INTRODUCTION

CHAPTER 1

1. INTRODUCTION

1.1 Introduction to Data Leakage Detection

In the course of doing business, sometimes sensitive data must be handed over to supposedly trusted third parties. For example, a hospital may give patient records to researchers who will devise new treatments. Similarly, a company may have partnerships with other companies that Require sharing customer data. Another enterprise may outsource its data processing, so data must be given to various other companies. We call the owner of the data the distributor and the supposedly trusted third parties the agents. Our goal is to detect when the distributor's sensitive Data have been leaked by agents, and if possible to identify the agent that leaked the data.

1.2 WHY DO WE USE DATA MINING

Data mining is the process of extracting patterns from data. Data mining is becoming an increasingly important tool to transform the data into information. It is commonly used in a wide range of profiling practices, such as marketing, surveillance, fraud detection and scientific discovery. Data mining can be used to uncover patterns in data but is often carried out only on samples of data. The mining process will be ineffective if the samples are not a good representation of the larger body of data. Data mining cannot discover patterns that may be present in the larger body of data if those patterns are not present in the sample being "mined". Inability to find patterns may become a cause for some disputes between customers and service providers. Therefore data mining is not foolproof but may be useful if sufficiently representative data samples are collected. The discovery of a particular pattern in a particular set of data does not necessarily mean that a pattern is found elsewhere in the larger data from which that sample was drawn. An important part of the process is the verification and validation of patterns on other samples of data

1.3 KEY TERMS

Data Leakage - A data breach is the unintentional release of secure information to an untrusted environment.

Data Privacy - Information privacy, or data privacy is the relationship between collection and dissemination of data, technology, the public expectation of privacy, and the legal and political issues surrounding them. Privacy concerns exist wherever personally identifiable information is collected and stored - in digital form or otherwise. Improper or non-existent disclosure control can be the root cause for privacy issues.

1.4 Existing System

Traditionally, leakage detection is handled by watermarking, e.g., a unique code is embedded in each distributed copy. If that copy is later discovered in the hands of an unauthorized party, the leaker can be identified. Watermarks can be very useful in some cases, but again, involve some modification of the original data. Furthermore, watermarks can sometimes be destroyed if the data recipient is malicious. The Existing System can detect the hackers but the total no of cookies (evidence) will be less and the organization may not be able to proceed legally for further proceedings due to lack of good amount of cookies and the chances to escape of hackers are high.

1.5 Proposed System

In the proposed system we study unobtrusive techniques for detecting leakage of a set of objects or records. Specifically, we study the following scenario: After giving a set of objects to agents, the distributor discovers some of those same objects in an unauthorized place. (For example, the data may be found on a website, or may be obtained through a legal discovery process.) At this point, the distributor can assess the likelihood that the leaked data came from one or more agents, as opposed to having been independently gathered by other means. In the proposed approach, we develop a model for assessing the "guilt" of agents. We also present algorithms for distributing

objects to agents, in a way that improves our chances of identifying a leaker. Finally, we also consider the option of adding "fake" objects to the distributed set. Such objects do not correspond to real entities but appear realistic to the agents. In a sense, the fake objects act as a type of watermark for the entire set, without modifying any individual members. If it turns out that an agent was given one or more fake objects that were leaked, then the distributor can be more confident that agent was guilty. In the Proposed System the hackers can be traced with good amount of evidence.

1.6 The type of employees that may leak data

- The security illiterate
  - Majority of employees with little or no knowledge of security
  - Corporate risk because of accidental breaches

- The gadget nerds
  - Introduce a variety of devices to their work PCs
  - Download software

- The unlawful residents
  - Use the company IT resources in ways they shouldn't
  - i.e., by storing music, movies, or playing games

- The malicious/disgruntled employees
  - Typically minority of employees
  - Gain access to areas of the IT system to which they shouldn't
  - Send corporate data (e.g., customer lists, R&D, etc.) to third parties

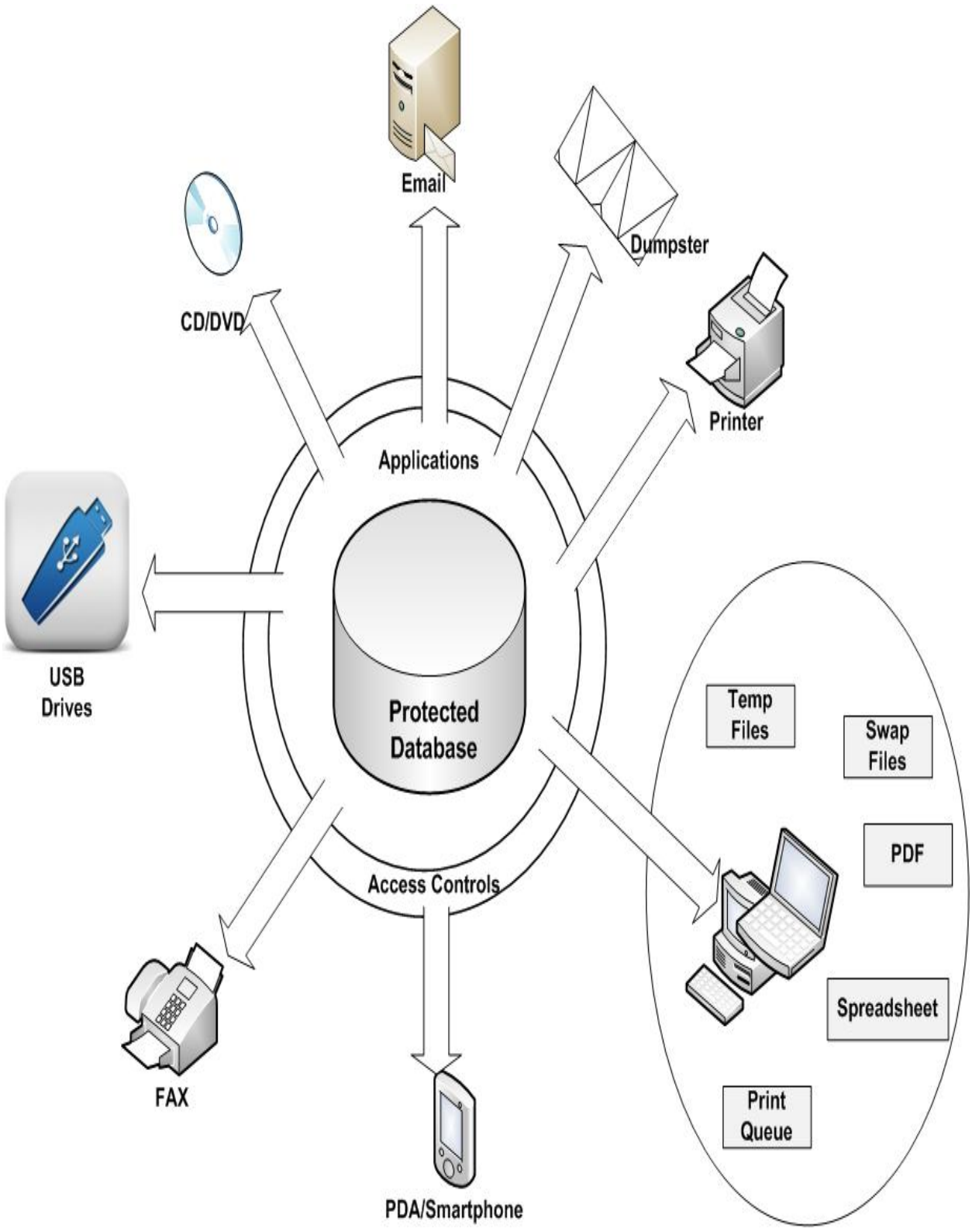

Fig 1 An Example of Data Leakage

LITERATURE SURVEY

CHAPTER 2

LITRATURE SURVEY

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

The guilt detection approach we present is related to the data provenance problem tracing the lineage of S objects implies essentially the detection of the guilty agents. And assume some prior knowledge on the way a data view is created out of data sources. Our problem formulation with objects and sets is more general As far as the data allocation strategies are concerned; our work is mostly relevant to watermarking that is used as a means of establishing original ownership of distributed objects. Finally, there are also lots of other works on mechanisms that allow only authorized users to access sensitive data through access control policies. Such approaches prevent in some sense data leakage by sharing information only with trusted parties. However, these policies are restrictive and may make it impossible to satisfy agent's requests.

2.1 Inference

In a perfect world there would be no need to hand over sensitive data to agents that may unknowingly or maliciously leak it. And even if we had to hand over sensitive data, in a perfect world we could watermark each object so that we could trace its origins with absolute certainty. However, in many cases we must indeed work with agents that may not be 100% trusted, and we may not be certain if a leaked object came from an agent or from some other source, since certain data cannot admit watermarks. In spite of these difficulties, we have shown it is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the probability that objects can be ―guessed by other means. Our model is relatively simple, but we believe it captures the essential trade-offs. The algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. We have shown that distributing objects judiciously can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive.

2.2 Aim of the project

Our goal is to detect when the distributor's sensitive data have been leaked by agents, and if possible to identify the agent that leaked the data.

2.3 Description of the project in short

In this project we are finding out the data is leaked or not. The agent will give the information to broadcast the data via a server to other agents. We will check whether the authorized user leaked the data to another agent.

2.4Process Summary

In this project authorized agent give the request for data request may be explicit or sample, according to request agent get the data. If agent leaked the data to another agent, then in our side we are checking whether the data matches our data. After that we will find out the agent who leaked the data.

2.5 Algorithm

Allocation for Explicit Data Requests In this request the agent will send the request with appropriate condition. Allocation for Sample Data Requests In this request agent request does not have condition. The agent sends the request without condition as per his query he will get the data.

2.6 Deliverables

1. Client & Distributor Software (Data Detection System) installed at the server along with its database.

2. Database backup

3. Agent machine installed with the software and connected to the database at the server.

2.7 Assumptions and Dependencies

Each and every agent has its unique data. Another agent will receive the data only from agent.

2.8 Project Plan

Plan of Execution

- Identification – Searching for different project ideas. Identifying and finalizing one of them for further implementation.
- Conceptualization and design – The concepts required for building the project are studied in detail. Also the high level designing is done at this stage. Preliminary presentation given for more clarification of project.
- Detailed design – At this stage, low level designing is done. User Interface is designed to give better visualization of the project idea.
- Coding – Actual implementation of the project starts at this stage. Coding for each module of the four modules will be done. Coding and testing will take approximately 8 to 10 weeks.
- Unit Testing –Initially the backend database will be tested over no of transactions. Then GUI for agent user as well as for server level (or Distributor) user is tested separately.
- Integration Testing –All modules will be integrated and then testing of whole integrated testing will be performed. It also includes evaluation of project.
- System Testing – The product was tested in the context of the entire system. Different Linux systems will be used for system testing and the performance will be monitored.
- Documentation – A detailed document about the project shall be prepared at this stage.

Sl. No	Module For	S/W Required for coding	Period for coding (Approx. weeks)	S/W Required for Testing	Period for Testing (Approx. weeks)
1.	Requirement gathering	N/A	1	N/A	N/A
2.	Analysis	N/A	1	N/A	N/A
3.	Creating Application	Visual Studio 2008	2	Visual Studio2008	1
4.	Integrating all the modules	Visual Studio 2008	1	Visual Studio 2008	N/A
5.	Final Testing	---------	1	N/A	N/A

Test plan of the project

REQUIREMENT ANALYSIS

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 SOFTWARE REQUIREMENTS

- The module is written in ASP .net and C# .net.
- It is developed in Visual Basics Platform.
- Windows is the operating system chosen for the module.
- The database used in the project is MS-SQL server 2005 or higher.

3.2 HARDWARE REQUIREMENTS

- **PROCESSOR:** Pentium 4 or above.
- **RAM:** 256 MB or more.
- **Hard disc Space:** 500 MB to 1GB.

SOFTWARE ENVIRONMENT

CHAPTER 4

SOFTWARE ENVIRONMENT

4.1 INTRODUCTION TO VISUAL STUDIO 2008

To get the most out of Visual Studio .NET, you will most likely wish to tailor it to suit your style of working. With the wide variety of configuration options, both familiar and new, you'll want to take the time examine some of the various options you can set. In this document, you will be introduced to many of the different configurations and learn about the various settings in Visual Studio .NET. You will also learn about the different types of windows, including Tool windows, which can be docked to the environment or free floating, and you'll learn about Document windows.

4.2 CONFIGURATION

The first time you use Visual Studio .NET, you will be prompted for some configuration information about how you will use the environment most often. Figure 1 shows an example of the My Profile screen.

The My Profile Screen allows you to set some overall environment defaults.

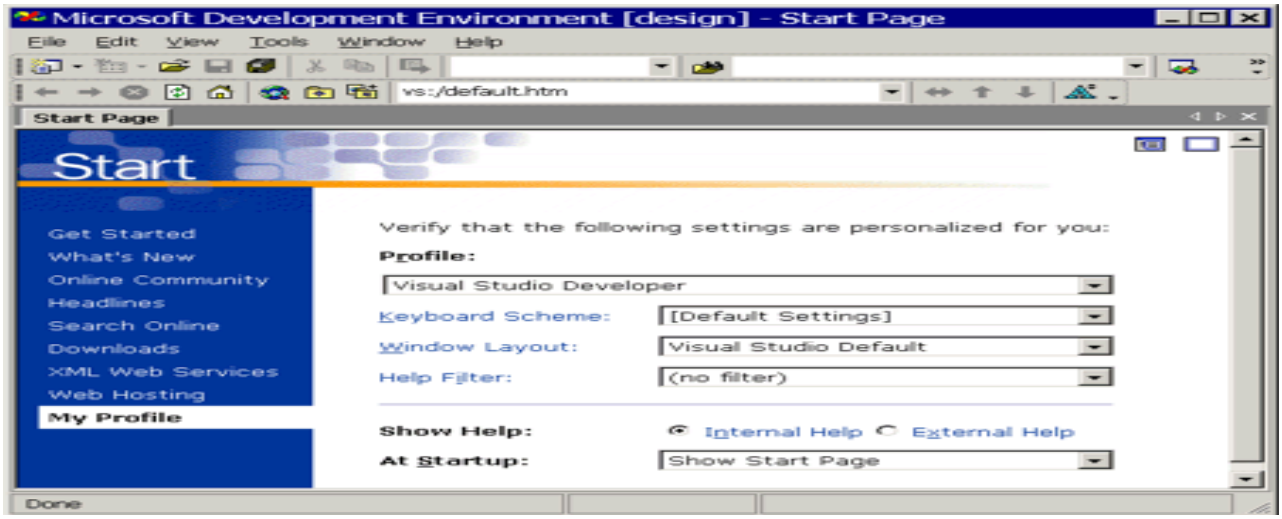

Figure 1. Set the configuration on the My Profile Screen

4.3VISUAL STUDIO START PAGE

If you choose to have the start-up page as the Visual Studio Start Page, you will see a screen that looks similar to Figure 2.

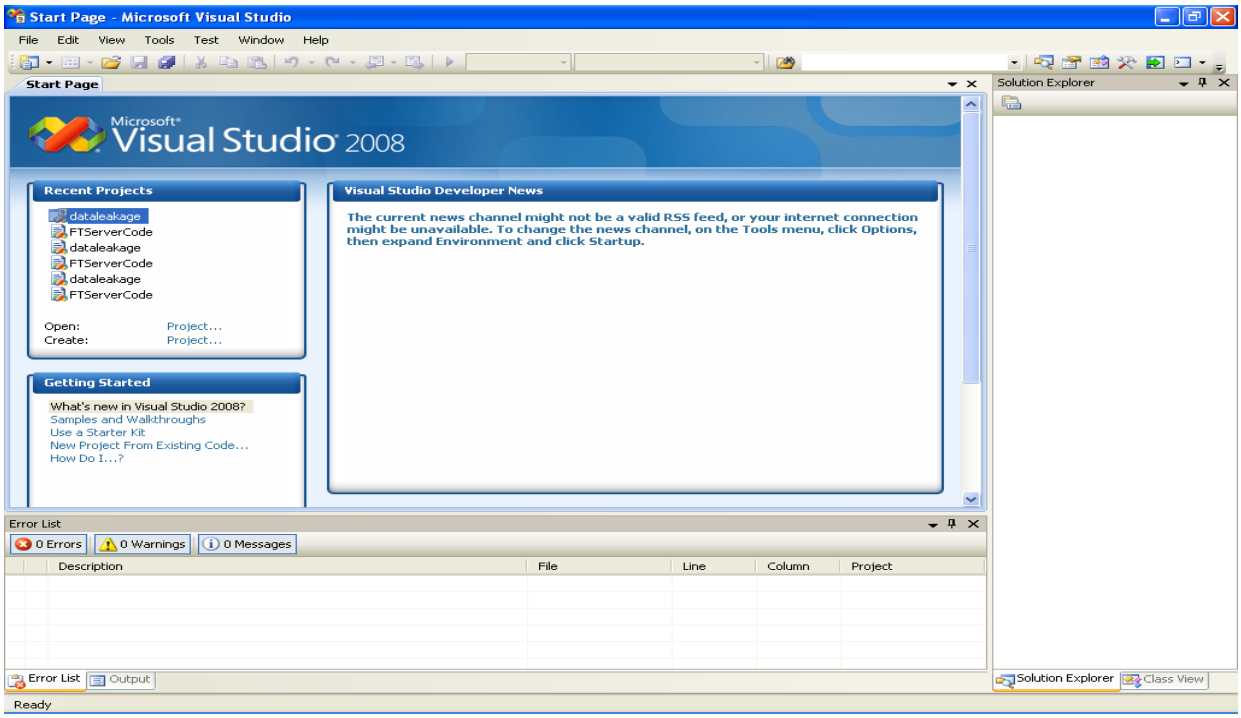

Figure 2. Visual Studio start page

Figure 2 The Visual Studio Start page allows you to start a recent project, open an existing project, or create a new project. On this screen, there is a menu on the left side that will let you link to what's New Help menu. You can see a list of online community links, where you can get assistance with Visual Studio .NET and many other Microsoft products. You can get the headlines for MSDN news, and you have the ability to Search the MSDN site for information related to Visual Studio. You can also reset your profile. On the Get Started page you can select a recent project, create a new project, or open an existing project.

4.4 CREATING A NEW PROJECT

In Visual Studio .NET, if from the File menu you click New and then Project, you will see a dialog that looks like Figure 3. When putting together an application in Visual Studio .NET, you may have multiple projects. The set of projects together make up what is called a Solution.

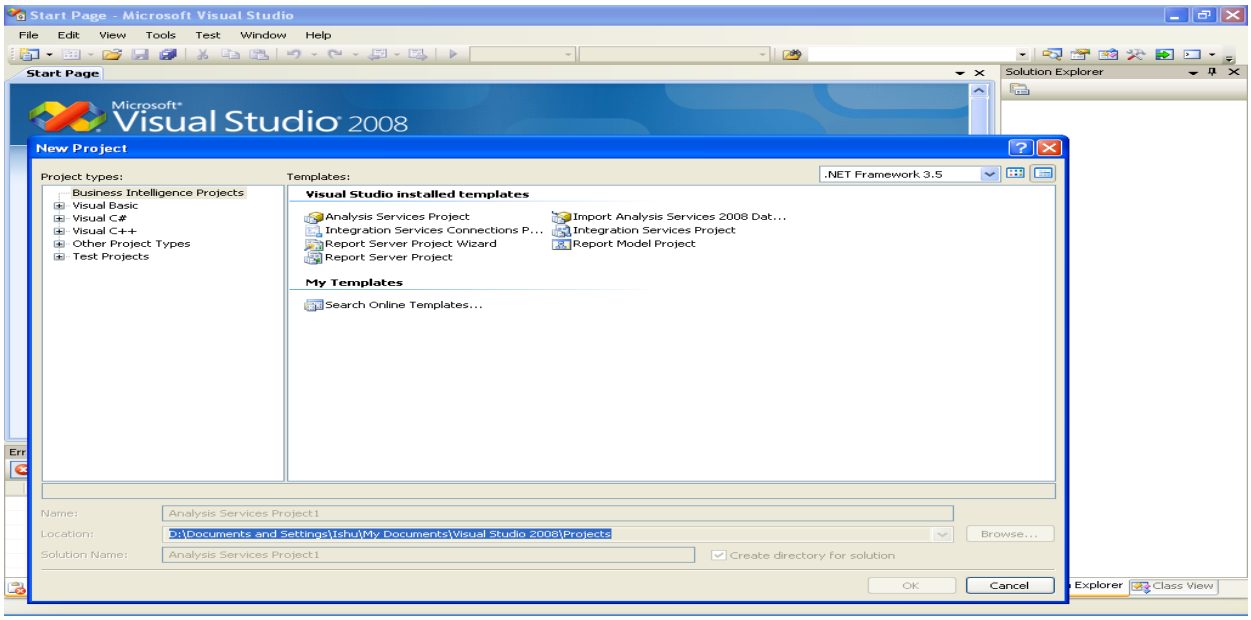

Figure 3. The New Project dialog box

Figure 3. The New Project dialog box allows you to create a new Solution of a particular project type

On the left side of this screen, you can choose what type of project you will be creating. Depending on the options you selected when you installed your Visual Studio environment, you can choose from a Visual Basic .NET, C#, C++, and possibly other programming languages. Not all of these languages for Visual Studio come from Microsoft; there are other companies developing applications that will use the .NET Framework.

On the right side of this screen, you can choose a default template for the type of project you will be creating. There are many different templates to choose from. Table 2 provides a list of some of these project template types.

Prior to adding a new project to this solution, you need to set the Name and Path where this project will reside on your hard drive. Fill in the path for where you want this project to reside in the Location text box. Visual Studio .NET creates the necessary path, and will create a folder name with the same name as the project. For example, if you fill in the Name **Login Test**, and set path to D:\MySamples, this solution will be created in D:\MySamples\LoginTest\LoginTest.sln.

4.5 SOLUTION EXPLORER WINDOW

A set of projects that are part of the same application in Visual Studio .NET is called a *Solution*. The Solution Explorer window shows you a tree view list of each project, each project's references, and each project's components. If this window is closed, you can open it from the **View** menu by clicking **Solution Explorer**. Components may be made up of forms, classes, modules, and any other file types it takes to create your application. Double-click on an item in order to edit that item within the IDE.

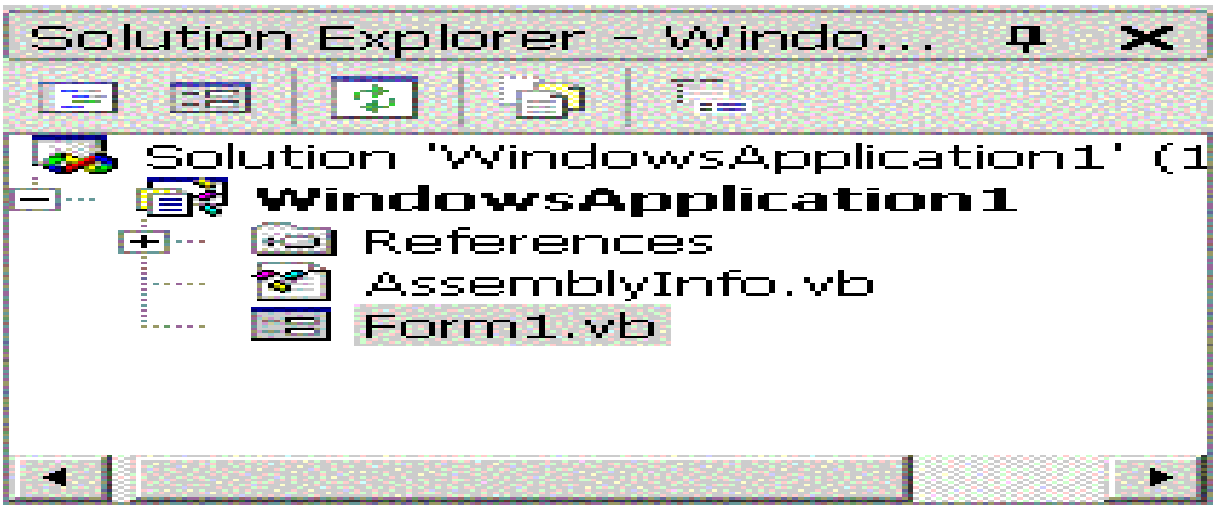

Figure 4. Solution Explorer

Figure 4. The Solution Explorer gives you a graphical representation of all of the files that make up your project(s)

5.6 PROPERTY WINDOW

When working with classes such as text boxes and forms, you will most likely need to change certain attributes about those classes. To bring up the Properties window, on the **View** menu click **Properties Window** (F4). (See Figure 5.) Once this window is up, you can either view the list alphabetically or categorized by attribute. Properties within this window can be selected either from a list or by clicking a button to bring up a dialog box. There may be others you type some text into, like the Text property that is used to change the title of a form.

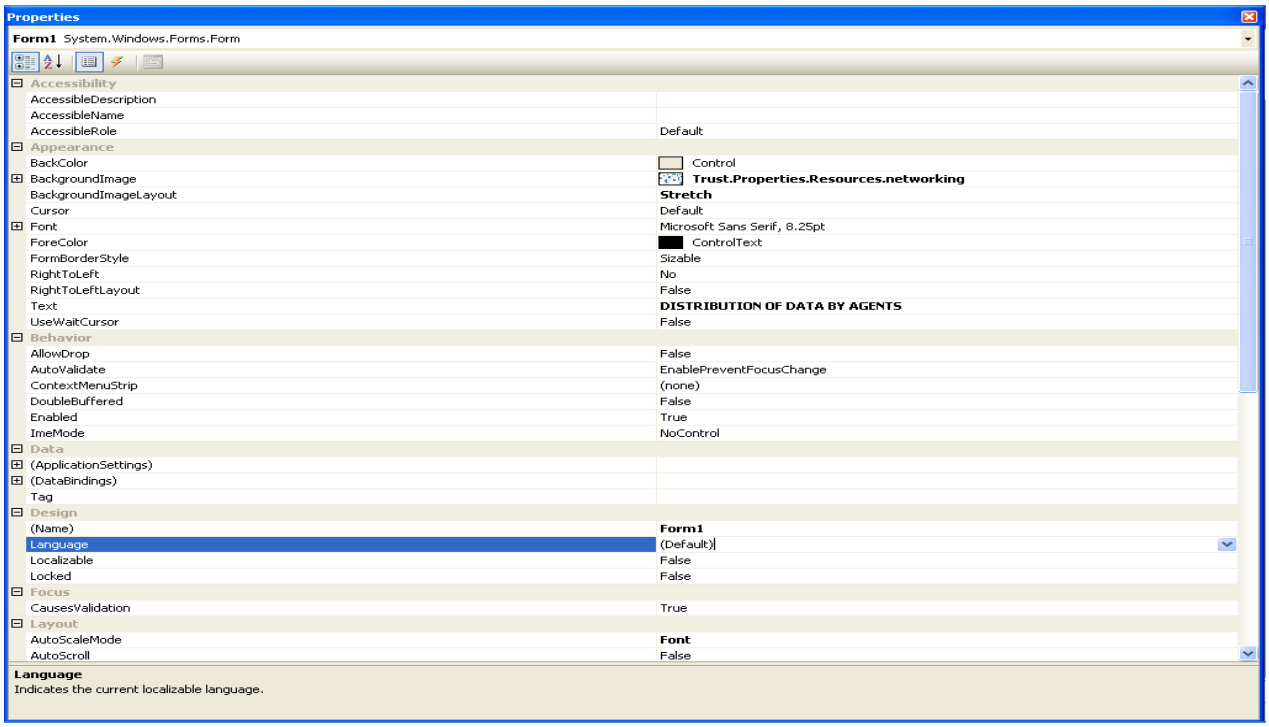

Figure 5. This shows us Property Window, used to set the parameters of our screen dialog box

MODULE ANALYSIS

CHAPTER 5

MODULE ANALYSIS

5.1 ARCHITECTURE DIAGRAM

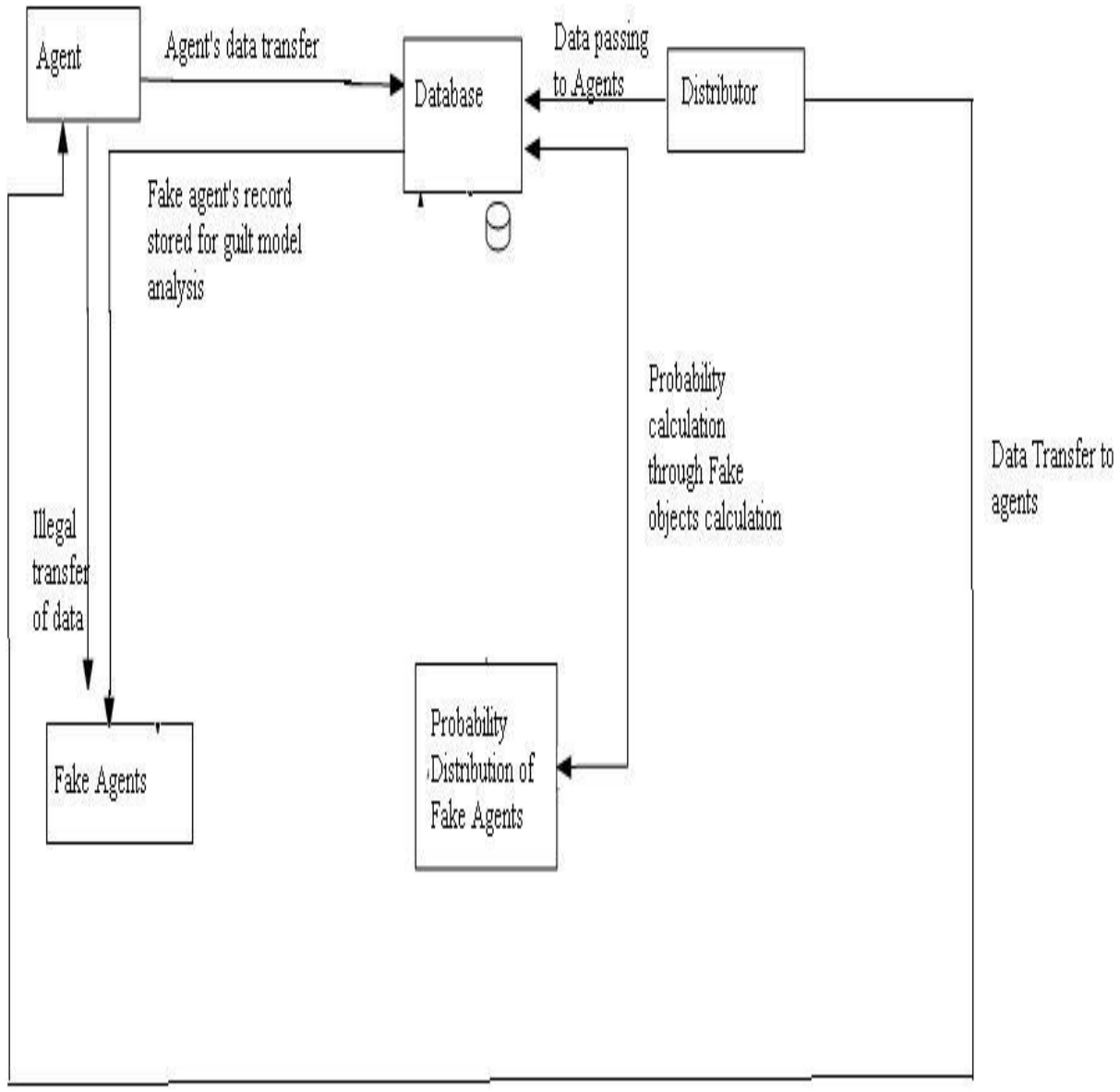

Figure 6. Architectural Structure of our Project

5.2 PROBLEM SETUP AND NOTATION

A distributor owns a set T = {t1 . . . t_m} of valuable data objects. The distributor wants to share some of the objects with a set of agents U1; U2; . . . ; Un, but does not wish the objects be leaked to other third parties. The objects in T could be of any type and size, e.g., they could be tuples in a relation, or relations in a database. An agent Uireceives a subset of objects Ri_ T, determined either by a sample request or an explicit request:

- Sample request Ri = **SAMPLE** (T, $m_{i)}$: Any subset ofmi records from T can be given to Ui.
- Explicit request Ri=**EXPLICIT** (T,$cond_i$): Agent Uireceives all T objects that satisfy condi.

5.3 GUILTY AGENTS

Suppose that after giving objects to agents, the distributor discovers that a set S (T)has leaked. This means that some third party, called the target, has been caught in possession of S. For example, this target may be displaying S on its website, or perhaps as part of a legal discovery process, the target turned over S to the distributor. Since the agents U1.. . . .Unhas some of the data, it is reasonable to suspect them leaking the data. However, the agents can argue that they are innocent, and that the S data were obtained by the target through other means. For example, say that one of the objects in S represents a customer X. Perhaps X is also a customer of some other company, and that company provided the data to the target or perhaps X can be reconstructed from various publicly available sources on the web. Our goal is to estimate the likelihood that the leaked data came from the agents as opposed to other sources. Intuitively, the more data in S, the harder it is for the agents to argue they did not leak anything. Similarly, the "rarer" the objects, the harder it is to argue that the target obtained them through other means. Not only do we want to estimate the likelihood the agents leaked data, but we would also like to find out if one of them, in particular, was more likely to be the leaker. For instance, if one of the S objects were only given to agent U1, while the other objects were given to all agents, we may suspect U1 more. The model we present next captures this intuition. We say an agent Uiis guilty

and if it contributes one or more objects to the target. We denote the event that agent Ui is guilty by Gi and the event that agent Ui is guilty for a given leaked set S by $G_i|$ S. Our next step is to estimate $P_r\{ G_i|$ S }, i.e., the probability that agent Ui is guilty given evidence S.

5.4 IMPLEMENTATION METHODS

5.4.1 DATA ALLOCATION –

The main focus of this paper is the data allocation problem: How can the distributor "intelligently" give data to agents in order to improve the chances of detecting a guilty agent? As illustrated in Fig. 1, there are four instances of this problem we address, depending on the type of data requests made by agents and whether "fake objects" are allowed.

5.4.2 FAKE OBJECT –

The distributor may be able to add fake objects to the distributed data in order to improve his effectiveness in detecting guilty agents. However, fake objects may impact the correctness of what agents do, so they may not always be allowable. The idea of perturbing data to detect leakage is not new, However, in most cases, individual objects are perturbed, e.g., by adding random noise to sensitive salaries, or adding a watermark to an image. In our case, we are perturbing the set of distributor objects by adding fake elements. In some applications, fake objects may cause fewer problems that perturbing real objects. Our use of fake objects is inspired by the use of "trace" records in mailing lists. For example In case, company A sells to company B a mailing list to be used once (e.g., to send advertisements). Company A adds trace records that contain addresses owned by company A. Thus, each time company B uses the purchased mailing list, A receives copies of the mailing. These records are a type of fake objects that help identify improper use of data. In many cases, the distributor may be limited in how many fake objects he can create. For example, objects may contain e-mail addresses, and each fake e-mail address may require the creation of an actual inbox (otherwise, the agent may discover that the object is fake). The inboxes can actually be monitored by the distributor: if e-mail is received from someone other than the agent who was given the address, it is

evident that the address was leaked. Since creating and monitoring e-mail accounts consumes resources, the distributor may have a limit of fake objects. If there is a limit, we denote it by B fake objects.

5.4.3 OPTIMIZATION –

The distributor's data allocation to agents has one constraint and one objective. The distributor's constraint is to satisfy agents' requests, by providing them with the number of objects they request or with all available objects that satisfy their conditions. His objective is to be able to detect an agent who leaks any portion of his data. We consider the constraint as strict. The distributor may not deny serving an agent request as and may not provide agents with different perturbed versions of the same objects. We consider fake object distribution as the only possible constraint relaxation.

5.5 ALLOCATION STRATAGIES

5.5.1 EXPLICIT DATA REQUEST –

In the first place, the goal of these experiments was to see whether fake objects in the distributed data sets yield significant improvement in our chances of detecting a guilty agent. In the second place, we wanted to evaluate our e-optimal algorithm relative to a random allocation.

ALGORITHM –

1: R ← ∅ *Agents that can receive fake objects*

2: for *i = 1, . . . , n do*

3: if *bi > 0 then*

4: R ← *R ∪{i}*

5: *Fi* ← ∅

6: while B > *0 do*

7: *i* ← *SELECTAGENT(R,R1, . . . , Rn)*

8: *f* ← *CREATEFAKEOBJECT(Ri, Fi, condi)*

9: *Ri* ← *Ri∪{f}*

10: *Fi* ← *Fi ∪{f}*

11: *bi* ← *bi − 1*

12: if *bi = 0 then*

13: R ← *R\{Ri}*

14: *B* ← *B – 1*

5.5.2 SAMPLE DATA REQUEST - With sample data requests agents are not interested in particular objects. Hence, object sharing is not explicitly defined by their requests. The distributor is "forced" to allocate certain objects to multiple agents only if the number of requested objects exceeds the number of objects in set T. The more data objects the agents request in total, the more recipients on average an object has; and the more objects are shared among different agents, the more difficult it is to detect a guilty agent.

ALGORITHM –

1: $a \leftarrow 0|T|$ *a[k]:number of agents who have received object tk*

2: $R1 \leftarrow \emptyset, \ldots, Rn \leftarrow \emptyset$

3: *remaining* ←

4: while *remaining* > *0 do*

5: for all $i = 1, \ldots, n : |Ri| < mi$ *do*

6: $k \leftarrow$ *SELECTOBJECT(i,Ri) May also use additional*Parameters

7: $Ri \leftarrow Ri \cup \{tk\}$

8: $a[k] \leftarrow a[k] + 1$

9: *remaining* ← *remaining* − *1*

SYSTEM DESIGN

CHAPTER 6

SYSTEM DESIGN

6.1 INTRODUCTION

System design is a process through which requirements are translated into a representation of software. Initially the representation depicts a holistic view of software. Subsequent refinement leads to a design representation that is very close to source code. Design provides us with representation of software development. Design is the only phase where user requirements are accurately translated into finished software product or system.

System design refers to modeling of a process. It is an approach to create a new system. It can be defined as a transition from user's view to programmer's view. The system design phase acts as a bridge between the required specification and the implementation phase. The design stage involves two sub stages namely:

- High - Level Design.
- Low - Level Design

6.2 HIGH LEVEL DESIGN

The purpose of the design phase is to plan a solution of the problem specified by the requirement document. This phase is the first step in moving from the problem domain to the solution domain. The design of the system is perhaps the most critical factor affecting the quality of the software. Here we build the System Block Diagram that will be helpful to understand the behavior of the system. Here we divide problem into modules. Data flow diagrams show flow of data between or among modules.

The High-level design has the following activities:

- Design Considerations: This section describes many issues, which need to be addressed or resolved before attempting to device a complete design solution.

- Assumptions and Dependencies: Describe any assumptions or dependencies regarding the software and its use.
- Development Methods: Describe the software development method used by the project.
- Data Flow Diagrams: This section describes the DFDs, which are the root part for any design.

6.3 DESIGN CONSIDERATIONS

The purpose of the design is to plan the solution of a problem specified by the requirements document. This phase is the first step in moving from problem to the solution domain. In other words, starting with what is needed design takes us to work how to satisfy the needs. The design of the system is perhaps the most critical factor affecting the quality of the software and has a major impact on the later phases, particularly testing and maintenance. System design aims to identify the modules that should be in the system, the specifications of these modules and to interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats as well as major modules in the system and their specifications are decided.

6.4 DATA FLOW DIAGRAM

A Data-Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process. Data flow diagrams are an intuitive way of showing how data is processed by a system At the analysis level, they should be used to model the way in which data is processed in the existing system. The notations used in these models represents functional processing, data stores and data movements between functions. Data flow models are used to show how data flows through a sequence of processing steps. The data is transferred at each step before moving on to the next stage. These processing steps or transformations are program functions when data flow diagrams are used to explain a software design.

6.5 DATA FLOW DIAGRAM

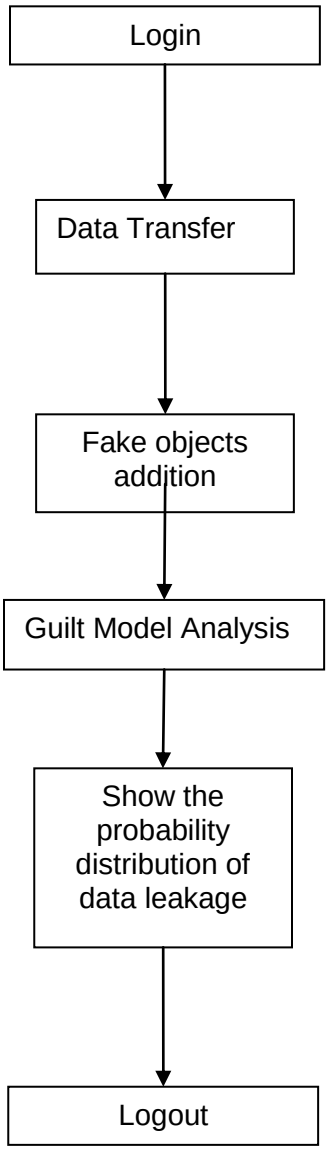

Figure 7. Data Flow Diagram

6.6 OBJECT DIAGRAM

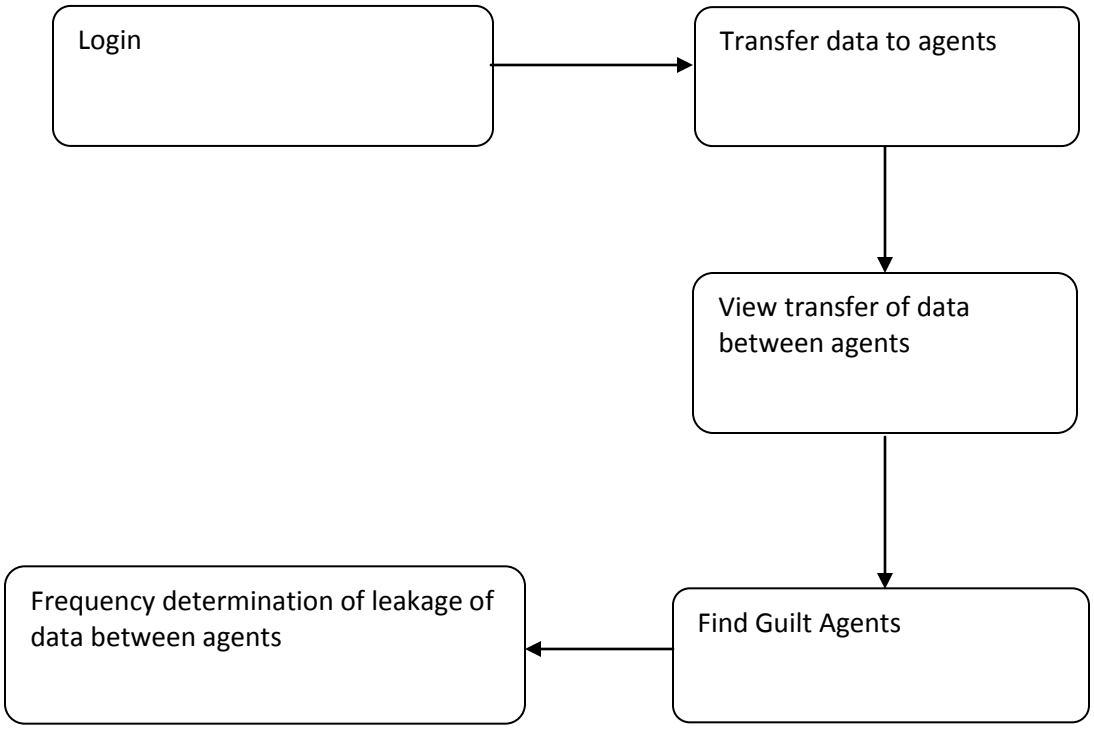

Figure 8. Object Diagram

6.6 PROJECT FLOW DIAGRAM

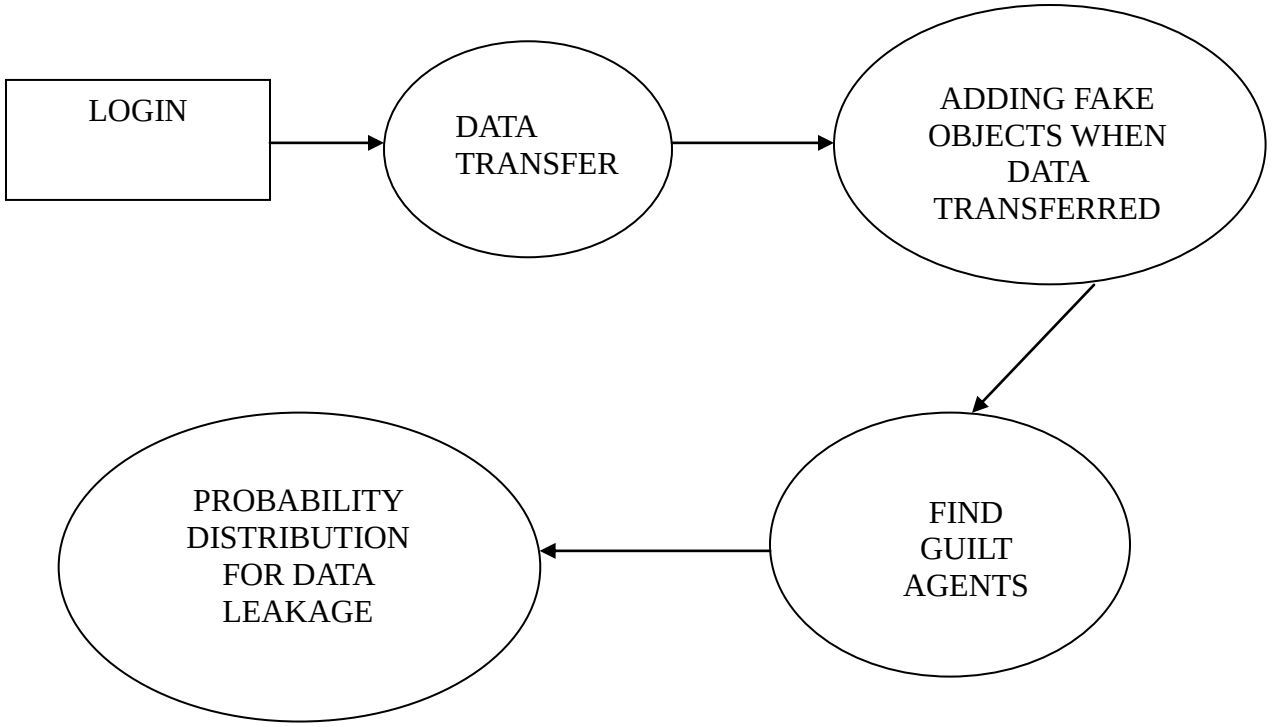

Figure .9 Project Flow Diagram

6.7 ACTIVITY DIAGRAM

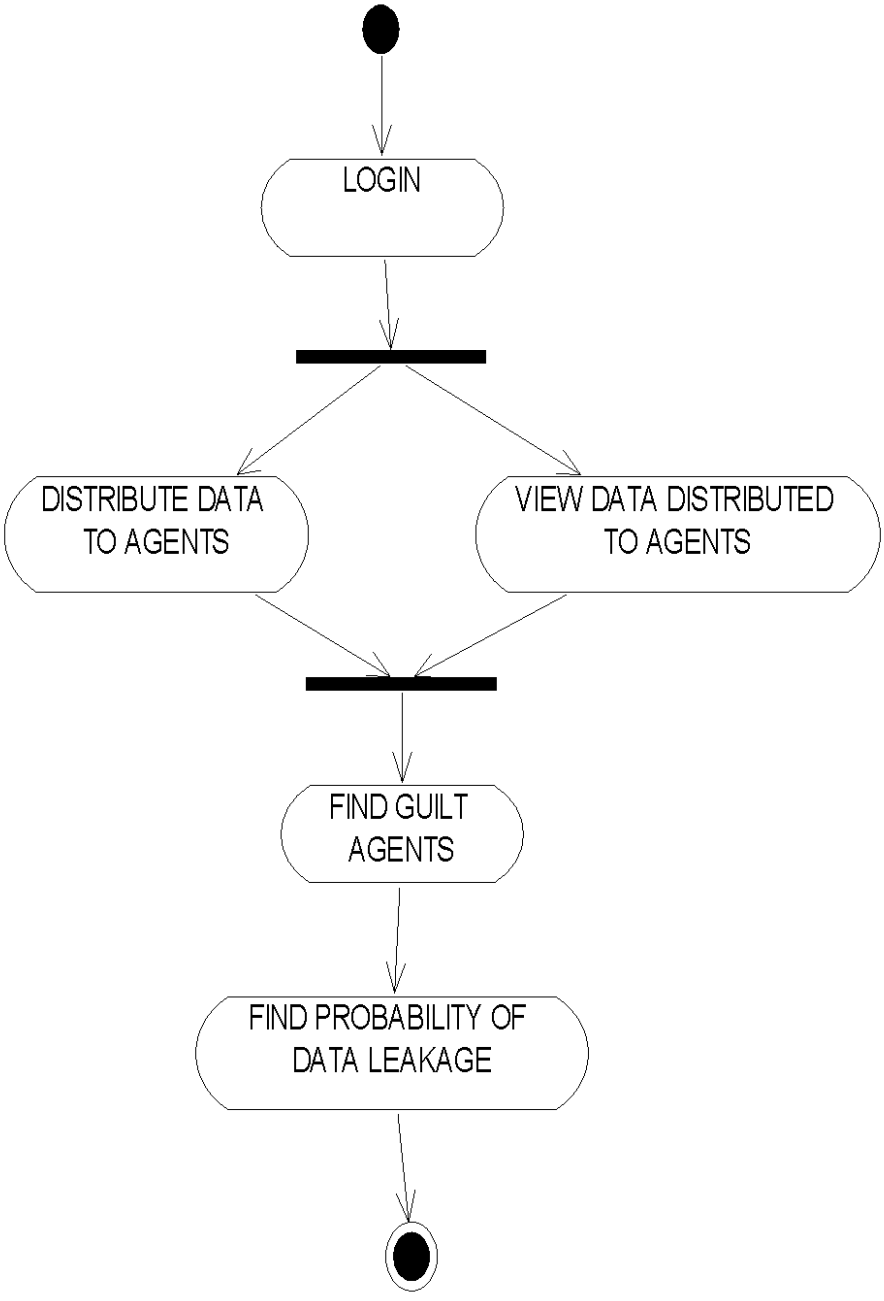

Figure 10. Activity Diagram followed in our Project

6.8 USECASE DIAGRAM

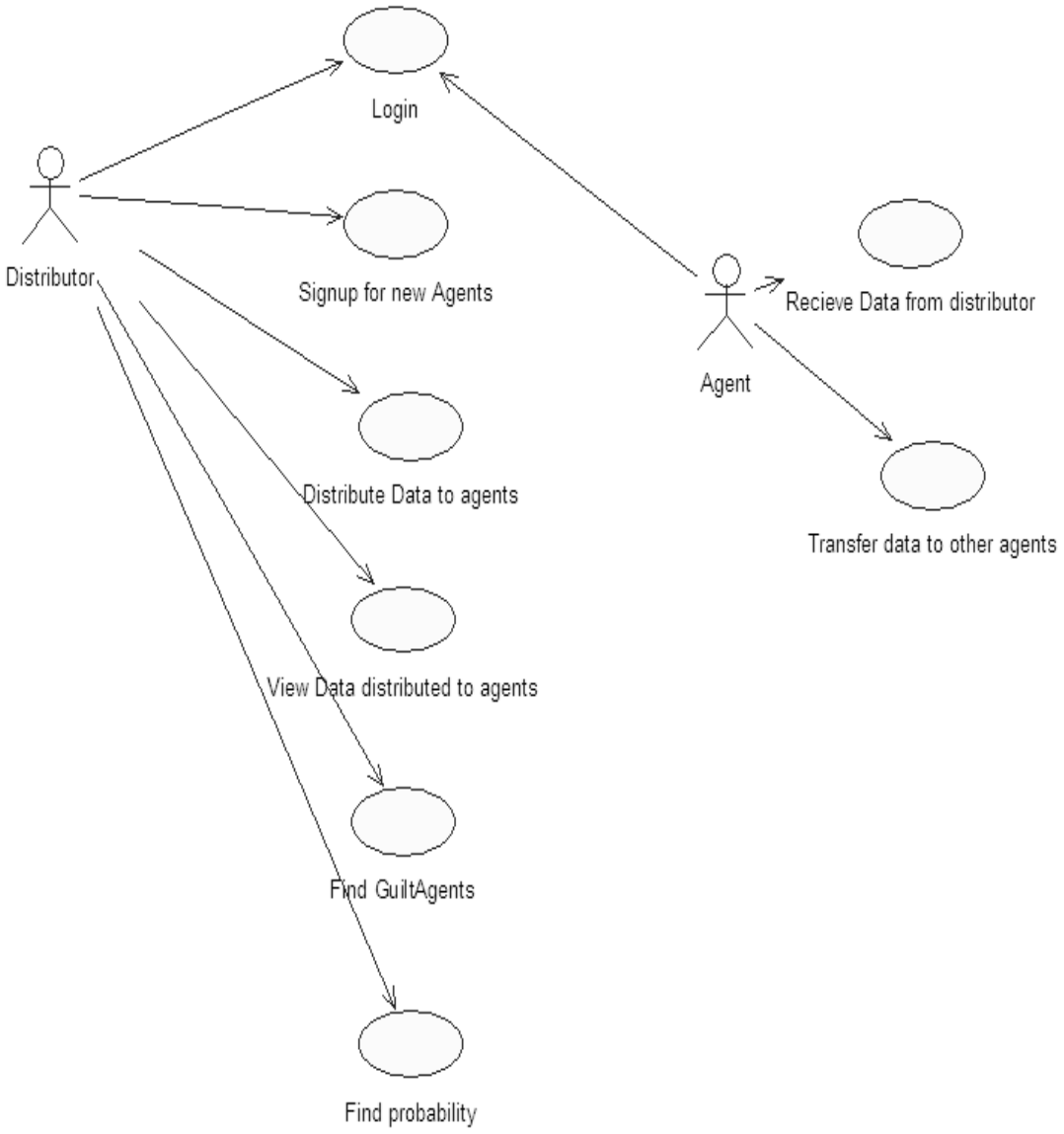

Figure 11. Use Case Diagram

6.9 SEQUENCE DIAGRAM

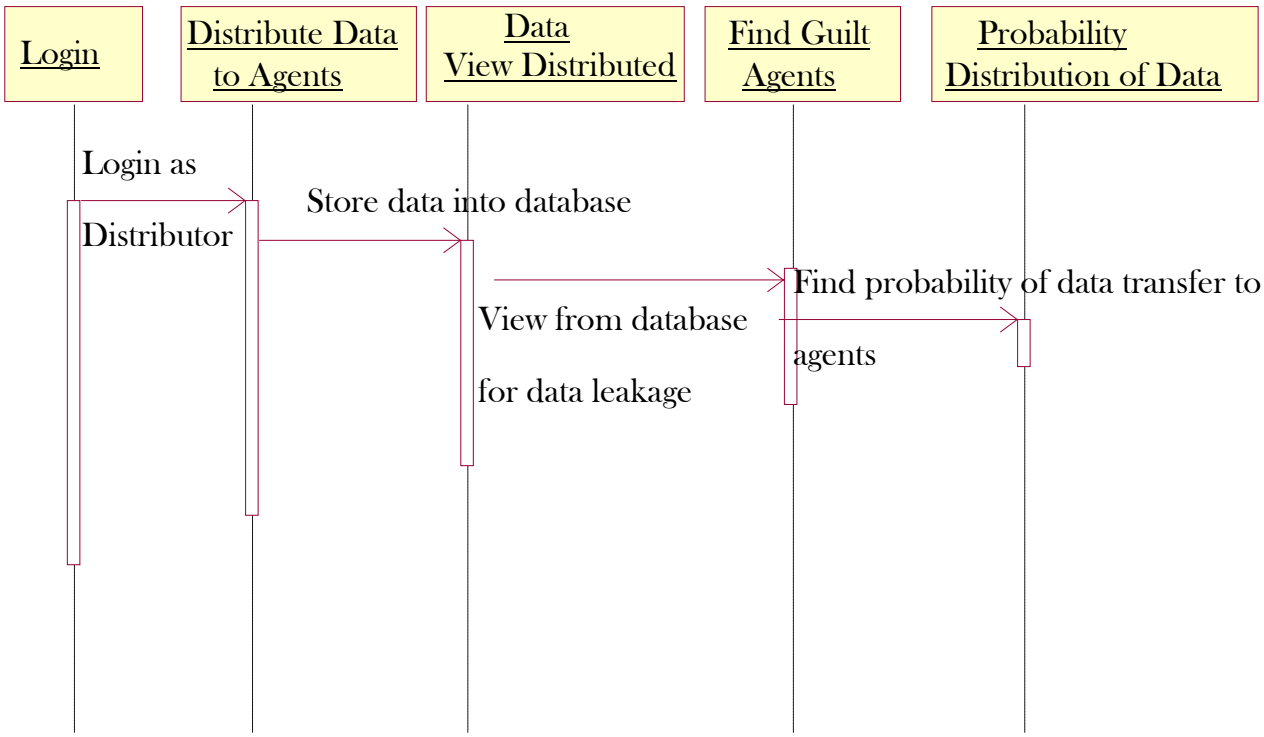

Figure 12. Sequence Diagram, this will brief us through the complete sequence in which our Project runs

6.10 COLLABORATION DIAGRAM

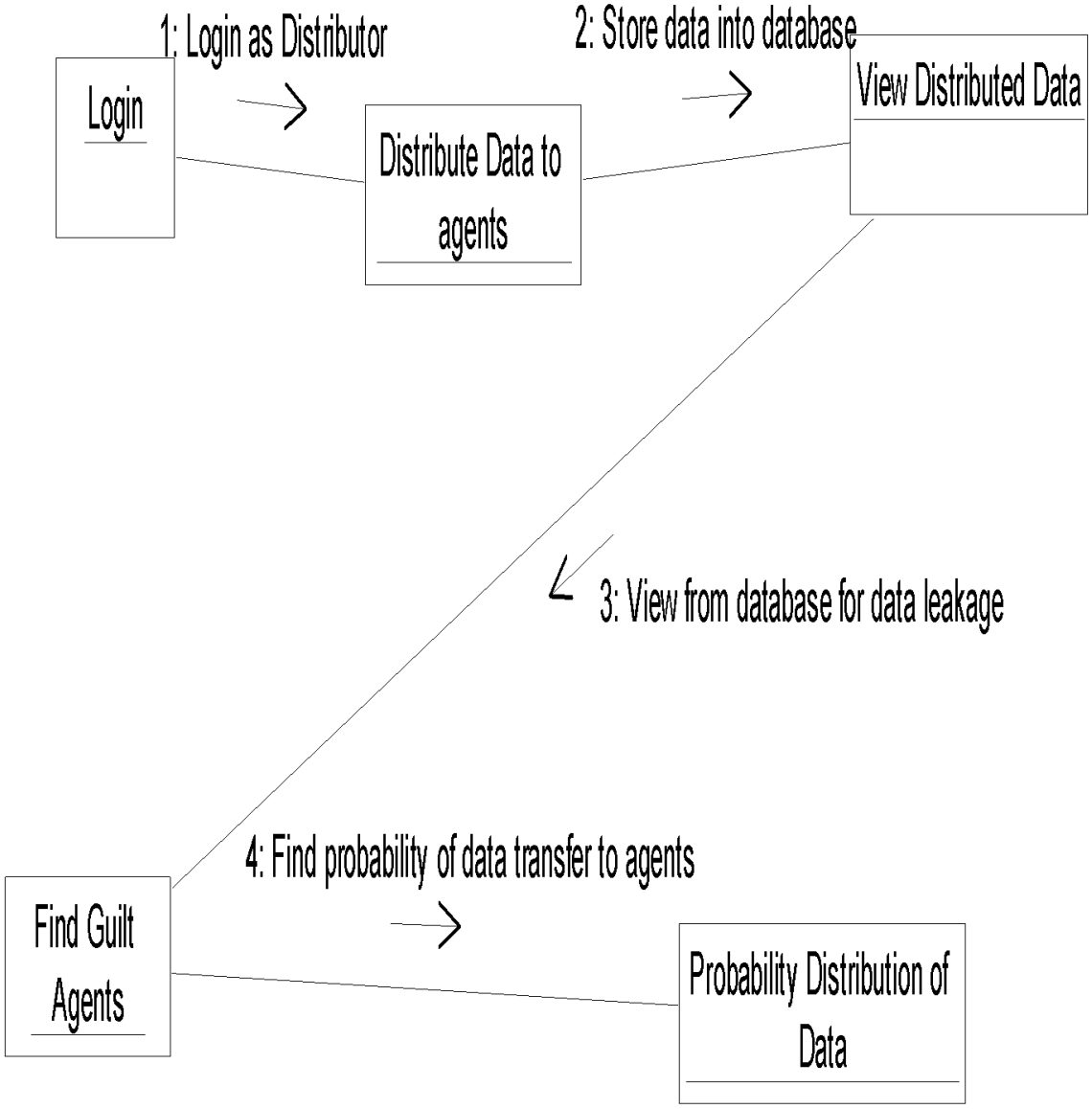

Figure 13. Collaboration Diagram

6.11 ENTITY RELATIONSHIP DIAGRAM

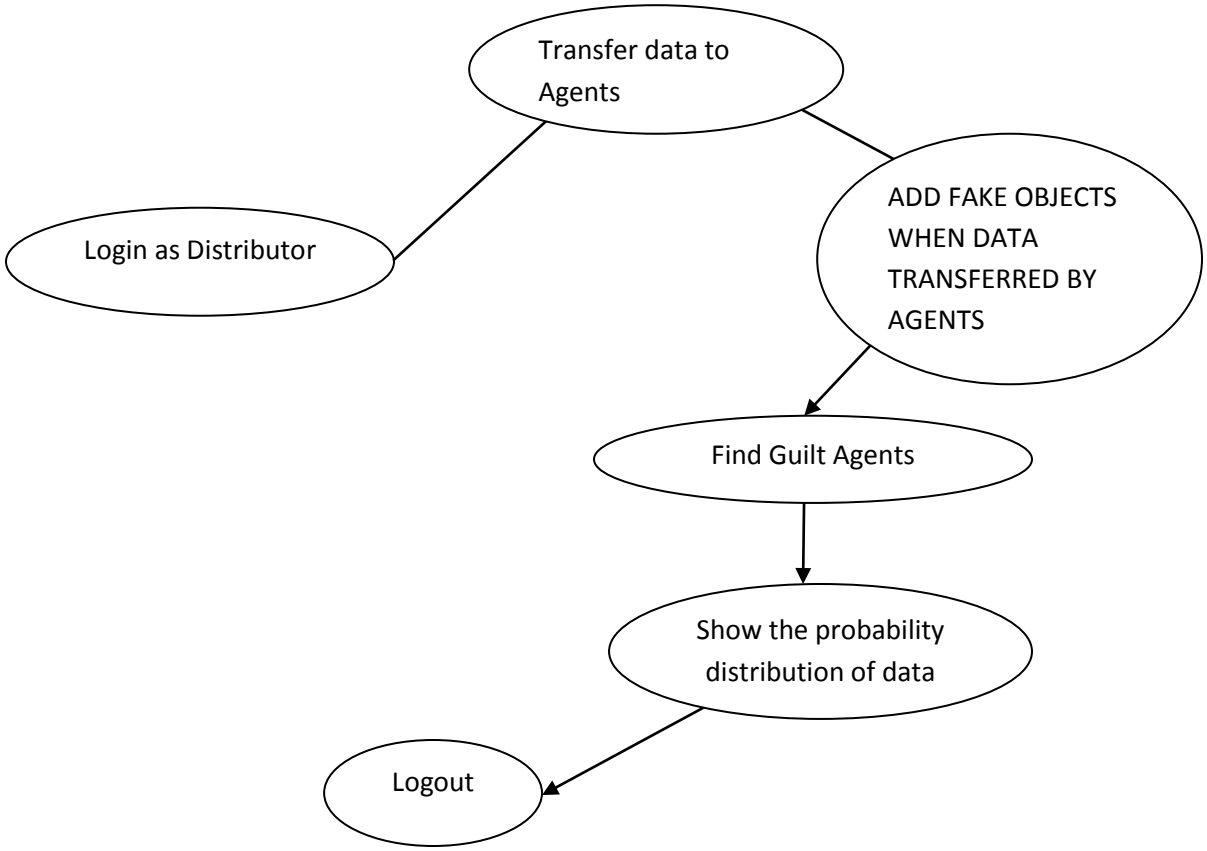

Figure 14. Entity Relation Diagram of Data Leakage Detection.

IMPLEMENTATION

7.1 SNAPSHOTS

Figure 15. Welcome Page, to login as Distributor or as Agents

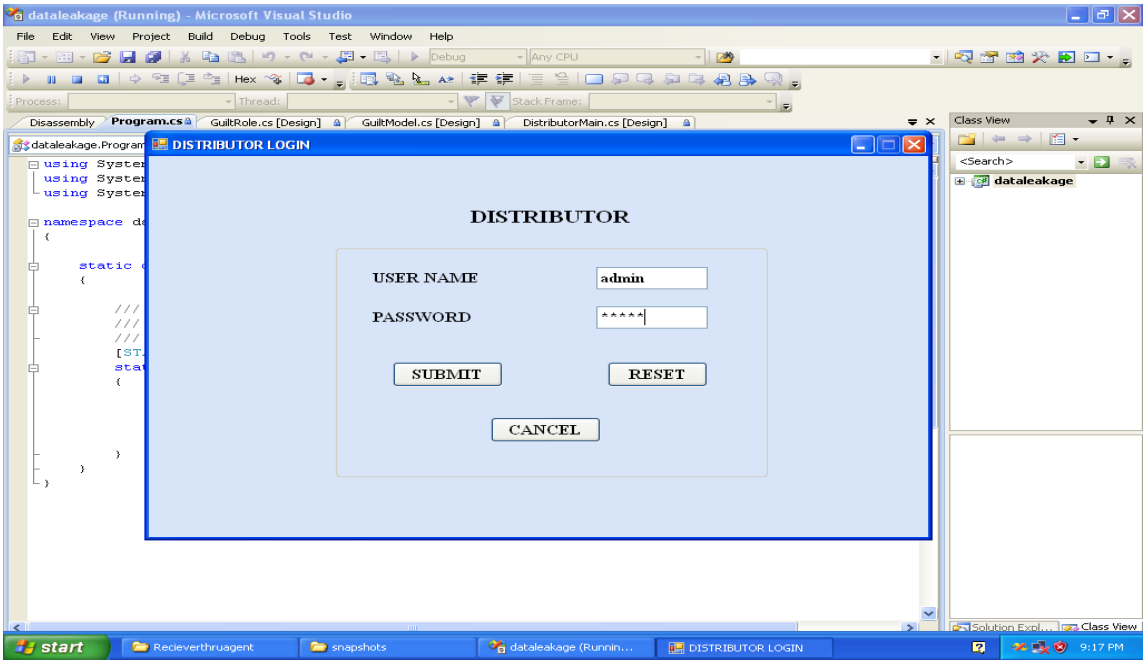

Figure 16. Distributor Login Page

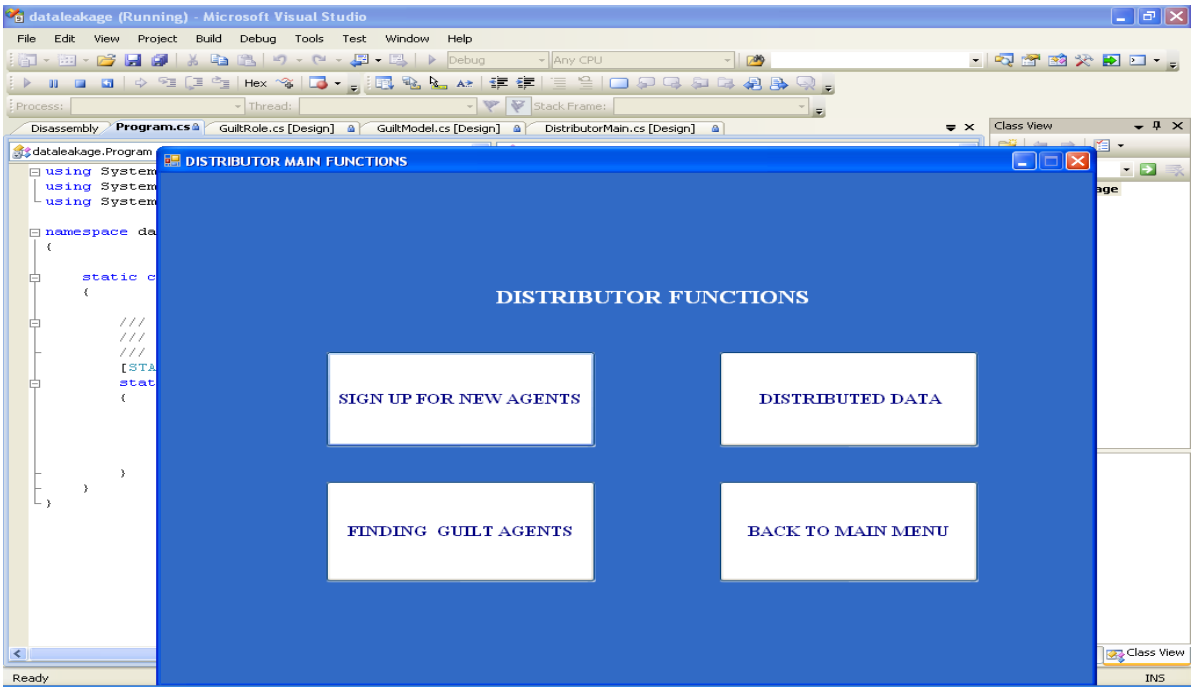

Figure 17. Distributor Functions

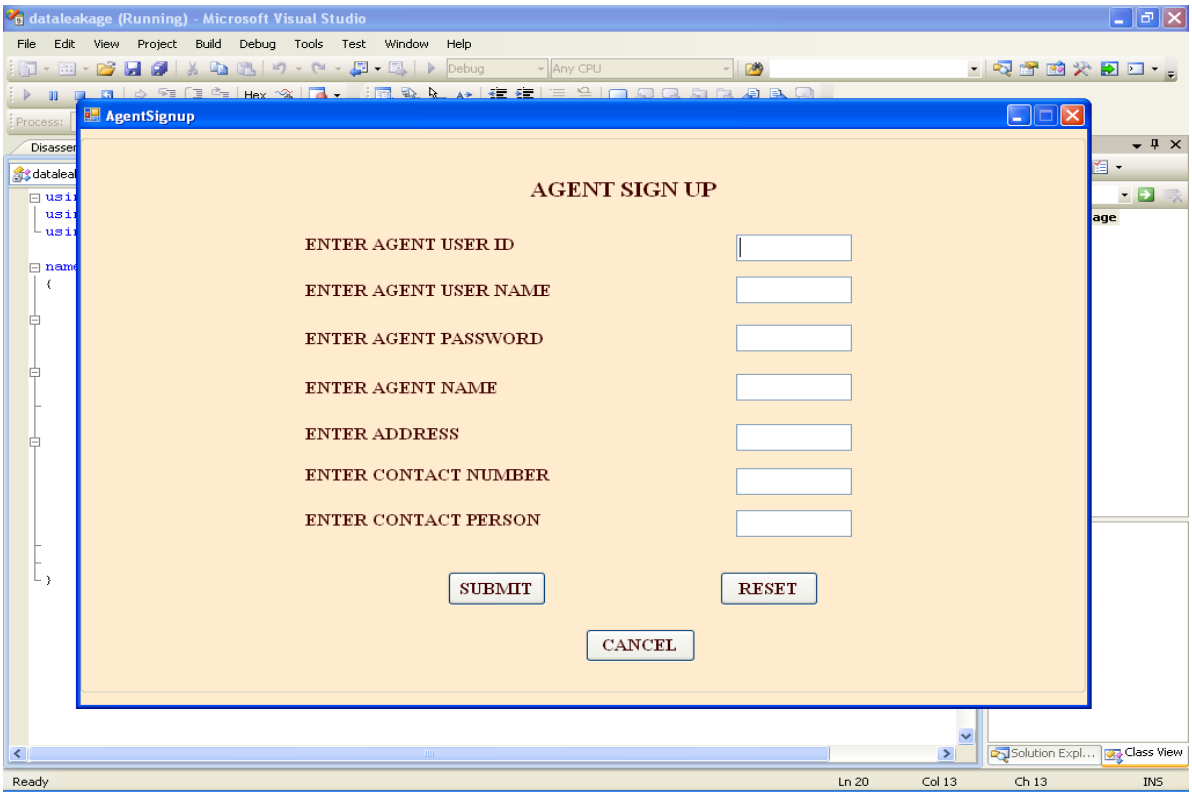

Figure 18. New Agent Signup or Creation

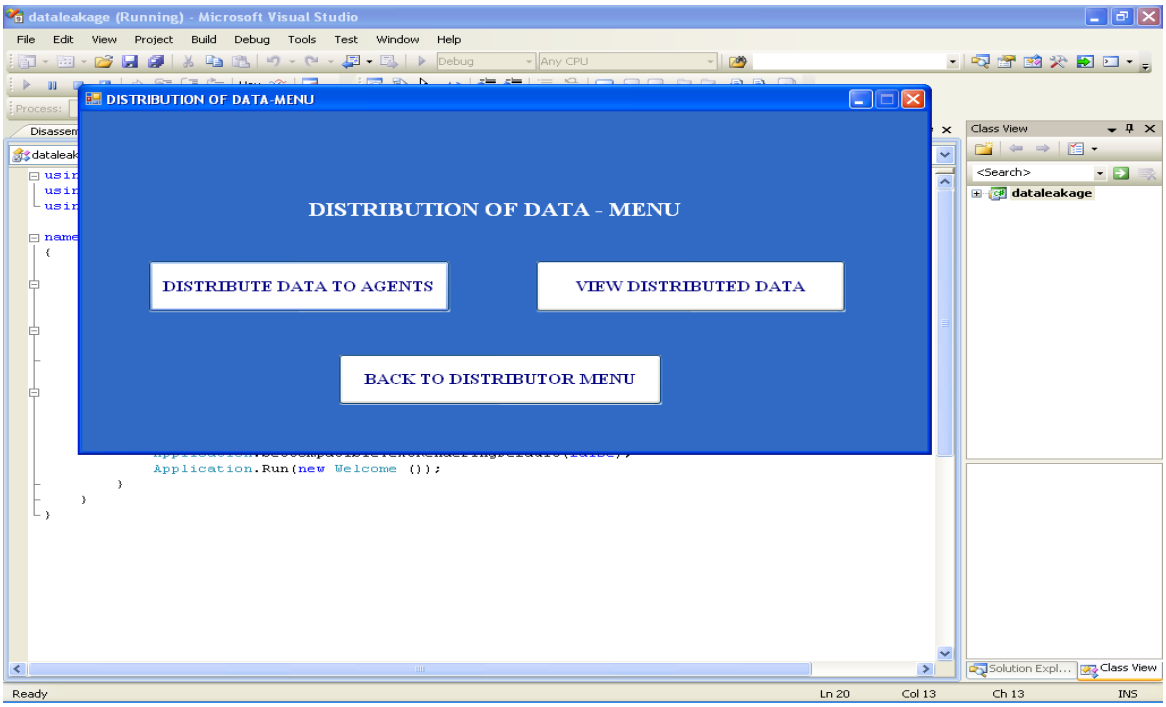

Figure 18. Distributor's Data Distribution Menu

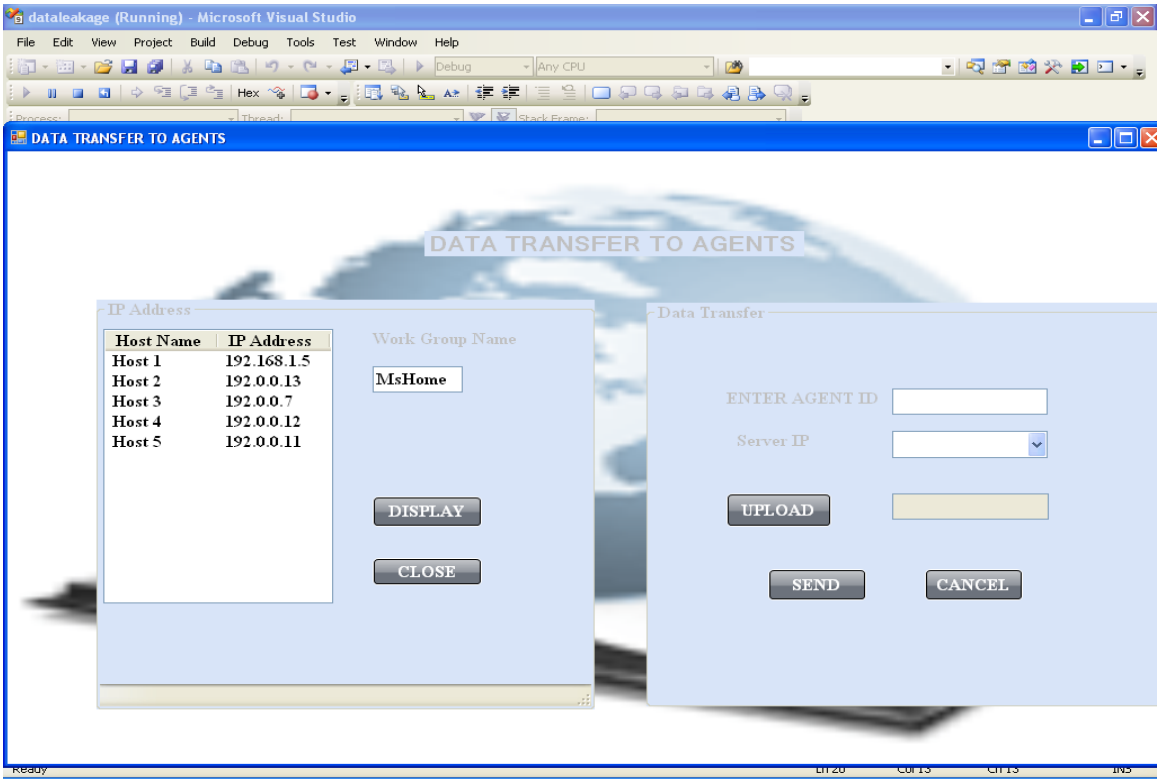

Figure 20. Distributor transferring Data to an Agent

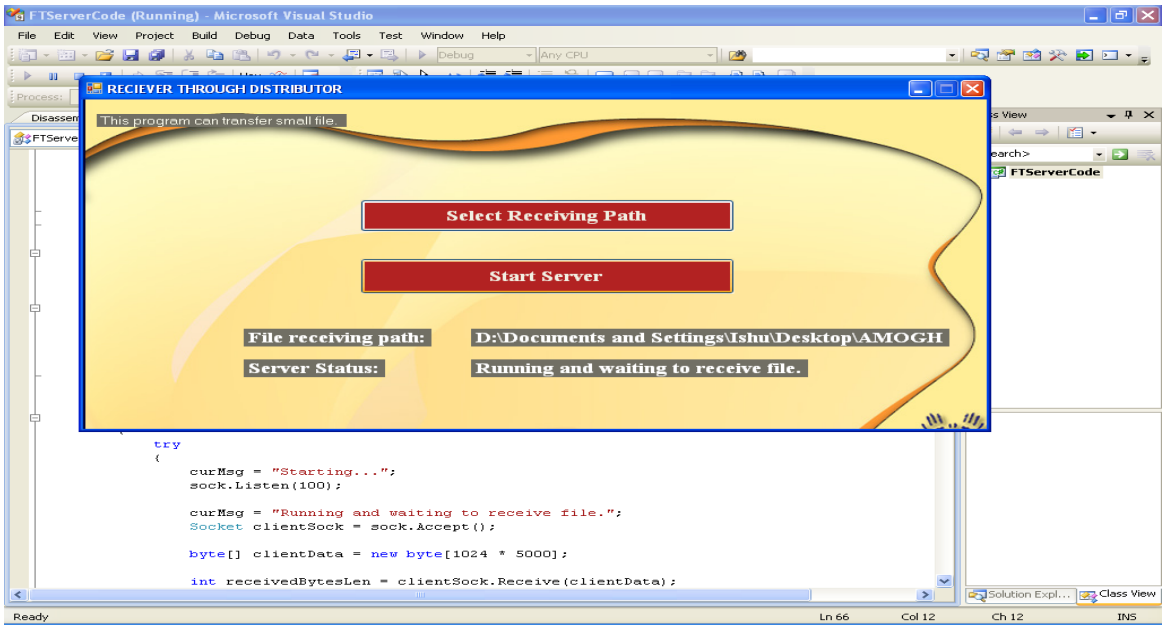

Figure 20. Distributor starting the server to send Data to an Agent to a selected path.

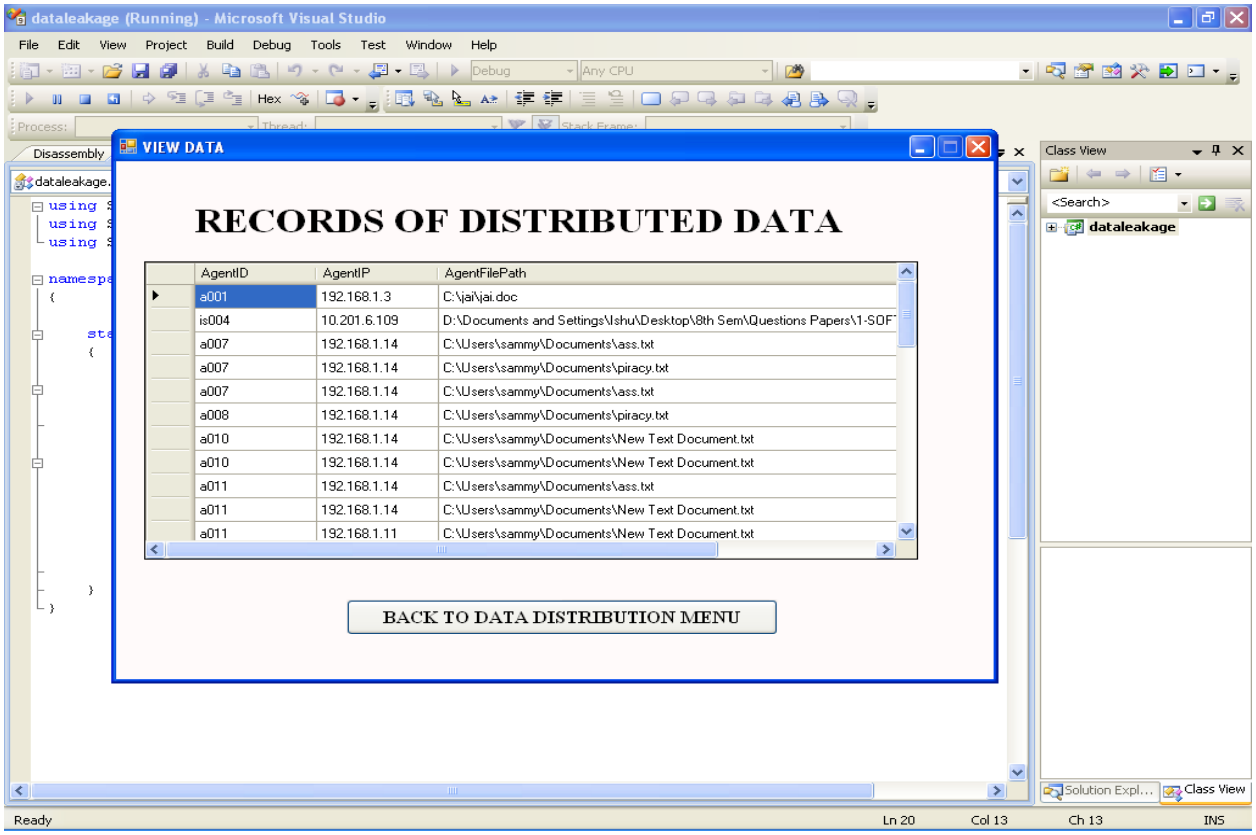

Figure 21. Distributed Data from Distributor

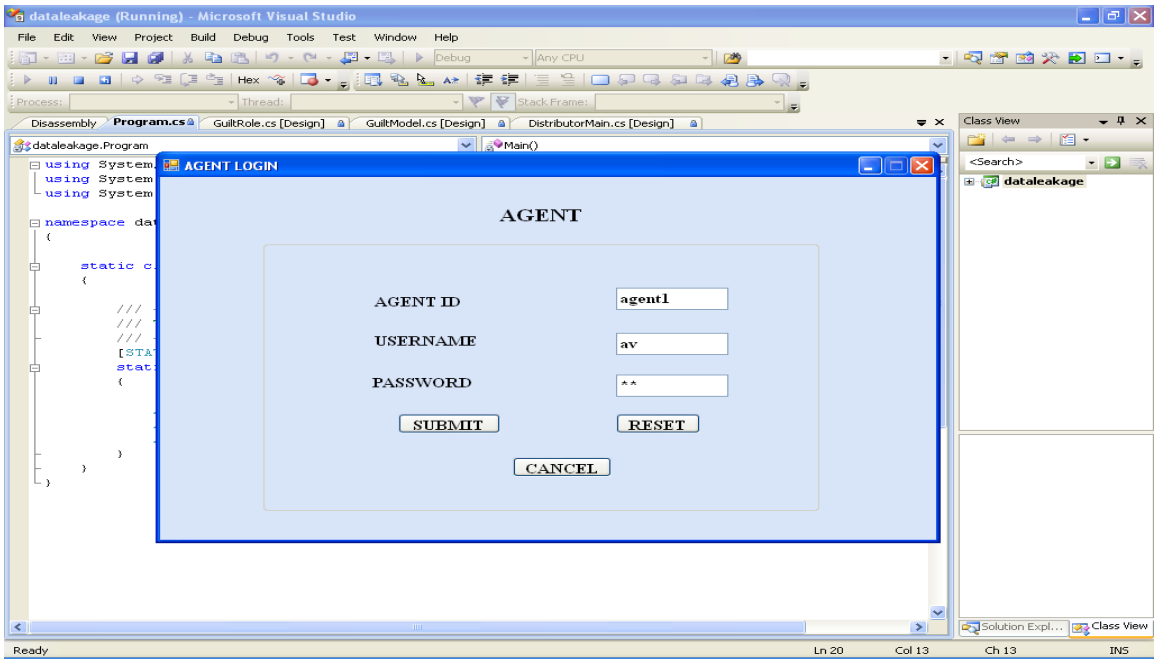

Figure 23. Agent Login Screen

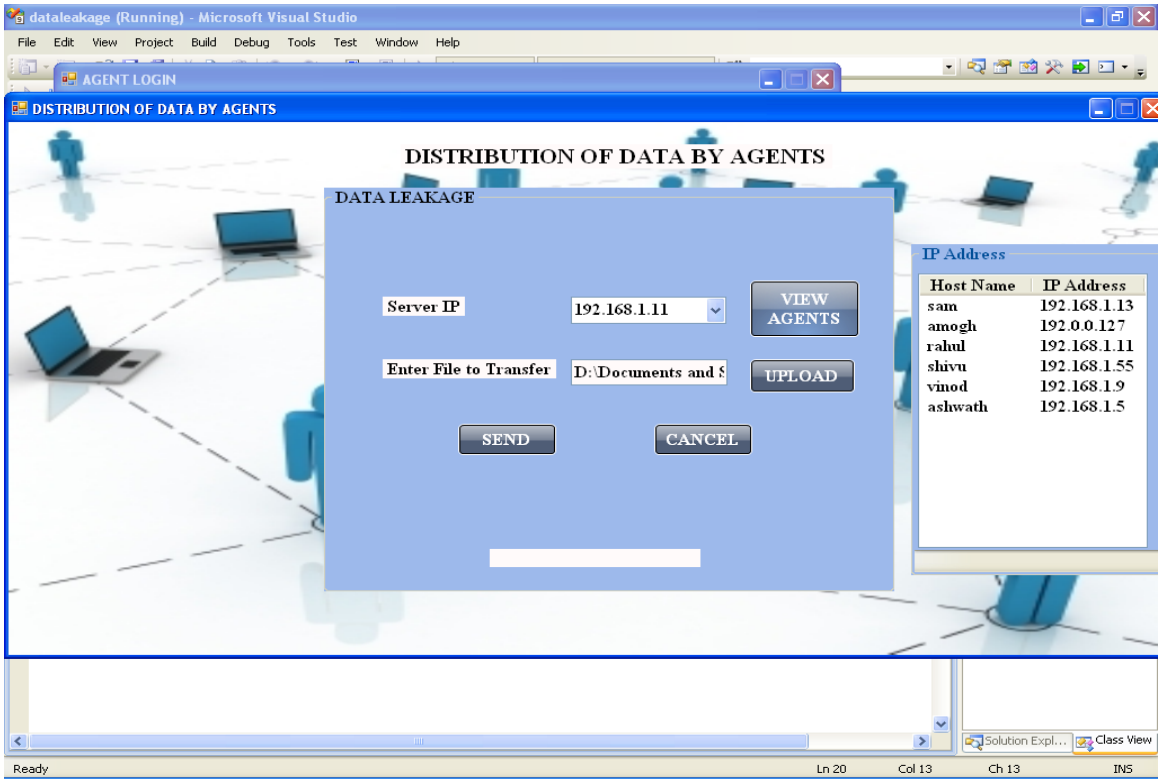

Figure 24. Agent forwarding Data (leaking data)

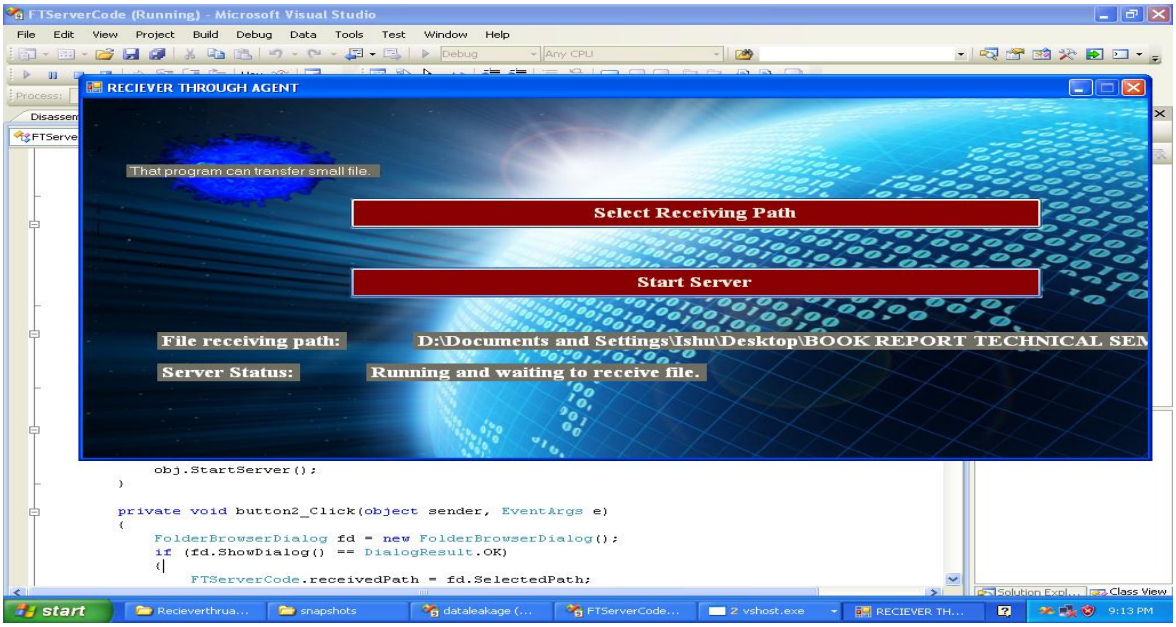

Figure 25. Agent starting Server to leaking the Data

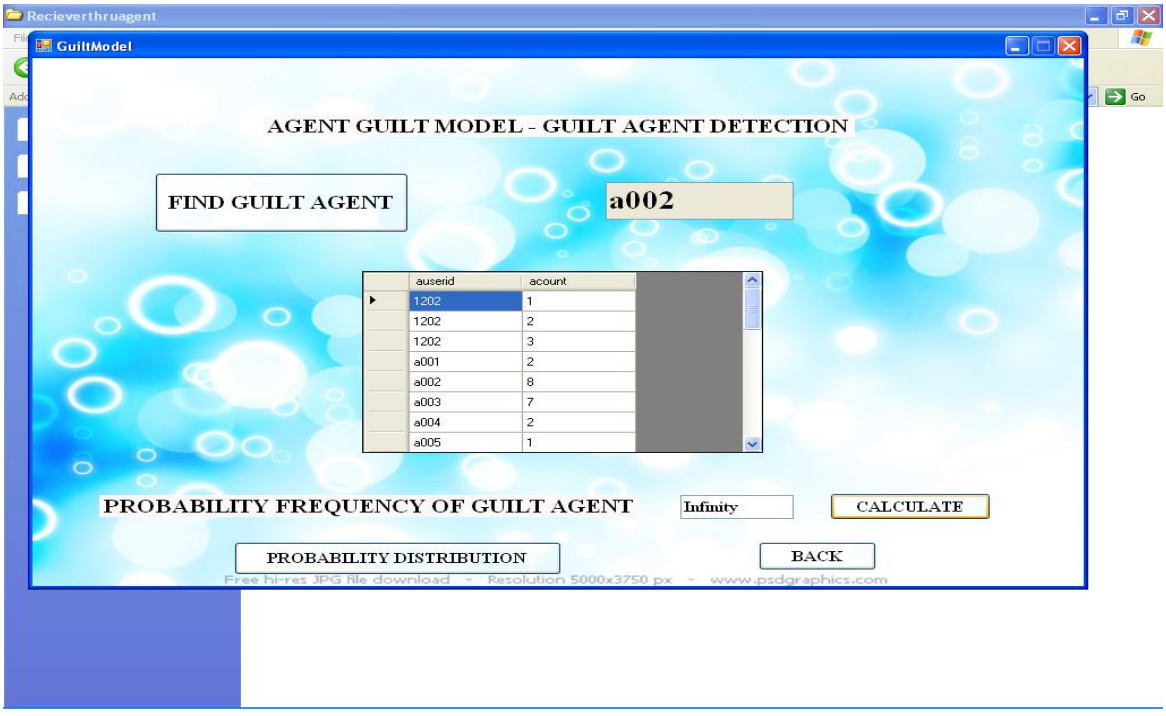

Figure 26. Distributor calculating the Probability of each Agent being Guilty

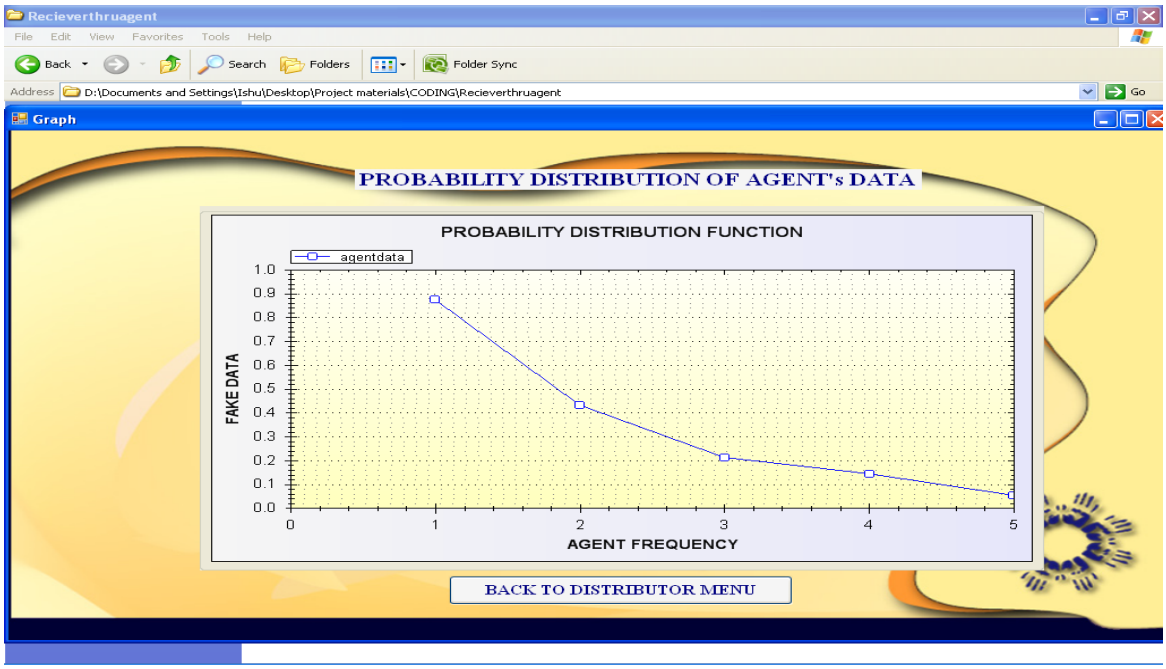

Figure 26. Graphical representation of Probability of Agent being Guilty

SYSTEM TESTING

CHAPTER 8

SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the

Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

8.1 TYPES OF TESTS

UNIT TESTING -Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

INTEGRATION TESTING -Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

FUNCTIONAL TEST -Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

SYSTEM TESTING-System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

WHITE BOX TESTING -White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

BLACK BOX TESTING -Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

8.2 UNIT TESTING

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Sl.no	Input	Expected Output	Actual Output	Status
01	Welcome Page	A page with distributor login and agent login	A page with distributor login and agent login	Pass
02	Distributor Login	Login page with user name and password arrives	Login Page with user name password arrives	Pass
03	Distributor Function	The 4 sub models of distributor login	The 4 sub models of distributor login	Pass
04	Sign up new agents	A sign up page where the user has to enter personal details	A sign up page where the user has to enter personal details	Pass
05	Distribute data via the distributor to agents	A page with a set of hosts arrive and can send data to various agents	A page with a set of hosts arrive and can send data to various agents	Pass
06	Distributor Server	Before sending data from the distributor this server has to be turned on	Before sending data from the distributor this server has to be turned on	Pass
07	Return to Welcome Page	A page with distributor login and agent login	A page with distributor login and agent login	Pass
08	Login as Agent	A page where the agent enters the agent details	A page where the agent enters the agent details	Pass

09	Agent Leaking data to different agents	A page where agent can leak data arrives	A page where agent can leak data arrives	Pass
10	Return to Welcome Page	A page with distributor login and agent login	A page with distributor login and agent login	Pass
11	Distributor Login	Login page with user name and password arrives	Login page with user name and password arrives	Pass
12	Find Guilty Agents	A Page for finding guilty agents arrive	A Page for finding guilty agents arrive	Pass
13	Find probability of guilty agent	Find the guilty agent	Find the guilty agent	Pass
14	Show the probability graph	Graph shown	Graph shown	Pass

8.3 INTEGRATION TESTING

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results - All the test cases mentioned above passed successfully. No defects encountered.

8.4 ACCEPTANCE TESTING

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

FUTURE ENHANCEMENT

CHAPTER 9

FUTURE ENHANCEMENT

Our future work includes the investigation of agentguilt models that capture leakage scenarios that are not studied in this project. For example, what is the appropriate model for cases where agents can collude and identify fake tuples? A preliminary discussion of such a model is available in. Another open problem is the extension of our allocation strategies so that they can handle agent requests in an online fashion (the presented strategies assume that there is a fixed set of agents with requests known in advance).

Any application does not end with a single version. It can be improved to include new features. Our application is no different from this. The future enhancements that can be made to Data Leakage Detection are:

- Providing support for other file formats.
- Creation of a web based UI for execution of the application.
- Improving the detection process based on user requirements.
- Provision of quality or accuracy variance parameter for the user to set.

CONCLUSION

CHAPTER 10

CONCLUSION

In a perfect world, there would be no need to hand over sensitive data to agents that may unknowingly or maliciously leak it. And even if we had to hand over sensitive data, in a perfect world, we could watermark each object so that we could trace its origins with absolute certainty. However, in many cases, we must indeed work with agents that may not be 100 percent trusted, and we may not be certain if a leaked object came from an agent or from some other source, since certain data cannot admit watermarks. In spite of these difficulties, we have shown that it is possible to assess the likelihood that an agent is responsible for a leak, based on the overlap of his data with the leaked data and the data of other agents, and based on the probability that objects can be "guessed" by other means. Our model is relatively simple, but we believe that it captures the essential trade-offs. The algorithms we have presented implement a variety of data distribution strategies that can improve the distributor's chances of identifying a leaker. We have shown that distributing objects judiciously can make a significant difference in identifying guilty agents, especially in cases where there is large overlap in the data that agents must receive.

BIBLIOGRAPHY

CHAPTER 11

BIBLIOGRAPHY

Good Teachers are worth more than thousand books, we have them in Our Department.

REFERENCES HAVE BEEN MADE FROM

- R. Agrawal and J. Kiernan, "Watermarking Relational Databases," Proc. 28th Int'l Conf. Very Large Data Bases (VLDB '02), VLDB Endowment, pp. 155-166, 2002.

- P. Bonatti, S.D.C. di Vimercati, and P. Samarati, "An Algebra for Composing Access Control Policies," ACM Trans. Information andSystem Security, vol. 5, no. 1, pp. 1-35, 2002.

- P. Buneman, S. Khanna, and W.C. Tan, "Why and Where: A Characterization of Data Provenance," Proc. Eighth Int'l Conf.Database Theory (ICDT '01), J.V. den Bussche and V. Vianu, eds.,pp. 316-330, Jan. 2001.

- P. Buneman and W.-C. Tan, "Provenance in Databases," Proc. ACM SIGMOD, pp. 1171-1173, 2007.

- Y. Cui and J. Widom, "Lineage Tracing for General Data Warehouse Transformations," The VLDB J., vol. 12, pp. 41-58, 2003.

- S. Czerwinski, R. Fromm, and T. Hodes, "Digital Music Distributionand Audio Watermarking," http://www.scientificcommons.org/43025658, 2007.

- F. Guo, J. Wang, Z. Zhang, X. Ye, and D. Li, "An Improved Algorithm to Watermark Numeric Relational Data," Information Security Applications, pp. 138-149, Springer, 2006.

- F. Hartung and B. Girod, "Watermarking of Uncompressed and Compressed Video," Signal Processing, vol. 66, no. 3, pp. 283-301, 1998.

- S. Jajodia, P. Samarati, M.L. Sapino, and V.S. Subrahmanian,"Flexible Support for Multiple Access Control Policies," ACMTrans. Database Systems, vol. 26, no. 2, pp. 214-260, 2001.

- Y. Li, V. Swarup, and S. Jajodia, "Fingerprinting RelationalDatabases: Schemes and Specialties," IEEE Trans. Dependable and Secure Computing, vol. 2, no. 1, pp. 34-45, Jan.-Mar. 2005.