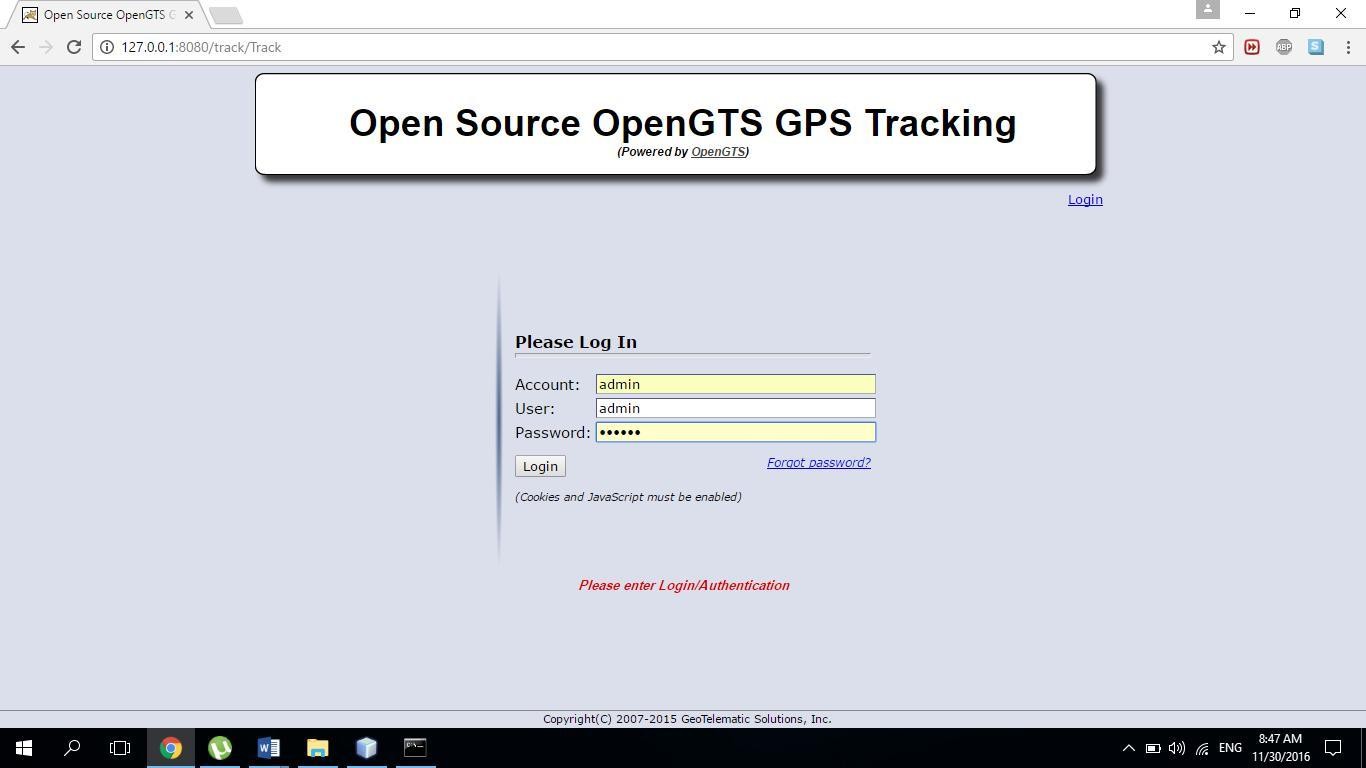
1. (type cd bin)Type command initdb.bat –rootUser=root –rootPass=root

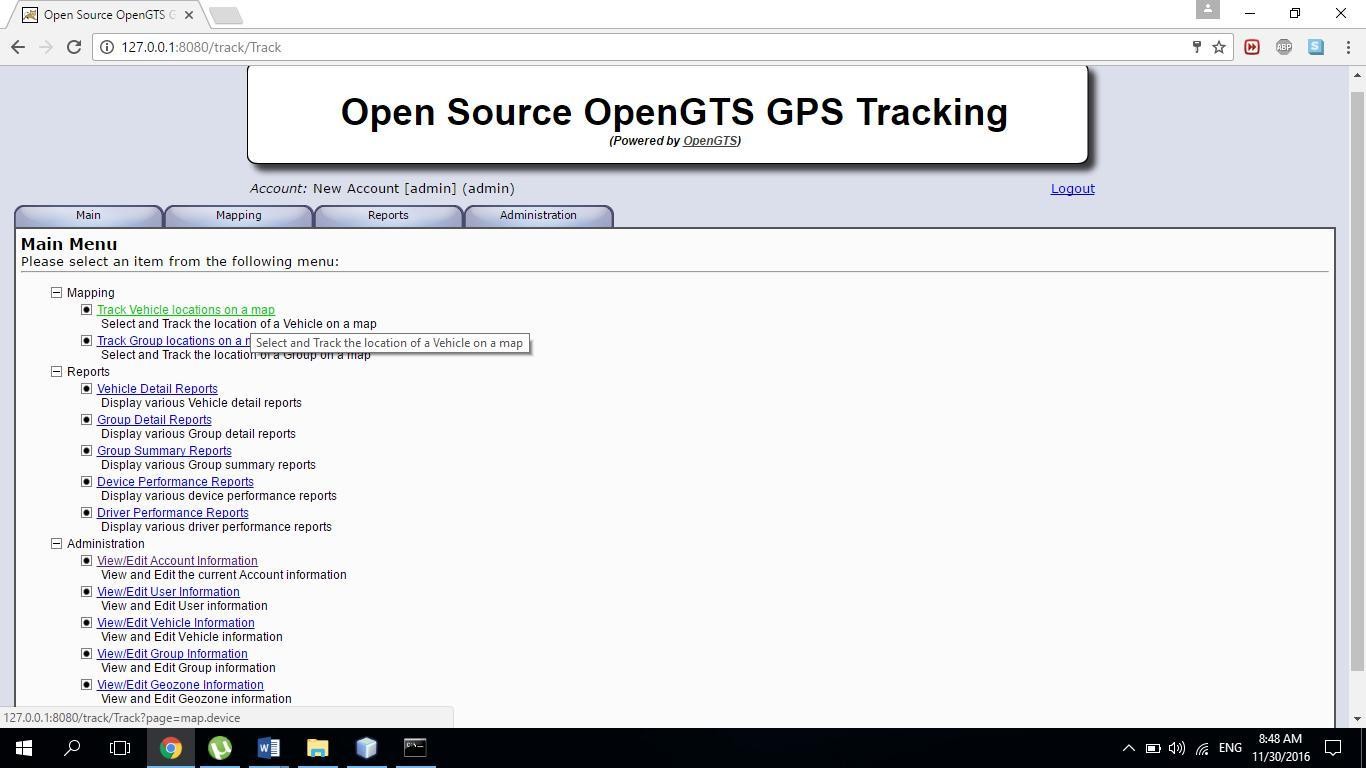
----- start Tomcat/MySQL.

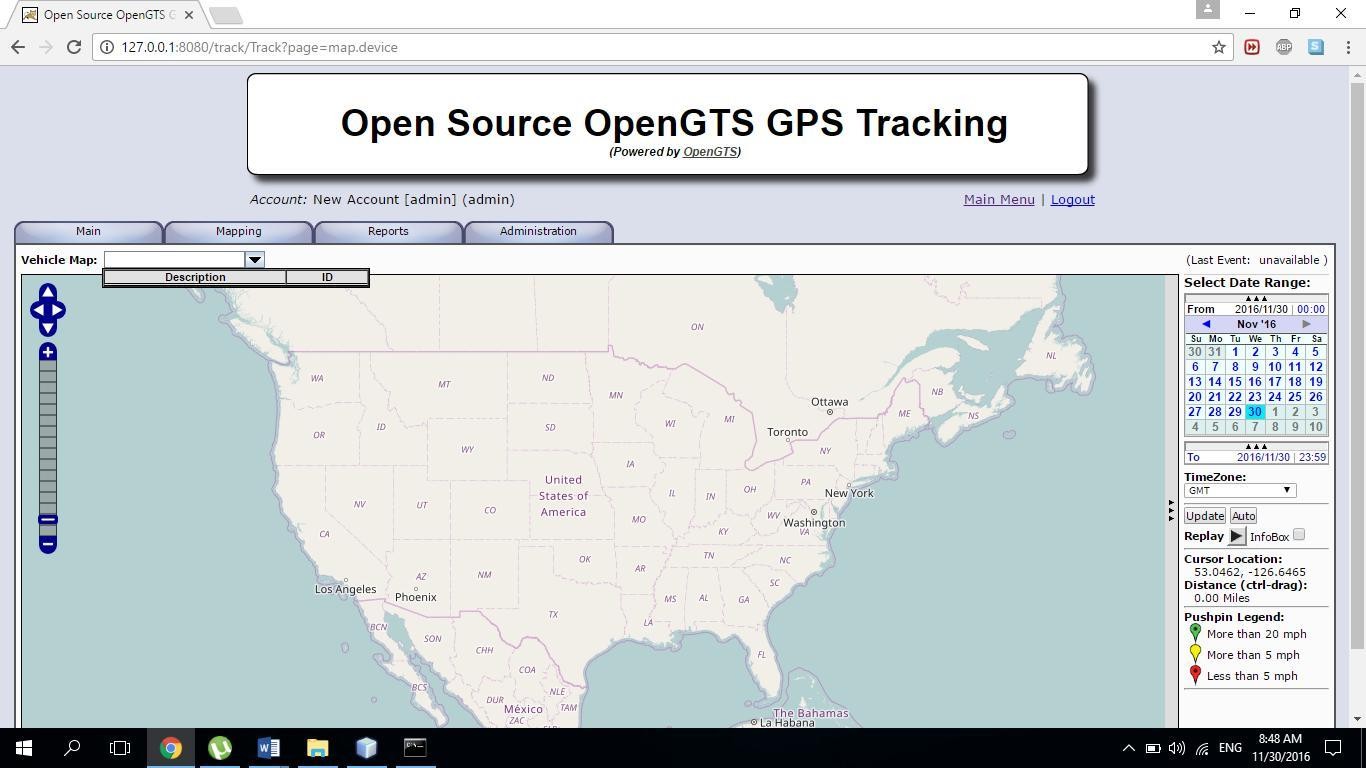


1. **Type Command admin.bat Account –account:admin –pass:123456 –create**



 7) **Type url 127.0.0.1:8080/track/Track and login with admin and 123456**





**Conclusion:**

In conclusion, OpenGTS (Open GPS Tracking System) is a powerful open-source platform for GPS tracking and fleet management. It offers a wide range of features for real-time and historical tracking of vehicles and assets equipped with GPS devices. Some of the key highlights of OpenGTS include its flexibility, support for various GPS hardware and protocols, geofencing capabilities, event notifications, reporting and analytics, and a user-friendly web-based interface.

**Practical No.4**

**Aim: Develop application demonstrating Human Computer Interaction. (Using Key Listener)**

**Theory:** Creating an application demonstrating Human-Computer Interaction (HCI) using a Key Listener can be a valuable project to showcase how users interact with software through keyboard input. Here's a theoretical framework for developing such an application:

**Objective:**

To develop an application that captures and analyzes keyboard input from users, providing insights into HCI patterns, usability, and user behavior.

**Key Features:**

**Key Logging**: Implement a Key Listener to capture keystrokes from the user.

Record timestamps for each key press event.

**User Authentication:**

* Create a login system to ensure data privacy and security.
* Only authorized users should be able to access the application.

**Program:**

package uc\_prac4;

import java.awt.\*;

import java.awt.event.\*;

public class UC\_Prac4 extends Frame implements KeyListener {

Label l;

TextArea area;

UC\_Prac4() {

l = new Label();

l.setBounds(20, 50, 100, 20);

area = new TextArea();

area.setBounds(20, 80, 300, 300);

area.addKeyListener(this);

add(l);

add(area);

setSize(400, 400);

setLayout(null);

setVisible(true);

}

@Override

public void keyPressed(KeyEvent e) {

l.setText("Key Pressed");

}

@Override

public void keyReleased(KeyEvent e) {

l.setText("Key Released");

}

@Override

public void keyTyped(KeyEvent e) {

l.setText("Key Typed");

}

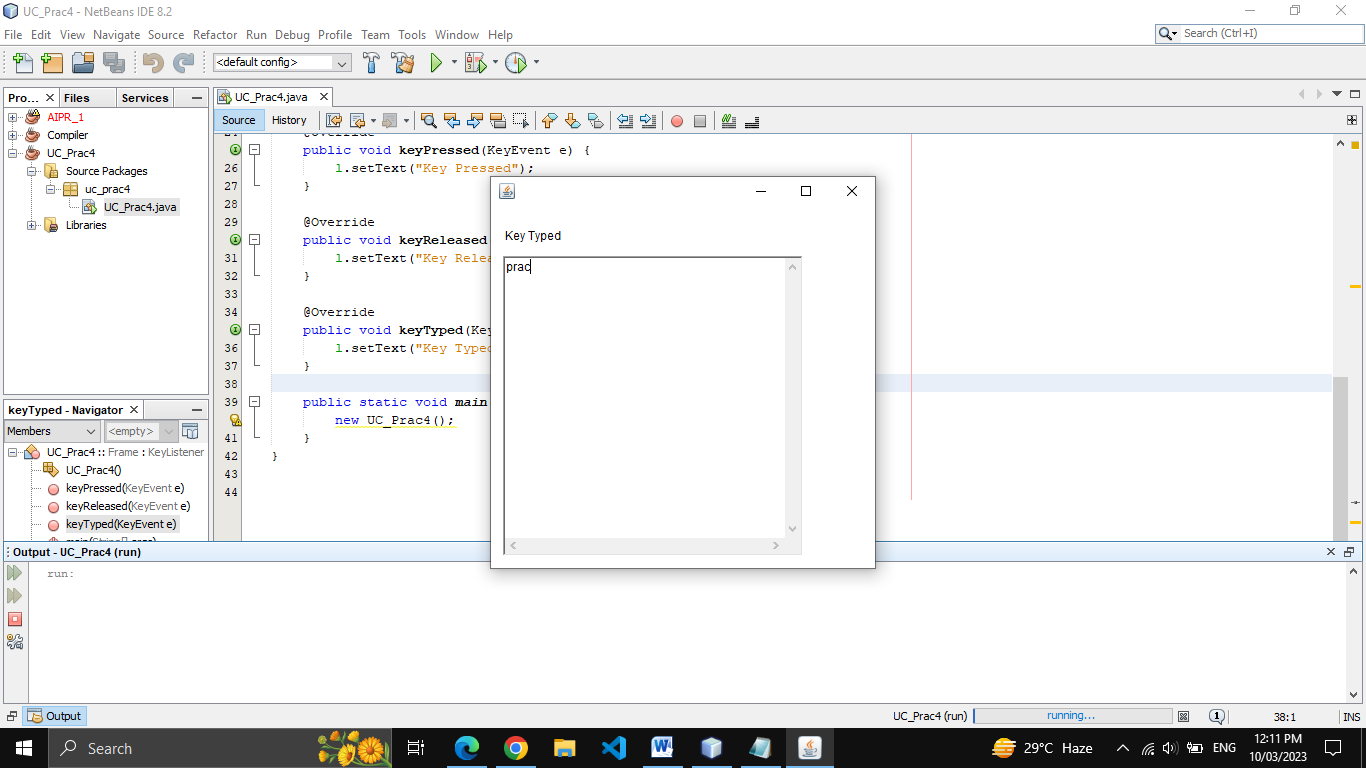
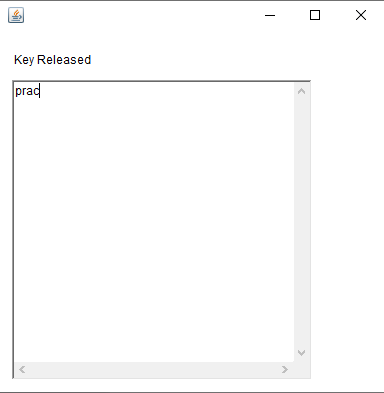
public static void main(String[] args) {

new UC\_Prac4();

}

}

**Output:**

 ****

**Conclusion:**

Developing an HCI application using a Key Listener can provide valuable insights into user behavior and typing patterns. It is essential to prioritize user privacy, security, and ethical considerations throughout the development process. Additionally, on-going testing and evaluation are crucial for refining the application and ensuring it meets user needs effectively.

**Practical No.5**

**Aim: Develop application demonstrating Human Computer Interaction. (Using Mouse Listener)**

**Theory:** Creating an application to demonstrate Human-Computer Interaction (HCI) using a Mouse Listener can be an interesting project. This application will capture and analyze mouse input from users to showcase various aspects of HCI. Here's a theoretical framework for developing such an application:

**Objective:**

To develop an application that captures and analyzes mouse input from users, providing insights into HCI patterns, usability, and user behavior.

**Key Features:**

Mouse Event Capture:

* Implement a Mouse Listener to capture mouse events, including clicks, movements, and scrolling.
* Record timestamps for each mouse event.

**User Authentication:**

* Create a login system to ensure data privacy and security.
* Only authorized users should be able to access the application.

**Program:**

package uc\_prac4;

import java.awt.\*;

import java.awt.event.\*;

public class UC\_Prac5 extends Frame implements MouseListener {

Label l;

UC\_Prac5() {

addMouseListener(this);

l = new Label();

l.setBounds(20, 50, 100, 20);

add(l);

setSize(300, 300);

setLayout(null);

setVisible(true);

}

public void mouseClicked(MouseEvent e) {

l.setText("Mouse Clicked");

}

public void mouseEntered(MouseEvent e) {

l.setText("Mouse Entered");

}

public void mouseExited(MouseEvent e) {

l.setText("Mouse Exited");

}

public void mousePressed(MouseEvent e) {

l.setText("Mouse Pressed");

}

public void mouseReleased(MouseEvent e) {

l.setText("Mouse Released");

}

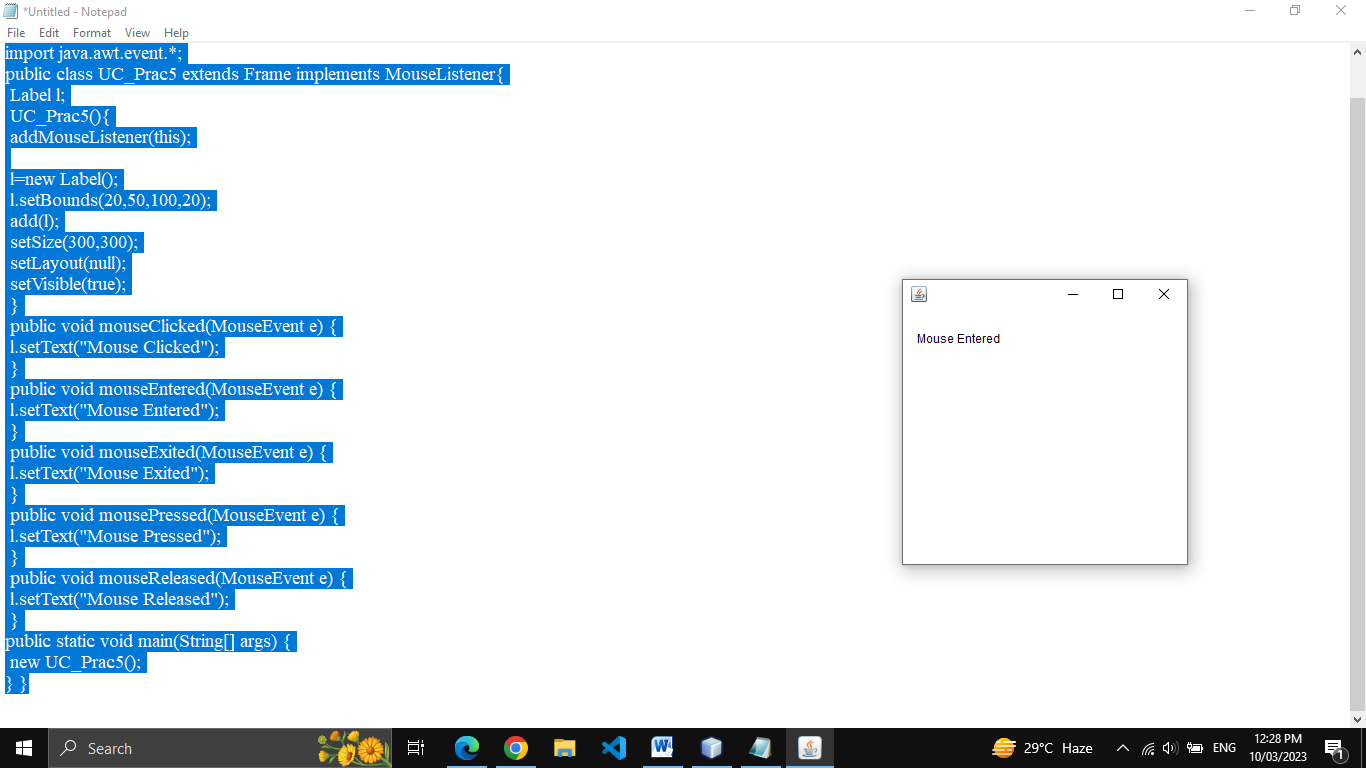
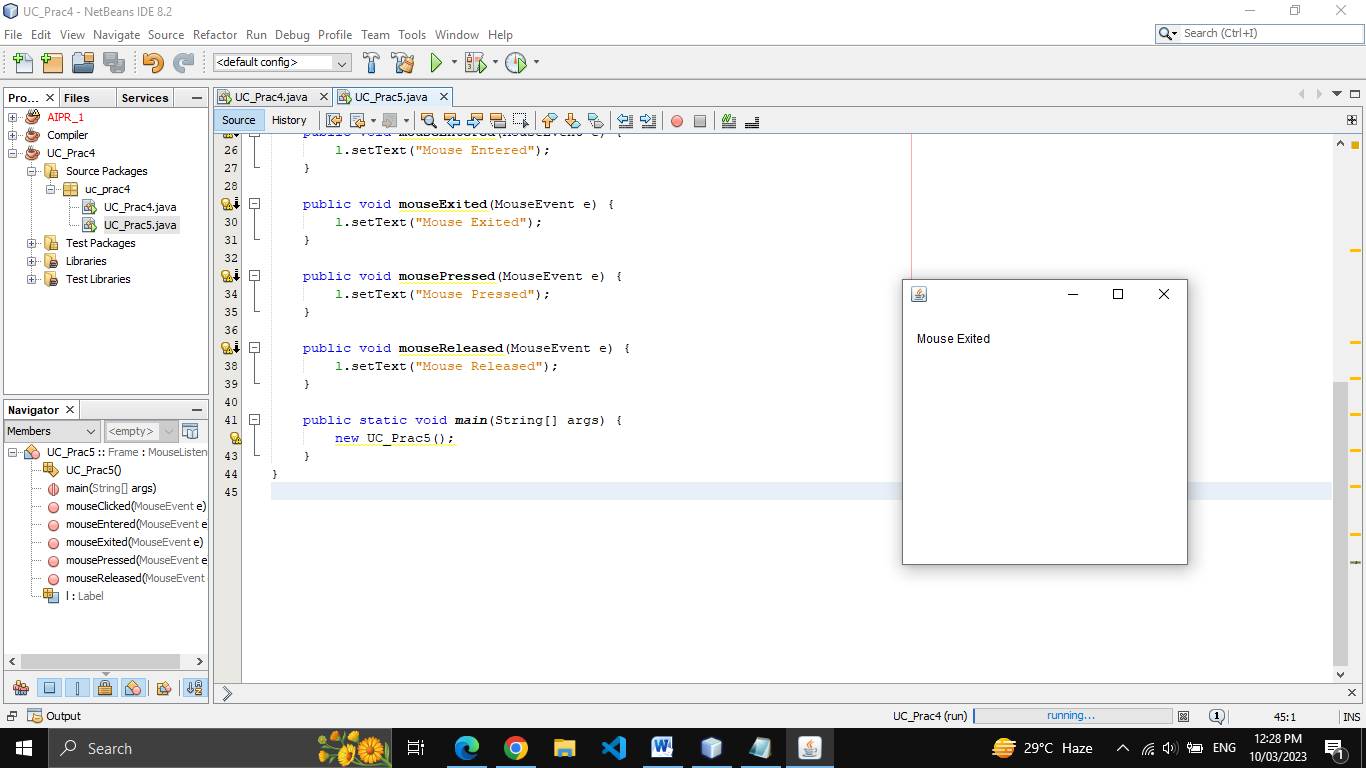
public static void main(String[] args) {

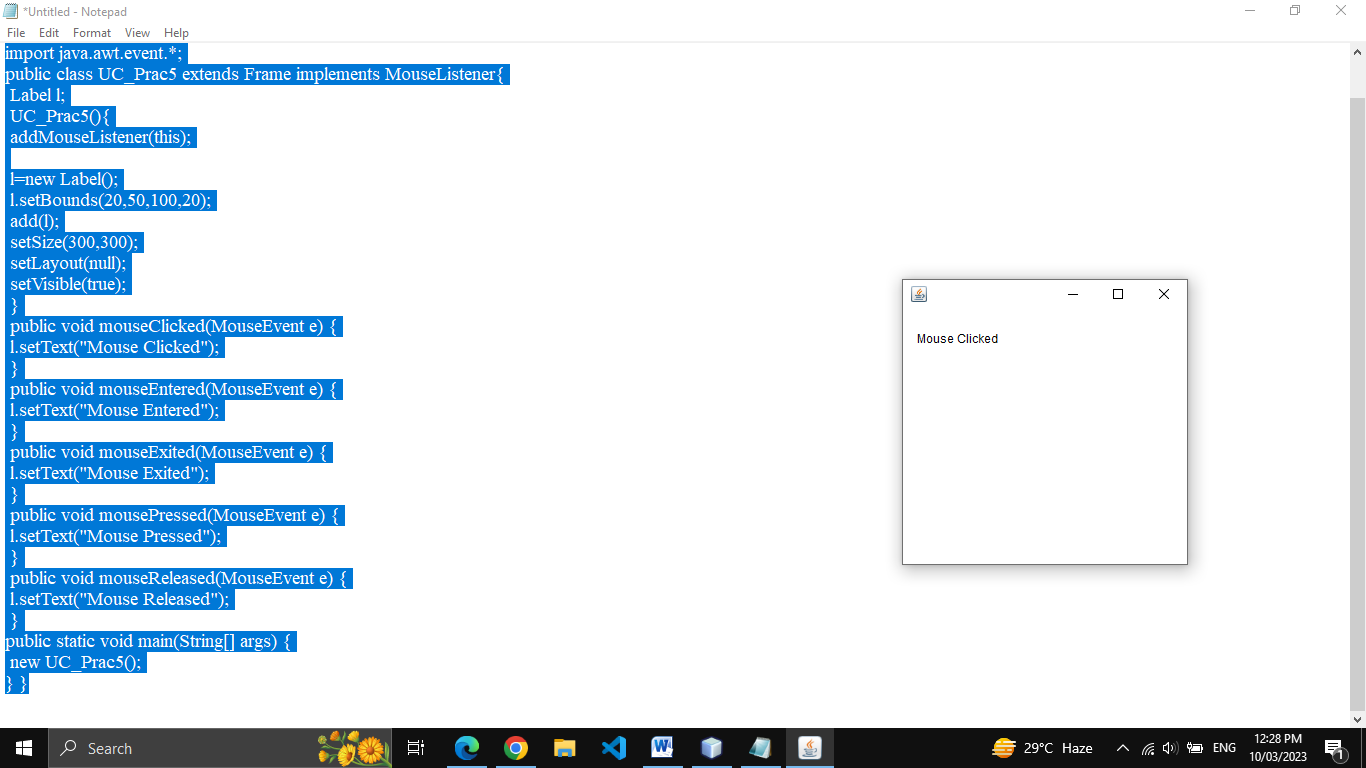
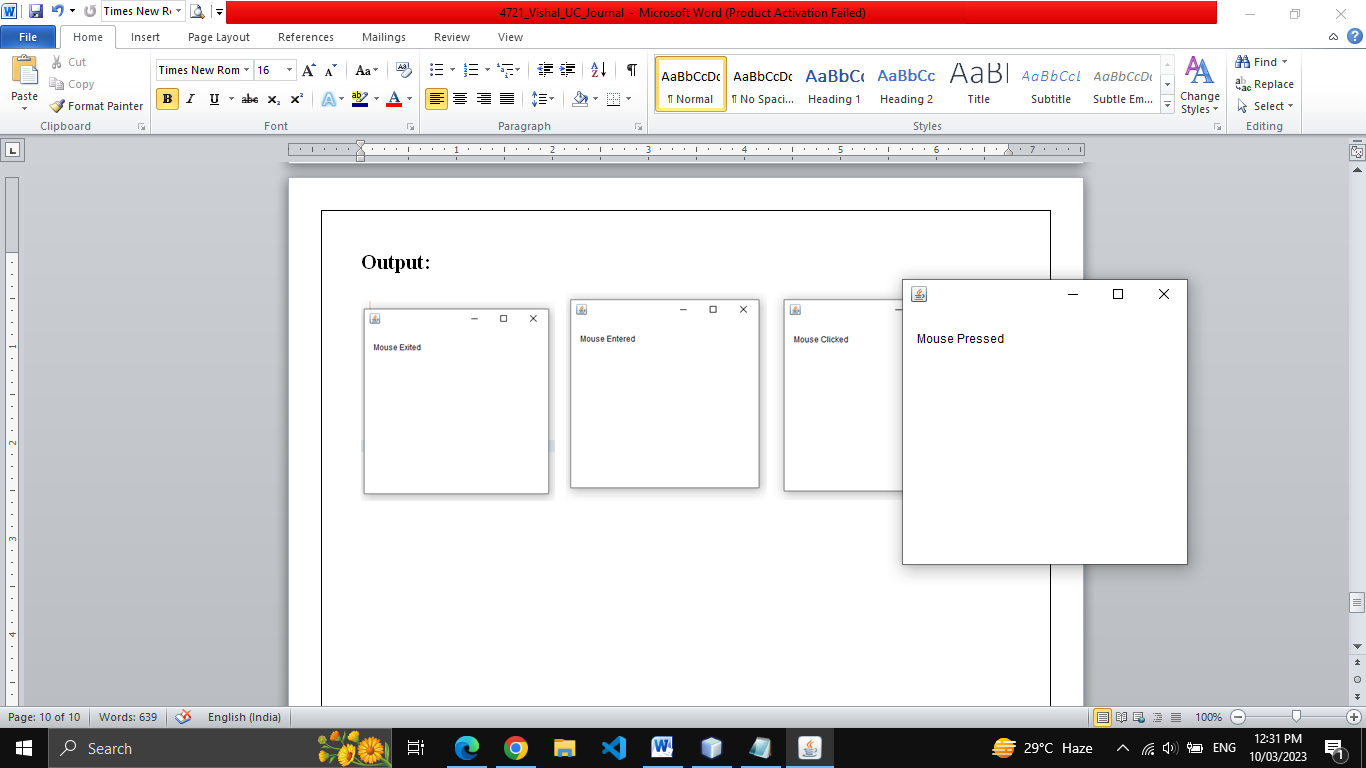
new UC\_Prac5();

}

}

**Output:**



**Conclusion:**

Developing an HCI application using a Mouse Listener can provide valuable insights into user behavior and interaction patterns. It's crucial to prioritize user privacy, security, and ethical considerations throughout the development process. Regular testing and evaluation are essential for refining the application and ensuring it effectively meets user needs.

**Practical No.6**

**Aim: Write a Java Card applet.**

**Theory:** A Java Card applet is a small, Java-based software application that runs on Java Card, a specialized platform for running Java applications on smart cards and other constrained devices with limited resources. Java Card technology allows developers to create secure, portable, and interoperable applications for various smart card and embedded systems.

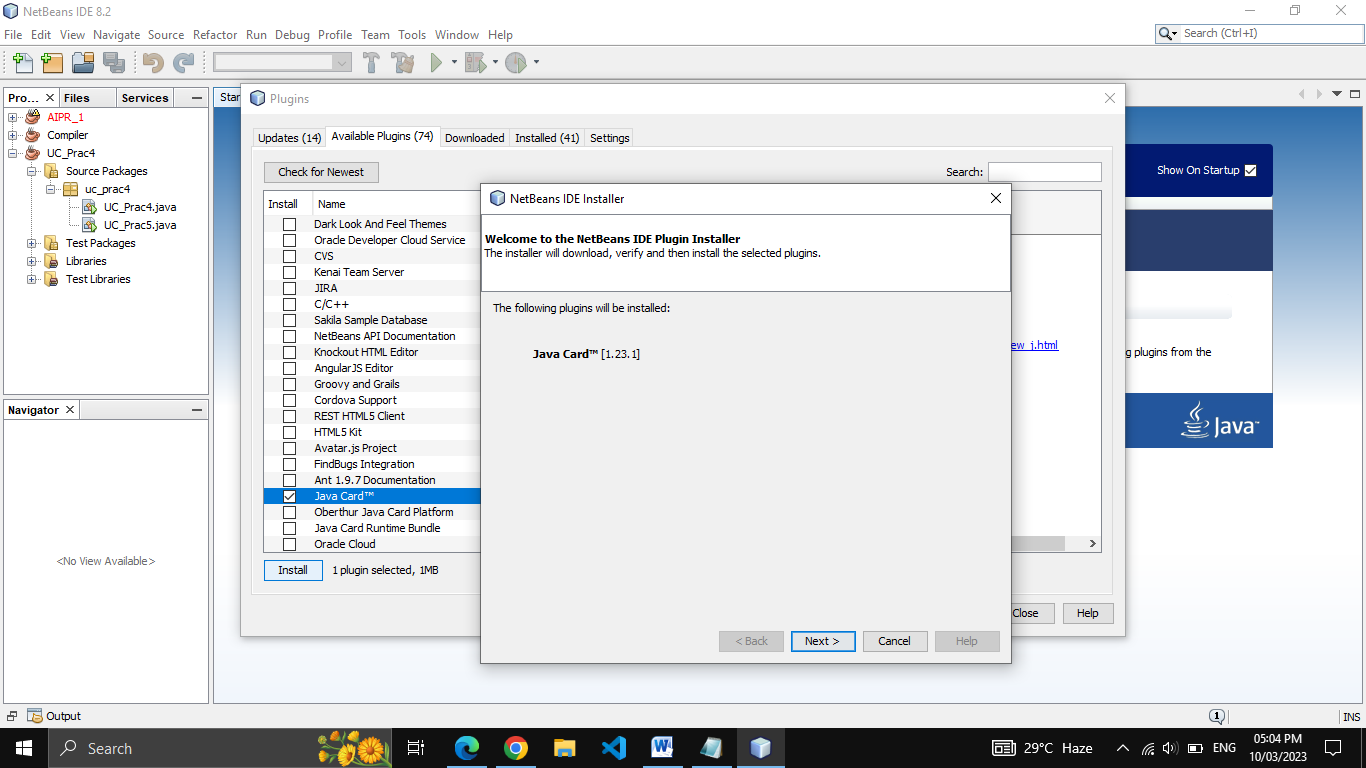
**Steps:**

Download Java Card Sdk from

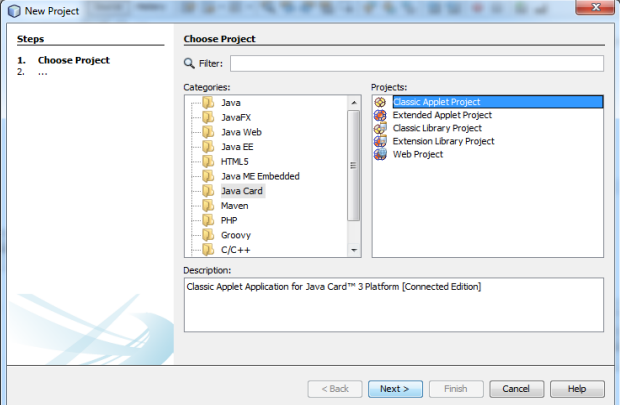
<https://www.oracle.com/java/technologies/javacard-sdk-downloads.html>

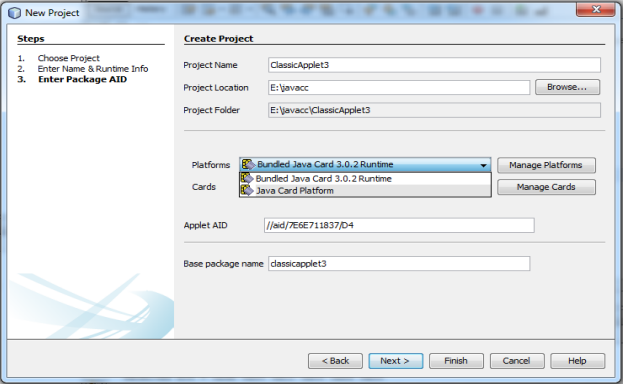
Install in Netbeans as plugin :-

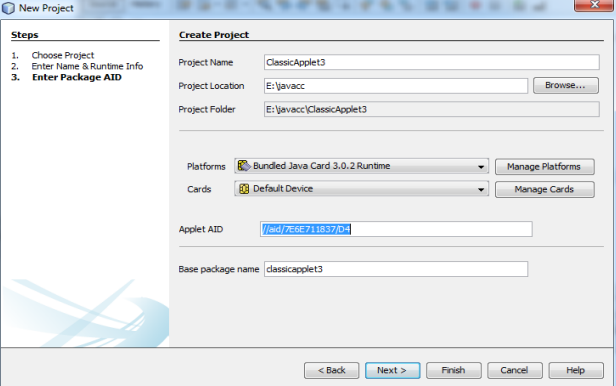
Tools -> Plugin

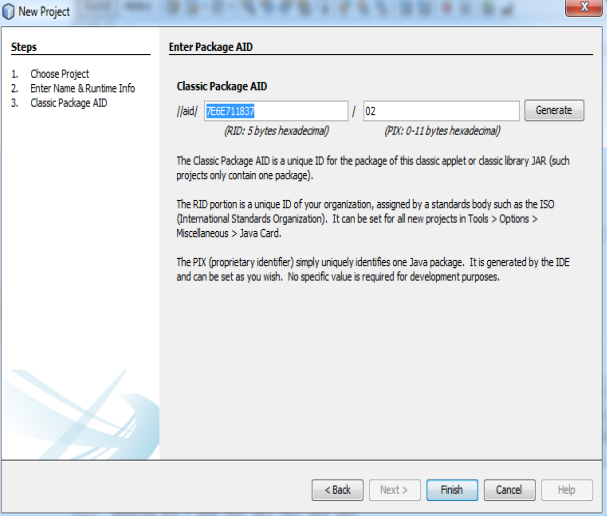


**Create an application**

****

****

****

****

**Program:**

package uc\_prac6;

import javacard.framework.\*;

public class UC\_Prac6 extends Applet {

private static final byte[] helloFidesmo

= {(byte) 'H', (byte) 'e', (byte) 'l', (byte) 'l', (byte) 'o', (byte) ' ', (byte) 'F', (byte) 'i', (byte) 'd', (byte) 'e', (byte) 's', (byte) 'm', (byte) 'o', (byte) '!'};

public static void install(byte[] bArray, short bOffset, byte bLength) {

new UC\_Prac6();

}

protected UC\_Prac6() {

register();

}

public void process(APDU apdu) {

byte[] buffer = apdu.getBuffer();

short length = (short) helloFidesmo.length;

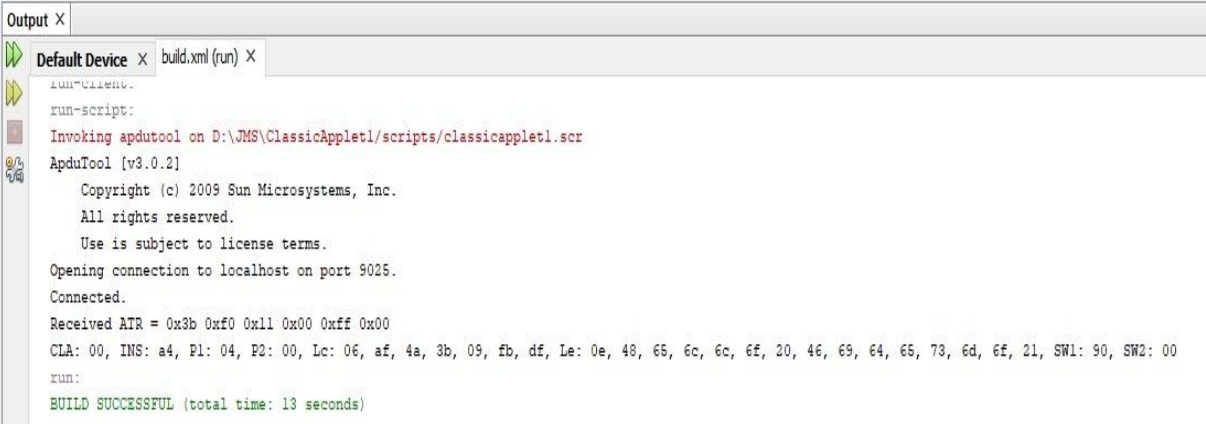
Util.arrayCopyNonAtomic(helloFidesmo, (short) 0, buffer, (short) 0, length);

apdu.setOutgoingAndSend((short) 0, length);

}

}

**Output:**

****

**Conclusion:**

In conclusion, a Java Card applet is a small, secure Java-based software application designed to run on smart cards and other constrained devices. It offers a platform for creating secure and interoperable applications for various purposes, such as payment systems, identity authentication, and secure access control. Key characteristics of Java Card applets include strong security features, well-defined lifecycles, communication through APDU commands, and efficient resource usage.