Trust based Identity Provisioning for Cloud Computing

Thesis submitted in partial fulfillment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Aradhana
(Roll No. 800931004)

Under the supervision of:
Name of Supervisor(s)
Dr. Inderveer Chana
Assistant Professor
CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

June 2011

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "Trust based Identity Provisioning for Cloud Computing", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Inderveer Chana* and refers other researcher's work which are duly listed in the reference section.

The matter presented in the thesis has not been submitted for award of any other degree of this or any other University.

Signature: Aradhana

(Aradhana)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Inderveer 17/06/11

(Dr.Inderveer Chana)
Assistant Professor,
CSED

Countersigned by

(Dr. Maninder Singh)
Head 17/6/11
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. K. Mohapatra)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

First of all, I want to thank Almighty who has always guided me to work on the right path of the life. My greatest thanks to my parents who bestowed the ability and strength needed to complete this work on me. I am deeply indebted to my friends Sonam, Ipneet, Vaneet, Ravneet Chawla, Ravneet Grewal, Aarti, Arpita, Vishonika and Aman for their ever encouraging moral support and help which enabled me to pursue my studies.

This work would not have been possible without the encouragement and able guidance of my supervisor, to **Dr Inderveer Chana**, Assistant Professor, Computer Science and Engineering Department, Thapar University, Patiala. Her enthusiasm and optimism made this experience both rewarding and enjoyable. Most of the novel ideas and solutions founded in this thesis are the result of our various stimulating discussion sessions. Her feedback and editorial comments were also valuable for writing the thesis.

I am also heartily thankful to Mrs Anju Bala, Assistant Professor, Computer Science and Engineering Department, Thapar University, Patiala for the motivation and inspiration that triggered me for my thesis work

I would also like to express my sincere gratitude towards the entire faculty and staff of the Computer Science & Engineering Department, for their help, cooperation, love and affection, which made my stay at Thapar University memorable.

.

Aradhana
(800931004)

Abstract

Cloud computing is current buzzword in the market. It is paradigm in which the resources can be leveraged on per use basis thus reducing the cost and complexity of service providers. Cloud computing promises to cut operational and capital costs and more importantly let IT departments focus on strategic projects instead of keeping datacenters running. It is much more than simple internet. It is a construct that allows user to access applications that actually reside at location other than user's own computer or other Internet-connected devices. There are numerous benefits of this construct. For instance other company hosts user application. This implies that they handle cost of servers, they manage software updates and depending on the contract user pays less i.e. for the service only.

Despite of so many cloud benefits its usage has been hindered due to security issues which need to be addressed. Identity management in cloud requires new dimensions to be understood. In a federated environment a seamless interaction is required between service providers and identity providers thus increasing user experience. This necessitates building of dynamic trust relationships for efficient federated identity management.

In this thesis current identity frameworks have been compared, importance of Trust in Identity management has been analyzed and risk factor has been incorporated in Trust Management. Further Trust policies have been designed and validated on Google App Engine using Python.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter introduces cloud computing and explains various related technologies like distributed computing, grid computing and various cloud categories and the services cloud offers along with the organisation of the thesis.

1.1 Evolution

In the 1980s and 1990s, with the rise of PCs, the shrinking costs of networking and computing infrastructure, and a need for more agility, client/server provided the ability to split the application tier away from the server tier [29]. This was done to support distributed clients running richer user interfaces and also to reduce costs by offloading the user handling, application workloads off monolithic servers.

2000 onwards, as data centers started to fill out, and power, space and cooling became more and more expensive, concepts such as commodity grid computing and virtualization started to become established [14]. Cloud computing takes these concepts further by allowing self-service, metered usage and more automated dynamic resource and workload management practices. Figure 1.1 shows the Cloud evolution.

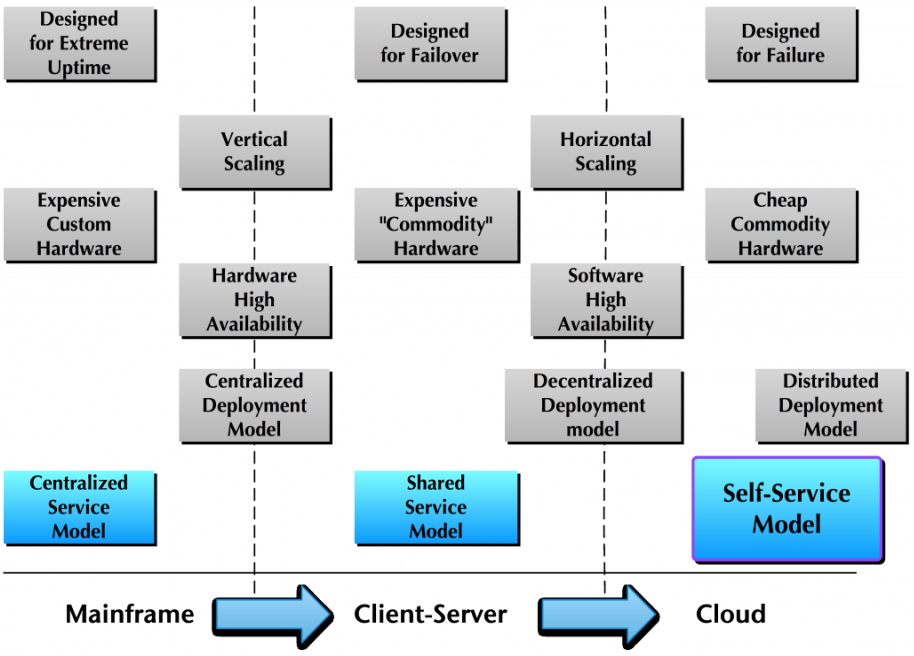

Figure 1.1: Computing paradigm shift [12]

1.2 Cloud Computing

The term 'Cloud' first appeared in the early 1990s, referring mainly to large ATM networks. Cloud computing began in the beginning of this century, just a short nine years ago with the advent of Amazon's web-based services. Less than three years ago, Yahoo and Google announced plans to provide cloud computing services to some countries' largest universities: Carnegie Mellon, University of Washington, Stanford and MIT [28]. The IBM quickly announced plans to offer cloud computing technologies. More recent entries into the encounter include well-known companies: Sun, Intel, Oracle, SAS and Adobe. All of these companies invested mightily in cloud computing infrastructure to provide vendor-based cloud services to the masses [13].

Cloud computing has become a buzzword of today. Cloud Computing is not a completely new concept; it has intricate connection to the established Grid Computing paradigm, and other relevant technologies such as utility computing, cluster computing, and distributed systems in general [12]. The term "cloud" is used as a metaphor for the internet.

Cloud Computing is a concept of computing in which dynamically scalable and often virtualized resources are provided as a service over the internet. Cloud Computing consists of hardware and software resources made available on the internet as managed by third-party services. These services typically provide access to advanced software applications and high-end networks of server computers [13].

To get Cloud Computing to work, three things are required: thin clients (or clients with a thick-thin switch), grid computing, and utility computing. Grid computing links disparate computers to form one large infrastructure, harnessing unused resources. Utility computing is paying for what users use on shared servers like consumers pay for a public utility such as electricity, gas, and so on [9].

1.2.1 Characteristics of Cloud

There is a level of consensus emerging around the characteristics of cloud computing, or the capabilities that must be adhered to an offering to be considered a cloud. These include [14]:

- Pay as you go – payment is variable based on the actual consumption by the customer.
- Highly abstracted – server hardware and related network infrastructure is highly abstracted from the users.

- Multi-tenant – multi-tenant architectures allow numerous customer enterprises to subscribe to the cloud computing capabilities while retaining privacy and security over their information.
- Immediately scalable – usage, capacity, and therefore cost, can be scaled up or down with no additional contract or penalties.

1.2.2 Cloud Categories

Public cloud

Public cloud or external cloud describes cloud computing in the traditional mainstream sense, whereby resources are dynamically provisioned on a fine-grained, self-service basis over the Internet, via web applications/web services, from an off-site third-party provider who share resources and bills on a fine-grained utility computing basis [6].

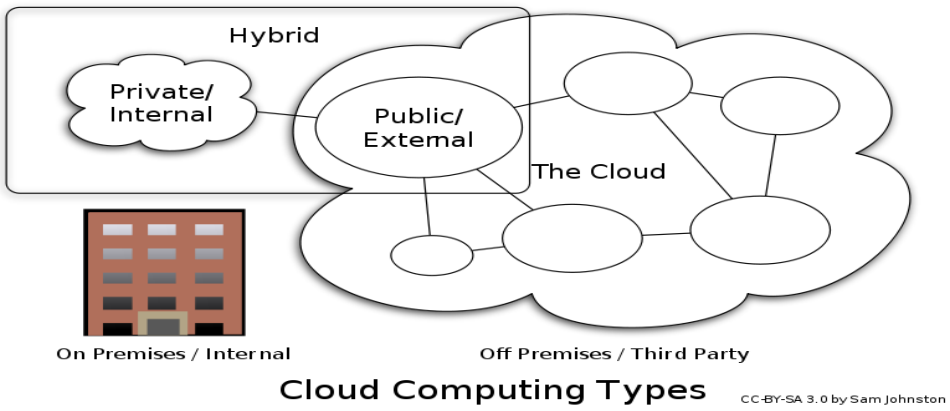

Figure 1.2: Cloud Computing Types [6]

Hybrid cloud

The cloud infrastructure is a composition of two or more clouds (private, community, or public) that remain unique entities but are bound together by standardized or proprietary technology that enables data and application portability (e.g., cloud bursting for load-balancing between clouds) [7].

Private cloud

Private clouds are built exclusively for a single enterprise. They aim to address concerns on data security and offer greater control, which is typically lacking in a public cloud. There are two variations to a private cloud:

- On-premise Private Cloud: On-premise private clouds, also known as internal clouds are hosted within one's own data center. This model provides a more standardized process and protection, but is limited in aspects of size and scalability. IT departments would also need to incur the capital and operational costs for the physical resources. This is best suited for applications which require complete control and configurability of the infrastructure and security [30].
- Externally hosted Private Cloud: This type of private cloud is hosted externally with a cloud provider, where the provider facilitates an exclusive cloud environment with full guarantee of privacy. This is best suited for enterprises that don't prefer a public cloud due to sharing of physical resources. [4]

1.2.3 Levels of Cloud Computing

Cloud computing is typically divided into three levels of service offerings: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a service (IaaS). These levels support virtualization and management of differing levels of the solution stack [8].

Software as a Service (SaaS)

A SaaS provider typically hosts and manages a given application in their own data center and makes it available to multiple tenants and users over the Web. Some SaaS providers run on another cloud provider's PaaS or IaaS service offerings. Oracle CRM on Demand, Salesforce.com, and Netsuite are some of the well known SaaS examples [8].

Platform as a Service (PaaS)

Platform as a Service (PaaS) is an application development and deployment platform delivered as a service to developers over the Web. It facilitates development and deployment of applications without the cost and complexity of buying and managing the underlying infrastructure, providing all of the facilities required to support the complete life cycle of building and delivering web applications and services entirely available from the Internet. This platform consists of infrastructure software, and typically includes a database, middleware and development tools [8].

Infrastructure as a Service (IaaS)

Infrastructure as a Service (IaaS) is the delivery of hardware (server, storage and network), and associated software (operating systems virtualization technology, file system), as a service. It is an evolution of traditional hosting that does not require any long term commitment and allows users to provision resources on demand. Unlike PaaS services, the IaaS provider does very little management other than keep the data center operational and users must deploy and manage the software services themselves – just the way they would in their own data center. Amazon Web Services Elastic Compute Cloud (EC2) and Secure Storage Service (S3) are examples of IaaS offerings [8].

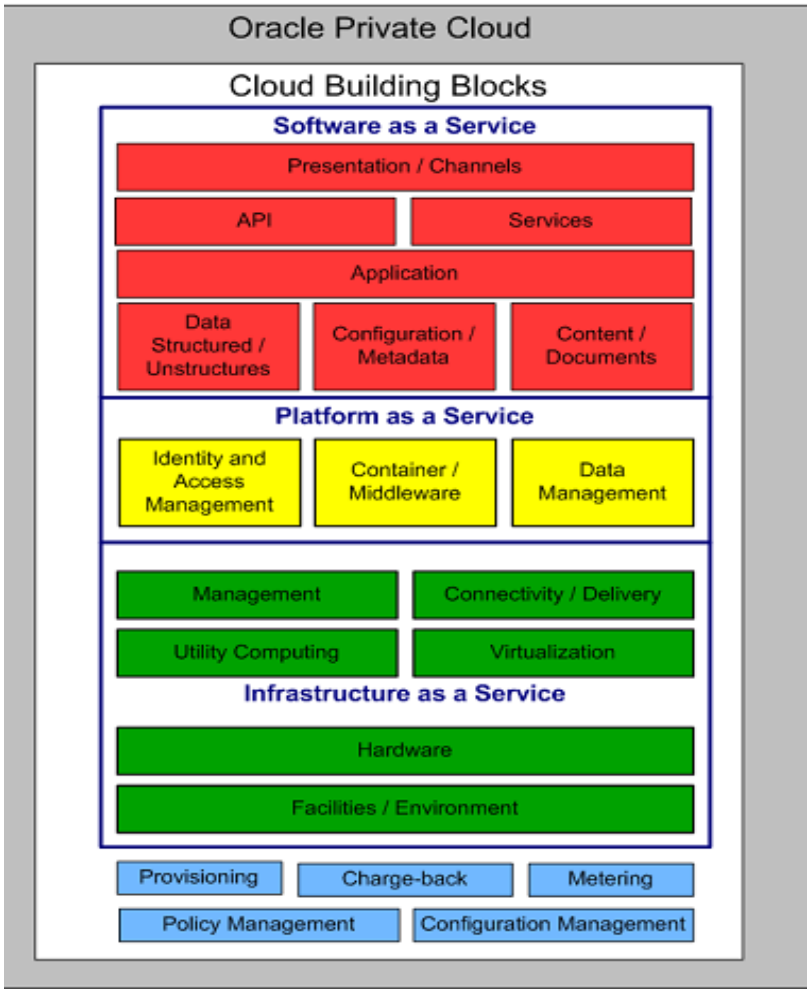

Figure 1.3: Levels of Cloud Computing [8]

1.3 Cloud Security Issues

Many new companies often lack the protection measures to weather off an attack on their servers due to the shortage of resources, poor programming that explores

software vulnerabilities (PHP, JavaScript, etc) open ports to firewalls. For this reason, new companies are encouraged to pursue cloud computing as an alternative to support their own hardware backbone. However, Cloud Computing does not come without its pitfalls. A cloud is a single point of failure for multiple resources. Even though network carriers such as AT&T believe a distributed cloud structure is right implementation, it faces major challenges in finding the optimal approach for low power transmission and high network availability [13]. Some people believe that major corporations will retire away from implementing cloud solutions in near future due to ineffective security measures [31]. One problem comes from the fact that different cloud providers have different ways of storing data, so creating a distributed cloud implies more challenges to be solved between the vendors. Cloud Computing has unique attributes that require risk assessment in areas such as data integrity, recovery, and privacy and an evaluation of legal issues in areas such as e-discovery, regulatory compliance, and auditing [10].

Amazon's EC2 service and Google App Engine are examples of cloud computing, which define a type of computing in which massively scalable IT enabled capabilities are delivered as a service to external customers using internet technologies.

Following are some cloud security issues that need to be addressed:

- **Privileged user access**

  Sensitive data processed outside the enterprise brings with it an inherent level of risk, because outsourced services bypass the physical, logical and personnel controls IT shops exert over in-house programs [5].

- **Regulatory compliance**

  Customers are ultimately responsible for the security and integrity of their own data, even when it is held by a service provider. Traditional service providers are subjected to external audits and security certifications. Cloud computing providers who refuse to undergo this scrutiny are signalling that customers can only use them for the most trivial functions [32].

- **Data location**

  When users use the cloud, they probably won't know exactly where their data will be hosted. In fact, they might not even know what country it will be stored in.

  Service providers need to be asked if they will commit to storing and processing

data in specific jurisdictions, and whether they will make a contractual commitment to obey local privacy requirements on behalf of their customers [11].

- **Data security**

  Security refers to confidentiality, integrity and availability, which pose a major issue for cloud vendors. Confidentiality refers to who stores the encryption keys data from company A, stored in an encrypted format at company B must be kept secure from employees of B, thus the client company should own the encryption keys. Integrity refers that no common policies exist for approved data exchanges [35]. One way to maintain data security on the client side is the use of thin clients that run with resources as possible and do not store any user data, so passwords cannot be stolen. The concept seems to be impervious to attacks based on capturing this data. However, companies have implemented systems with unpublished API's claiming that improves security [15].

- **Identity Management**

  Identity Management in cloud has to manage provisioning/deprovisioning, policies, delegation, password maintenance task, etc as cloud deployments are dynamic with servers launched or terminated and services started or decommissioned. So, as in traditional Identity Management, merely managing users and services is not sufficient in Cloud environment [5].

- **Data recovery**

  Even if we don't know where your data is, a cloud provider should tell us what will happen to our data and service in case of a disaster. Any offering that does not replicate the data and application infrastructure across multiple sites is vulnerable to a total failure [32][34].

- **Investigative support**

  Investigating inappropriate or illegal activity may be impossible in cloud computing. Cloud services are especially difficult to investigate, because logging and data for multiple customers may be co-located and may also be spread across an ever-changing set of hosts and data centers. If user cannot get a contractual commitment to support specific forms of investigation, along with evidence that the vendor has already successfully supported such activities, then our safe assumption is that investigation and discovery requests will be impossible [33].

1.4 Organisation of Thesis

The chapters in thesis are organised as follows:

Chapter 2 This chapter describes in detail the literature survey about Cloud identity management dimensions.

Chapter 3 This chapter describes gap analysis and need for dynamic trust policies.

Chapter 4 This chapter describes in detail the solution of problem, UML diagrams, risk management, notion of trust management, formulation of trust polices and a decision table representing the same. Alternative tools for setting Cloud environment have also been discussed in this chapter.

Chapter 5 This chapter focuses on the implementation of proposed trust policies using Google App Engine and Python.

Chapter 6 This chapter discusses conclusion and future scope of the work done.

Chapter 2
Literature Review

Previous chapter gave a brief introduction about Cloud Computing, its categories and different levels and various Cloud security issues. This chapter discusses in detail about identity management in Cloud and various identity federation frameworks available.

2.1 Cloud Identity Management

Managing identities and access control for enterprise applications remains one of the greatest challenges facing IT today. While an enterprise may be able to leverage several Cloud Computing services without a good identity and access management strategy, in the long run extending an organization's identity services into the cloud is a necessary precursor towards strategic use of on-demand computing services. Supporting today's aggressive adoption of an admittedly immature cloud ecosystem requires an honest assessment of an organization's readiness to conduct cloud-based Identity and Access Management (IAM), as well as understanding the capabilities of that organization's Cloud Computing providers [16].

Major IAM functions that are essential for successful and effective management of identities in the cloud are [16]:

• Identity provisioning/deprovisioning- secure and timely management of on-boarding (provisioning) and off-boarding (deprovisioning) of users in the cloud.

• Authentication & federation - When organizations utilize cloud services, authenticating users in a trustworthy and manageable manner is a vital requirement. Federated Identity Management plays a vital role in enabling organizations to authenticate their users of cloud services using the organization's chosen identity provider (IdP).

• Authorization & user profile management -The requirements for user profiles and access control policy vary, depending on whether the user is acting on their own behalf (such as a consumer) or as a member of an organization.

• Support for compliance- For customers who rely on cloud services, it is important to understand how Identity Management can enable compliance with internal or regulatory requirements.

2.1.1 Identity Lifecycle Management

Lifecycle management incorporates an integrated and comprehensive solution for managing the entire lifecycle of user identities and their associated credentials and entitlements. Functionally, it is divided into two components — the provisioning component and the administrative component. Administrative component defines delegations rules, providing self-service components to change personal details or make requests to the users. Delegation of administrative rights to local group or process-in-charge is crucial for a volatile and dynamic cloud based scenarios. Decentralizing the tasks will reduce the load on the authenticator component and also save time in making access control decisions [3]. Figure 2.1 illustrates the various components of lifecycle management. In cloud, provisioning means just-in-time or on-demand provisioning and de provisioning stands for real time de-provisioning. Just-in time provisioning indicates the federation of user accounts without sharing prior data, based on some trust model.

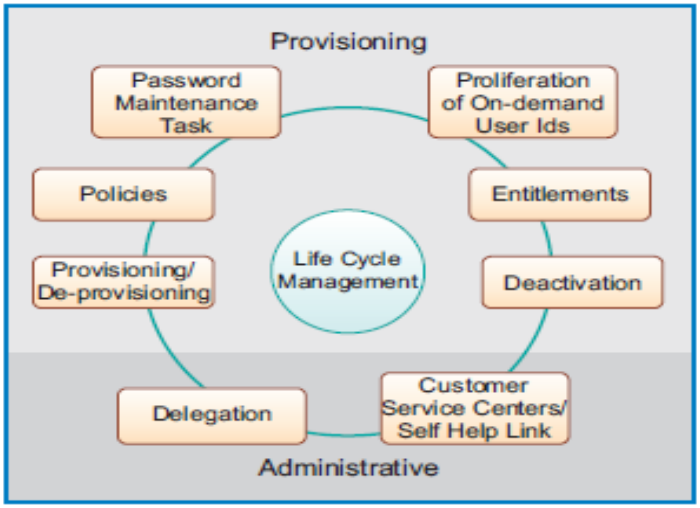

Figure 2.1: The Identity Lifecycle Management [3]

On the internet, it is likely that each user ends up with multiple credentials and multiple access permissions across different applications provided by different service providers. These fragmented logins present a challenge to the users and service providers, in forms of synchronization of shared identities, security, etc. There is a strong need for an intrinsic identity system that is trusted across the web and within enterprises and unambiguously identifying users. Another issue is the trust relation setup between the service providers of the federated world. Currently it is based on policy files framed by the local authority, depending on various factors like the domain trust information automatically fed in by the trust authorities. This is not a

scalable or flexible model that can meet cloud computing demands. Cloud scenarios require dynamic trust propagation and dynamic authorization [3].

2.2 Current Identity Frameworks

Identity federation can be achieved by means of various technologies and open source technologies. Few of them are Security Assertion Markup Language (SAML), Liberty Alliance Initiative, OpenID and WS-Federation.

2.2.1 Security Assertion Markup Language (SAML)

The Organization for the Advancement of Structured Information Standards (OASIS) developed SAML as an XML-based specification for exchanging security information. Currently at Version 2, SAML defines syntax and exchange mechanisms for three kinds of assertions:

1. Authentication assertions, which are declarations about a user's identity
2. Attribute assertions, which contain particular details about a user
3. Authorization decision assertions, which specify what the user is allowed to do on a particular site

SAML is a flexible and extensible protocol designed to be used—and customized if necessary—by other standards [6].

SAML defines a trust mechanism for Single Sign On in which interaction takes place only if there is predefined existing relationship between the relying party and asserting party. The trust mechanism in SAML is based on Public Key Infrastructure. PKI based approach provides secure and trusted lifecycle management of certificates. In this named entity is binded to public key by a certificate. In this approach long lists of providers have to be maintained which makes it difficult to deploy.

2.2.2 OpenID

OpenID is another way to achieve identity federation. It is centric around user, open and decentralized framework. It makes single sign on very easy to be achieved as user can have multiple logins and there is no requirement of predefined trust. It is mainly authentication protocol mainly achieved through attribute exchange.

However, a new OpenID specification called PAPE (Provider Authentication Policy Extension) has been recently approved in order to enforce trust mechanisms. This extension provides means for a RP to request previously agreed upon authentication policies being applied by the OpenID Provider and for an OpenID Provider to inform an RP what policies were used. With PAPE, OpenID moves from a *trust-all-comers*

philosophy to a situation in which the decision to trust can be based in the knowledge of the employed authentication mechanism. In other words, there is no trust model specified by OpenID, RPs must decide for themselves which providers are trustworthy, being their possibility to implement any policies related to the OpenID Provider's response. For these reasons OpenID is simple, lighter and more scalable [17].

2.2.3 Liberty Alliance Initiative

It is closely related to SAML. It was formed with the aim to establish open standards to easily conduct online transactions while protecting the privacy and security of identity information. The Liberty specifications, built on top of SAML, enable identity federation and management through features such as identity/account linkage, single sign on, and simple session management [17]. Liberty Alliance Initiative takes into account the concept of Cot (Circle of Trust). In this there must be trust and business agreements that parties must comply to for any future interactions. There are different kinds of relationships that can exist between two parties for authentication and business relationships in CoTs specified by LA. If the context is business relationship then the two types of relationships can exist- pairwise or community trust relationship. Former describes the case where two entities have direct business agreements with each other or it can be brokered trust [18] i.e where two entities do not have direct business agreements with each other, but do have agreements with one or more intermediaries so as to enable a business trust path to be constructed between the entities. The intermediary brokers operate as active entities, and are invoked dynamically via protocol facilities when new paths are to be established. Latter trust is when no kind of relationship exists.

So Liberty entities have a TAL or Trust Anchor List with the trustworthy entities for authentication purposes, and also have a BAL or Business Agreement List, containing those parties which are related to the entity via a business agreement. Authority lists just allow us to take Boolean decisions, which mean that if the list contains an entry for an authority or a trustworthy path to reach it, then the decision will be positive. On the other hand, if the authority is unknown and there is no path to it, the decision will be negative. These mechanisms limit interaction in open environments, where the presence of unknown users is common and there is no previous configuration before interaction [17].

2.2.4 WS-Federation

In WS-Federation, an administrator or other trusted authority may designate that all tokens of a certain type are trusted (e.g. all X.509 tokens from a specific CA). The security token service maintains this as a trust axiom and can communicate this to trust engines to make their own trust decisions. In this IdPs are equivalent to CAs [17].

Table 1.1: Summary of Trust models in Identity Federation [17]

IdM Technology	Trust Model
OpenID	No trust model defined, *trust-all-comers* philosophy, no preconfiguration required.
SAML	PKI recommended. Typically implemented with trust lists.
Liberty Alliance	Trust architecture based on CoTs. Follows SAML recommendation (PKI). Two relationship contexts: authentication, business. Hierarchical, peer-to-peer, mesh and hybrid topologies considered. Typically implemented with trust lists.
WS-Federation	Trust architecture based on WS-Trust. Peer-to-peer, mesh and hybrid topologies considered. Typically implemented with trust lists.

The technologies analysed above handle trust through PKI with trust lists except OpenId which has no trust mechanism and is based on trust all customers' philosophy. It can be noted that none of the above identity management technologies include efficient trust models for dynamic environments, which implies an important challenge. Furthermore, the problem of establishing trust relationships between previously unknown entities willing to interact is not covered by any of the current frameworks or specifications [17].

2.3 Dynamic Trust Establishment

For seamless interaction between user and service provider identity provisioning needs to be dynamic. This will increase user experience and reduce complexity of service providers. In the following it is determined how trust can be incorporated in federated environment in a dynamic way.

2.3.1 Through the use of service specific XACML policies

An algorithm [19] has been designed which computes trust values according to past experiences i.e if there is any direct or indirect relationship between the two parties. These computed values are stored within trust relationship sub objects. The computed

trust level values serve as parameters for ARP's threshold to determine whether the user data should be released or not.

Recursive Trust Algorithm

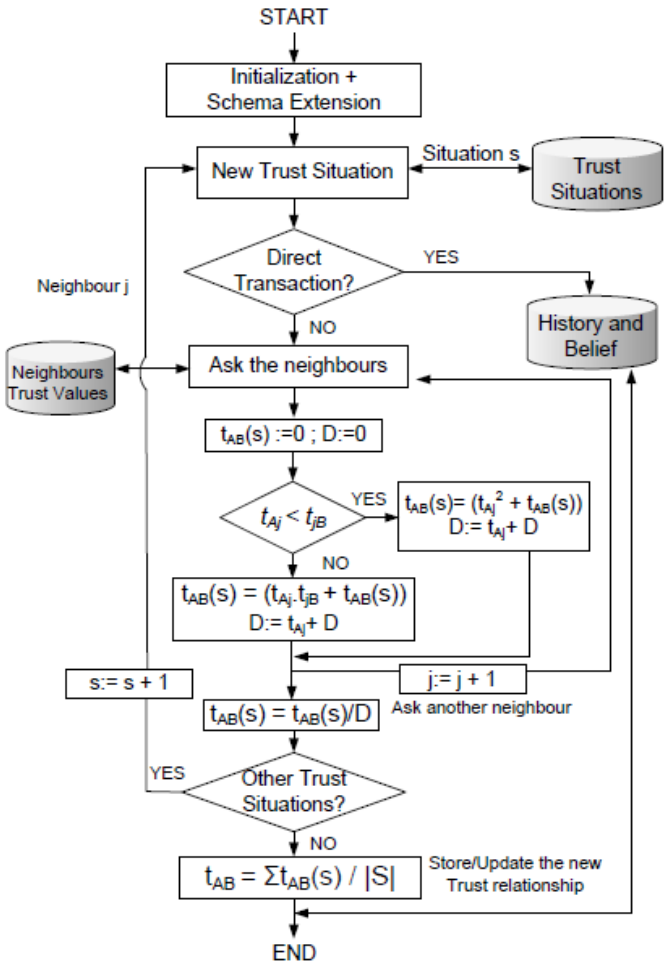

Figure 2.2: Recursive Trust Algorithm [19]

The workflow involves three main steps on deciding upon the request made, and permits the users data release if the requester has been accepted as trustworthy partner:

1. First, a search mechanism is specified in the form of a search filter based on the trust attributes, in addition to the name and the Identifier of the Service Provider who attempts to access the user's data by his IdP. The search filter strives to collect machine readable encoded information about past interactions with the new Service Provider and make it available to the trust algorithm.

2. Secondly, the recursive trust algorithm is applied to make estimation about the level of trust. The algorithm performs incremental alignment in real time of two

series, as one is performed on the set of the neighbours' recommendations and the second is performed on the set of the trust situations.

3. Finally, the computed trust level value will serve as a parameter for the XACML based ARPs reception thresholds in the policy repository in order to assess the limits imposed by the IdP for releasing the user's data, for example, if the trust level for the requesting service is <= 6 [19].

Schema Definition

Unlike many traditional directory conventions which aim at storing user objects as leaf objects, the user objects are stored into container objects so that further objects representing the trust relationships of the user to the corresponding Service Provider, can be stored beneath them, usually beside other sub-objects representing for instance the user addresses. This design shows a lot of flexibility regarding storing multiple objects of the same type without having to extend the schema and define new attributes for each new created object separately [19].

2.3.2 SAML extension for Dynamic Federation

Security Assertion Markup Language (SAML) is XML-based specification for exchanging security information.It is the most flexible to add extensions in order to achieve dynamic federation in a generic way. In addition, SAML is the mostly deployed federation solution and has been adopted by many well known providers (e.g. Google Apps). Furthermore, while all the solutions are mainly concerned with web scenarios and the SSO use case, SAML offers abstraction enough to be applied to a wider range of situations [17]. So by including modifications in the abstract level we can assure its later application in more specific use cases. Also, SAML is the only standard nowadays and LA is based in its specifications, so it is more logical to introduce modifications in SAML that could be later adopted by other technologies based on it. A new component the Trust Engine is added which will be in charge of processing every trust-related data to decide if an entity is trustworthy or not. The trust information can be obtained from external or internal sources. The Trust Engine will allow entities to maintain a dynamic store instead of a static list with trust data, that we call Dynamic Trust List or DTL [17].

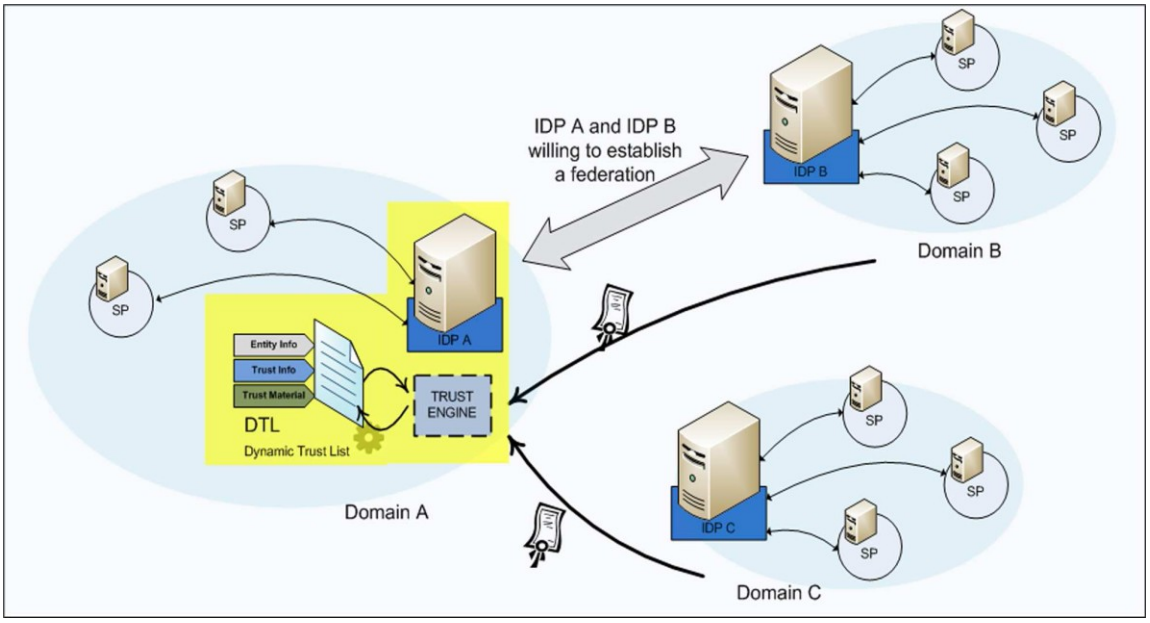

Figure 2.3: SAML Extension for Dynamic Trust Establishment [17]

Trust Engine

In order to include dynamic features in the process of trust establishment, SAML entities must be extended with a Trust Engine. This component is the responsible of processing external and internal trust information and performing DTL updating. Also, decisions to trust will be made by this logical block. The internal trust information can be obtained from the DTL. Furthermore, other data as internal policies could be useful when applying custom trust levels. e.g. transient or attribute federation may have less requirements than permanent federation as it implies less personal user identity information disclosure. On the other hand, external trust information can be obtained from other entities [17].

Dynamic Trust List

In SAML implementations every entity is usually configured with a static TAL before any interaction between parties takes place. This list contains the digital certificates associated to every other entity which is considered trustworthy. Protocol messages whose digital signature trust does not evolve over time, because interaction experiences are not taken into account, community knowledge is not exploited, distrust and ignorance are treated in the same way, and the automatic establishment of trust relationships between unknown entities is impossible. The preconfigured TAL model poses important obvious limitations in dynamic open environments. Instead of a static TAL, the system maintains an enhanced Dynamic Trust List with more complete information: entity data and its associated trust information (e.g. reputation

scores, trust level, previous interaction results, etc.) and trust material (e.g. keys, credentials, etc.). The list will be dynamically updated by the Trust Engine under specific events such as receiving recommendations from other entities or when a successful interaction ends. In order to allow the exchange of trust related data, it is required to define new protocols and messages that extend the SAML specifications [17].

Reputation Exchange over SAML

Adding reputation support to SAML implies modifications to both Assertions and Protocols. As it was explained before, specifications contemplate three different kinds of Assertions: Authentication, Attribute and Authorization. For the reputation exchange to be possible, we have defined a new one, called Reputation Assertion. It has been defined according to the extensions mechanisms explained in the SAML core specification, so compatibility is assured. Basically, the defined Assertion contains a new custom <Statement> type, called <ReputationStatement>. On the other hand, the reputation exchange protocol is based on the standard SAML Authentication protocol, so query/response formats are compliant with the rules defined for extending the schema. Thus, the communication flow has two steps:

1) the entity in the requester role sends a <ReputationRequest> and 2) the entity in the responder role returns a <Response> containing a Reputation Assertion in case of success, or an error message in case of failure [17].

```
<Assertion ID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
   IssueInstant="2009-09-09T00:46:02Z" Version="2.0"
   xmlns="urn:oasis:names:tc:SAML:2.0:assertion">
   <Issuer>
      Entity1.com
   </Issuer>
   <Subject>
      <NameID
        Format=
        "urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">
        IdP.domain2.com
      </NameID>
   </Subject>
   <ReputationStatement>
      <Score>
        <ScoreValue>
           6
         </ScoreValue>
      </Score>
      <RepContext>
           "authentication"
      </RepContext>
   </ReputationStatement>
</Assertion>
```

Figure 2.4: Reputation Assertion [17]

Chapter 3
Problem Statement

Previous chapter analyzed various Identity Federation frameworks available. This chapter focuses on gaps in Literature that has been reviewed.

3.1 Gap Analysis

Table 3.1 shows the comparison of the Identity Federation frameworks discussed in previous chapter.

Table 3.1: Comparison of Current Identity Frameworks

Identity Framework	Advantages	Limitations	Common Limitation
SAML	It is based on open industry standards, has extensible schema, it is simple and ensures integrity and privacy.	It uses trust model which is hard to deploy and maintain.	None of them include efficient trust models for dynamic trust establishment.
OpenID	It is open, flexible and free framework. It helps in single sign on.	It is too much flexible and does not have any trust model and uses symmetric encryption which is not much secure.	
Liberty Alliance Initiative	Supports single sign on and has strong authentication capability.	Trust lists are maintained manually.	
WS-Federation	It is based on established standards, is extensible and supports interoperability	It is complex and has constraints in deployment.	

Models discussed in Literature Survey help in seamless interaction of two entities in a federated environment by dynamic trust establishment. But they too have limitations. They are also compared in the Table 3.2.

Table 3.2: Comparison of Current Dynamic Trust Establishment Approaches

Dynamic Trust Establishment Approach	Common Limitations
Through the use of service specific XACML policies	• They do not know what trust value should be assigned to a completely unknown entity. • Also more dimensions of trust, such as risk management, are not considered in the present approaches.
SAML extension for Dynamic Federation	

Therefore a new approach is required overcome these limitations and will be more efficient in federated identity management scenarios.

3.2 Need for Dynamic Trust Policies

Cloud Identity Management unlike traditional identity management requires just in time provisioning which means sharing data based on some trust model and currently it is based on policy files framed by some local authority like SLA. But for dynamic environments like Cloud static agreements are not enough to ensure security. There is a strong need for dynamic trust establishment among the different service providers of the federated world. A few frameworks have been defined by Cummings and McKnight [6][7] that provide a classification system for different aspects of trust. On the basis of this work different facets of trust can be defined showing how trust can influence behaviour.

Trust Policies can help service provider to take better decision whether to trust the user or not. Ineffective trust policies for provisioning could fuel security and legal threats and can have negative effect on the reputation of the service provider. So trust policies should be carefully designed and all the possible risks should be taken into account while doing so.

This thesis aims to design, develop and implement Trust Policies for cloud environment and verify and validate them on Google App Engine using python.

Chapter 4

Design of Trust Policies

This chapter discusses about how Trust Policies can be designed with the help of UML diagrams. Also decision table is used represent designed Trust Policies.

4.1 Design of Solution

Following section shows modelling of Trust Policies through Unified Modelling Language.

4.1.1 UML Diagrams

Use case diagram shows the functionality provided by a system in terms of user, administrator and its peers. It is used to gather requirements of the system including internal and external influences. In this work, focus is on trust policies and specifically on user provisioning. Therefore, following UML diagrams have been drawn.

- Use case diagram: Use case diagram representing login for free trial has been shown in Figure 4.1

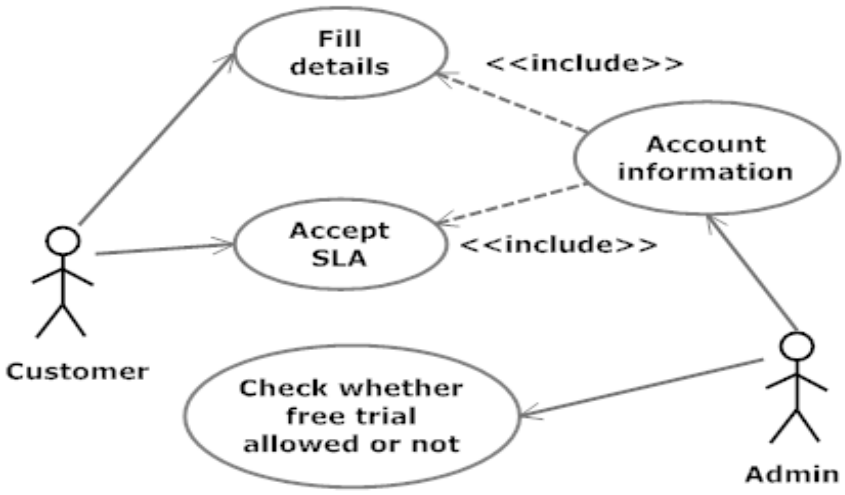

Figure 4.1: Use case diagram showing free trial login

- Use case diagram: Use case diagram representing login after registration has been shown in Figure 4.2.

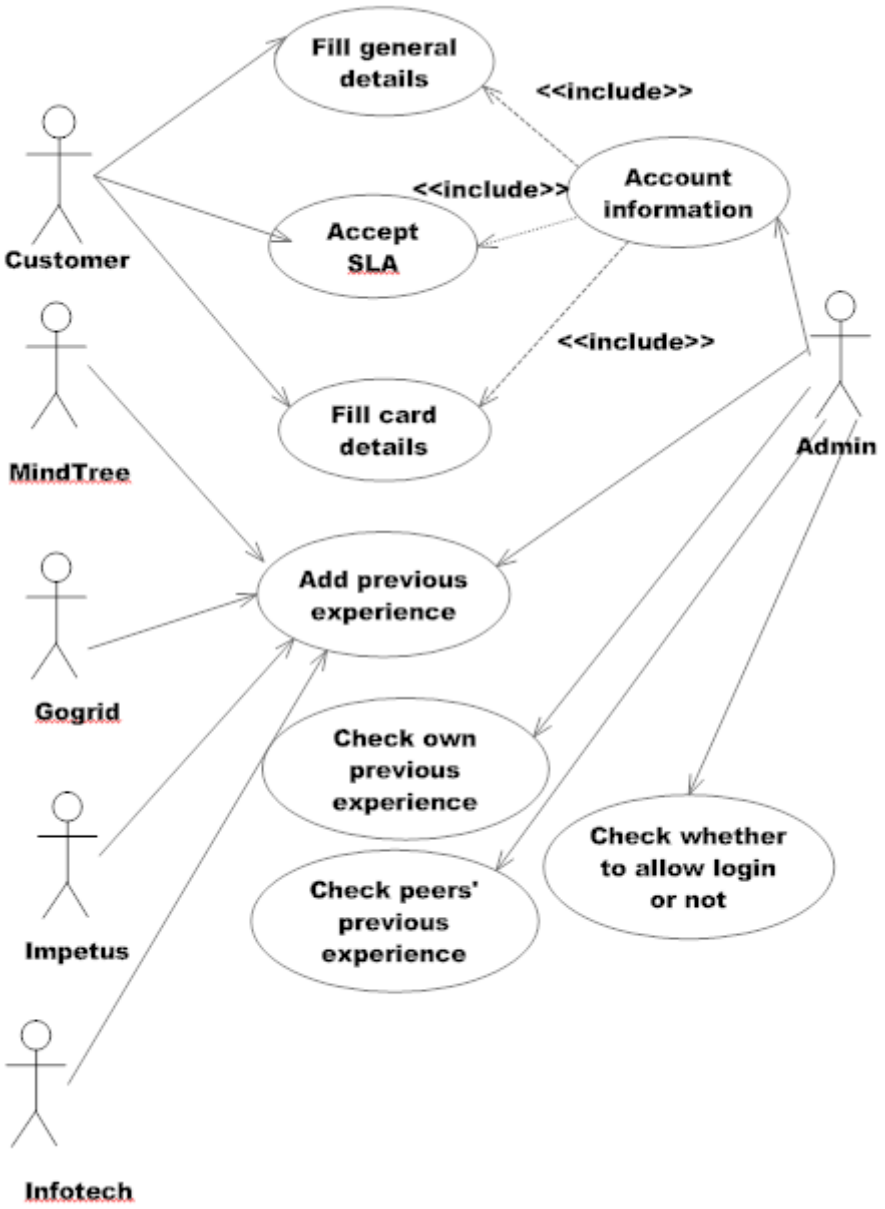

Figure 4.2: Use case diagram showing login after registration

- Sequence Diagram: Sequence diagram representing login after registration has been shown in Figure 4.3.

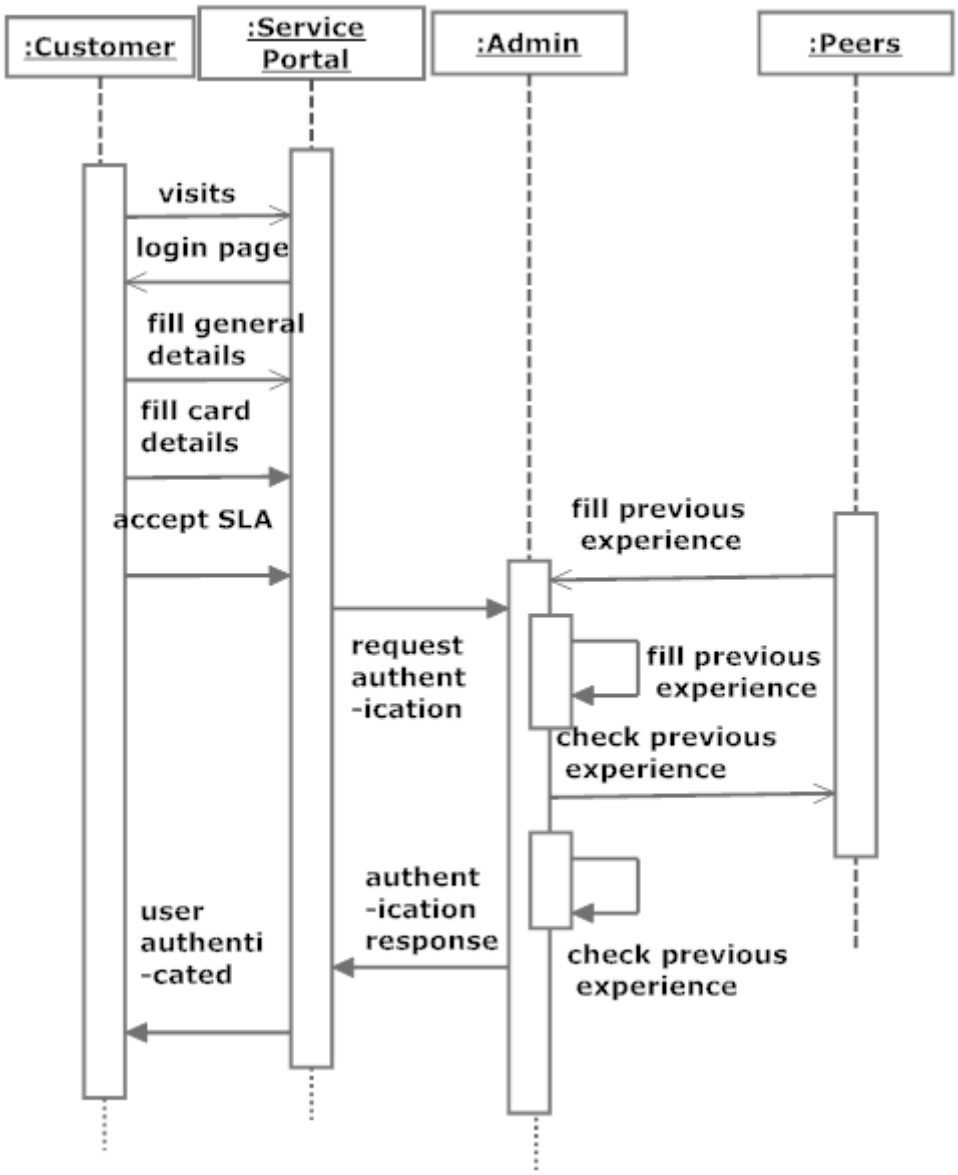

Figure 4.3: Sequence diagram showing login after registration

- Sequence Diagram: Sequence diagram representing login for free trial has been shown in Figure 4.4.

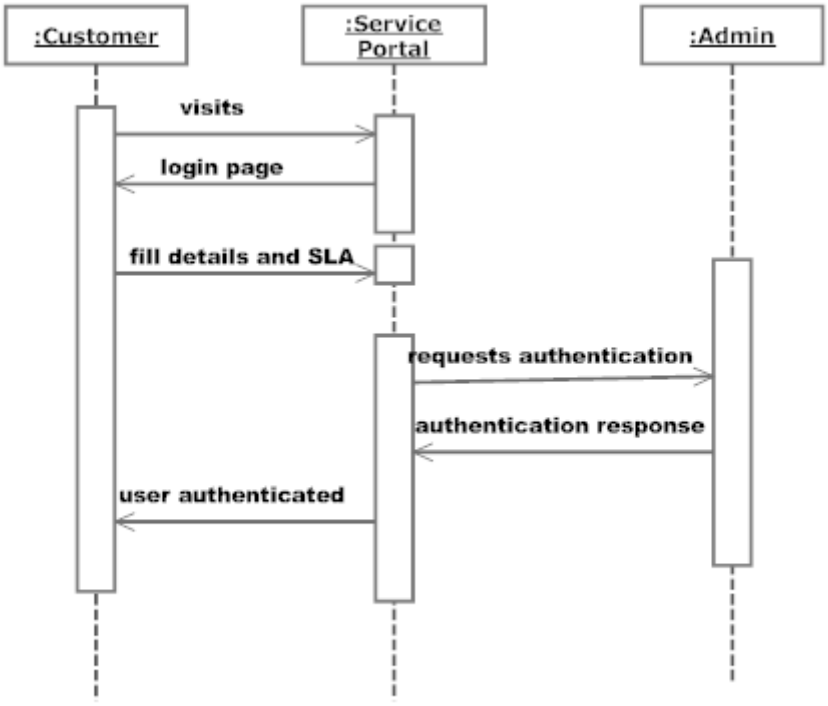

Figure 4.4: Sequence diagram showing login for free trial

- Activity Diagram: Activity diagram representing login after registration has been shown in Figure 4.5.

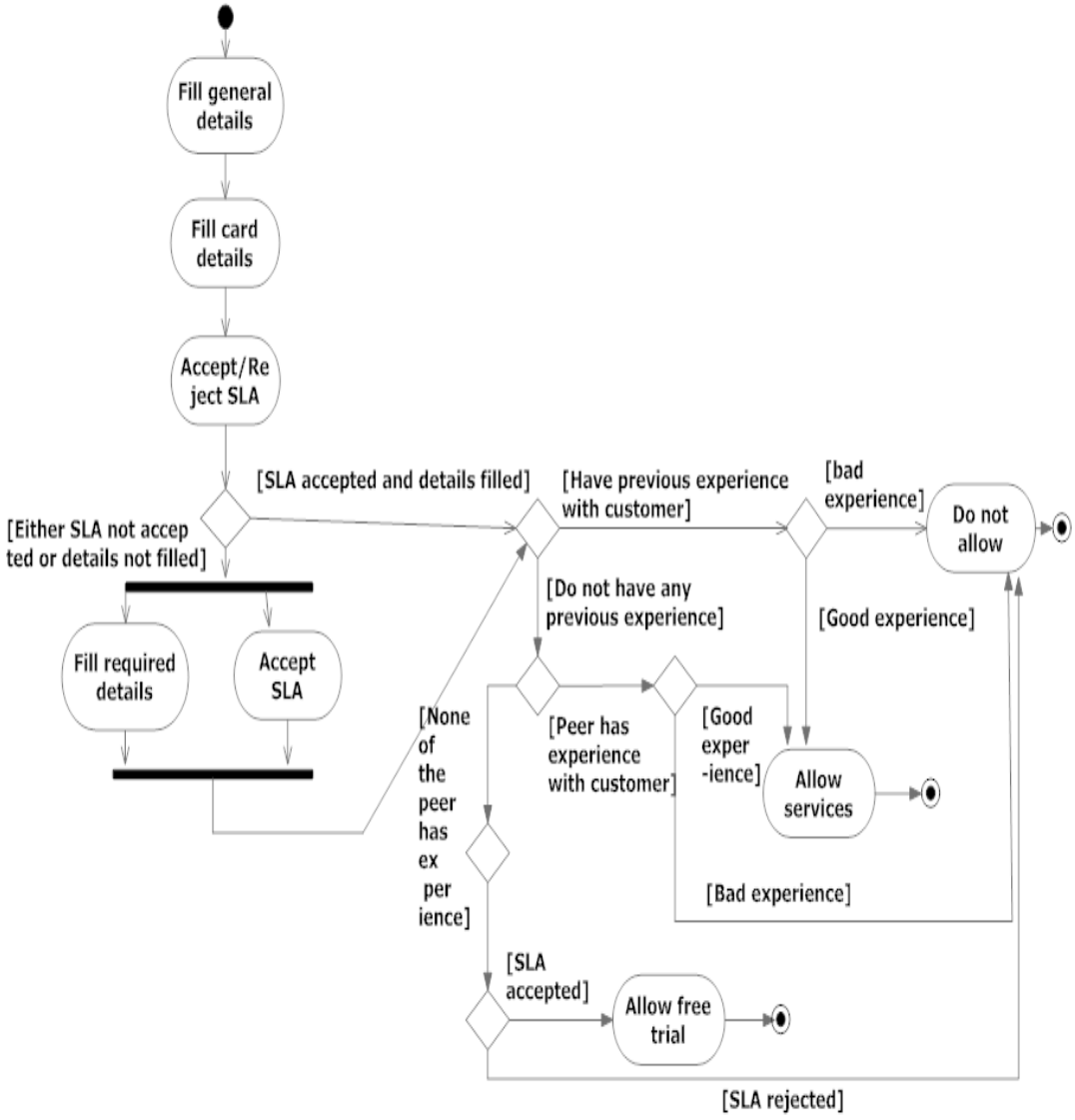

Figure 4.5 Activity diagram showing login after registration

- Activity Diagram: Activity diagram representing login for free trial has been shown in Figure 4.6.

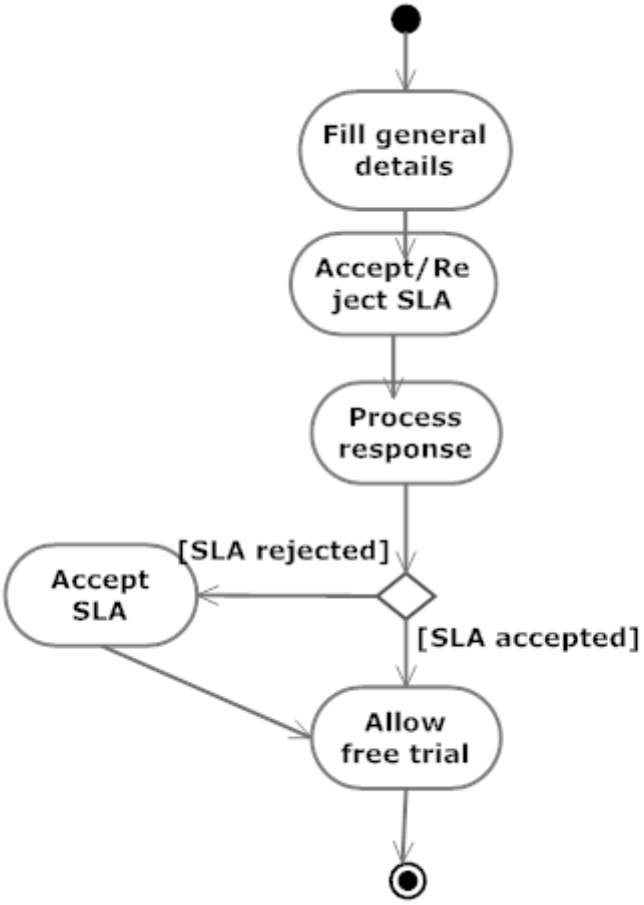

Figure 4.6: Activity diagram showing login for free trial

- Deployment Diagram: Deployment diagram depicting hardware components where software components are deployed is shown in Figure 4.7.

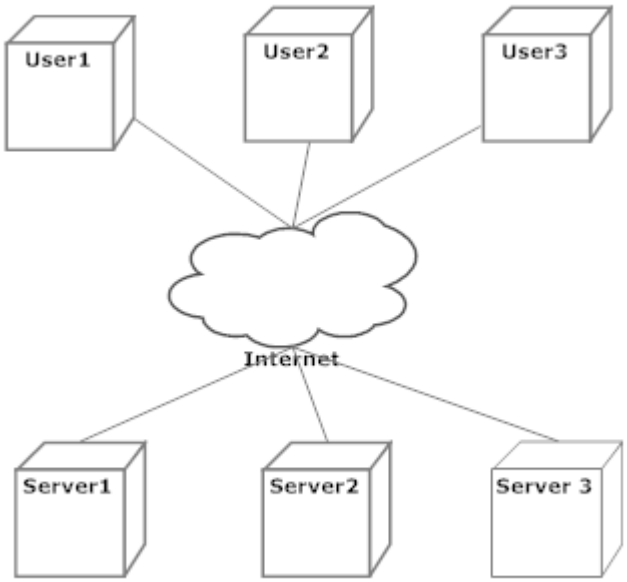

Figure 4.7: Deployment diagram showing hardware components where software components are deployed

4.2 Risk Management

Risk management is methodology of identifying, controlling and minimizing the effects of unpredictable events [20].

Risk management method includes:

- identifying assets to be protected;
- finding vulnerabilities that can be exploited;
- determining likelihood of their occurrences;
- determining their impact.

Risk is treated using countermeasures which seek to reduce either the likelihood or impact of the risk. Implementing the countermeasure has a cost associated with it, which must be balanced with the expected utility of implementing the measure. Integrating risk management into trust management process is therefore useful, as it will enable us to leverage off existing body of work. Figure 4.8 shows different relationships in risk management.

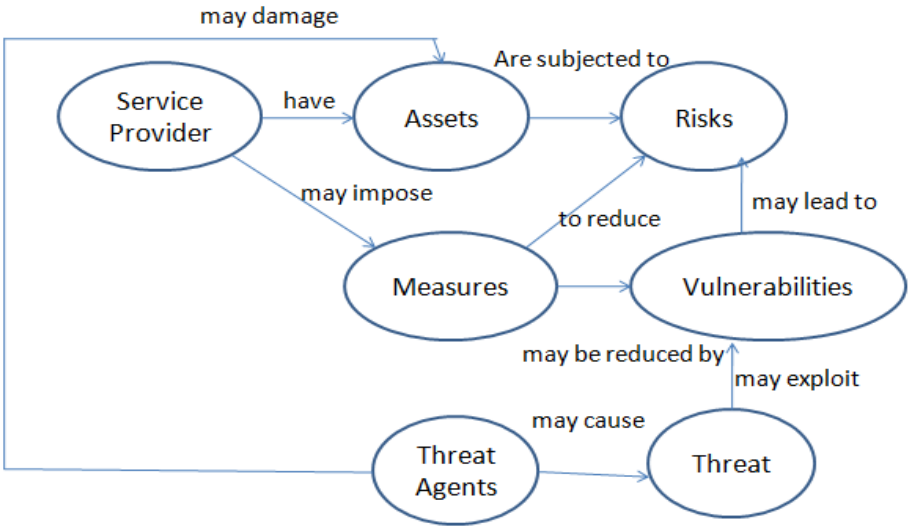

Figure 4.8: Relationships in Risk Management

4.3 Trust Management

For secure identity provisioning and smooth communication both service provider and the user need to trust each other and if service provider has strong trust policies security will be more and thus he will be more trusted by the users. Following section presents design of trust policies which will help in secure identity provisioning.

4.3.1 Notion of Trust

Trust is a key concept for providing seamless interaction between different trust domains and improved user experience. Figure 4.9 shows conceptual view of trust.

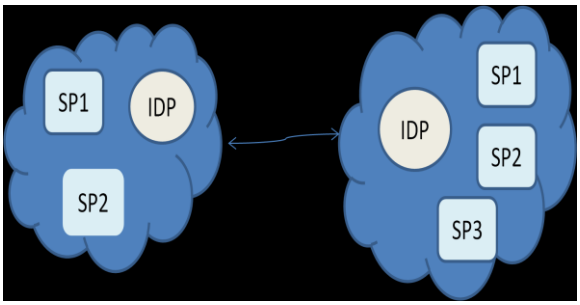

Figure 4.9: Conceptual view of Trust

- **Definition of Trust**

Trust is a directional relationship between two parties that can be called trustor and trustee. The trustor must be a “thinking entity” in some form, meaning that it has the ability to make assessments and decisions based on received information and past experience. The trustee can be anything from a person, organisation or physical entity, to abstract notions such as information or a cryptographic key [21].

- **Facets of Trust**

To integrate trust based identity provisioning with risk management a framework needs to be defined. Different facets of trust have been identified to define framework for risk along with trust based identity provisioning:

  - Trust conduct: It is the degree to which the trustor voluntarily trusts the trustee in a given context. It depends on reasonable confidence of the trustor that the trustee will behave in a beneficial way and also there is a risk to the trustor if the trustee will not behave in that way.
  - Trust point: It is the degree to which the trustor is willing to trust the trustee in a given context. Trust point directly relates to trust polices which determine how trustee are trusted in the system. A Trust point usually leads to trust conduct. It determines the willingness of the trustor to the trustee after it has analyzed benefits and risks arising of this trust.
  - Trust assumption: It is the degree to which the trustor feels confident about trusting the trustee that he will behave in a beneficial way towards him. It is mainly based on trustor's subjective confidence about the trustee.

Trust assumption plays a major role in determining the information by which a trustor can make decision whether to trust or not. Various metrics like brand name, recommendation etc are available to measure trust and make decisions accordingly.

Various factors affect assumption formation process (The process by which confidence of the trustor is increased in the trustee) which includes:

  - Static trust: This is simple trust based on agreements signed between two principals.
  - Dynamic trust: Dynamic trust is calculated by principal's own experience or peers' experience with the trustee.
  - Context: Different situations affect the trust making decision because each situation has different assumptions and risks associated with it.

These concepts do not exist in isolation. They are clearly related to each other. Trust conduct relies on trust point which is derived from trust assumption, static trust or dynamic trust. Figure 4.10 shows these concepts and their relationships.

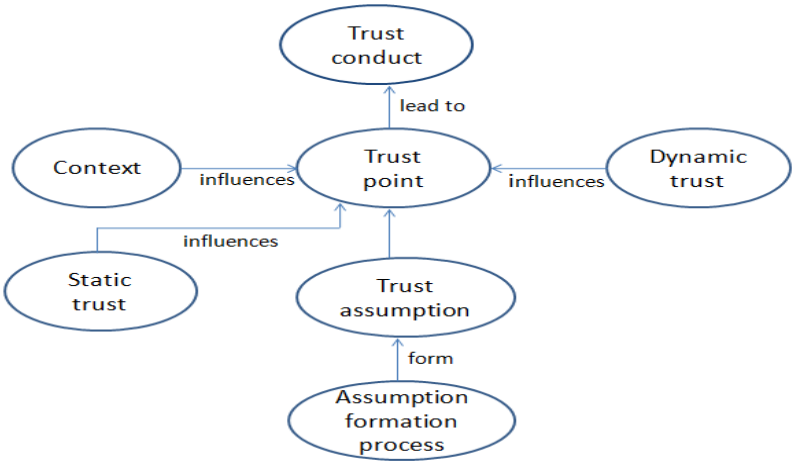

Figure 4.10: Trust concepts and their relationships

4.3.2 Incorporating risk factor in Trust management

Trust management is a process of identifying relationships between different trust concepts and analyzing them to identify different types of risks they are exposed to while trusting the trustees. Risk management is process of identifying the risks, likelihood of their occurrences and determining their impact. Therefore both are closely related to each other. Trust management can be related to risk acceptance decision [1] i.e. trustor is prepared to accept risks if he trusts the trustee that can expose them. So clearly more the trustor trusts the trustee more risk he is exposing himself to.

By combining risk management with the concepts defined in section 4.3.2 form the basis to define trust and develop a trust policy.

The trust point forms the basis of trust policy, which is essentially a statement of the conditions under which a trustor is prepared to trust to trustee. These trust points are formed from number of sources-agreement signed between two principals, confidence in the trustee, trustor's own or its peers experience with trustee. To develop trust policy different risks that are associated with these sources should be considered. Figure 4.11 shows how risk factor can be included in Trust Management.

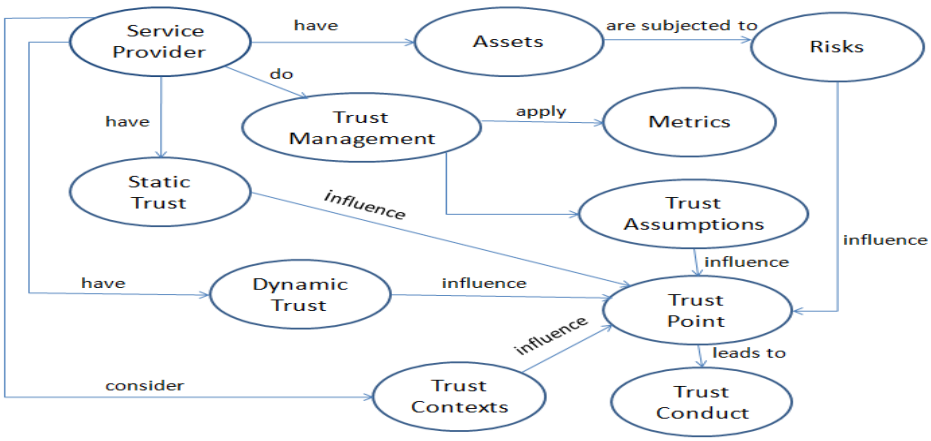

Figure 4.11: Including risk factor in Trust Management

4.4 Formulating Trust Policies

Trust Policies refer to rules and statements to ascertain if unknown entities are to be trusted for a given context and to what extent (trust level). The policy should include [1]:

- Trust metrics: A list of the metrics used in a trust policy, the trust assumptions they measure and their appropriateness for given trust contexts.
- Confidence levels: A description of our level of confidence in a given trust assumption. Trust levels may be expressed in range of [0,100] where high trust is represented in [90,100], low trust in [5, 20], default initial trust in [0, 50]. Negative values represent distrust [23].
- Trust contexts: These are all the situations and environments that are under consideration for the trust policy.

Trust Management Process for Cloud Scenarios

As cloud provides limitless resources to the users on consumption basis through the internet the trend of buying resources is losing attraction but in a federated environment like cloud seamless interaction between two domains trust is required and to establish trust dynamic trust policies are required so that a service provider can decide whom to trust and whom to not. Following is the process for setting a trust policy for cloud environment.

A. Cloud Trust Management Context

There may be different situations that a service provider should consider while trusting the other principals because each trust context has different types of risks

associated with it. There are number of trust points which a service provider must have before he takes a decision to trust or not:

- Trustor may have previous experience with the trustee.
- Trustor's peers may have some previous experience with the trustee.
- Trustee may be totally unknown.

In addition, service provider must also consider threats from the following sources:

- Hackers/Crackers, who wish to steal sensitive information or credentials or can have unauthorized system access;
- Computer Criminals who may do fraudulent acts, information bribery or system intrusion;
- Terrorists who may cause distributed denial of service (DDOS), system penetration or system tampering.

B. Expected benefits:

By leveraging resources on consumption basis the service provider may aim to benefit to a large extent.

C. Discover Assets to be protected:

In this scenario main assets that are under threat are:

- User's basic information;
- User's sensitive credentials;
- User's sensitive data;
- Over usage of resources

D. Vulnerability Analysis:

By analyzing assets to be protected and different threats to them service provider determines the set of vulnerabilities which may lead to these threats:

- User might try to hamper other user's data;
- User might not pay the bill;
- Terminated users' identifiers are not removed from the system.

E. Risk Analysis:

For the above vulnerabilities identified service providers determine likelihood levels of these vulnerabilities and their impact. Likelihood level is measured as High, Medium and Low depending on the motivation and capability of the threat agent and level controls to prevent the vulnerabilities to be exercised. Similarly magnitude of

impact is measured as High, Medium and Low depending on cost incurred due to loss and extent to which organisation's reputation is harmed. Table 4.1 shows this analysis.

Table 4.1: Risk Analysis Summary

Assets to be protected	**Threat agents**	**Impact**	**Likelihood**
Users' basic information	Hacker/Cracker	Low	Low
Users' sensitive credentials	Hacker/Cracker, Computer Criminals, Terrorists	High	High
Users' sensitive data	Hacker/Cracker, Computer Criminals, Terrorists	High	High
Over usage of resources	Customer	Medium	Medium

F. Identify Required Assumptions and Confidences:

Service provider now needs to determine the level of confidence in the assumptions. Following are the decisions for the identified vulnerabilities:

- User might try to hamper other user's data:

  Service provider has to trust the user about his intention with [90,100] level of confidence. In order to trust the user, service provider has to know that the user is competent, honest and predictable and confidence in these assumptions should also be of [90,100] level.

- User might not pay the bill:

  According to the level of risk identified service provider decides to have only [50, 60] confidence on the user and allow only a few days trial and require the user to submit his account details.

G. Identify and evaluate metrics

Metrics are ways to measure the extent to which a service provider can trust the user. Different metrics can be used:

- Previous experience with the user ([60,90] level of confidence);
- Previous experience of a peer with the user ([50,70] level of confidence);
- Delegation ([60, 70] level of confidence);
- Abidance by the Service Level Agreement ([90,100] level of confidence);
- Certifications ([50,70] level of confidence)

4.5 Decision Table representing Trust Policies

Table 4.2 shows decision table specifying trust policies for the cloud environment. Decision management process and actions determine whether to allow user to access resources or not. For example let there be a case in which a user wants to access services but service provider's peer's previous experience with that user was not good then according to the Trust Policies developed user will not be permitted to access the services. Similarly if the user does not agrees to service provider's SLA he will not be allowed to become member hence he too will not be eligible to access the services. Hence this shows these Trust Policies developed can be quite helpful in deciding more accurately whom to trust.

Table 4.2: Decision Table representing Trust Policies

		Rules								
Conditions	Is user admin?	Y	-	-	-	-	-	-	-	-
	Is user member?	-	Y	Y	Y	Y	Y	N	N	N
	Does service provider have any previous experience with the user?	-	Y	Y	N	N	Y	N	N	N
	Was previous experience good?	-	Y	N	-	-	Y	-	-	-
	Do any of the peers have previous experience with the user?	-	N	N	Y	Y	N	N	N	N
	Was peer's experience with user good?	-	-	-	Y	N	-	-	-	-
	Does user has any certification?	-	Y	Y	Y	Y	N	Y	Y	N
	Does user abides to SLA?	-	Y	Y	Y	Y	Y	Y	N	Y
	Is user totally unknown?	-	N	N	N	N	N	N	Y	Y
Actions	Service provider trusts the user and allows him to use resources	X	X		X		X			
	Service provider trusts the user partially and allows him a free trial							X		X
	Service provider does not trusts the user			X		X			X	

4.6 Tool Alternatives for setting Cloud Environment

Cloud applications have different composition, configuration, and deployment requirements. So different tools are available for setting cloud environment. Following section gives brief description of popular tools commonly utilized for creating Cloud environment.

4.6.1 OpenNebula [24]

OpenNebula is one of the key technologies of reservoir plan and the flagship research project in virtualization infrastructure and cloud computing of European Union. It allows user deploy and manage virtual machines on physical resources and it can set user's data centers or clusters to flexible virtual infrastructure that can automatically adapt to the change of the service load.

OpenNebula is also an open and flexible virtual infrastructure management tool, which can use to synchronize the storage, network and virtual techniques, and let users dynamically deploy services on the distributed infrastructure according to the allocation strategies at data center and remote cloud resources. Through the interior interfaces and OpenNebula data center environment, users can easily deploy any types of clouds. OpenNebula is mainly used to manage the data center of private cloud and infrastructure of cluster and it also support hybrid cloud to connect the local and public infrastructure. This is very useful to build high scalable cloud computing environment. Besides, OpenNebula also supports public cloud platform by providing interfaces and functions to virtual machines, storage and network management and so on. Through the control interfaces, users can access services provided by OpenNebula cloud computing platform.

4.6.2 Hadoop [25]

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using a simple programming model. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

The project includes these subprojects:

- Hadoop Common

The common utilities that support the other Hadoop subprojects. It provides access to the filesystems supported by Hadoop. The Hadoop Common package contains the necessary JAR files and scripts needed to start Hadoop. The package also provides source code, documentation, and a contribution section which includes projects from the Hadoop Community.

- Hadoop Distributed File System (HDFS)

A distributed file system that provides high throughput access to application data. It is a distributed, scalable, and portable filesystem written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single datanode; a cluster of datanodes form the HDFS cluster. The HDFS stores large files across multiple machines. It achieves reliability by replicating the data across multiple hosts, and hence does not require RAID storage on hosts.

- Hadoop MapReduce

A software framework for distributed processing of large data sets on compute clusters. MapReduce is a framework for processing huge datasets on certain kinds of distributable problems using a large number of computers (nodes), collectively referred to as a cluster (if all nodes use the same hardware) or as a grid (if the nodes use different hardware). Computational processing can occur on data stored either in a filesystem (unstructured) or within a database (structured).

  - "Map" step: The master node takes the input, partitions it up into smaller sub-problems, and distributes those to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes that smaller problem, and passes the answer back to its master node.
  - "Reduce" step: The master node then takes the answers to all the sub-problems and combines them in some way to get the output – the answer to the problem it was originally trying to solve.

The advantage of MapReduce is that it allows for distributed processing of the map and reduction operations.

4.6.3 CloudSim [26]

Specifically in the case of Cloud computing, where access to the infrastructure incurs payments in real currency, simulation-based approaches offer significant benefits to Cloud customers by allowing them to: (i) test their services in repeatable and

controllable environment free of cost; and (ii) tune the performance bottlenecks before deploying on real Clouds.

CloudSim is an extensible simulation toolkit that enables modelling and simulation of Cloud computing environments. The CloudSim toolkit supports modelling and creation of one or more virtual machines (VMs) on a simulated node of a Data Center, jobs, and their mapping to suitable VMs. It also allows simulation of multiple Data Centers to enable a study on federation and associated policies for migration of VMs for reliability and automatic scaling of applications.

4.6.4 Google App Engine [27]

Google App Engine is "a system that exposes various pieces of Google's scalable infrastructure so that you can write server-side applications on top of them". Simply this is a platform which allows users to run and host their web applications on Google's infrastructure. These applications are easy to build, easy to maintain and easy to scale whenever traffic and data storage needed. By using Google's App Engine, there are no servers to maintain and no administrators needed. The idea is user just to upload his application and it is ready to serve its own customers. User has a choice either his product to be served by the free domain appspot.com or to allow Google Apps to serve it from domain chosen by the customer. Google also provide the user with the option to limit the access of the application within the members of his own organization or to share it with the rest of the world. The starting packet is free of charge and additional obligation. All the user have to do is to sign up for a free account, and then to develop and publish his own application. The starting package includes up to500MB of storage and enough CPU power and bandwidth to serve 5 million page views per month.

- The Application Environment

Several key features are included in the environment:

- dynamic web serving, with full support for common web technologies
- persistent storage with queries, sorting and transactions
- automatic scaling and load balancing
- APIs for authenticating users and sending email using Google Accounts
- a fully featured local development environment that simulates Google App Engine on user's computer

- The Python Runtime Environment

The Runtime environment provided by App Engine uses Python and Java programming language, but other languages as well are taken into account to be supported in the near future. The current supported version of the Python runtime environment is 2.5 and 2.7, which includes the standard Python library. All methods, except those ones which violates the sandbox restriction, like attempting to open a socket or write into a file, are included in the library. The modules which features are not supported by the standard are disabled, and the part of code which imports them will raise an error. All applications code must be written entirely only in Python language and code with extensions written in C is not supported. Rich APIs for the Google Accounts, URL fetch, Datastore and email servers are provided by the Python environment.

Simple web application framework called "webapp" which makes easy to start building applications is provided as well by App Engine. In addition Django Web application framework, version 0.96.1 is also included in App Engine, but it should be take into account that App Engine datastore is not a relational database, which is required by some Django components. The option to be included other third-party libraries is also supported. As long as they are implemented in Python programming language and if they not require any unsupported library modules, it should not be a problem to be used.

This chapter presented modelling of Trust Policies through UML diagrams, explained notion of trust and its different facets. Risk factor is included Trust Management to make Trust Policies more effective and finally Trust Policies are represented through Decision Table. Different tools for setting Cloud environments have also been discussed.

Chapter 5

Experimental Results

This chapter focuses on implementation of proposed design of trust policies on Google App Engine in Python and experimental results of this approach. Installation of Google App Engine and Python has been shown in appendix.

5.1 Implementation of Trust Policies

In this thesis an application has been developed for providing secure login on Google App Engine using Python language. This application can be used by any cloud provider in order to secure his login by deciding through these designed policies whether to trust the customer or not. Here, we describe the functionalities of the designed system in three parts:

i) Sign in

ii) Register

iii) Free trial

Figure 5.1 shows user interface for login and registration.

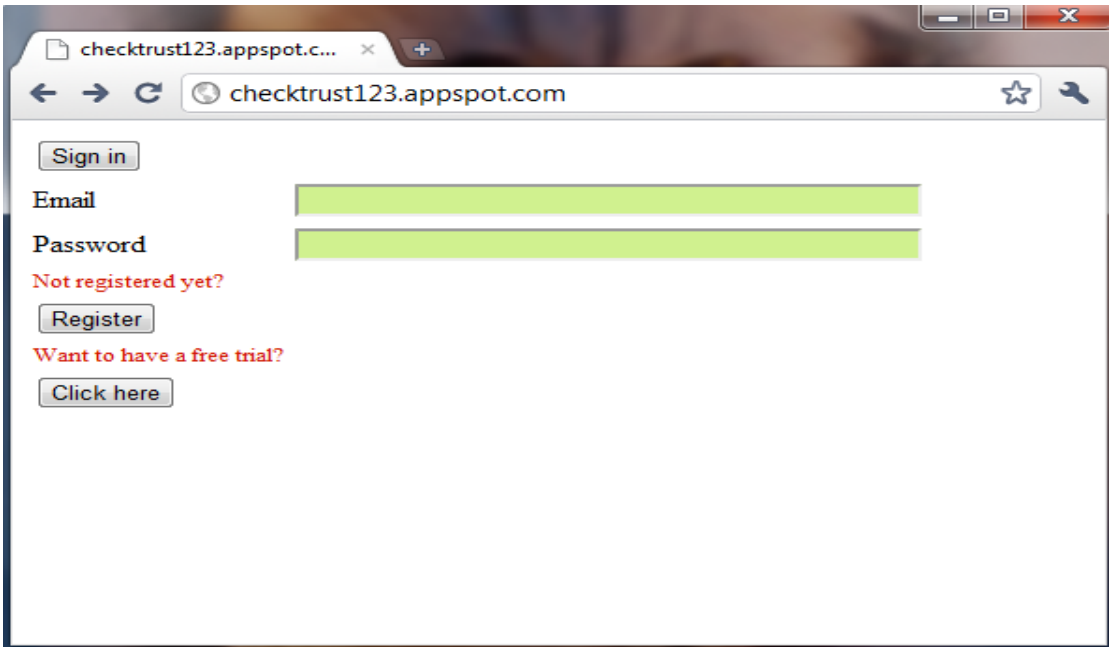

Figure 5.1: User interface for login and registration

Administrator and its peers MindTree, Gogrid, Imeptus and Infotech have been assigned roles and they can rate their previous experience with the customer. So before making any decision administrator checks his previous experience with the customer and that of its peers too. Following figures show admin and peers' login and form to fill their previous experience.

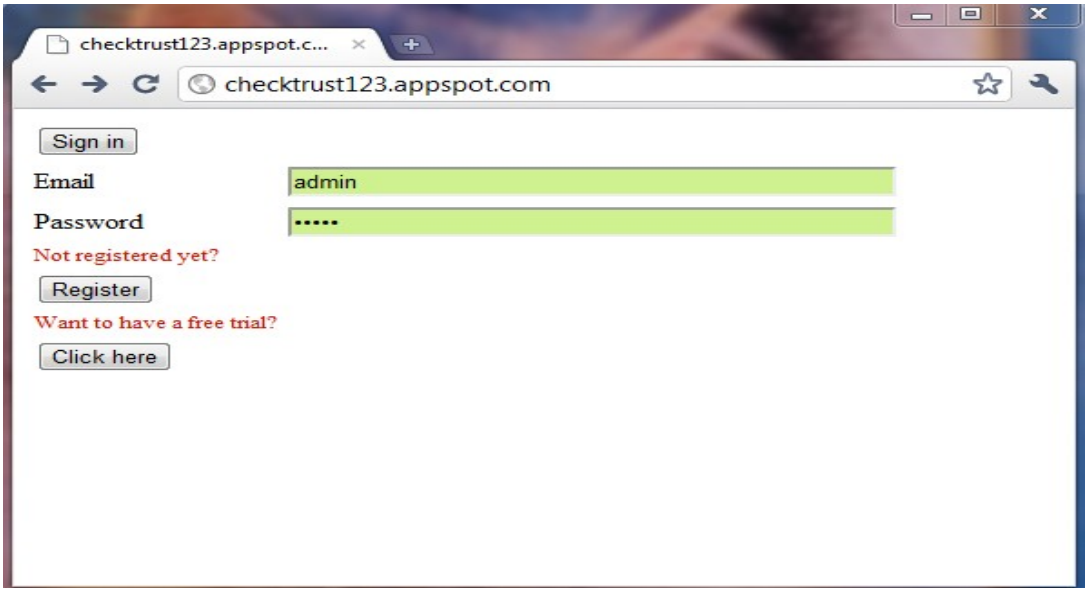

Figure 5.2: Admin login page

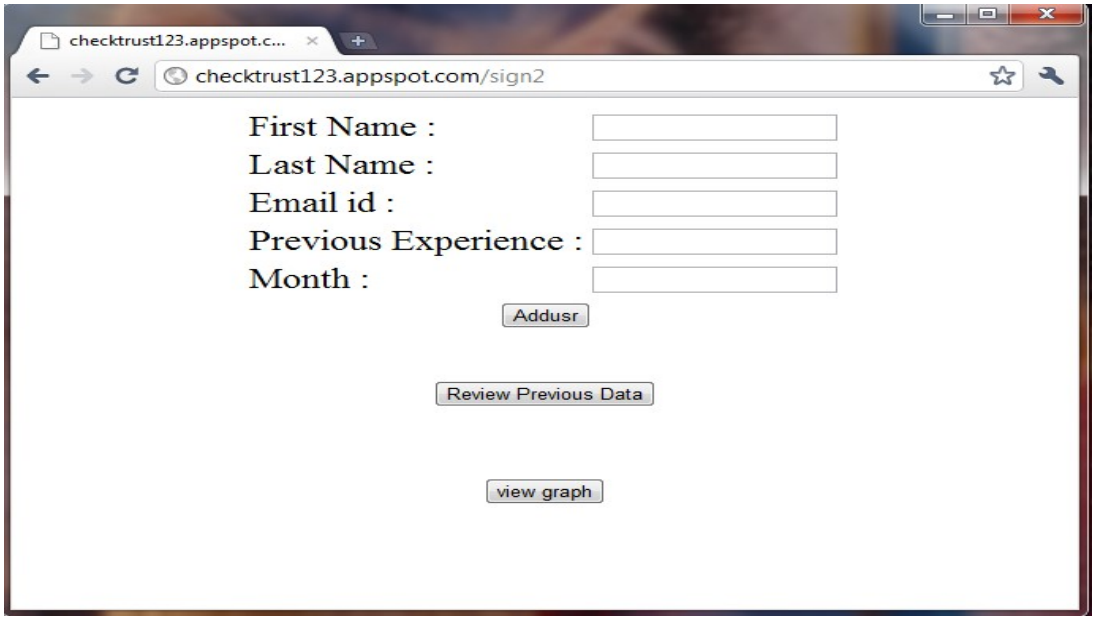

Figure 5.3: Form for filling previous experience

Addusr button allows admin to add details of the customer and experience he had with him. Review previous data allows admin to see details of all the previous customers. Figure 5.4 shows sample of database filled by the admin. View graph allows admin to

see how many good and bad experiences were encountered per month and analyze accordingly. Figure 5.5 shows the graph.

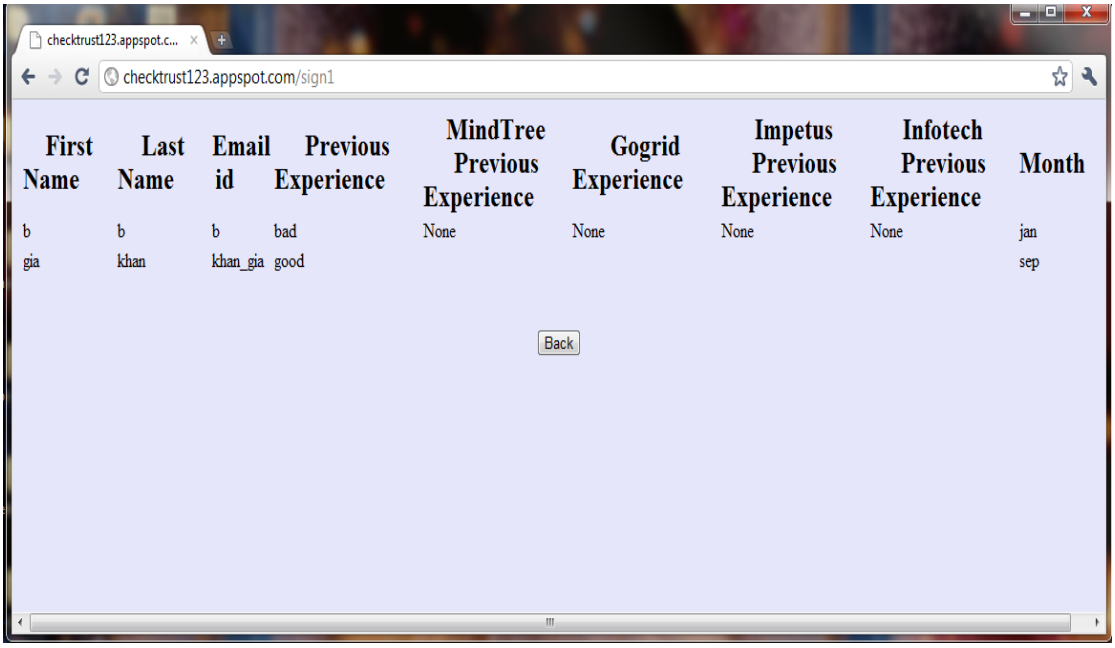

Figure 5.4: A sample database

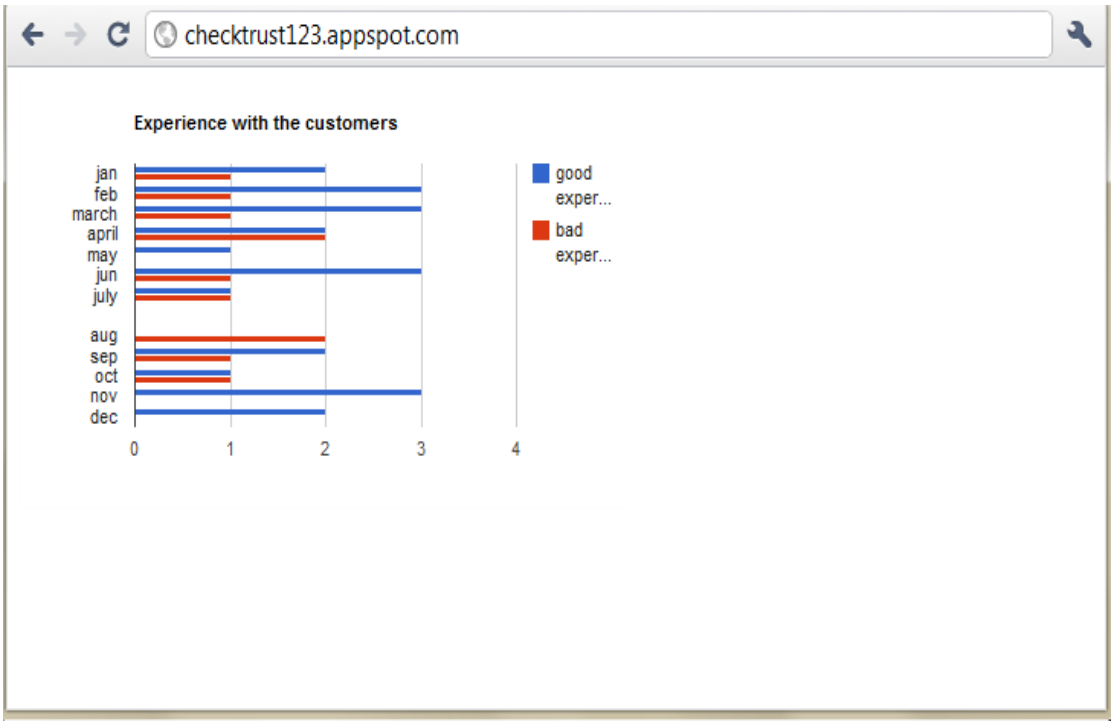

Figure 5.5: Good experience v/s bad experience graph

Figure 5.6 shows a peer Gogrid's login which quite similar to that of admin. Other peers MindTree, Impetus and Infotech have similar interfaces.

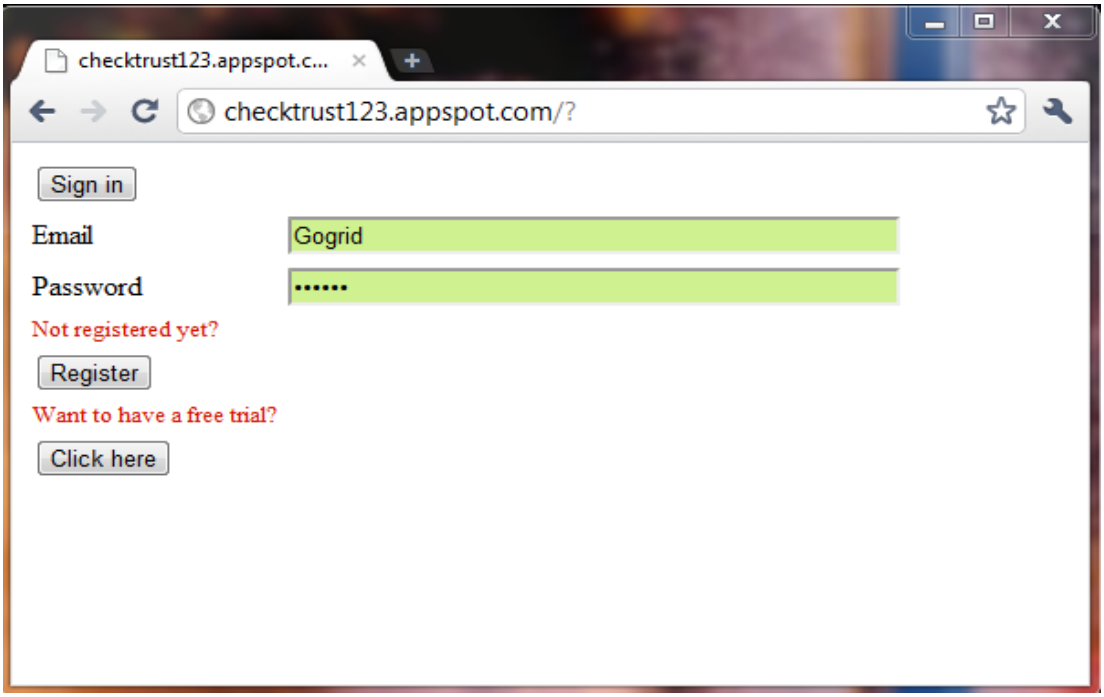

Figure 5.6: Gogrid login page

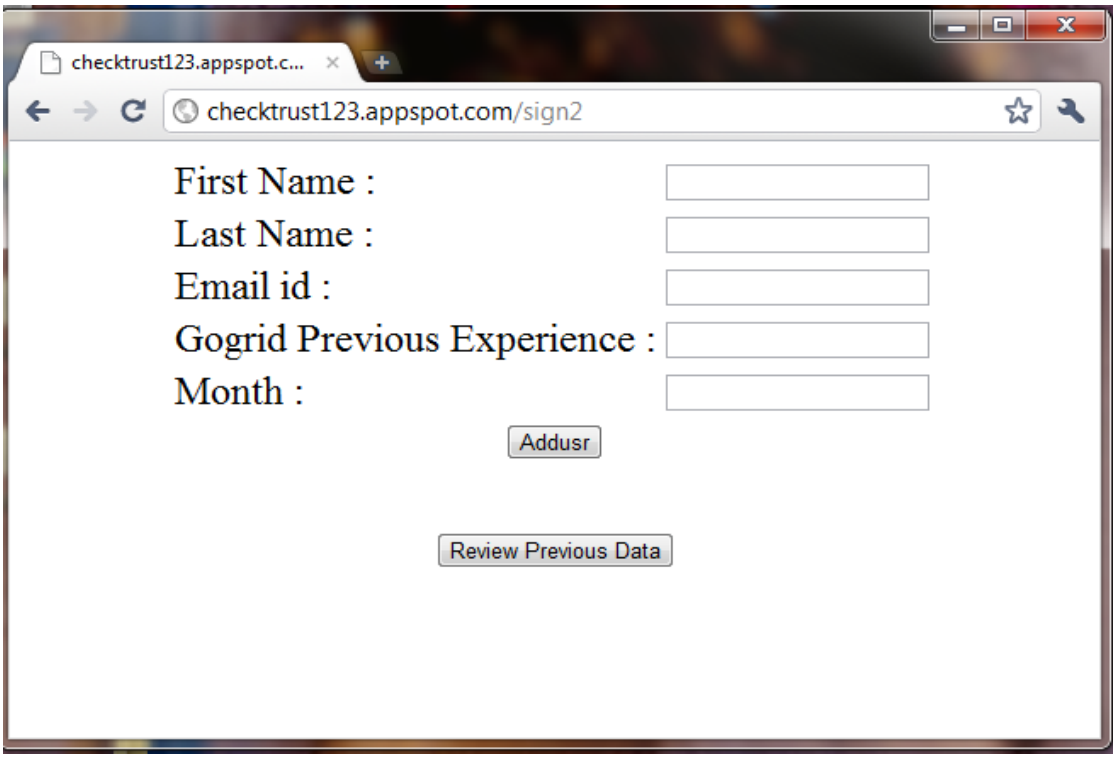

Figure 5.7: Form for filling Gogrid previous experience

Figure 5.8 shows registration page which opens on clicking Register button. By filling registration form completely customer can access cloud services. Few fields have been made mandatory to be filled otherwise form will not be submitted. Another

important thing in this form is SLA agreement which if not agreed by the customer will not let him to continue.

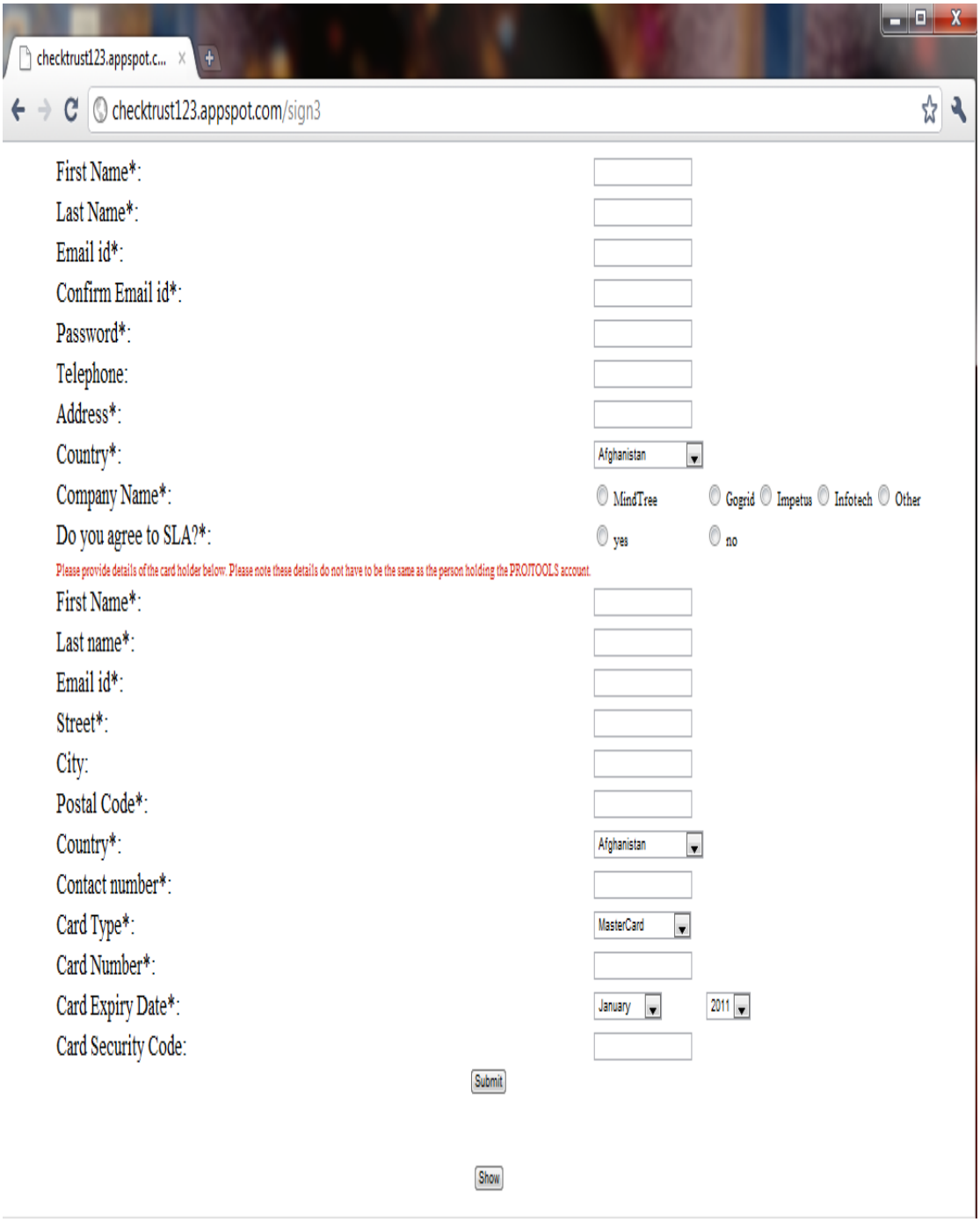

Figure 5.8: Registration form

Figure 5.9 shows a completely filled registration form by a customer and Figure 5.10 shows message sent by the administrator to the customer after receiving the form.

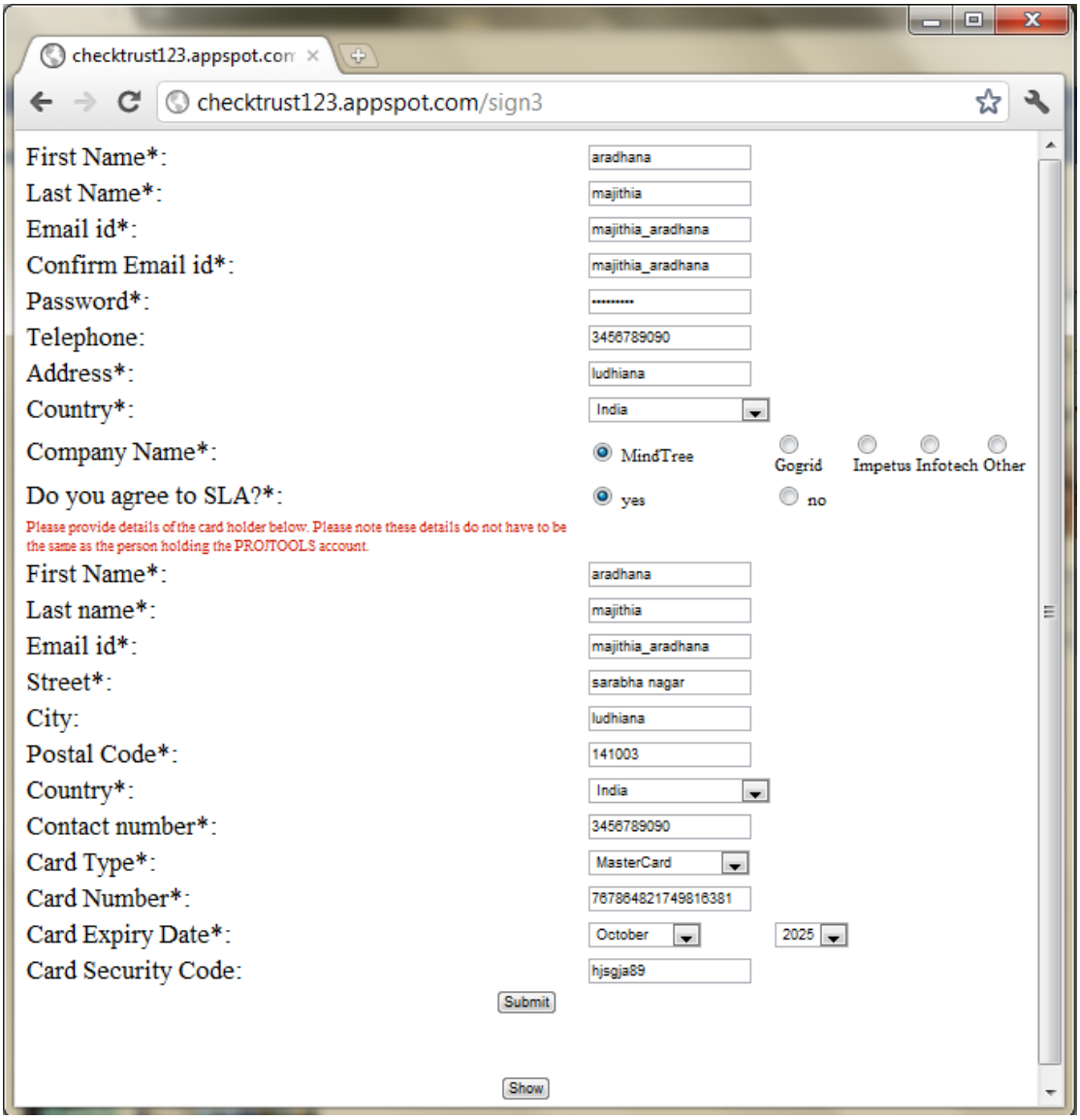

Figure 5.9: Completely filled registration form

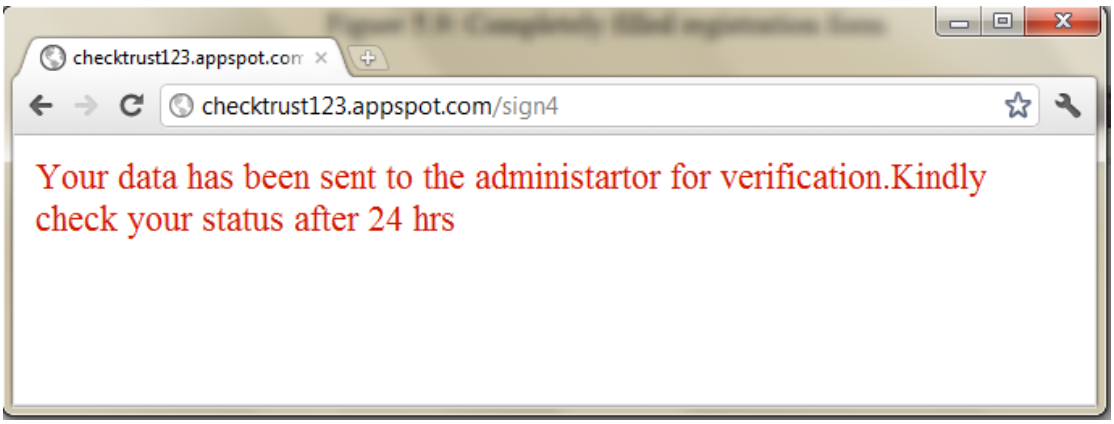

Figure 5.10: Message by administrator

Figure 5.11 shows registration form in which one of mandatory fields is not filled by the customer and Figure 5.12 shows registration form in which SLA is not agreed by the customer.

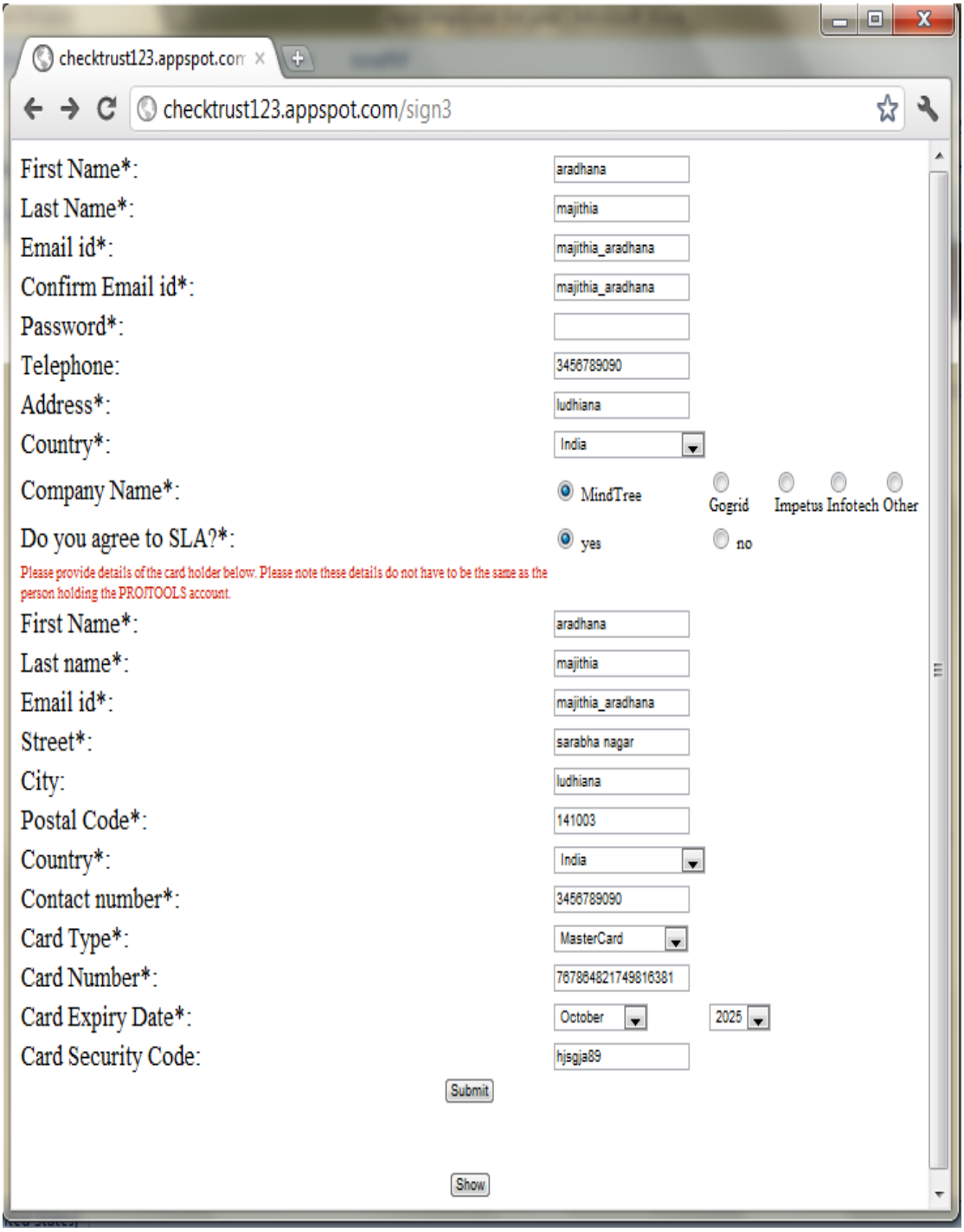

Figure 5.11 Mandatory fields missing in form

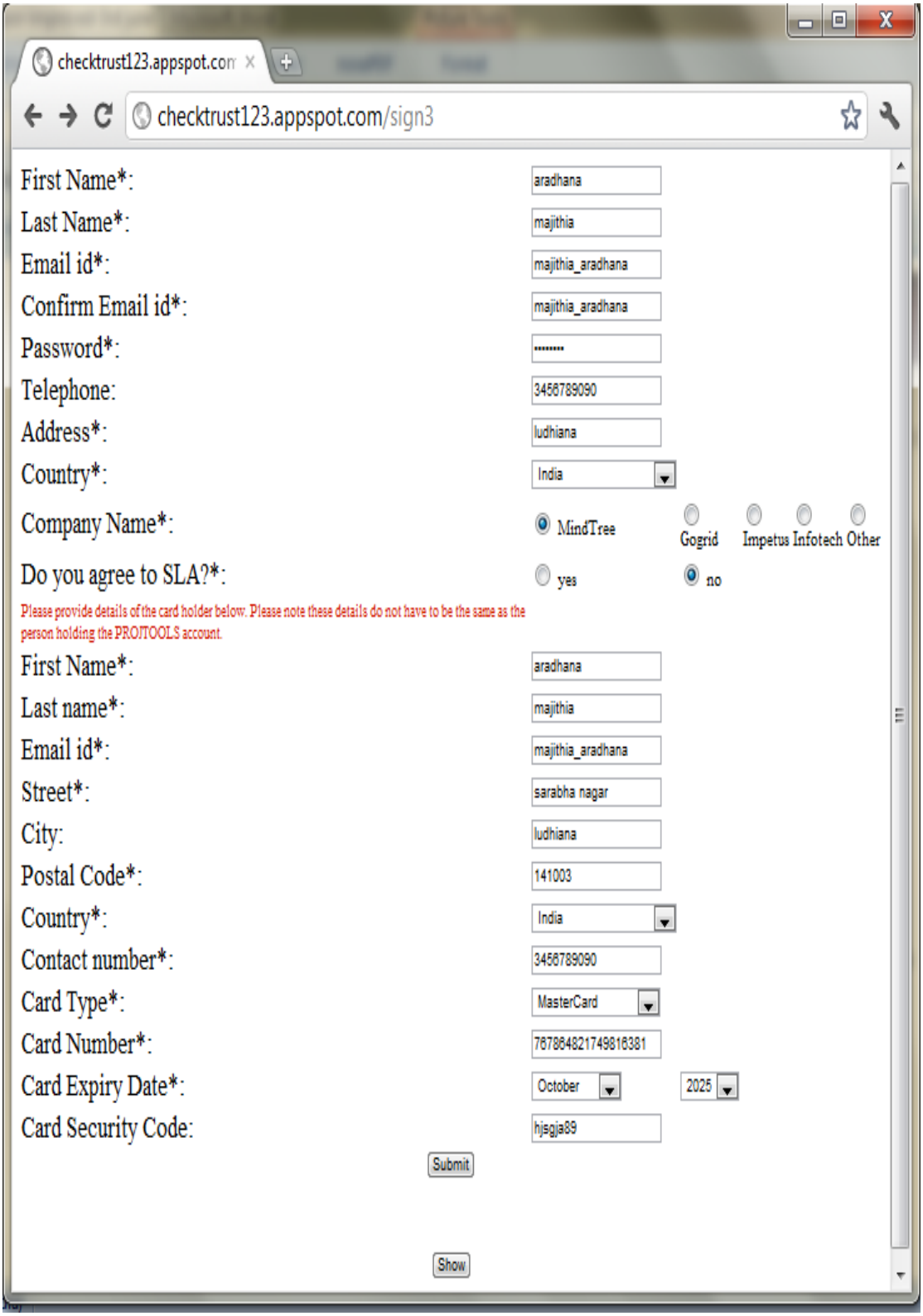

Figure 5.12 SLA not agreed by the customers

Figure 5.13 shows the form displaying the message that mandatory fields are missing. This form opens when previous form was submitted incomplete or when SLA was not agreed.

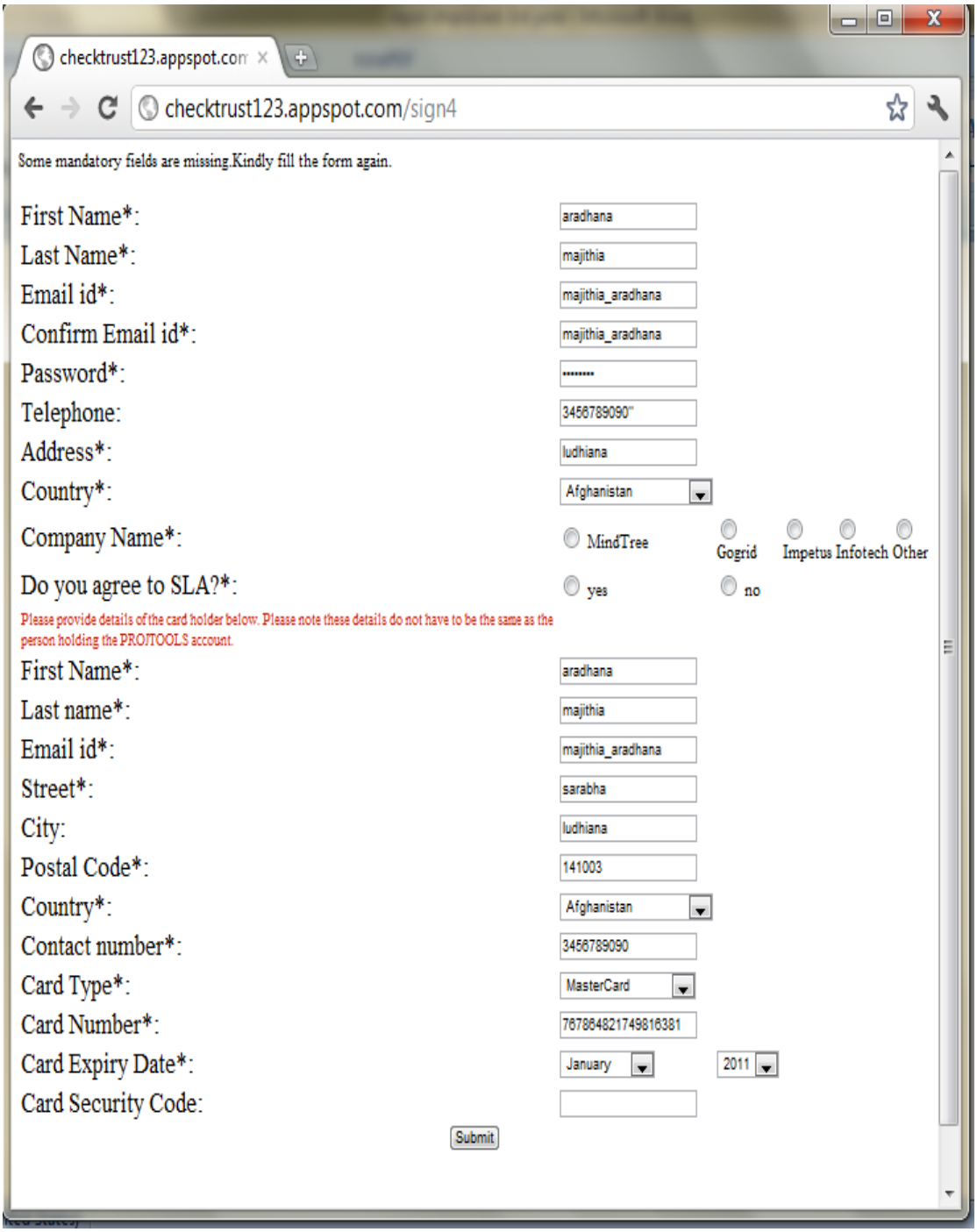

Figure 5.13: Form displaying message mandatory fields not filled

Two fields are there in the registration form Email id and Confirm Email id. Entries in both the fields should be same. If they they don't match registration form will not be submitted. Figure 5.14 shows mismatch in email id.

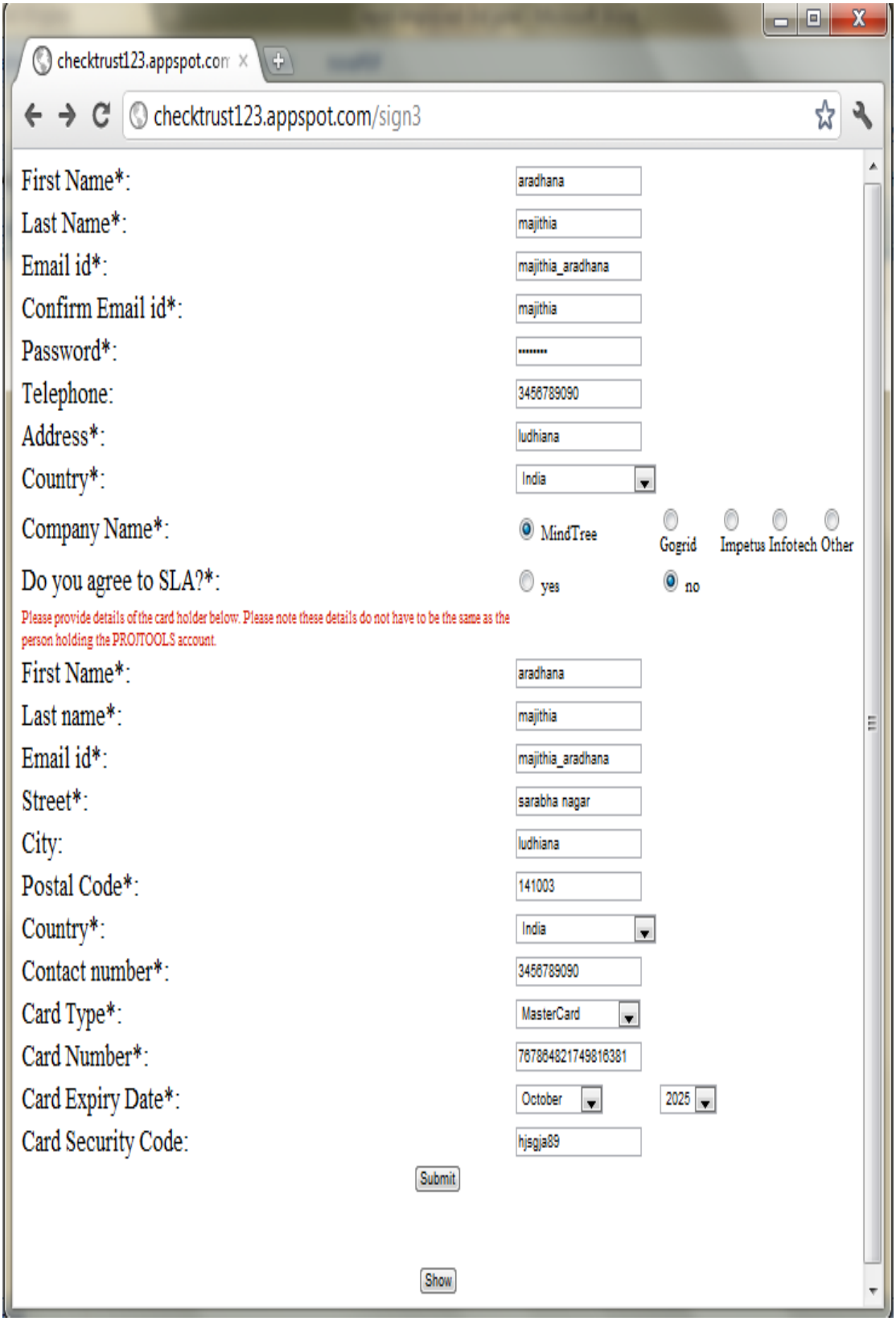

Figure 5.14 Email id mismatch

Figure 5.15 shows form displaying message that email id don't match. So customer needs to fill the form again.

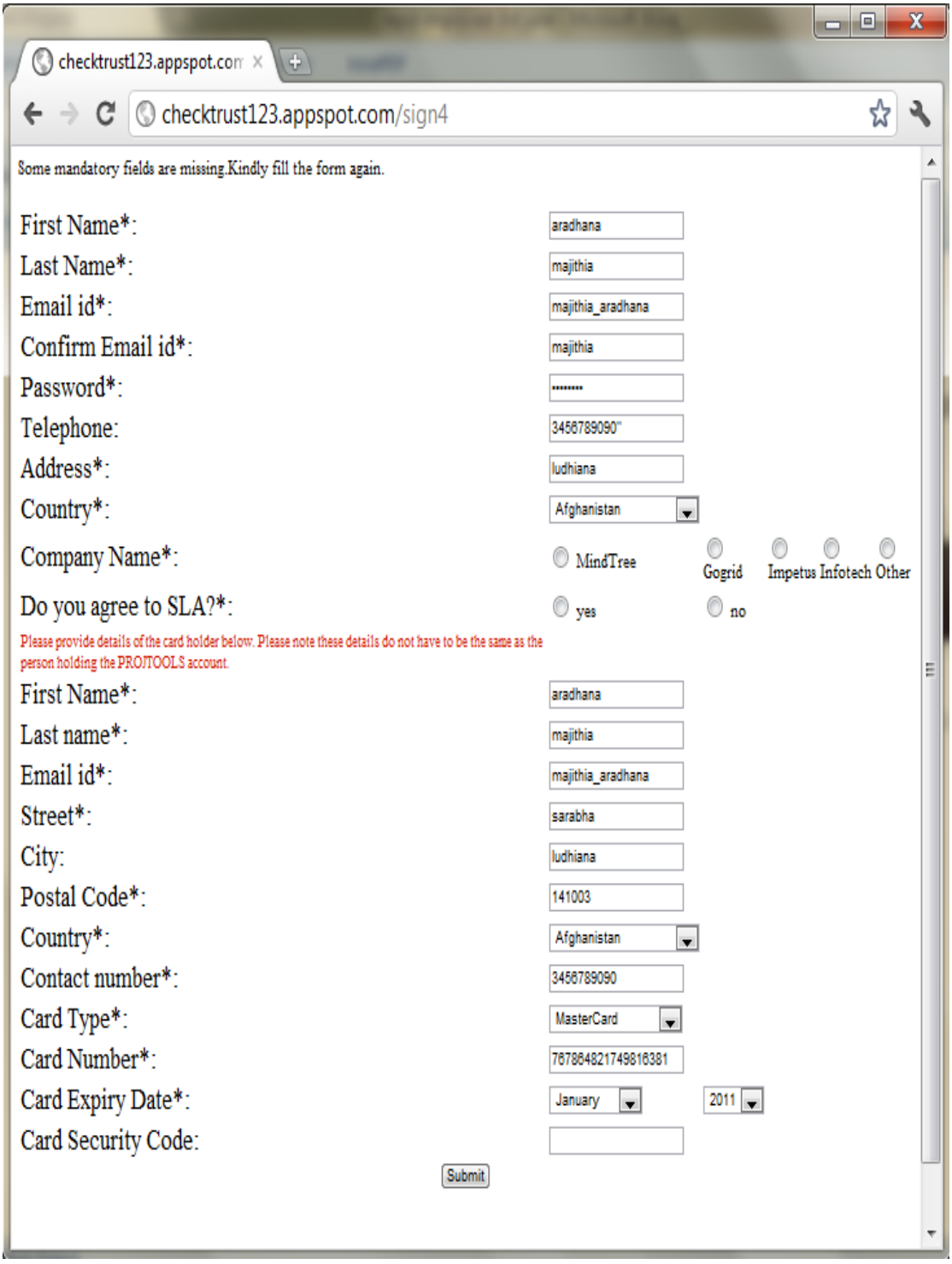

Figure 5.15: Form displaying message that email id don't match

Now before allowing customer to access cloud services administrator will check whether he has any previous experience with the customer or not. If he has interacted previously and the experience was good customer is allowed to use cloud services

otherwise he is denied of services. Figure 5.16 shows that service provider has good previous interaction with the customer.

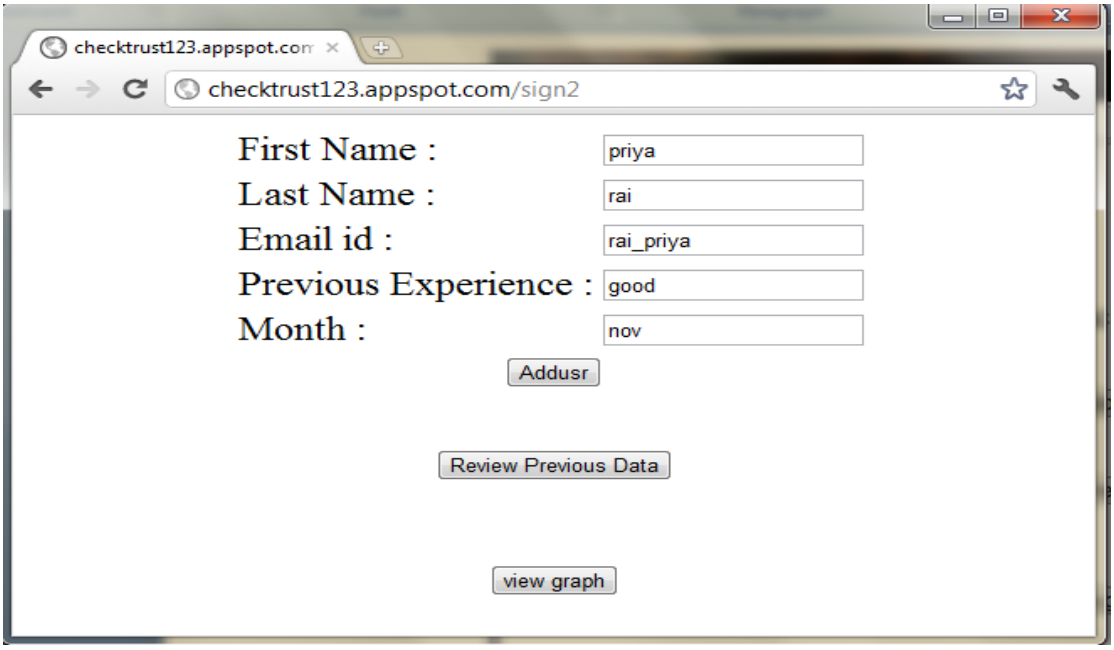

Figure 5.16: Previous experience good with customer

Now when customer tries to login as shown in Figure 5.17 administrator allows customer to access services because of good experience with him. Figure 5.18 shows successful login.

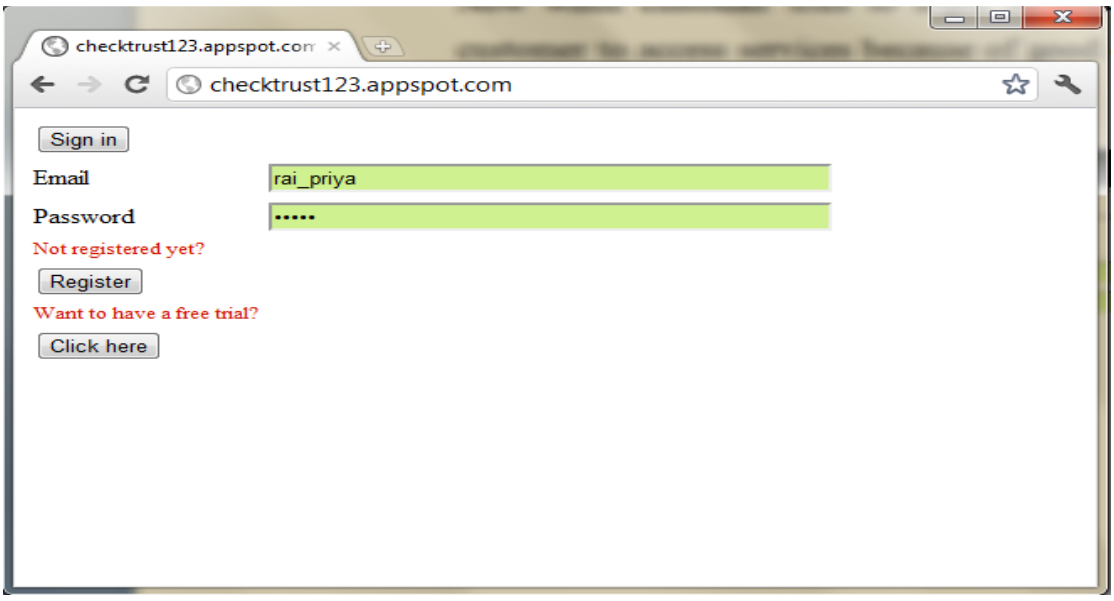

Figure 5.17: Customer with good experience login

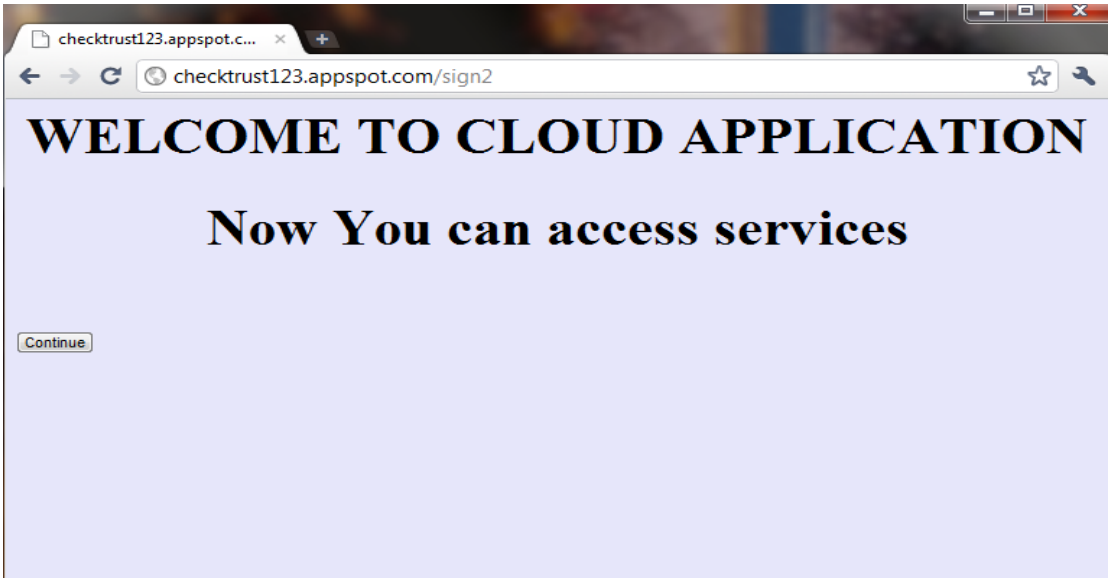

Figure 5.18: Successful login

Now if service provider's experience with customer was not good he will not be allowed to use the services. Figure 5.19 shows that service provider has bad previous experience with the customer. Figure 5.20 shows that customer with whom service provider had bad experience tries to login. Figure 5.21 shows unsuccessful login.

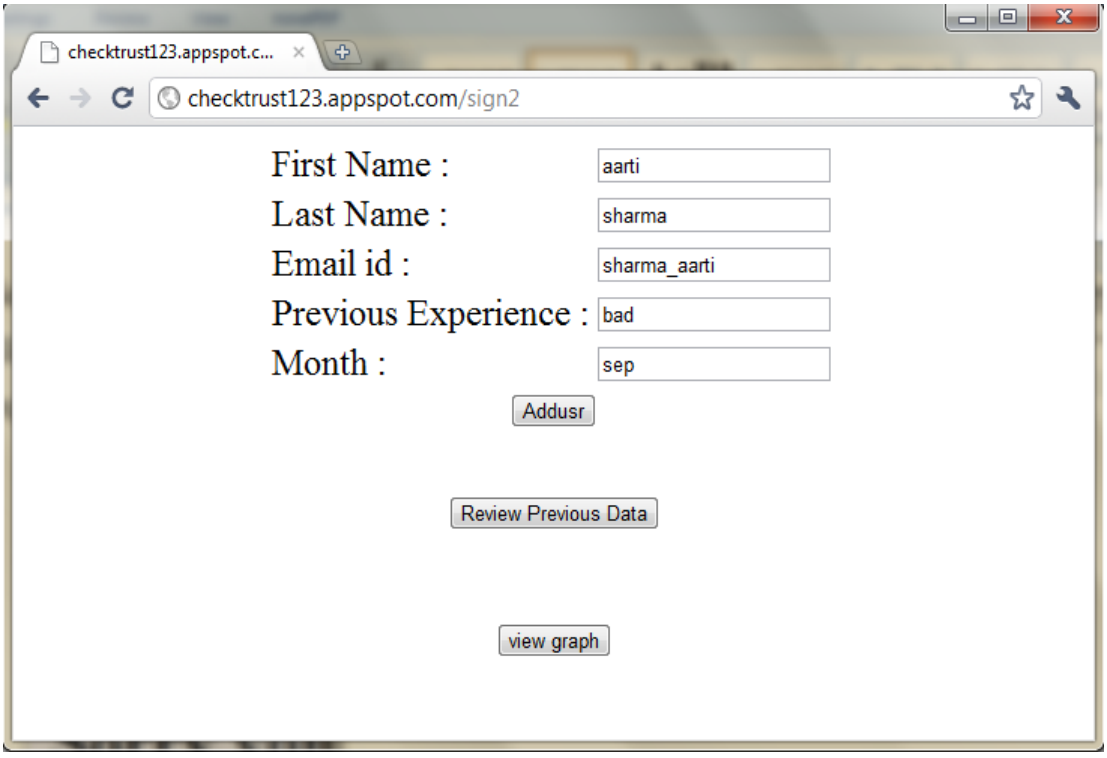

Figure 5.19: Bad experience filled by the administrator

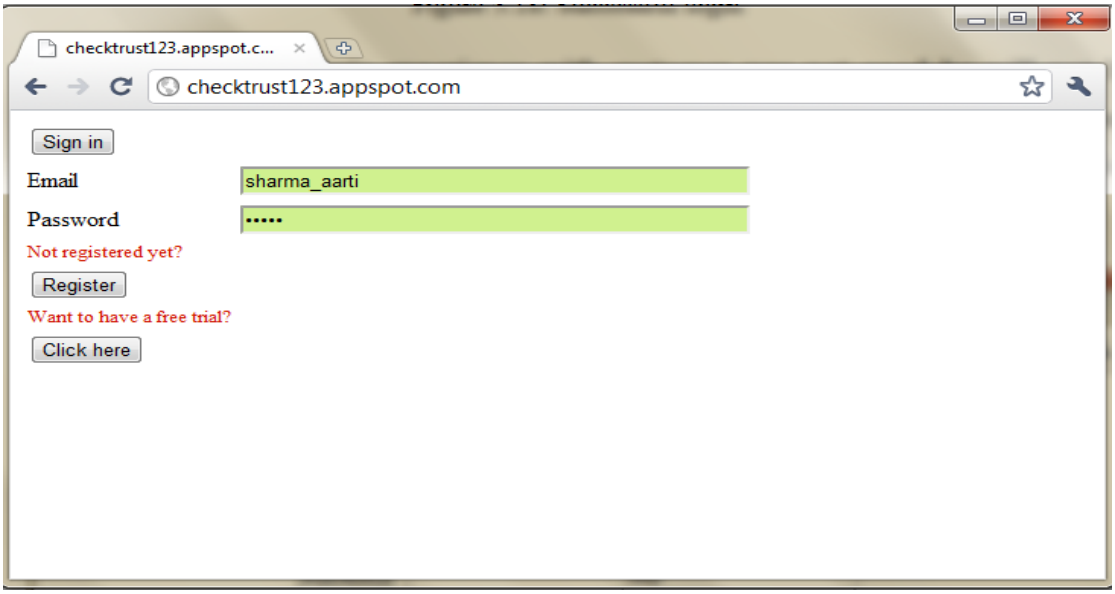

Figure 5.20: Customer with bad experience login

Figure 5.21: Unsuccessful login

If service provider does not have any previous interaction with customer he will check if any of his peers have previous interaction with him. If previous interaction with any of the peer is good customer will be allowed to access cloud services else he will be denied of them. Figure 5.22 shows a peer's good experience with the customer and Figure 5.23 shows message sent by the administrator when he tries to login.

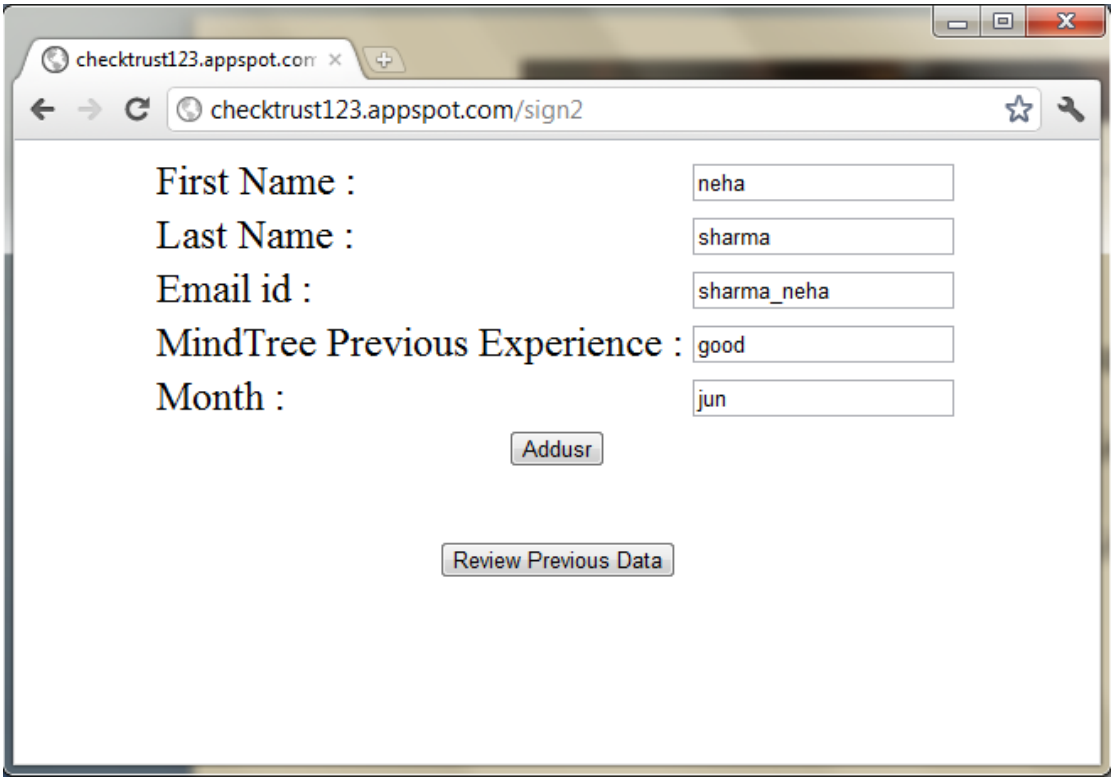

Figure 5.22: Peer's good experience with the customer

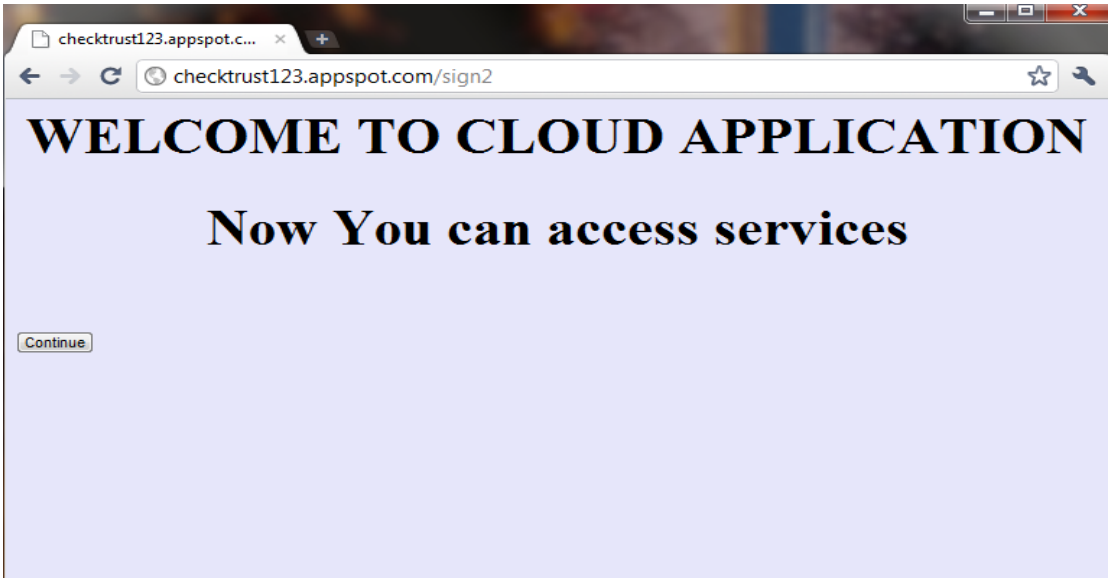

Figure 5.23: Successful login because of good experience with a peer

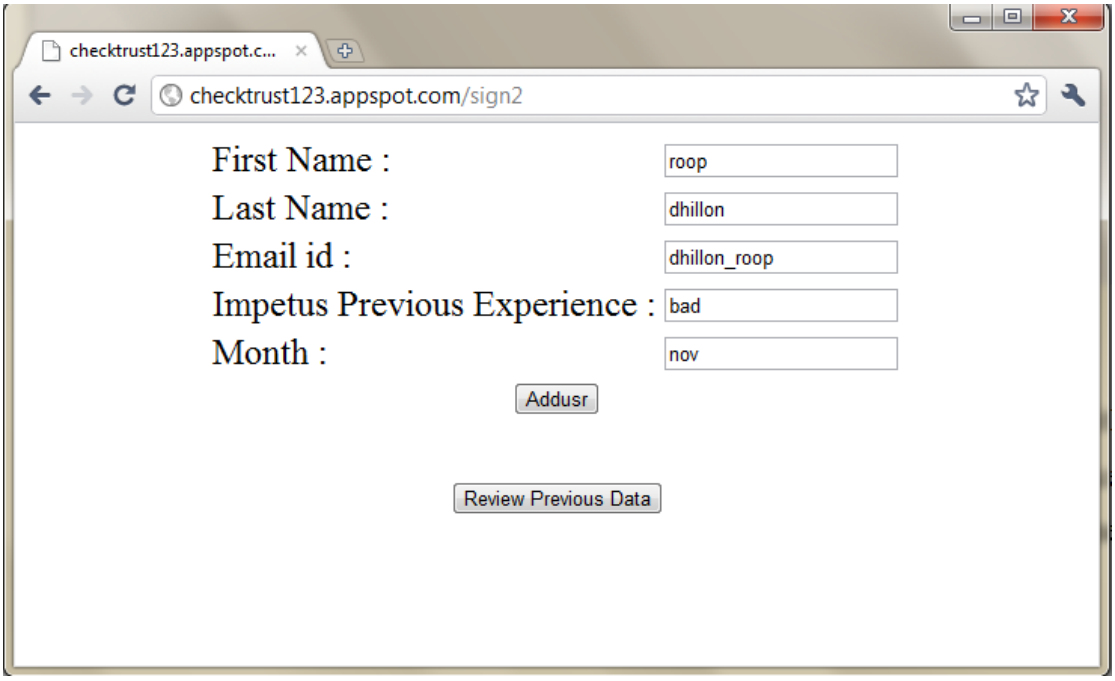

Figure 5.24: Peer's bad experience with the customer

Figure 5.25 displays message sent by the administrator when roop tries to login.

Figure 5.25: Unsuccessful login because of bad experience with a peer

It may happen that both service provider and its peer have previous experience with the customer. In that case service provider's rating will be given preference. Following figures show a case where peer has good interaction with the customer previously but peer had a bad one.

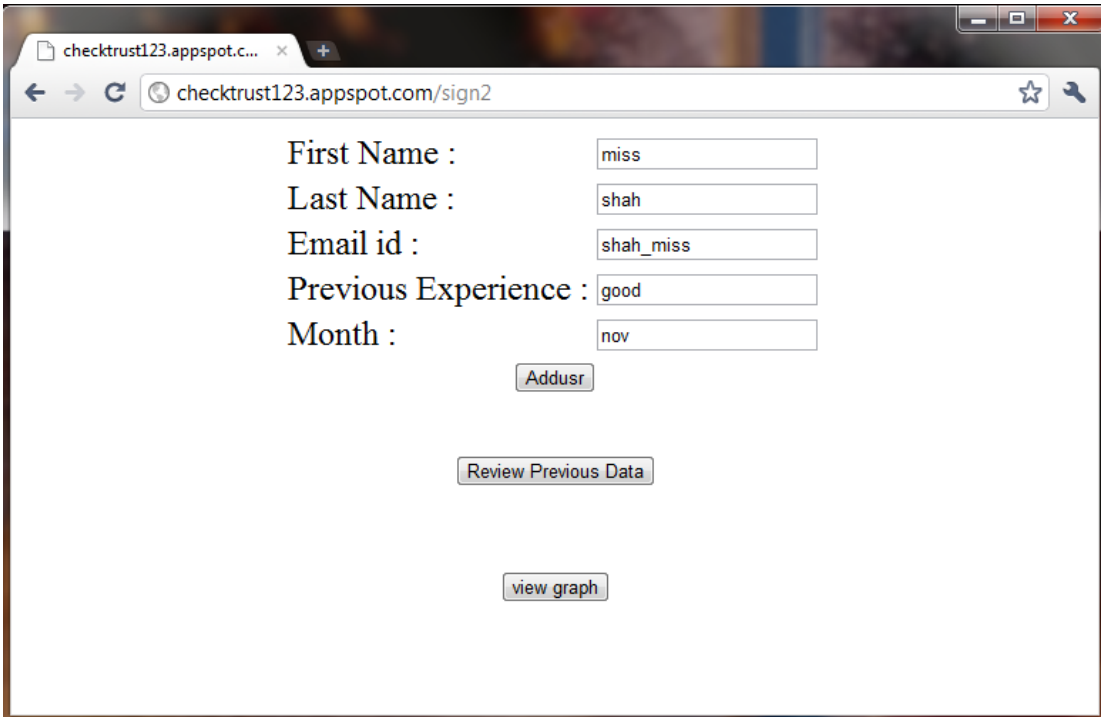

Figure 5.26: Service Provider's good experience with the customer

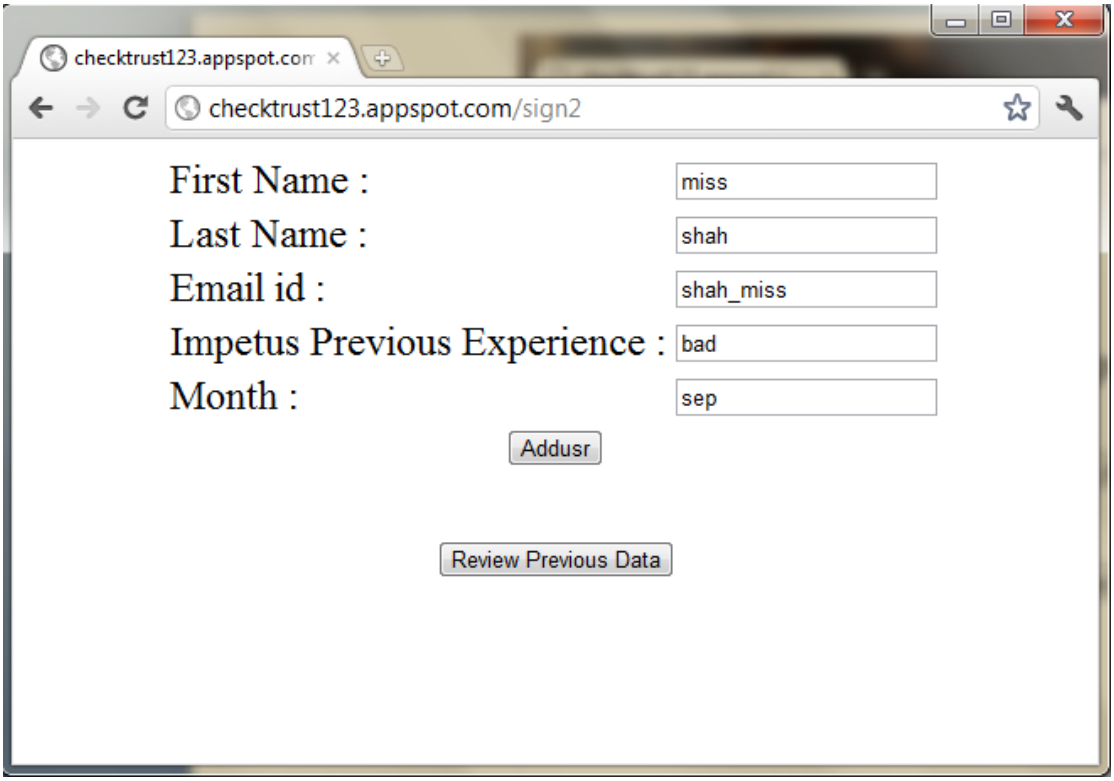

Figure 5.27: Peer's bad experience with the same customer

Figure 5.28 shows when that customer tries to login with the intent of using cloud services. Figure 5.29 shows decision sent by the administrator.

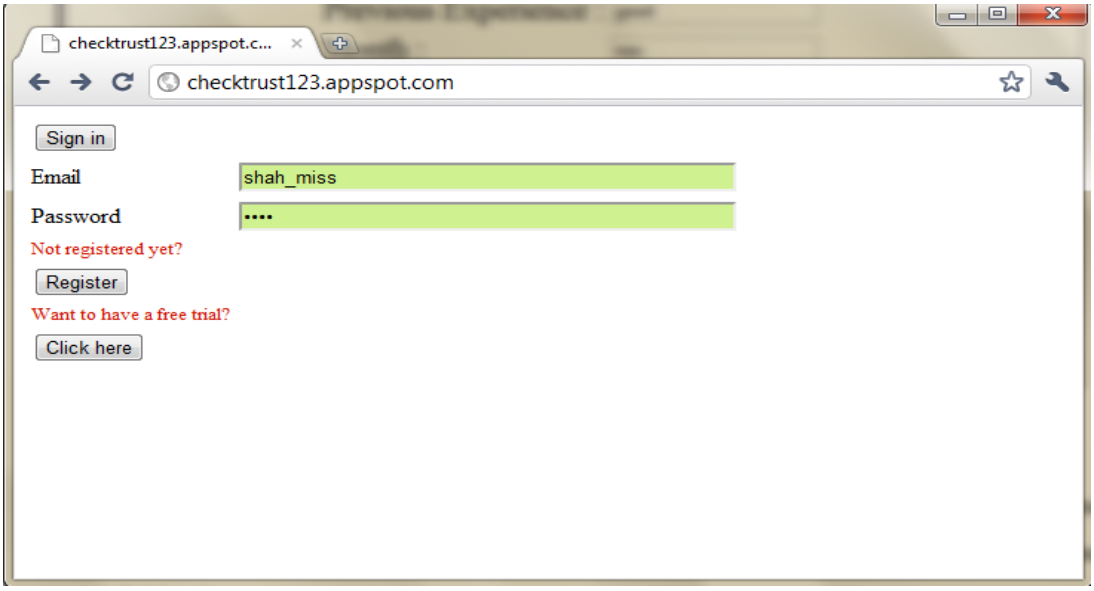

Figure 5.28: Customer with whom service provider had good experience and peer had bad experience tries to login

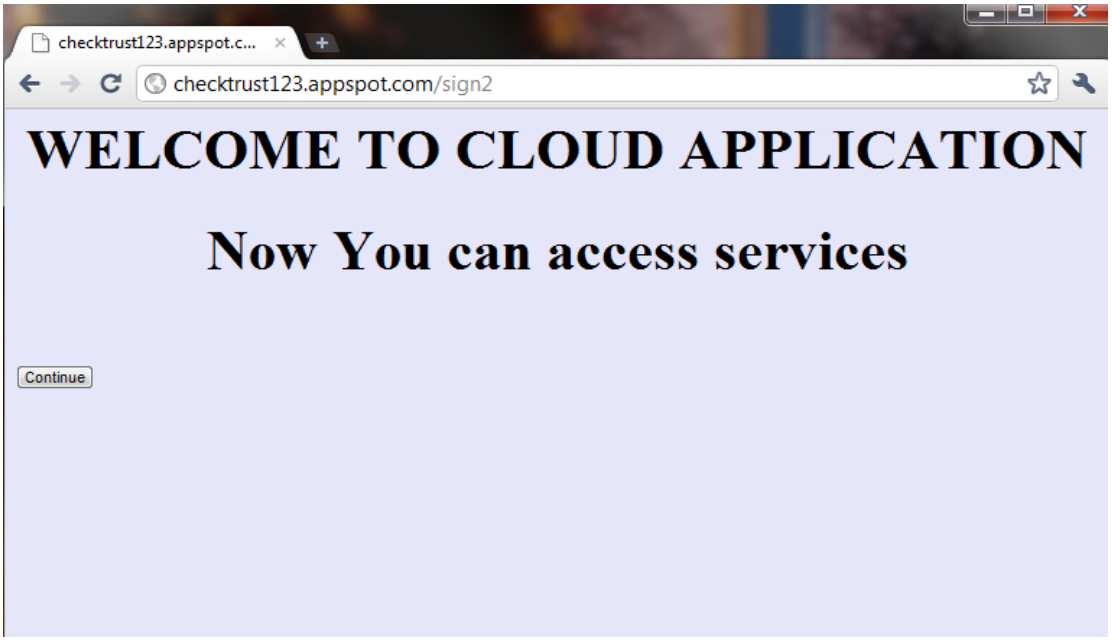

Figure 5.29: Successful login because of good experience with the service provider

Next case is when though one of peer's experiences with customer was good but service provider's own experience with him was bad. Figure 5.30 shows service provider had bad experience with the customer and Figure 5.31 shows peer's good experience with him and finally Figure 5.30 shows decision made by the administrator.

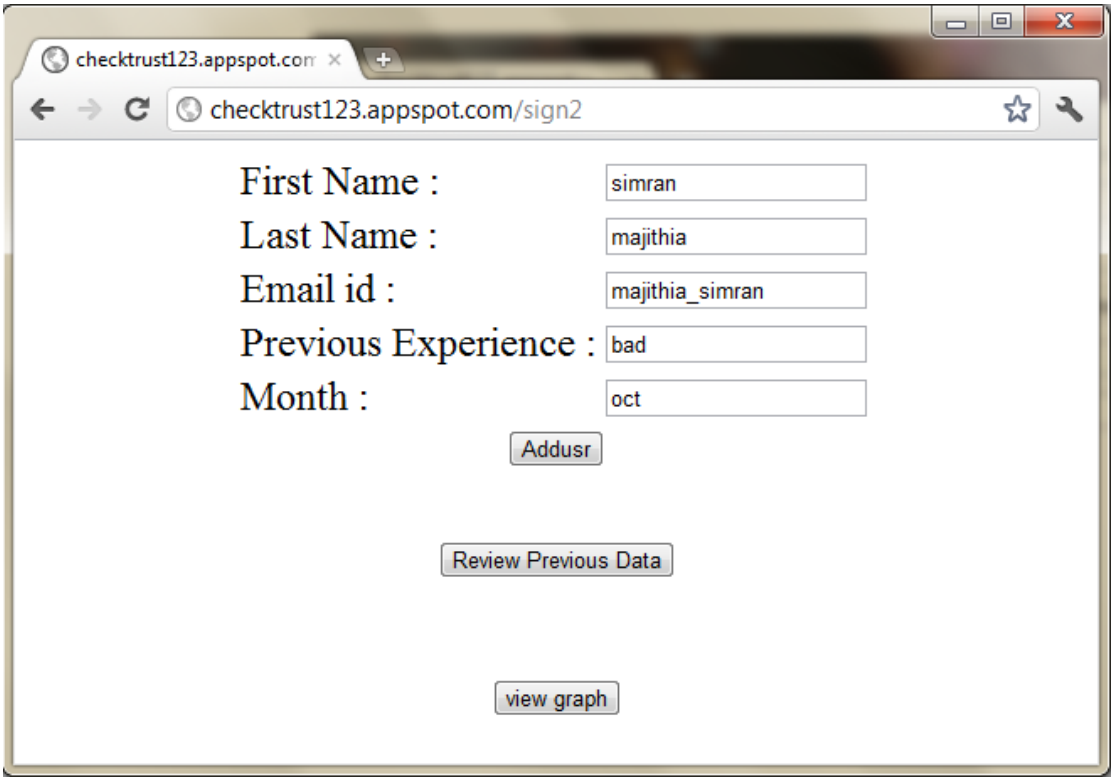

Figure 5.30: Service provider's bad experience with the customer

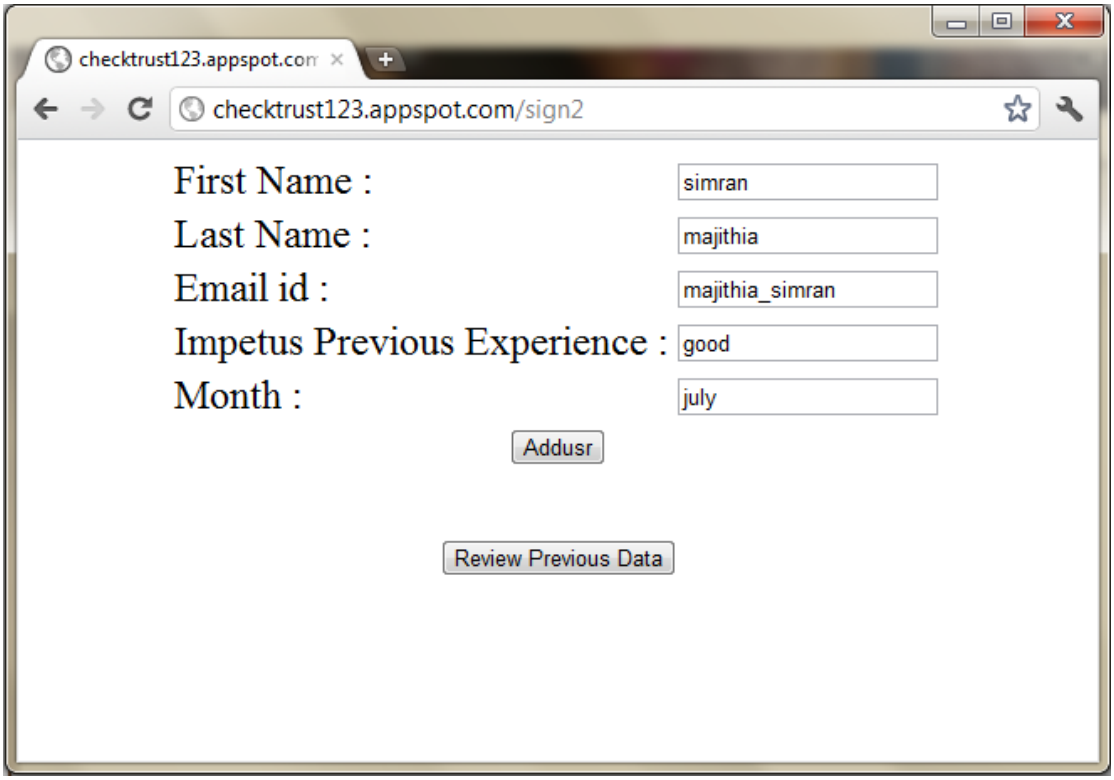

Figure 5.31: Service provider's peer's good experience with the same customer

Figure 5.32: Unsuccessful login because of bad experience of service provider with the customer

It may happen that customer wants to use cloud services on trial basis. For this customer needs to fill a small form avail free services. This form just like registration form contains few compulsory fields and SLA agreement which needs to be agreed upon. Figure 5.33 shows free trial form.

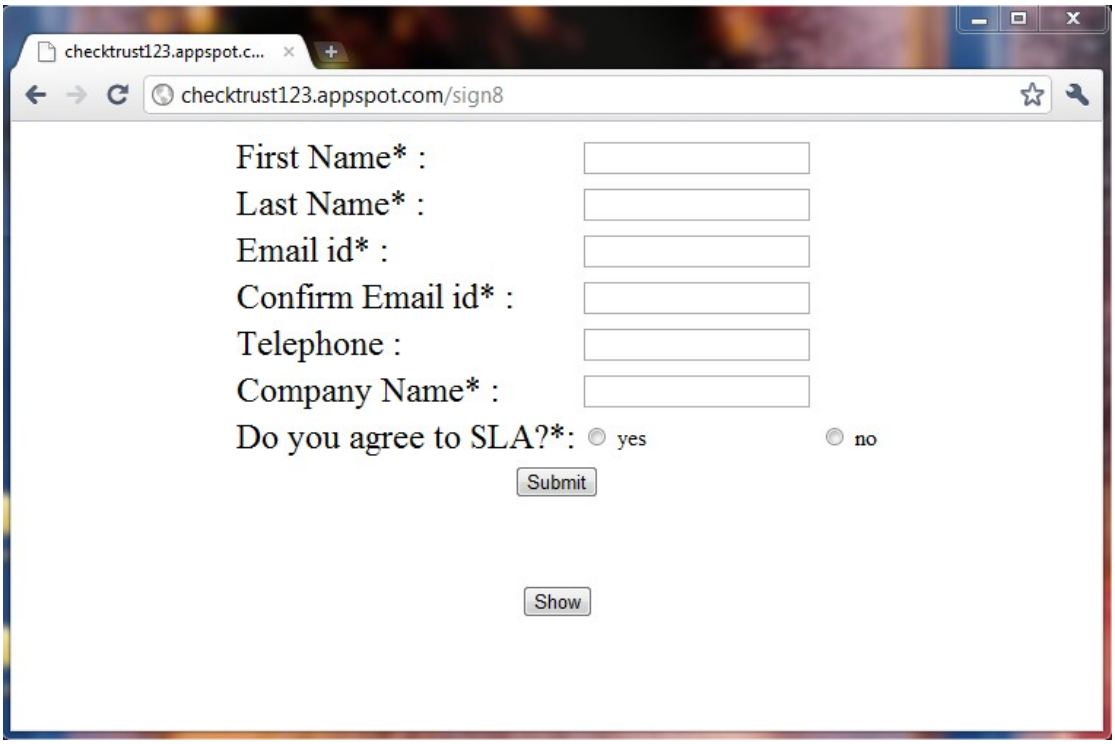

Figure 5.33: Free Trial form

Figure 5.34 shows a completely filled form along with SLA agreed upon. Administrator allows customer to use free trial services.

Figure 5.34: Completely filled free trial form along with SLA agreed upon

When customer tries to login after completely filling the free trial form administrator sends him his decision.

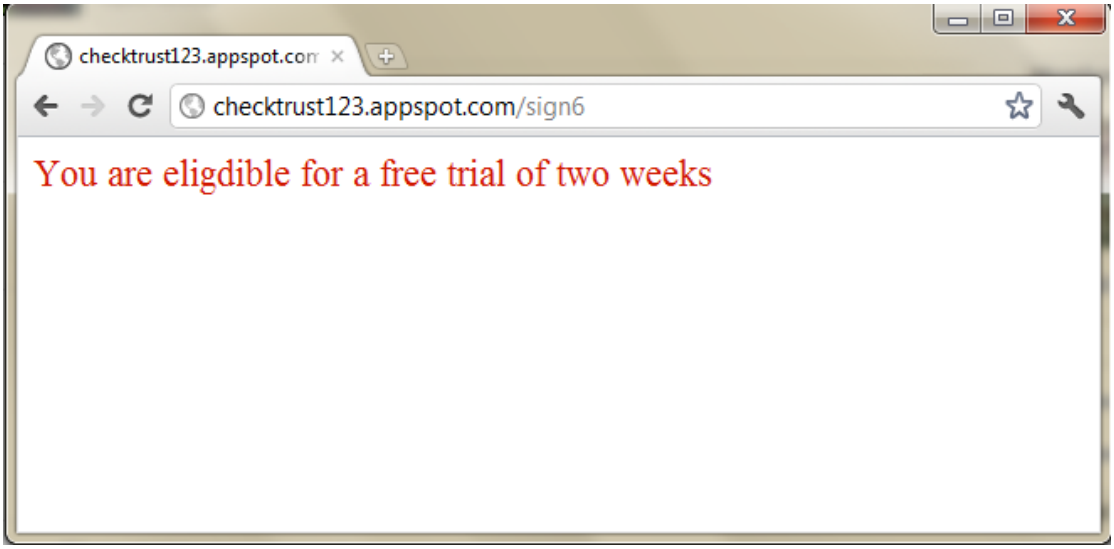

Figure 5.35: Administrator decision in favour

Figure 5.36 shows that SLA is not agreed by customer while fillilng the form. If SLA is not agrred by the customer he will not be allowed to use free cloud services for trial.

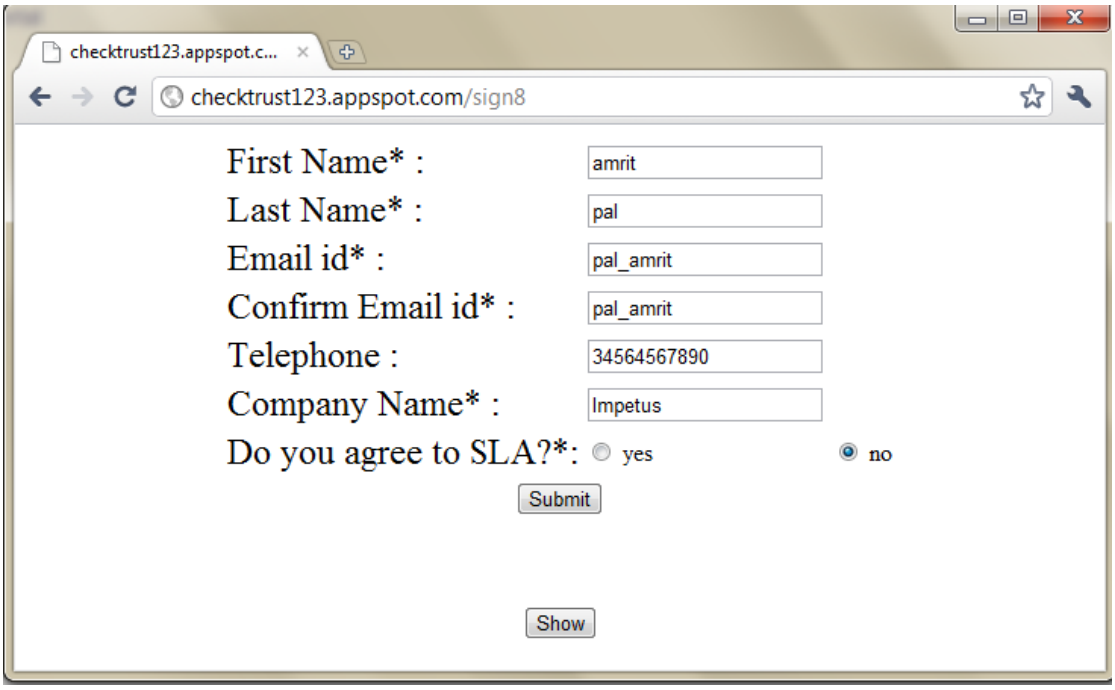

Figure 5.36: SLA not agreed by the customer in free trial form

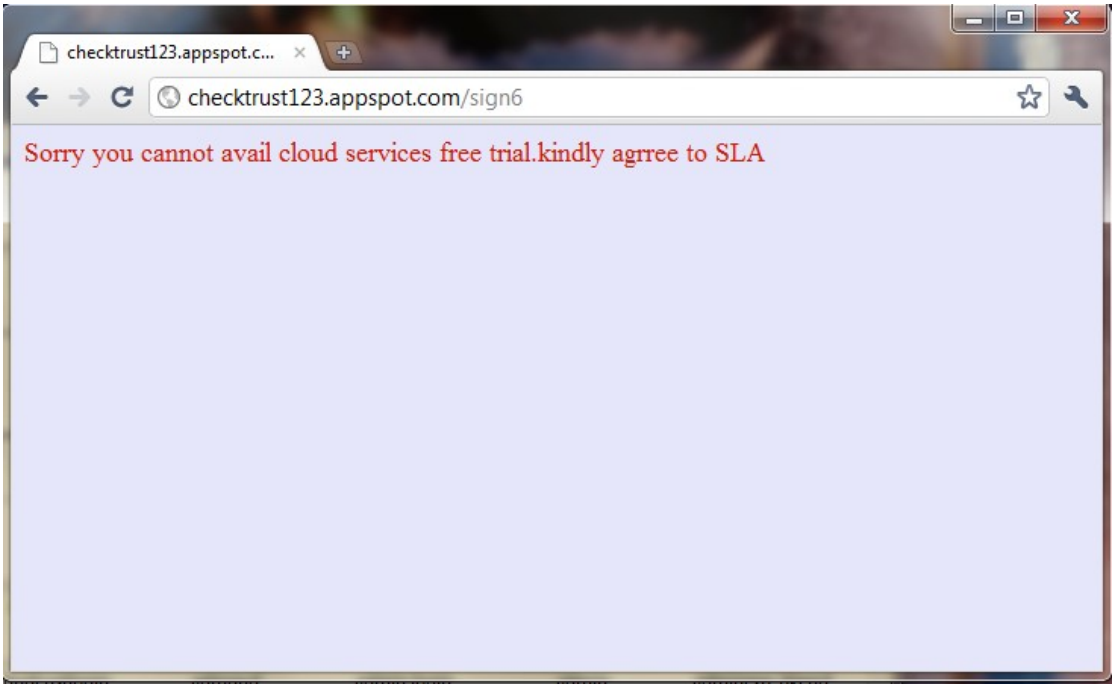

Figure 5.37: Message sent by administrator to agree to SLA

Now few things are similar to registration form such if email id and confirm email id fields do not match then customer will again have to fill the form correctly and resend.

This chapter focussed on the implementation of the designed Trust Policies in Cloud environment. Various contexts were taken into account to make Cloud Identity Provisioning more effective. Next chapter discusses conclusion and future scope of thesis work.

Chapter 6

Conclusion and Future scope

This chapter discusses the conclusion of work presented in this thesis. The chapter ends with a discussion of future direction and scope of research.

6.1 Conclusion

Cloud computing is the next step in utility computing. To leverage its advantages cloud security needs to be addressed as it poses to be one of the key challenges. In this regard Identity provisioning based on Trust Policies can benefit the cloud providers to a large extent.

In this thesis current Identity Federation Frameworks have been compared, importance of Trust in Cloud Identity Management has been analyzed, solution has been designed through UML diagrams, notion of Trust Management has been explained and risk factor has been incorporated in Trust Management to make Trust Policies more efficient. This work presented the decision table for Trust Policies along with its implementation in Cloud environment. A Cloud environment has been setup by using Google App Engine and Python.

6.2 Thesis Contribution

a) In this thesis Current Identity Federation Frameworks (like SAML, OpenID, Liberty Alliance Initiative, etc) have been compared.
b) Risk factor has been incorporated in Trust Management to make Trust Policies more effective.
c) Trust Policies have been designed for dynamic trust establishment and therefore for better decision making.
d) Trust policies have been implemented and validated on Google App Engine for Python.

6.3 Future Scope

a) Trust Policies developed can be extended to include more decisions for Trust Management.
b) More trust metrics can be included for better decision making.
c) Reputation updates can also be included by developing reputation update engine.

References

[1] D. Povey, “Developing Electronic Trust Policies Using a Risk Management Model”. In Proc. of the Secure Networking - CQRE (Secure)’99, International Exhibition and Congress, LNCS 1740, D¨usseldorf, Germany, pp. 1–16, Nov 30 - Dec 2 1999.

[2] D. Harrison McKnight, L. Larry and L. Norman, “Trust formation in new organisational relationships”. In Information and Decision Sciences Workshop, pp.11-16, Oct 1995.

[3] A. Gopalkrishnan, “Cloud Computing Identity Management” SETLabs Briefings vol7 no7, 2009.

[4] Deloitte, “Cloud computing-information security briefing” CPNI, March.2010. Available at http://www.cpni.gov.uk/Documents/Publications/2010/2010007-ISB_cloud_computing.pdf

[5] J. Brodkin. “cloud computing security issues”. Available at http://www.infoworld.com/d/security-central/gartner-seven-cloud-computing-security-risks-853, July 2008.

[6] T. Grob, “Security analysis of the SAML single sign on browser/artifact profile”. In Proc. of 19th Annual Computer Security Applications Conference, pp.298-308, 2003.

[7] P. Jadhwani, J. Mackinnon, M. Elrefal. “Cloud Computing-Building a Framework for Successful Transition”. GTSI, Northern Virginia, 2009.

[8] S. Bennett, M. Bhuller, R. Covington. “Oracle-Architectural strategies for cloud computing”, august 2009.

[9] A. Velte, T. Velte, R. Elsenpeter .“Your organization and cloud computing”. In Cloud Computing-A practical Approach, McGraw-Hill, pp.35-36, 2010.

[10] J. Vaos, J. Zhang. “Cloud Computing, New Wine or Just a New Bottle?”. IT Pro,vol11, pp.15-17, March/April 2009.

[11] F. Agmerich, G. Fenu, S. Surcis. “An Approach to Cloud Computing Network”. In proc. of ICADIWT-2008, pp.113-118, 2008.

[12] I. Foster, Y. Zhao, I. Raicu, S. Lu. “Cloud Computing and Grid Computing 360-Degree Compared”. In Proc. of IEEE Grid Computing Environments Workshop, pp.1-10, 2008- 2009.

[13] A. Croll. “Why Cloud Computing Needs Security”. 2008. Available at http://gigaom.com/2008/06/10/the-amazon-outage-fortresses-in-the-clouds

[14] A. Lenk, M. Klems, J. Nimis, S. Tai, T. Sandholm. “ What’s Inside the Cloud? An Architectural Map of the Cloud Landscape”, CLOUD’09 ISCE Workshop, Canada, May 2009.

[15] J. Brodkin. “Seven Cloud-Computing Security Risks”. 2008. Available at http://www.cio.com/article/423713/Gartner_Seven_Cloud_Computing_Security_Risk

[16] Guidance for Identity & Access Management V2.1, domain 12, Cloud Security Alliance, 2010. Available at http://www.cloudsecurityalliance.org/guidance/csaguide-dom12-v2.10.pdf

[17] P. Arias, F. Almenarez, “Dynamic Trust Relationship Establishment in Federated Identity Management”. Available at http://www.it.uc3m.es/ariasp/tfm_patricia _arias.pdf

[18] S. Boeyen, G. Ellison, et al. “Trust Models Guidelines” sstc-saml-trustmodels-2.0-draft-01, OASIS, 2004.

[19] L. Boursas and H. Reiser. “Propagating Trust and Privacy Aspects in Federated Identity Management Scenarios”. In Proc. of the 14th Annual Workshop of HP Software University Association, Leibniz Supercomputing Center, Munich, Germany, July 2007.

[20] Common Criteria for Information Technology Security Evaluation-Part 1: Introduction and general model, pp.40-43, May 1998.

[21] A. Josang, “The right type of trust for distributed systems”. In C. Meadows, editor, Proc. of the 1996 New Security Paradigms Workshop. ACM, 1996.

[22] D. Harrison McKnight and L. Norman, “The meanings of trust”. Technical report, MISRC Working Papers Series, pp.21-41, 1996.

[23] M. Sloman, “Trust Management in Internet and Pervasive Systems.”IEEE Intelligent Systems, pp.77–79, September 2004.

[24] J. Peng, X. Zhang, Z. Lei, et al. “Comparison of Several Cloud Computing Platforms”. In proc. of Second International Symposium on Information Science and Engineering. China.2009.

[25] D. Orgizovie, B. Svilicic, E. Tijan. “Open Source Science Clouds”. In proc of MIPRO 2010, Opatija, May 24-28, 2010.

[26] R. Buyya, R. Ranjan, R. Calherios. “Modeling and Simulation of Scalable Cloud Computing Environments and the CloudSim Toolkit: Challenges and Opportunities”. Available at arxiv.org/pdf/0907.4878.

[27] A. Zahariev. “Google App Engine”. Seminar on Internetworking. 2009. Availabe at www.cse.tkk.fi/en/publications/B/5/papers/1Zahariev_final.pdf

[28] B. Raghavan, K. Vishwanath, S. Ramabhadran, K.Yocum, A.C. Snoeren. “Cloud control with distributed ratelimiting”. SIGCOMM Computer Communication Review, ACM, pp. 337-348 , October 2007.

[29] B. Behsaz, P. Jaferian, M.R. Meybodi. “Comparison of Global Computing with Grid Computing”. In proc. of International Conference on Parallel and Distributed Computing, Applications and Technologies, pp. 531-534, December 2006.

[30] A. Weiss, “Computing in the Clouds”. networker, ACM, pp.16-25 December 2007.

[31] B. Kandukuri, R. Paturi, A. Rakshit. "Cloud Security Issues". In Proc. of The 2009 IEEE International Conference on Services Computing, pp.517-520, 2009.

[32] M. Morsy, J. Grundy, I. Müller. “An Analysis of The Cloud Computing Security Problem”. In proc. of APSEC 2010 Cloud Workshop, Sydney, Australia, November 2010.

[33] K. Popovic , Z. Hocenski. "Cloud computing security issues and challenges". In Proc. of The Third International Conference on Advances in Human-oriented and Personalized Mechanisms, Technologies, and Services, pp.344-349, 2010.

[34] M. Jensen, J. Schwenk, et. al. "On Technical Security Issues in Cloud Computing." In Proc. of IEEE ICCC, pp.109-116, Bangalore 2009.

[35] B. Grobauer, T. Walloschek, E. Stöcker. "Understanding Cloud-Computing Vulnerabilities". In Proc. of IEEE Security and Privacy, vol. 99, 2010.

List of Publications

1. Aradhana, Inderveer Chana "Implementing Trust Policies for Cloud: A Case Study" published in the proceedings of International Conference on Networks and Computer Communications (ETNCC 2011), Udaipur, 22-24 April, 2011.
2. Aradhana, Inderveer Chana "Trust Based Identity Federation in Cloud" accepted by IEEE Cloud Forum for Practitioners (ICFP 2011), USA, 3-5 July, 2011.
3. Aradhana, Inderveer Chana "Developing Trust Policies for Cloud Scenarios" accepted in 2nd International Conference on Computer and Communication Technology (ICCCT-2011), Allahabad, 15-17 September, 2011.

Appendix A

Installation of Google App Engine for Python

This appendix shows the installation of Google App Engine and Python 2.7.

A.1. Installation of Python 2.7

Python 2.7.msi file can be downloaded from http://www.python.org/getit/. Installation steps are shown in figure A.1.1 to A.1.5

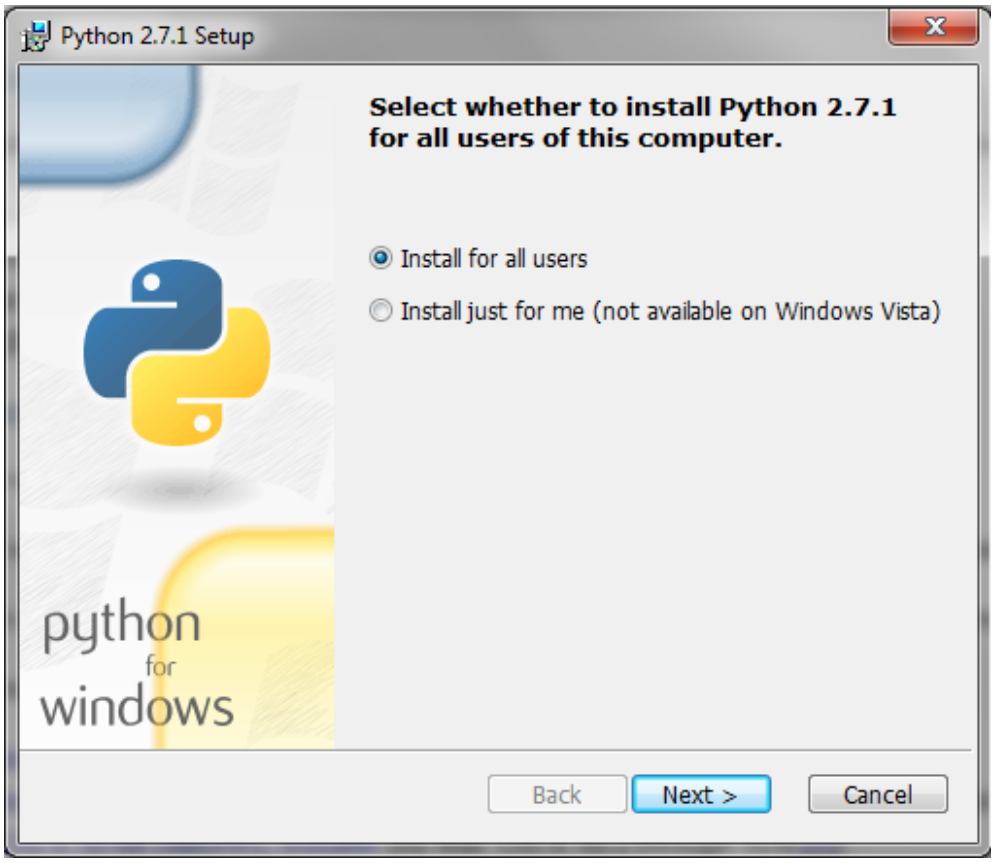

Figure A.1.1 Start page of installation of python 2.7

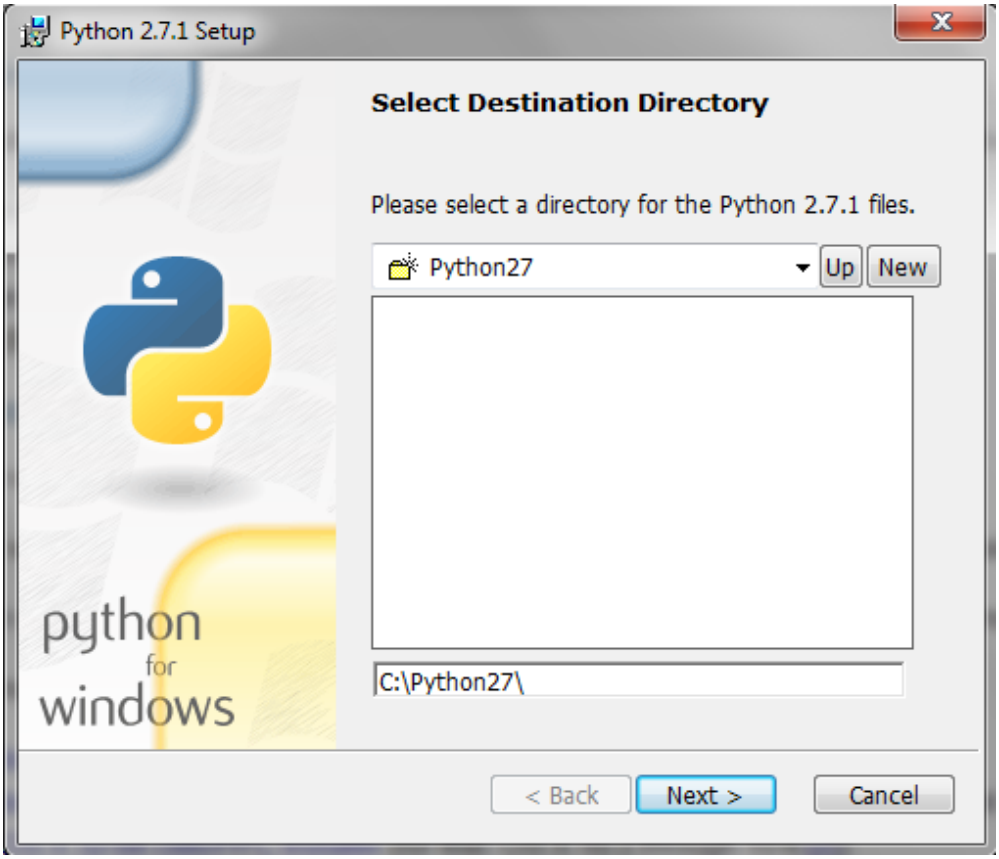

Figure A.1.2 Select destination directory.

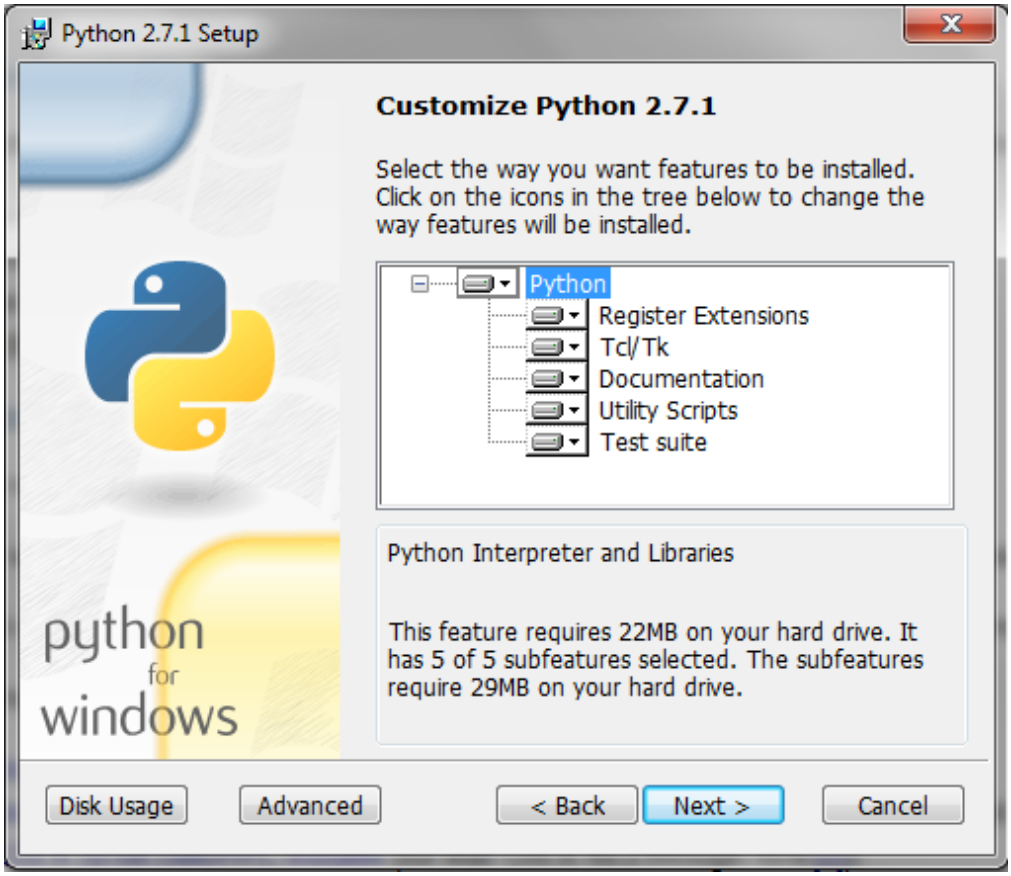

Figure A.1.3 Customize Python 2.7

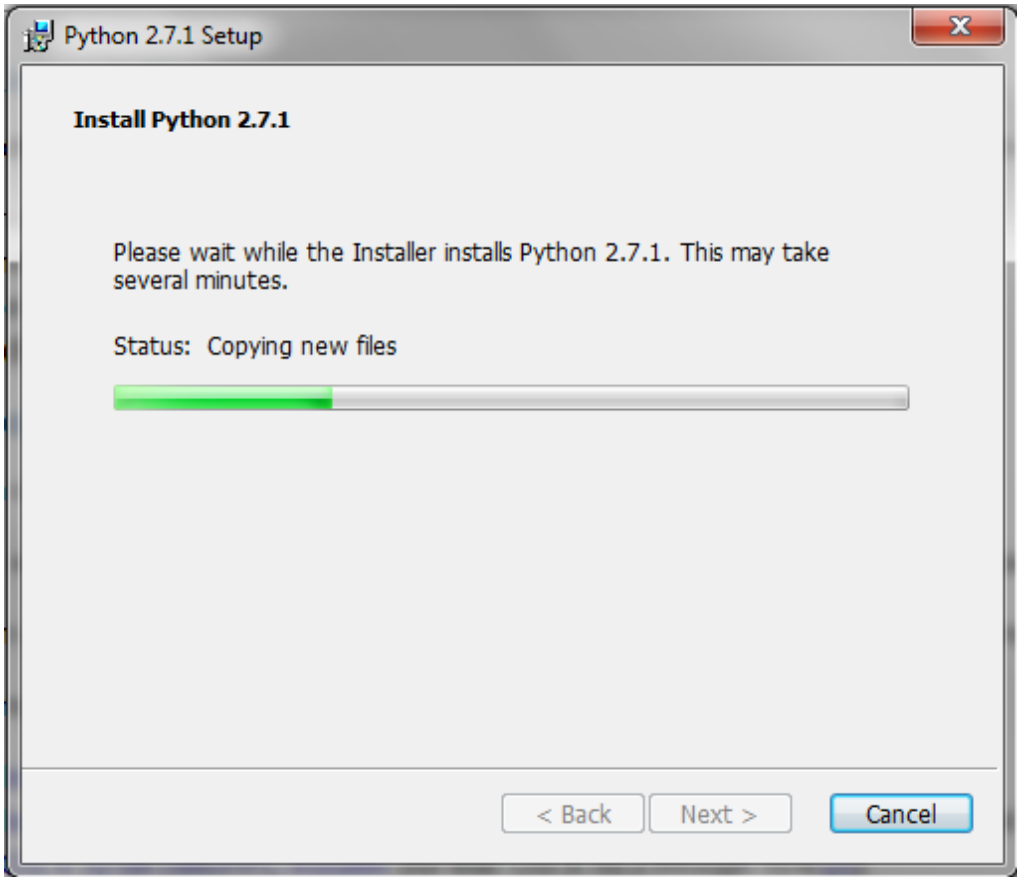

Figure A.1.4 Installing Python

Figure A.1.5 Completing Installation

A.2. Installation of Google App Engine

Google App Engine can be installed by downloading the .msi file of Google App Engine SDK for Python from http://code.google.com/appengine/. Steps for installing the Google App Engine are shown as figure A.2.1 – A.2.6

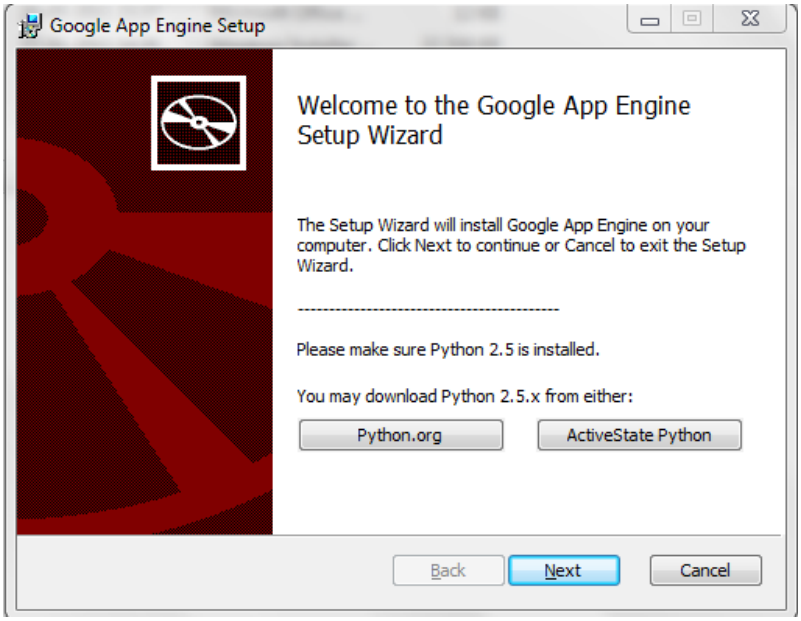

Figure A.2.1 Home page of the Google App Engine setup wizard.

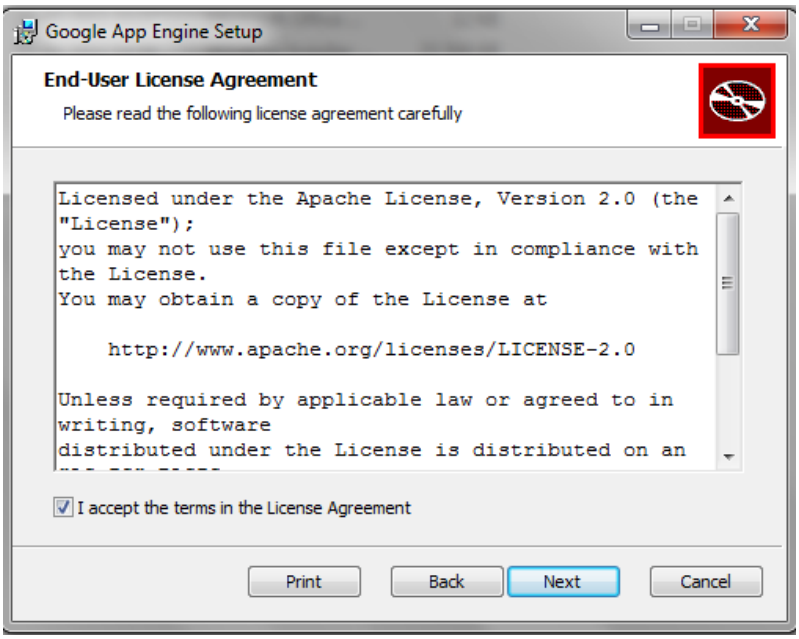

Figure A.2.2 End-user license agreement.

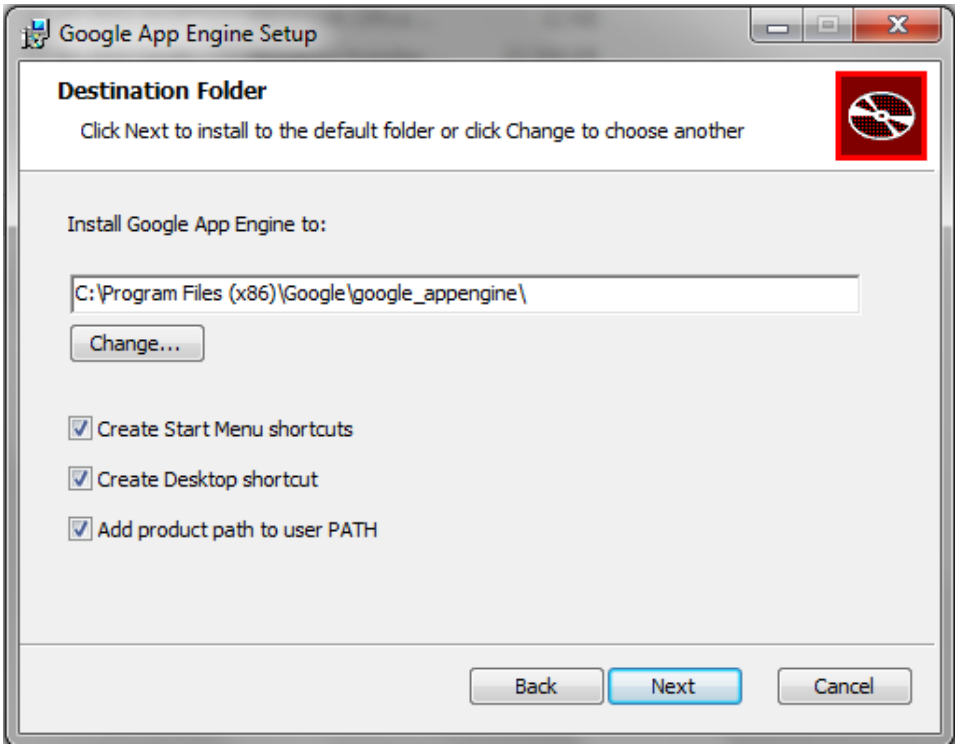

Figure A.2.3 Selection of destination folder.

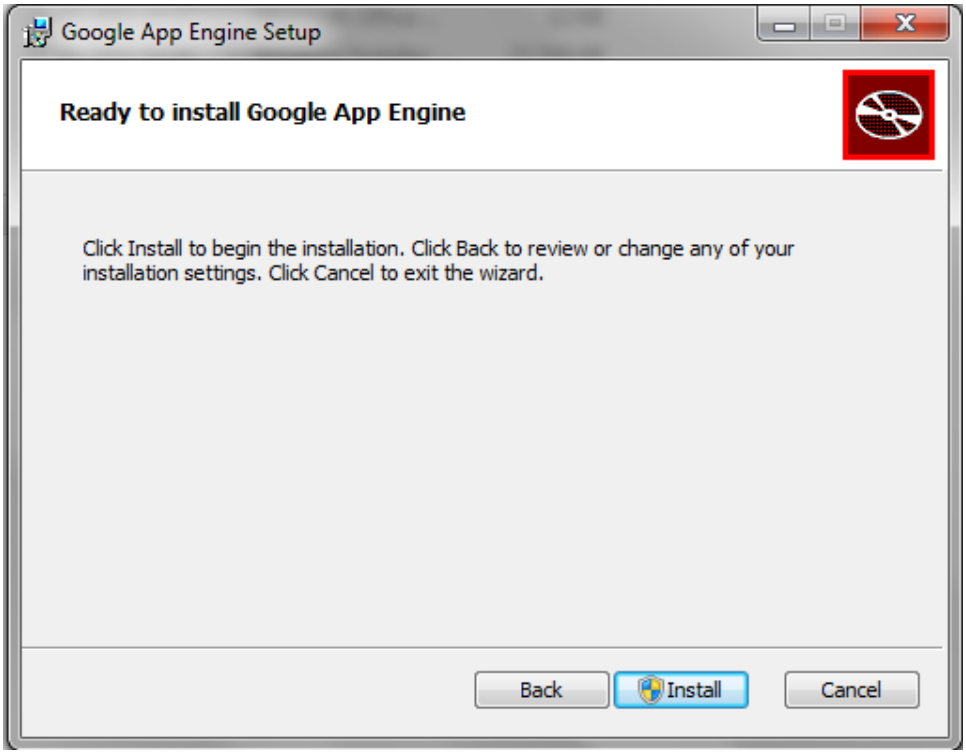

Figure A.2.4 Page to start the installation.

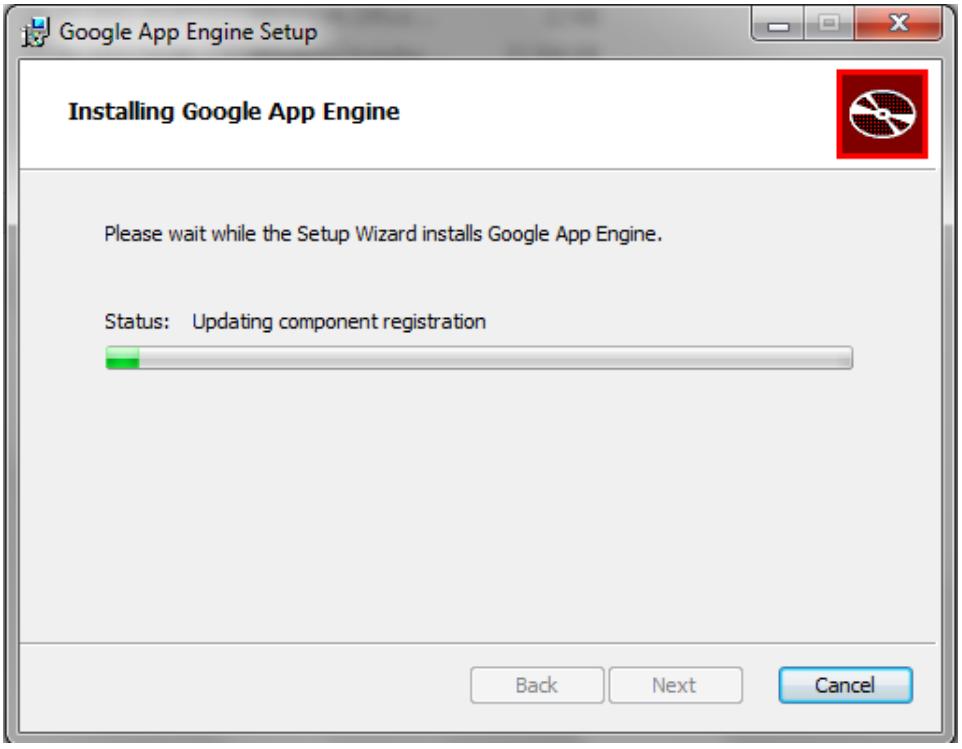

Figure A.2.5 Installing Google App Engine.

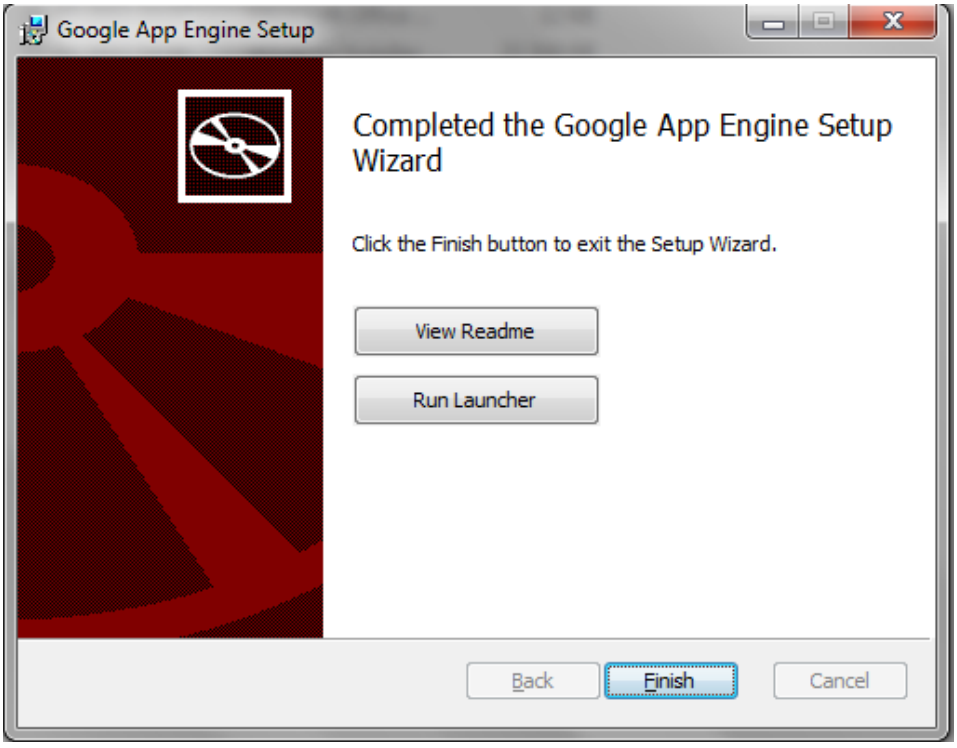

Figure A.2.6 Completed the Google App Engine Setup.