**Trust based Collaborative Privacy Management in Online Social Networks**

**ABSTRACT:**

Online social networks have now become the most popular platforms for people to share information with others. Along with this, there is a serious threat to individuals’ privacy. One privacy risk comes from the sharing of co-owned data, i.e., when a user shares a data item that involves multiple users, some users’ privacy may be compromised, since different users generally have different opinions on who can access the data. How to design a collaborative management mechanism to deal with such a privacy issue has recently attracted much attention. In this paper, we propose a trust-based mechanism to realize collaborative privacy management. Basically, a user decides whether or not to post a data item based on the aggregated opinion of all involved users. The trust values between users are used to weight users’ opinions, and the values are updated according to users’ privacy loss. Moreover, the user can make a trade-off between data sharing and privacy preserving by tuning the parameter of the proposed mechanism. We formulate the selecting of the parameter as a multi-armed bandit problem and apply the upper confidence bound policy to solve the problem. Simulation results demonstrate that the trust-based mechanism can encourage the user to be considerate of others’ privacy, and the proposed bandit approach can bring the user a high payoff.

I.INTRODUCTION

Online social networks (OSNs), such as Facebook, Google+, and Twitter, have become the most important platforms for people to make social connections with others. Thousands of millions of users post data about their daily lives in terms of text messages, photos, or videos on OSNs. Such data often contain sensitive information of users. If the data can be accessed by unauthorized entities, users’ privacy will be compromised. The privacy issue has always been a major concern in studies related to OSNs. To protect users’ privacy, on one hand, the service providers of OSNs need to take measures to prevent data breach. On the other hand, users themselves can control the access to their data by using the privacy setting function implemented in OSNs. An access control policy, also referred to as the privacy policy, defines which users are allowed to access a user’s data. Current OSNs often utilize user relationship to distinguish between authorized users and unauthorized users. For example, Facebook users can specify if their data can be accessed by friends, specific groups or everyone. The privacy control mechanisms implemented in current OSNs only impose restrictions on users who want to access others’ data. While there is no strict restriction on users who post data. A consequence of this one-side restriction is that the user who posts data may unintentionally violate another users’ privacy

Consider the following example. Suppose that a user A posts a photo of him/her playing with a friend B, and user A specifies that the photo can be accessed by his/her colleagues. If user B considers this photo to be sensitive and user B is not familiar with user A’s colleagues, then user B’s privacy will be violated. In the above case, the photo is actually co-owned by the two users. Hence, the privacy policy specified by user A should be compatible with user B’s privacy policy, otherwise, user B will suffer a loss in privacy. Data which are co-owned by multiple users are quite common in OSNs. Privacy management of such data requires a collaboration of all involved users. The problem of collaborative privacy management in OSNs has attracted much attention in recent years. Most studies deal with this problem by first detecting the conflicts among different users’ privacy policies, and then generating an aggregated policy that can resolve the conflicts to the largest extent. Given a data item (e.g. a photo), a user’s privacy policy is generally represented by a set of users with whom the user wants to share the data. Usually there is a mediator who collects users’ policies and makes a group decision via some aggregation scheme

In most cases, the conflicts among users’ privacy policies cannot be completely eliminated, which means the aggregated policy may still cause a privacy loss to some of the users. How to make a trade-off between data sharing and privacy preserving is an important question for the design of the conflict resolution method. Different from previous studies which rely on a mediator to coordinate among multiple users, in this project we assume that it is the user who wants to post data makes a collective decision based on other users’ privacy requirements. Previous studies usually assume that the user who posts the data will tag all the users involved, or the involved users can be identified via some technique (e.g. face reorganization). In such a case, the mediator is able to notify the involved users about the posting of the data. However, in practice, it is likely that the user posts the data without tagging other users and the users are hard to be identified automatically.

Considering this, I propose a mechanism which requires the user to solicit other users’ opinions before posting data. And a trust-weighted voting scheme is applied to aggregate different users ‘opinions. Specifically, given the data item that a user wants to post and the privacy policy specified by the user, every involved user makes a “vote” to state whether he/she approves of the privacy policy. The importance of the vote depends on the trust value between the two users. Only when the aggregation of the votes satisfies a certain condition, the data can be posted. Moreover, the trust values between users are not fixed

A user will lose the trust of others if he/she posts a data item that incurs privacy loss of others. Also, a user can gain more trust from others if he/she adopts others’ opinions. The interaction between the trust value and the privacy loss implies that if the user wants to reduce his/her privacy loss, then when posting a co-owned data item, the user should always consider others’ privacy requirements rather than taking a unilateral decision. In the proposed trustbased privacy management mechanism, I introduce a threshold based on which the user makes the final decision on data posting. Simply speaking, a high threshold indicates that the user has a relatively low tendency to share the data with others, and only when the majority of the involved users or users that are highly trusted agree to post the data, the data can finally be posted. By tuning the threshold, the user can make a trade-off between data sharing and privacy preserving. Considering that a user continually posts data items in an OSN, I model the threshold selecting problem as a sequential decisionmaking problem. More specifically, I formulate the problem’s multi-armed bandit problem and apply the upper confidence bound (UCB) policy to solve the problem.

**System Analysis**

## EXISTING SYSTEM:

* In Existing System Rathore and Tripathy proposed a trust-based access control method which utilizes the trust values to define access conditions. That is, a user can specify the minimum trust level that is required for another user to access his/her data.
* In previous scheme, Sun et al. proposed a trust-weighted voting scheme to aggregate different users’ privacy policies. In this paper, we also use trust values to indicate how much influence a user’s opinion will have on the aggregated decision. While, different from Sun et al.’s work where the trust values are fixed, the trust values in the proposed mechanism are related to users’ privacy loss, and hence they change over time.

## DISADVANTAGES OF EXISTING SYSTEM:

* In most cases, the conflicts among users’ privacy policies cannot be completely eliminated, which means the aggregated policy may still cause a privacy loss to some of the users.
* It is likely that the user posts the data without tagging other users and the involved users are hard to be identified automatically.

## PROPOSED SYSTEM:

* We propose a mechanism which requires the user to solicit other users’ opinions before posting data. And a trust-weighted voting scheme is applied to aggregate different users’ opinions.
* A trust-based mechanism is proposed for collaborative privacy management in OSNs.
* In the proposed trust-based privacy management mechanism, we introduce a threshold based on which the user makes the final decision on data posting. Simply speaking, a high threshold indicates that the user has a relatively low tendency to share the data with others, and only when the majority of the involved users or users that are highly trusted agree to post the data, the data can finally be posted

## ADVANTAGES OF PROPOSED SYSTEM:

* The trust values between users are associated with users’ privacy loss, and the proposed mechanism can encourage users to be more considerate of other users’ privacy.
* A bandit approach is proposed to adjust the parameter of the trust-based mechanism. By applying the UCB policy, the user can make a rational trade-off between data sharing and privacy preserving.
* The performance of the proposed methods is evaluated via a series of simulations. By conducting comparison among different methods, we demonstrate the advantage of the proposed methods.

**IMPLEMENTATION**

**MODULES:**

* Trust-Weighted Voting Scheme.
* Dataset and Experiment Setting.
* Trust-based Privacy Management

**MODULE DESCRIPTION:**

Given a data item *d*, the corresponding owner *od*and the set of stakeholders *Sd*, we considering the following two cases:

1. *The owner directly posts the data without asking the stakeholders for permission.*
2. *The owner solicits the stakeholders’ opinions before posting the data.*

***The owner directly posts the data without asking the stakeholders for permission:***

In such a case, it is very likely that the stakeholders’ privacy will be disclosed. Suppose that all the stakeholders can perceive the privacy disclosure (if exists) after *d*is posted. According to (3), the stakeholder *s 2 Sd*needs to know *Uao*to evaluate the privacy loss *ls*. However, generally the set *Uao*can not be fully observed by *s*. Only the owner *od*and the service provider know exactly which users are authorized by the owner. Considering this, we propose to use a binary number to indicate the privacy loss: if the stakeholder *s*feels that his/her privacy is violated, then *ls*= 1, otherwise *ls*= 0. If *ls >*0, then the trust of the stakeholder in the owner will decrease. A high privacy loss causes a large decrease in trust. Let *tso*denote the trust of *s*in *od*before *d*is posted. After *d*is posted, the new trust value *t0so*is computed by

***The owner solicits the stakeholders’ opinions before posting the data:***

In such a case, whether a stakeholder will suffer a privacy loss depends on the final decision made by the owner. We assume that the owner adopts a voting scheme to aggregate the stakeholders’ opinions. Before posting the data item *d*, the owner tells all the stakeholders in *Sd*his/her privacy policy in terms of relationship types. It should be noted that if the owner has tagged all the stakeholder in *d*, then the service provider can act as an agent of the owner to notify the stakeholders. Each stakeholder *s 2 Sd*, based on his/her own privacy policy regarding with *d*, decides whether to approve of the owner’s policy. We use a binary variable *bs*to indicate the opinion of *s*: *bs*= 1 means that *s*approves of the owner’s policy, and *bs*= 0 means disapproval. The opinion *bs*can be seen as the stakeholder’s *vote*. The outcome of the voting, represented as an *aggregated opinion*¯*b*, is computed by

**Dataset and Experiment Setting:**

1. *Data Sharing Simulation*
2. *Trust Evaluation*

***Data Sharing Simulation:***

Given a network, we simulate users’ data sharing behaviors via the following way. Suppose that time evolves in rounds. At each round *t 2 f*1*;*2*; · · · ; Tg*a certain number of users are selected as *owners*, i.e., they want to post data items that involve multiple users. To select the owners, we first pick a number *probi*uniformly at random from [0*;*1] for every user *vi*. If *probi*is smaller than a prespecified threshold *ρ*, then user *vi*will act as an owner. The threshold *ρ*actually denotes the ratio of owners to all users. Given an owner *o*, we determine the corresponding stakeholders via the following two approaches respectively: randomly select several users from the set of *o*’s friends (i.e. users who are directly connected to *o*); randomly select several users from the whole user set (other than the owner himself/herself). In the latter case, we define that the number of stakeholders is at most 20, considering the average node degree of the network. Each stakeholder’s opinion is picked from *f*0*;*1*g*uniformly at random. For every user, either an owner or a stakeholder, we utilize the distance defined in Section III-A to specify the user’s privacy policy. That is, given a pair of users *vi*and *vj*, where *vi*is the owner or the stakeholder of a data item, only if the distance between users *vi*and *vj*is no greater than a threshold *disth*, user *vj*is allowed to access the data item.

***Trust Evaluation:***

Initially, for any two users *vi*and *vj*, we set the trust value as follows. If the two users are directly connected, then we set *tij*= *tji*= 0*:*8. Otherwise, we use (1) to compute the trust value. That is, we first find the shortest path between the two users, and then set *tij*= *tji*= 0*:*8*dij*, where *dij*denotes the length of the shortest path. It should be noted that though we assign the same initial value to *ti*and *tij*, it doesn’t mean that the trust is reciprocal. After the initialization, the trust values are updated in each round, according to the rules defined by (4), (5), (8) and (9). The parameter in (9) is set to 0.5.

**Trust-based Privacy Management:**

Given a network and the setting of parameters *fT; ρ; disth; bthg*, we simulate the trust-based mechanism proposed in Section IV-A via the following way. At each round, we first determine the initial trust values, the owners, the corresponding stakeholders, and their privacy policies by using the method described above. Then for each owner, we generate a random number *γ 2*[0*;*1]. If *γ*is larger than a threshold *γth*, then the owner solicits the stakeholders’ opinions to decide whether to post the data, otherwise the user directly posts the data. The threshold *γth*can be set to different values. Statistically speaking, a high *γth*implies that most of the owners don’t care about the stakeholders’ privacy and post data directly. In an extreme case that *γth*= 1, none of the owners solicits the stakeholders’ opinions before posting data. If the owner decides to post the data, then the stakeholder *s*who has voted against the posting (*bs*= 0) will suffer a privacy loss. And the loss *ls*is set to 1. Every user may suffer some privacy loss at each round. During the simulation, each user’s privacy loss is accumulated. At the end of round *T*, we compute the average accumulated privacy loss of all users.

## SYSTEM REQUIREMENTS:

**HARDWARE REQUIREMENTS:**

* System : Pentium Dual Core.
* Hard Disk : 120 GB.
* Monitor : 15’’ LED
* Input Devices : Keyboard, Mouse
* Ram : 1 GB

**SOFTWARE REQUIREMENTS:**

* Operating system : Windows 7.
* Coding Language : JAVA/J2EE
* Tool : Netbeans 7.2.1
* Database : MYSQL

**INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user

will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**Software Environment**

## Java Technology

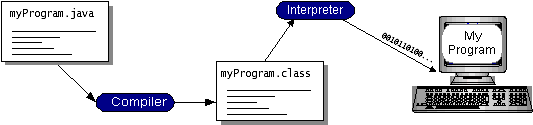
Java technology is both a programming language and a platform.

### The Java Programming Language

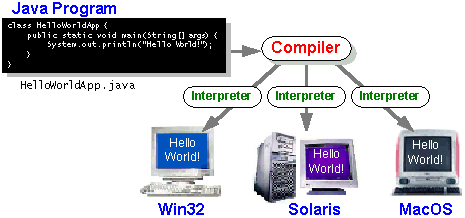
### The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

* + - Simple
    - Architecture neutral
    - Object oriented
    - Portable
    - Distributed
    - High performance
    - Interpreted
    - Multithreaded
    - Robust
    - Dynamic
    - Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make “write once, run anywhere” possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



### The Java Platform

A platform is the hardware or software environment in which a program runs. We’ve already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it’s a software-only platform that runs on top of other hardware-based platforms.

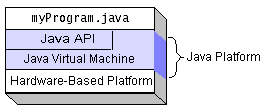
The Java platform has two components:

* The Java Virtual Machine (Java VM)
* The Java Application Programming Interface (Java API)

You’ve already been introduced to the Java VM. It’s the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as packages. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

## What Can Java Technology Do?

The most common types of programs written in the Java programming language are applets and applications. If you’ve surfed the Web, you’re probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

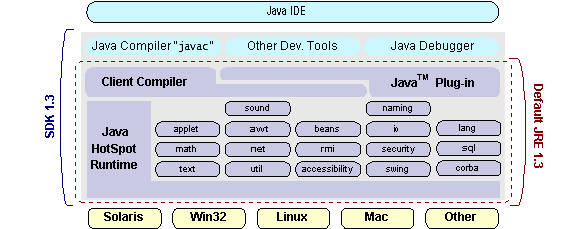
However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a server serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a servlet. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

* **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
* **Applets**: The set of conventions used by applets.
* **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
* **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
* **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
* **Software components**: Known as JavaBeansTM, can plug into existing component architectures.
* **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
* **Java Database Connectivity (JDBCTM)**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



## How Will Java Technology Change My Life?

We can’t promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

* **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it’s easy to learn, especially for programmers already familiar with C or C++.
* **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
* **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people’s tested code and introduce fewer bugs.
* **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
* **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure JavaTM Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
* **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
* **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded “on the fly,” without recompiling the entire program.

### ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn’t change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn’t as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

**JDBC**

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of “plug-in” database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC’s framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

### JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. ***SQL Level API***

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

1. ***SQL Conformance***

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

1. ***JDBC must be implemental on top of common database interfaces***The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.
2. ***Provide a Java interface that is consistent with the rest of the Java system***

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

1. ***Keep it simple***

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

1. ***Use strong, static typing wherever possible***

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

1. ***Keep the common cases simple***

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

Finally we decided to proceed the implementation using Java Networking.

And for dynamically updating the cache table we go for MS Access database.

Java ha two things: a programming language and a platform.

Java is a high-level programming language that is all of the following

Simple Architecture-neutral

Object-oriented Portable

Distributed High-performance

Interpreted multithreaded

Robust Dynamic

Secure

Java is also unusual in that each Java program is both compiled and interpreted. With a compile you translate a Java program into an intermediate language called Java byte codes the platform-independent code instruction is passed and run on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The figure illustrates how this works.

**Java Program**

**Compilers**

**Interpreter**

**My Program**

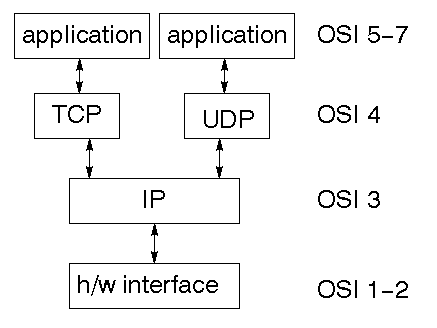
You can think of Java byte codes as the machine code instructions for the Java Virtual Machine (Java VM). Every Java interpreter, whether it’s a Java development tool or a Web browser that can run Java applets, is an implementation of the Java VM. The Java VM can also be implemented in hardware.

Java byte codes help make “write once, run anywhere” possible. You can compile your Java program into byte codes on my platform that has a Java compiler. The byte codes can then be run any implementation of the Java VM. For example, the same Java program can run Windows NT, Solaris, and Macintosh.

## Networking

### TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

### IP datagram’s

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

### UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

### TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

### Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

### Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

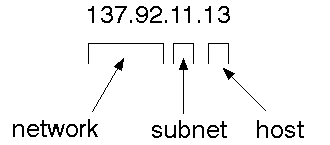
### Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

### Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

### Total address



The 32 bit address is usually written as 4 integers separated by dots.

### Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

### Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

#include <sys/types.h>

#include <sys/socket.h>

int socket(int family, int type, int protocol);

Here "family" will be AF\_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

**JFree Chart**

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

JFreeChart is "open source" or, more specifically, [free software](http://www.gnu.org/philosophy/free-sw.html). It is distributed under the terms of the [GNU Lesser General Public Licence](http://www.gnu.org/licenses/lgpl.html) (LGPL), which permits use in proprietary applications.

## 1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

## 2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

## 3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

## 4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

**J2ME (Java 2 Micro edition):-**

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices.

**1. General J2ME architecture**



J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

**2.Developing J2ME applications**

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role preverification plays in this process.

**3.Design considerations for small devices**

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

\* Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.

\* Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.

\* Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

**4.Configurations overview**

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

\* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

\* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

**5.J2ME profiles**

**What is a J2ME profile?**

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations.

A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

**Profile 1: KJava**

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is com.sun.kjava. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

**Profile 2: MIDP**

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application

development. 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

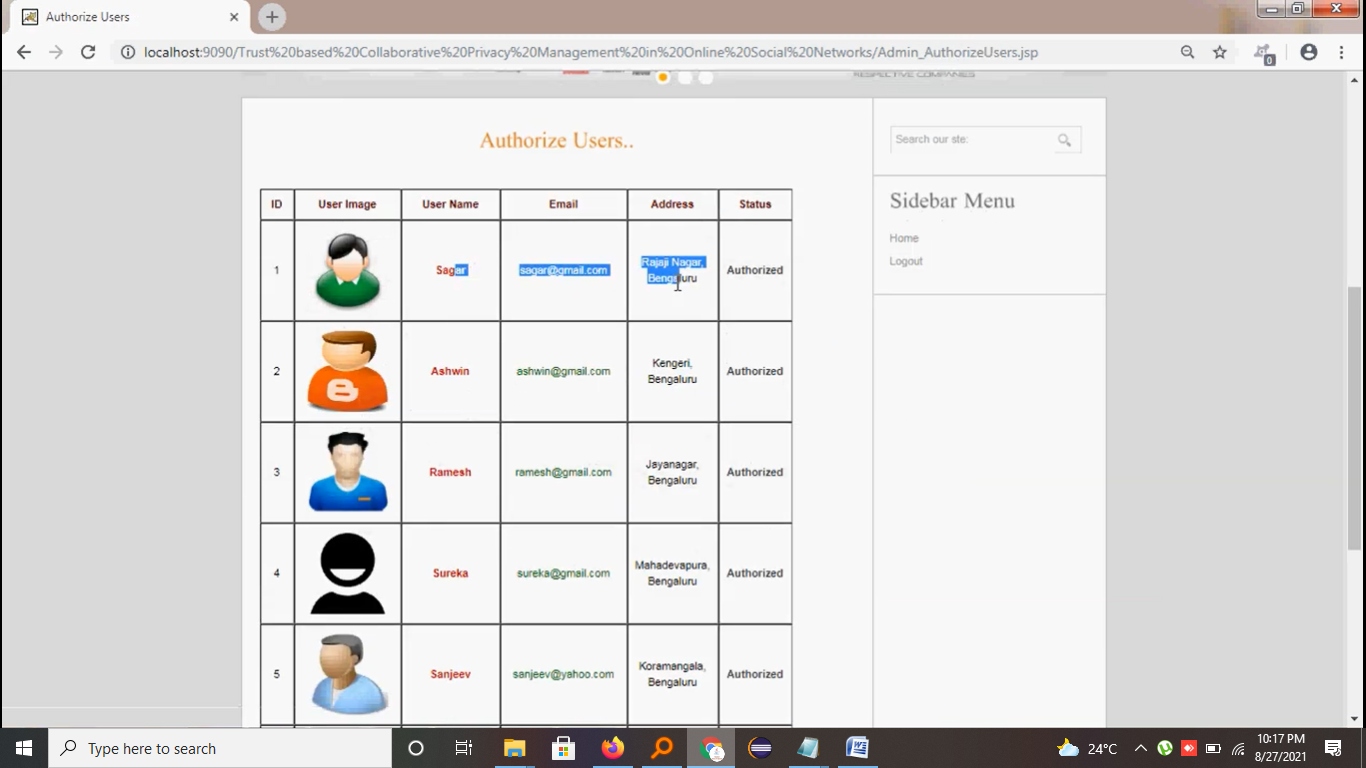
\* javax.microedition.midlet

\* javax.microedition.rms

**SCREENSHOTS:**

****

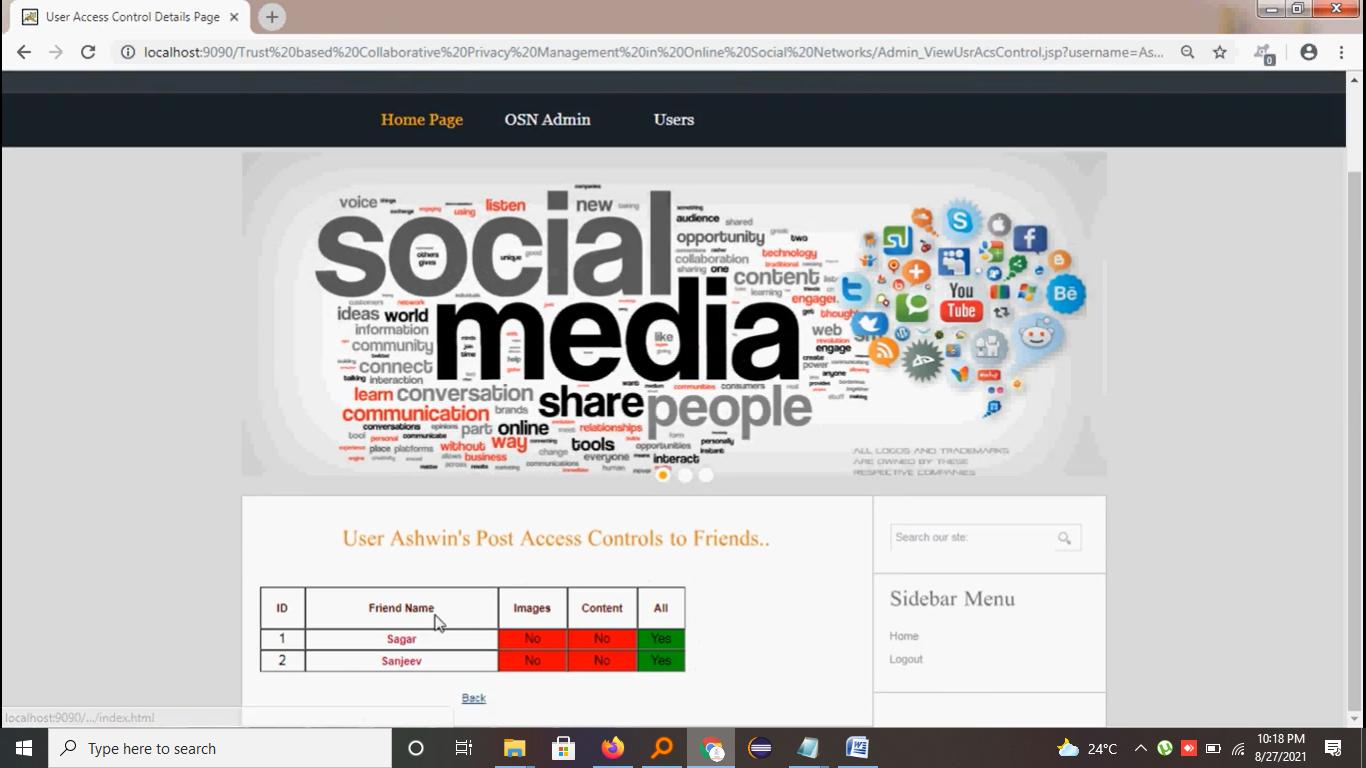
****

****

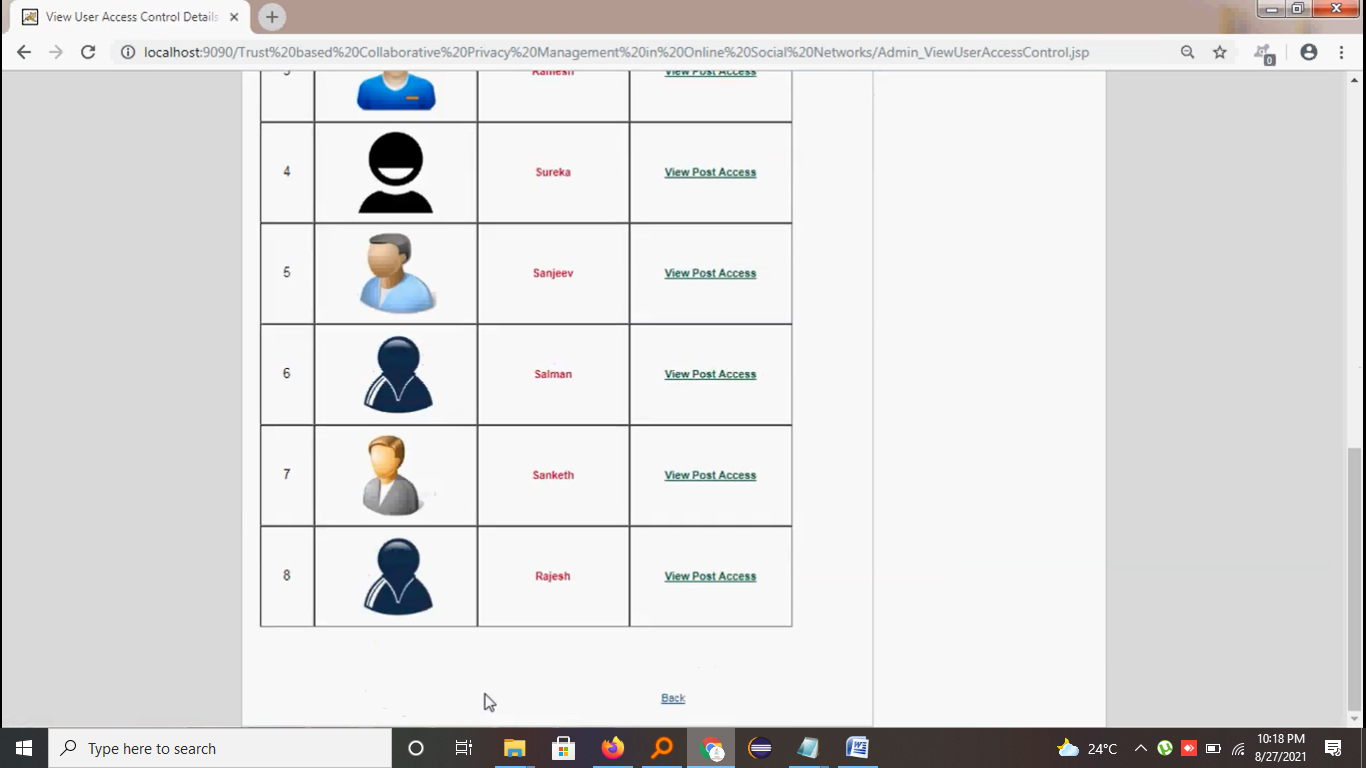
****

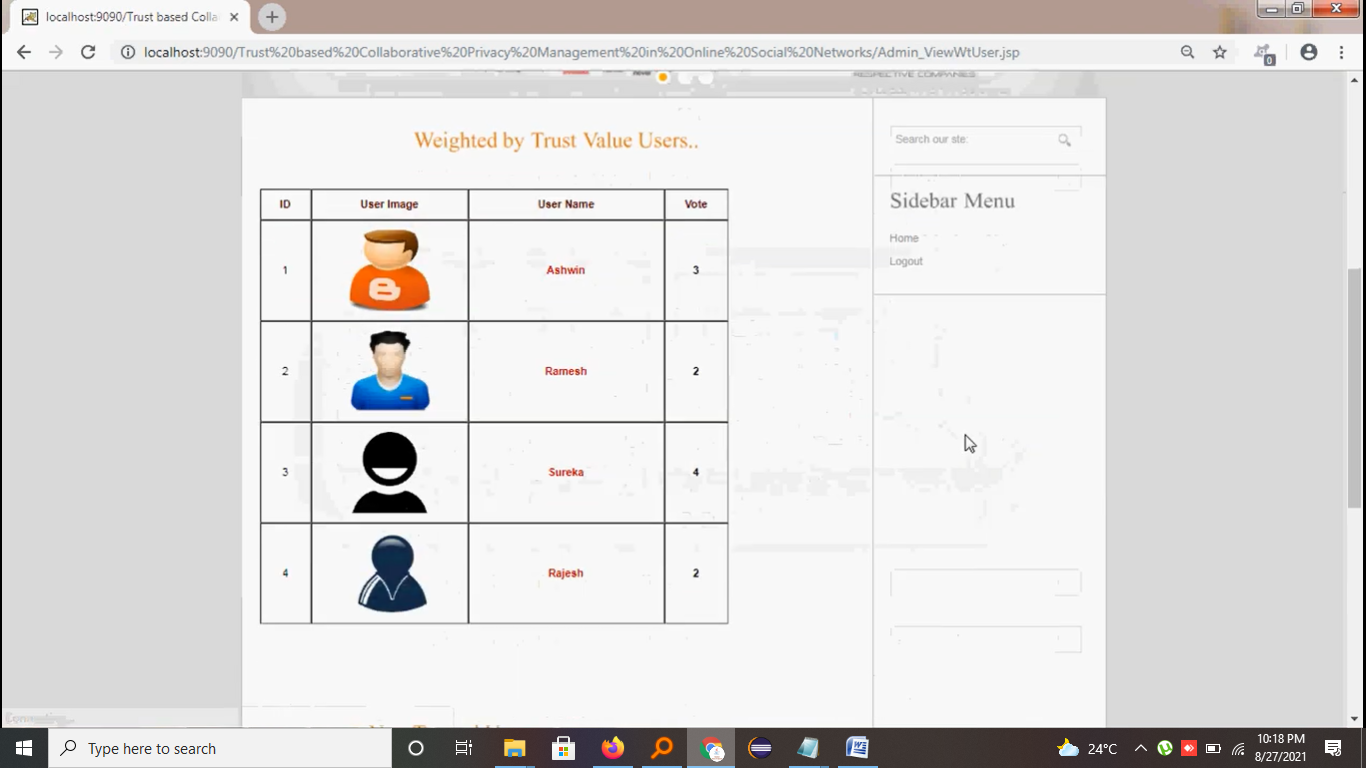
****

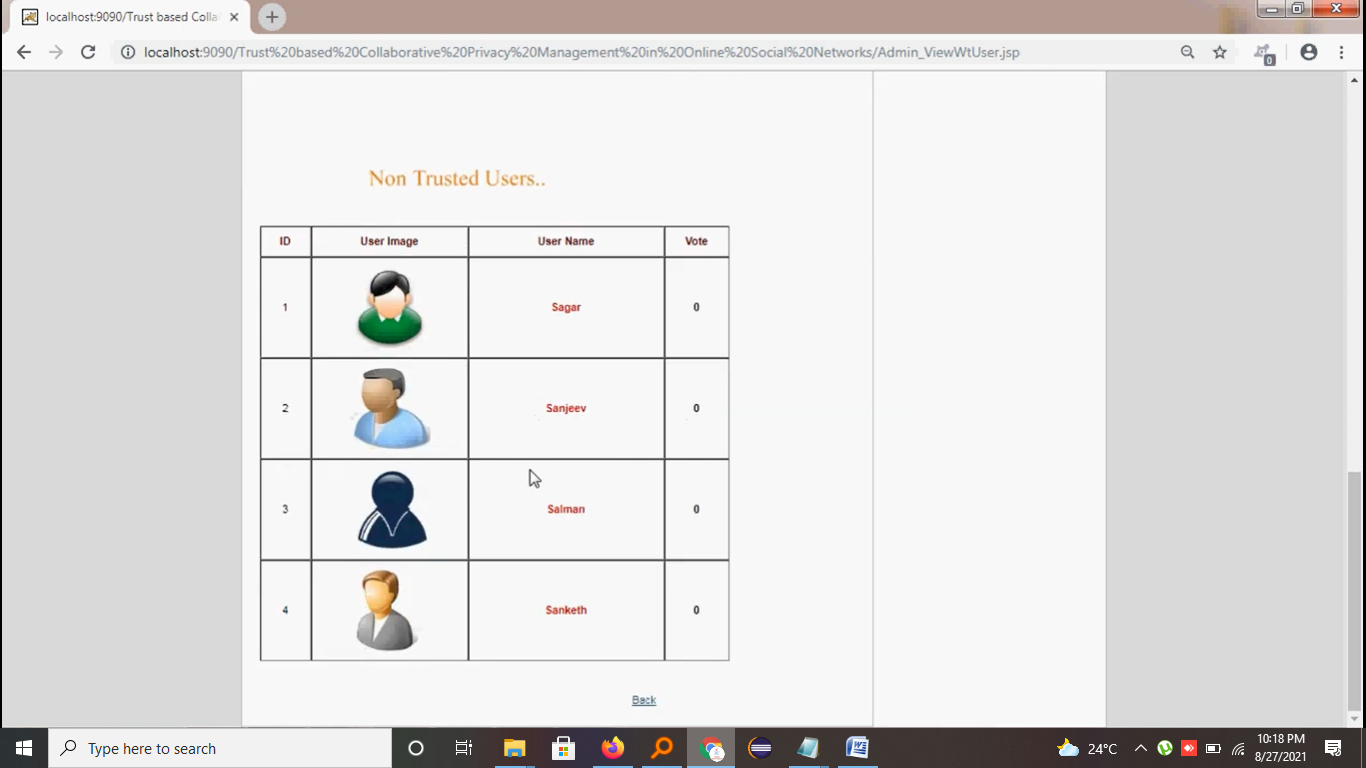
****

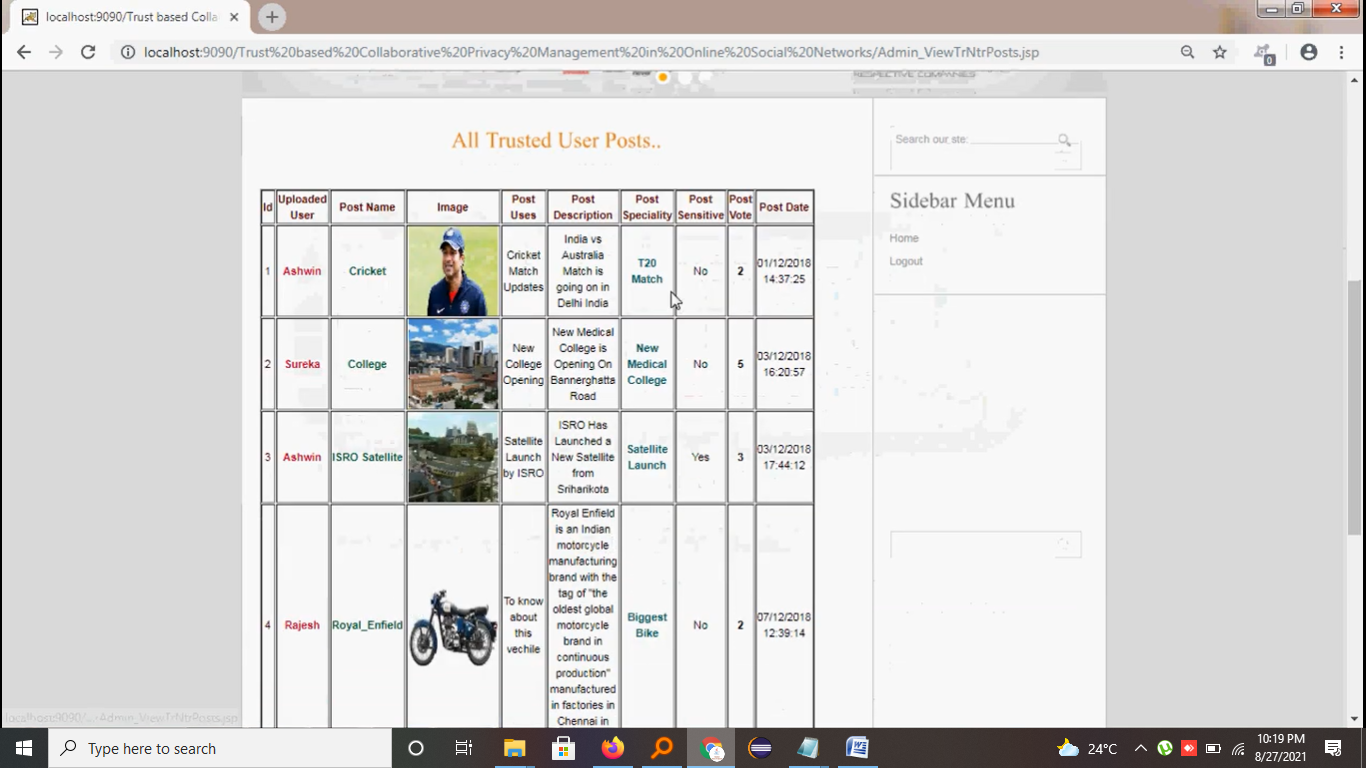
****

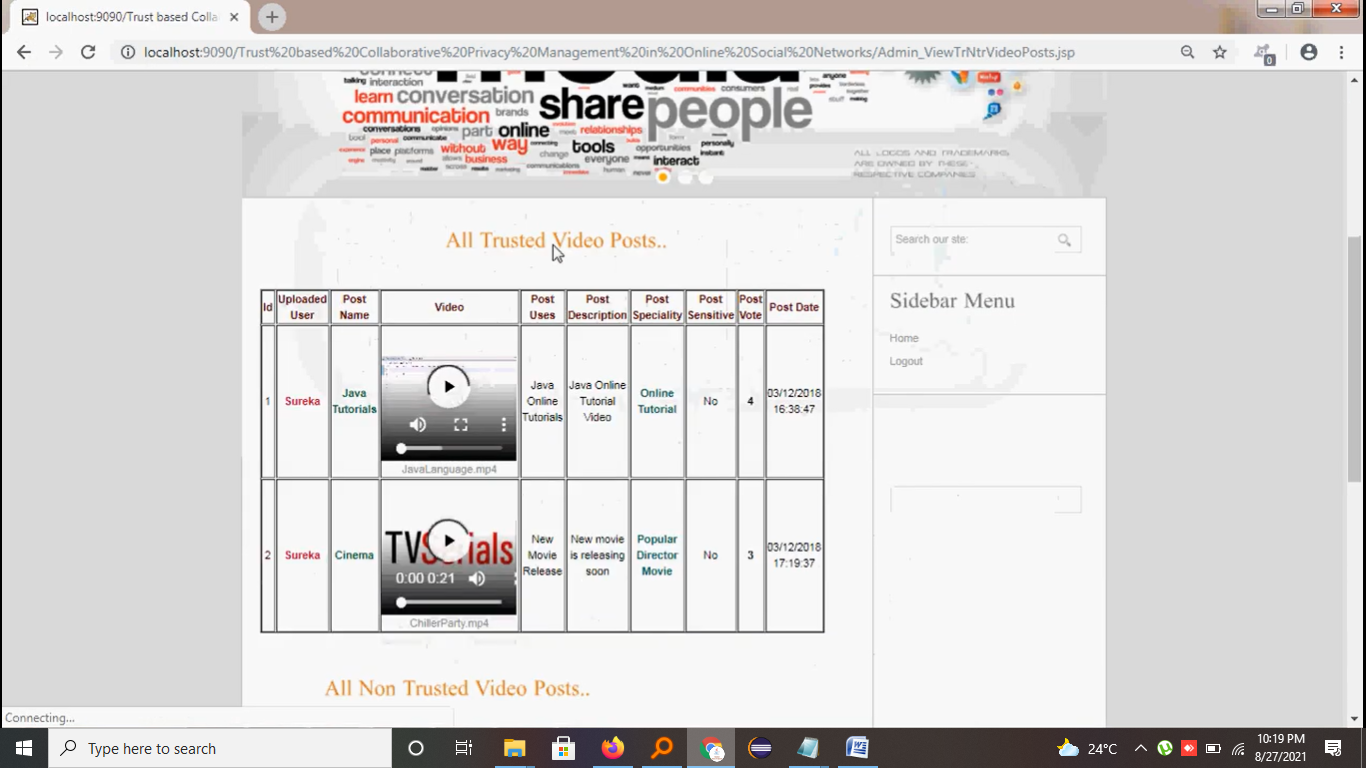
****

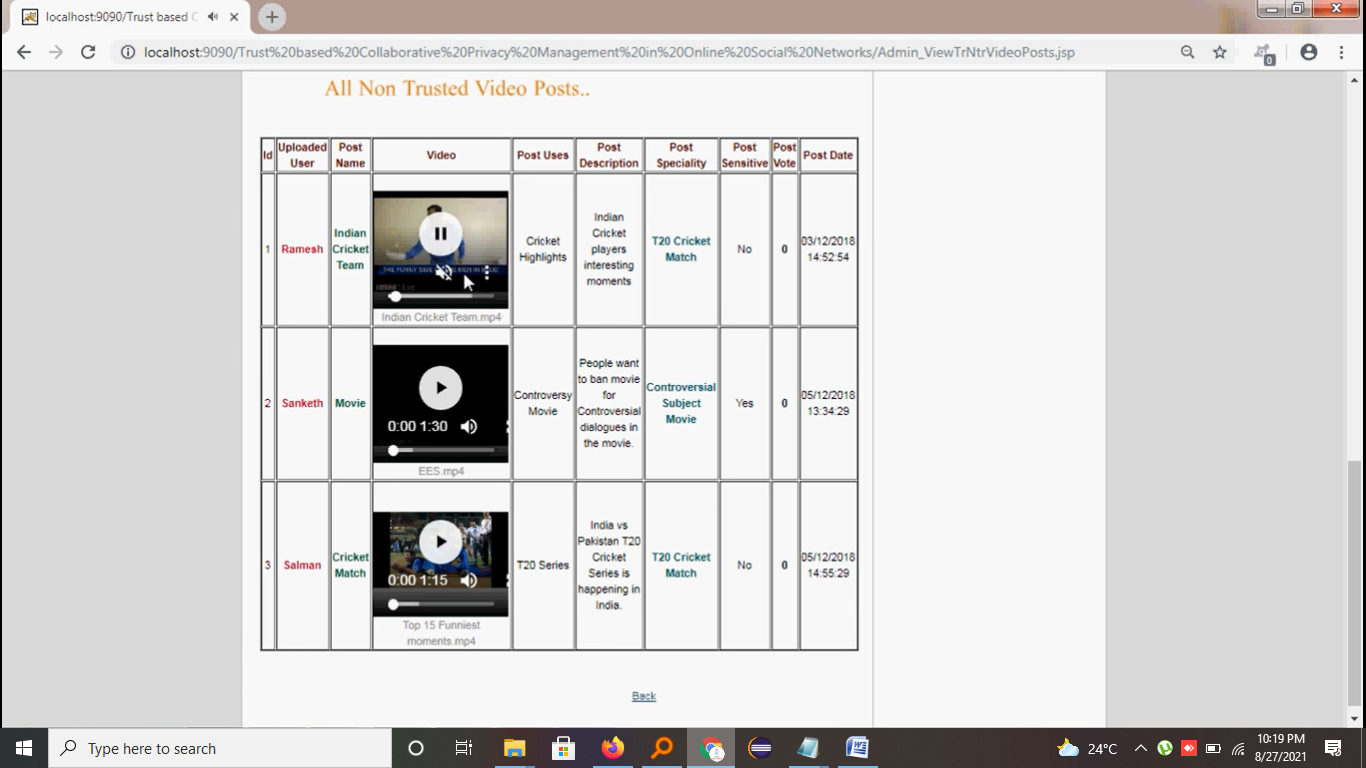
****

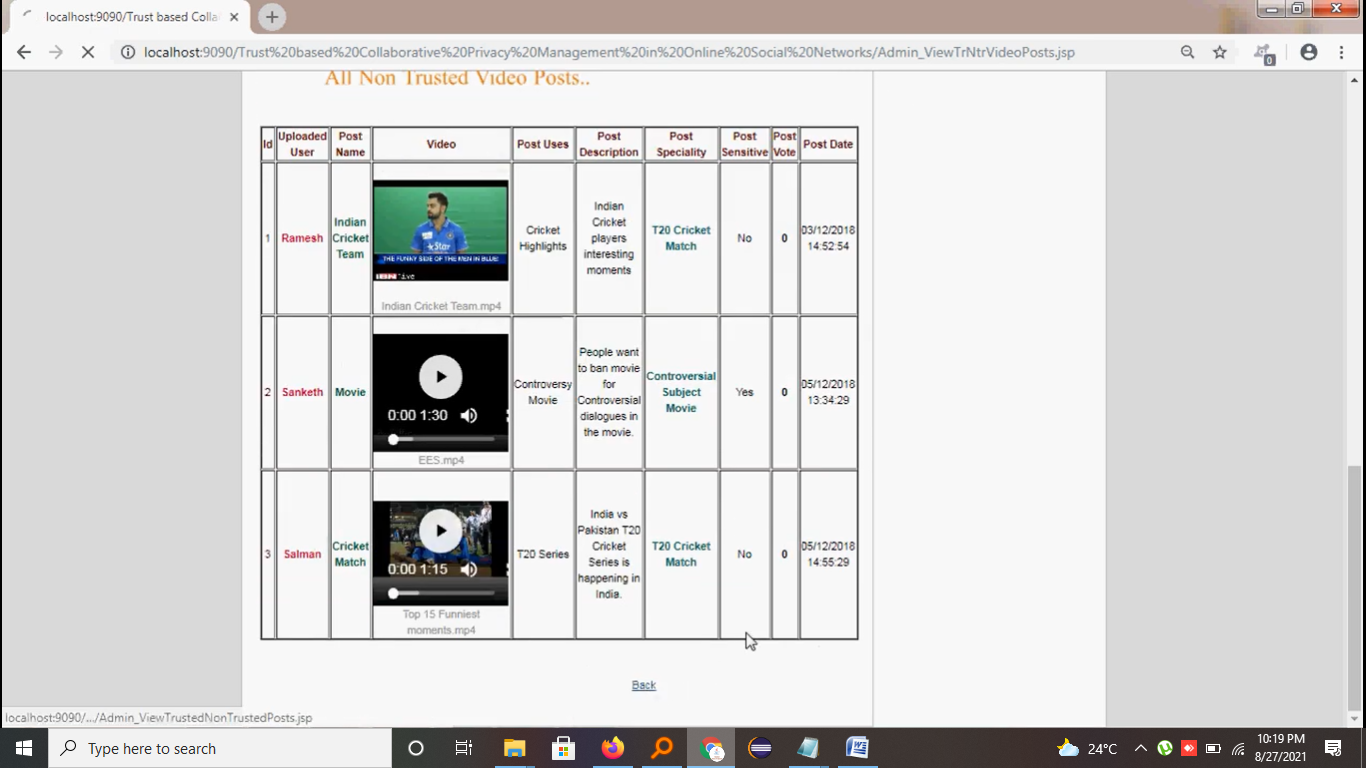
****

****

****

****

****

****

CONCLUSION

In this project, the privacy issue caused by the sharing of co-owned data in OSNs is studied. To help the owner of data collaborate with the stakeholders on the control of data sharing, I propose a trust-based mechanism. When a user is about to post a data item, the user first solicits the stakeholders’ opinions on data sharing, and then makes the final decision by comparing the aggregated opinion with a pre-specified threshold. The more the user trusts a stakeholder, the more the user values the stake holder’s opinion. If a user suffers a privacy loss because of the data sharing behavior of another user, then the user’s trust in another user decreases. On the other hand, considering that the user needs to balance between data sharing and privacy preserving, I apply a bandit approach (UCB Policy) to tune the threshold in the proposed trust-based mechanism, so that the user can get a benefit from posting data and privacy loss caused by the users.

REFERENCES

[1] C. Zhang, J. Sun, X. Zhu, and Y. Fang, “Privacy and security for online social networks: challenges and opportunities,” IEEE Network, vol. 24, no. 4, pp. 13–18, July 2010.

[2] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, “Information security in big data: Privacy and data mining,” IEEE Access, vol. 2, pp. 1149–1176, 2014.

[3] L. Xu, C. Jiang, Y. Chen, J. Wang, and Y. Ren, “A framework for categorizing and applying privacypreservation techniques in big data mining,” Computer, vol. 49, no. 2, pp. 54–62, Feb 2016.

[4] M. Qiu, K. Gai, and Z. Xiong, “Privacypreserving wireless communications using bipartite matching in social big data,” Future Generation Computer Systems, 2017. [Online]. Available: [http://www.sciencedirect.com/science/article/pii/S01 67739X17301449](http://www.sciencedirect.com/science/article/pii/S01%2067739X17301449)

[5] C. Fiesler, M. Dye, J. L. Feuston, C. Hiruncharoenvate, C. Hutto, S. Morrison, P. Khanipour Roshan, U. Pavalanathan, A. S. Bruckman, M. De Choudhury, and E. Gilbert, “What (or who) is public?: Privacy settings and social media content sharing,” in Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, March 2017, pp. 567– 580.

[6] A. C. Squicciarini, M. Shehab, and F. Paci, “Collective privacy management in social networks,” in Proceedings of the 18th ACM International Conference on World Wide Web, April 2009, pp. 521–530.

[7] H. Hu, G.-J. Ahn, and J. Jorgensen, “Detecting and resolving privacy conflicts for collaborative data sharing in online social networks,” in Proceedings of the 27th ACM Annual Computer Security Applications Conference, December 2011, pp. 103– 112.

[8] J. M. Such and N. Criado, “Resolving multi-party privacy conflicts in social media,” IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 7, pp. 1851–1863, July 2016.

[9] P. Auer, N. Cesa-Bianchi, and P. Fischer, “Finite-time analysis of the multiarmed bandit problem,” Machine learning, vol. 47, no. 2-3, pp. 235–256, 2002.

[10] H. Hu, G.-J. Ahn, Z. Zhao, and D. Yang, “Game theoretic analysis of multiparty access control in online social networks,” in Proceedings of the 19th ACM Symposium on Access Control Models and Technologies, New York, NY, June 2014, pp. 93– 102.