**Abstract**

Cloud computing provides many types of services such as software as a service (SaaS), platform as a service (PaaS), infrastructure as a service (IaaS), etc. This project mainly concentrates on infrastructure as a service in cloud computing. In that the client’s files are programmatically managed through a fixed set of simple system administration commands. Traditionally, if the user or client may delete their data, then it will not be recovered from their storage whatever the time or data. It is a main drawback of IaaS Cloud environment. The proposed system overcome the aforementioned problem by automatic rollback mechanism with the help of state capturing algorithm and check points for handles data for each user in Multi-cloud environment. When the user uses their services provided by cloud service providers such as resource management, memory management, etc. Using this system the user can recover the data, if they delete data by manually or by automatically by Artificial Intelligence (AI) planner. Rollback process is carried out by using manual check point or intermediate check point. Check point process is analyzing the user handing data whenever they used and what type data is used. Through this one can roll back their data whether it is deleted by manually or by automatically.

**CHAPTER I**

**Introduction**

**1.1 Cloud Computing**

The facility to rollback a collection of changes, i.e., reverting to a previous acceptable state, a checkpoint, is widely recognized as valuable support for dependability. This project considers the particular needs of users of cloud computing resources, wishing to manage the resources. Cloud computing provides infrastructure programmatically managed through a fixed set of simple system administration commands. For instance, creating and configuring a virtualized Web server on Amazon Web services (AWS) can be done with a few calls to operations that are offered through the AWS management API. This improves the efficiency of system operations; but having simple powerful system operations may increase the chances of human-induced faults, which play a large role in overall dependability. Catastrophic errors, like deleting a disk volume in a production environment, can happen easily with a few wrong API calls. To support dependability in a cloud platform, it would be helpful if the platform made it easy for a user to rollback to recover from failure. However, the nature of a cloud platform introduces particular difficulties for this approach. The user cannot alter the set of operations provided in a management API, nor can he tailor or even examine its implementation – users have to accept a given API, which is not necessarily designed to support undo. Thus, restoring a previous acceptable state can only be achieved by choosing an appropriate set of API operations, and calling them in a particular order, where constraints between the operation calls are non-obvious and state-specific. In workflow and business process communities, a wide-spread approach to rollback for long-running transactions is Sagas, where system designers provide a compensating action for each operation (the term compensation here differs from its usage in dependability literature). To undo the effects of a sequence of operations, the system executes the corresponding compensating actions in the reverse order. On cloud platforms, this is not always feasible. There are operations for which no compensating operation is provided in the API.

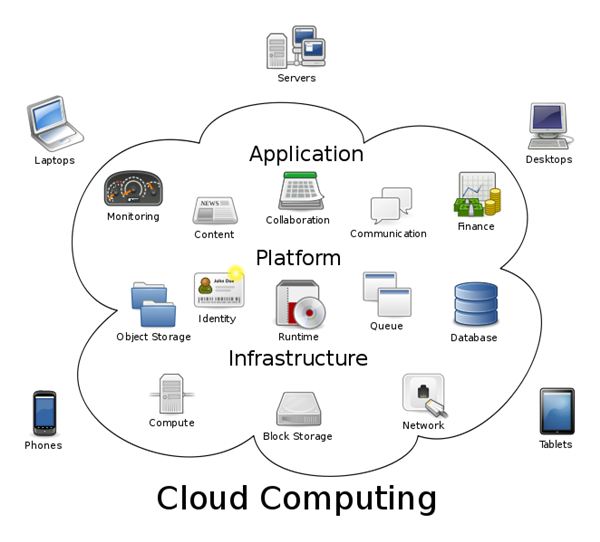
Even when an operation seems like an inverse for another, there may be non-obvious constraints and side-effects, so that executing the apparently compensating operations in reverse chronological order would not restore the previous system state properly, and a different order, or even different operations, is more suitable. Moreover, cloud API operations are often themselves error-prone: we have frequently observed failures or timeouts on most major commercial cloud platforms. The rollback therefore must handle failures that occur during the undo. This may require flexibility and executing alternative operations within the rollback. To improve the dependability of cloud-based systems, we use an AI planner to automate discovering an appropriate sequence of available operations from an API, in order to rollback to a checkpoint. Choosing a sequence of operations is a search in the space of possible solutions; highly optimized heuristics solve common cases of this computationally hard problem in reasonable time. A planner finds a sequence of calls while minimizing their number or cost, using knowledge of current and check pointed states of the system, and a model of operations. Some variants of AI planning also allow for finding alternative sequences when failures occur during the rollback. Our approach requires some changes to non-reversible forward operations such as deleting a disk volume. It also relies on a suitable abstract model of the domain, where each operation has a precise representation in its effects on each aspect of the abstract state. Rollback comes with little overhead, and restores the abstract state of resources, but not the financial charges. This work was inspired by our experience in developing tool support for users of cloud platforms. A technical report contains details omitted here for space limitations.

Cloud computing, also on-demand computing, is a kind of Internet-based computing that provides shared processing resources and data to computers and other devices on demand. It is a model for enabling ubiquitous, on-demand access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services), which can be rapidly provisioned and released with minimal management effort. Cloud computing and storage solutions provide users and enterprises with various capabilities to store and process their data in third-party data centers. It relies on sharing of resources to achieve coherence and economy of scale, similar to a utility (like the electricity grid) over a network.

Advocates claim that cloud computing allows companies to avoid upfront infrastructure costs, and focus on projects that differentiate their businesses instead of on infrastructure. Proponents also claim that cloud computing allows enterprises to get their applications up and running faster, with improved manageability and less maintenance, and enables IT to more rapidly adjust resources to meet fluctuating and unpredictable business demand. Cloud providers typically use a "pay as you go" model. This can lead to unexpectedly high charges if administrators do not adapt to the cloud pricing model.

The present availability of high-capacity networks, low-cost computers and storage devices as well as the widespread adoption of hardware virtualization, service-oriented architecture, and autonomic and utility computing have led to a growth in cloud computing. Companies can scale up as computing needs increase and then scale down again as demands decrease. Cloud computing has become a highly demanded service or utility due to the advantages of high computing power, cheap cost of services, high performance, scalability, accessibility as well as availability. Some cloud vendors are experiencing growth rates of 50% per year, but being still in a stage of infancy, it has pitfalls that need to be addressed to make cloud computing services more reliable and user friendly.

Cloud computing is a kind of grid computing; it has evolved by addressing the qos (quality of service) and reliability problems. Cloud computing provides the tools and technologies to build data/compute intensive parallel applications with much more affordable prices compared to traditional parallel computing techniques.



Cloud computing shares characteristics with:

**Client–server model**

Client–server computing refers broadly to any distributed application that distinguishes between service providers (servers) and service requestors (clients).

**Grid computing**

"A form of distributed and parallel computing, whereby a 'super and virtual computer' is composed of a cluster of networked, loosely coupled computers acting in concert to perform very large tasks."

**Fog computing**

Distributed computing paradigm that provides data, compute, storage and application services closer to client or near-user edge devices, such as network routers. Furthermore, fog computing handles data at the network level, on smart devices and on the end-user client side (e.g. Mobile devices), instead of sending data to a remote location for processing.

**Dew computing**

In the existing computing hierarchy, the Dew computing is positioned as the ground level for the cloud and fog computing paradigms. Compared to fog computing, which supports emerging iot applications that demand real-time and predictable latency and the dynamic network reconfigurability, Dew computing pushes the frontiers to computing applications, data, and low level services away from centralized virtual nodes to the end users.

**Mainframe computer**

Powerful computers used mainly by large organizations for critical applications, typically bulk data processing such as: census; industry and consumer statistics; police and secret intelligence services; enterprise resource planning; and financial transaction processing.

**Utility computing**

The "packaging of computing resources, such as computation and storage, as a metered service similar to a traditional public utility, such as electricity."

**Peer-to-peer**

A distributed architecture without the need for central coordination. Participants are both suppliers and consumers of resources (in contrast to the traditional client–server model).

**1.2 Cloud Computing Models**

Cloud Providers offer services that can be grouped into three categories.

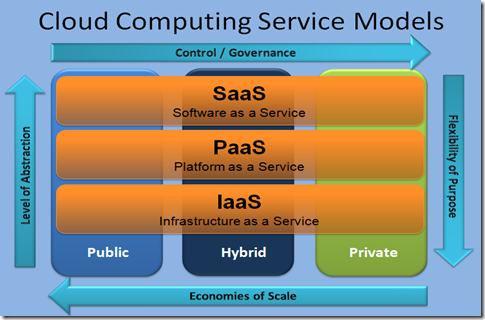
1. **Software as a Service (saas)**

In this model, a complete application is offered to the customer, as a service on demand. A single instance of the service runs on the cloud & multiple end users are serviced. On the customers‟ side, there is no need for upfront investment in servers or software licenses, while for the provider, the costs are lowered, since only a single application needs to be hosted & maintained. Today saas is offered by companies such as Google, Salesforce, Microsoft, Zoho, etc.

1. **Platform as a Service (Paas)**

Here, a layer of software, or development environment is encapsulated & offered as a service, upon which other higher levels of service can be built. The customer has the freedom to build his own applications, which run on the provider‟s infrastructure. To meet manageability and scalability requirements of the applications, paas providers offer a predefined combination of OS and application servers, such as LAMP platform (Linux, Apache, mysql and PHP), restricted J2EE, Ruby etc. Google‟s App Engine, Force.com, etc are some of the popular paas examples.

1. **Infrastructure as a Service (Iaas)**

Iaas provides basic storage and computing capabilities as standardized services over the network. Servers, storage systems, networking equipment, data centre space etc. Are pooled and made available to handle workloads. The customer would typically deploy his own software on the infrastructure. Some common examples are Amazon, gogrid, 3 Tera, etc.

**1.3 Public and Private Clouds**

Enterprises can choose to deploy applications on Public, Private or Hybrid clouds. Cloud Integrators can play a vital part in determining the right cloud path for each organization.

**1.3.1 Public Cloud**

Public clouds are owned and operated by third parties; they deliver superior economies of scale to customers, as the infrastructure costs are spread among a mix of users, giving each individual client an attractive low-cost, “Pay-as-you-go” model. All customers share the same infrastructure pool with limited configuration, security protections, and availability variances. These are managed and supported by the cloud provider. One of the advantages of a Public cloud is that they may be larger than an enterprises cloud, thus providing the ability to scale seamlessly, on demand.

**1.3.2 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.

- 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.

**Hybrid Cloud**

Hybrid Clouds combine both public and private cloud models. With a Hybrid Cloud, service providers can utilize 3rd party Cloud Providers in a full or partial manner thus increasing the flexibility of computing. The Hybrid cloud environment is capable of providing on-demand, externally provisioned scale. The ability to augment a private cloud with the resources of a public cloud can be used to manage any unexpected surges in workload.

**CHAPTER II**

**Literature Survey**

Ingo Weber, Hiroshi Wada, Alan Fekete, Anna Liu, and Len Bass [1] presented Supporting undoability in systems operations. When managing cloud resources, many administrators operate without a safety net. For instance, inadvertently deleting a virtual disk results in the complete loss of the contained data. The facility to undo a collection of changes, reverting to a previous acceptable state, is widely recognized as valuable support for dependability. A positive outcome of this check means a formal guarantee that any sequence of calls to such operations can be undone. A negative outcome contains information on the properties preventing undo ability. At runtime we can then warn the user intending to use an irreversible operation. We demonstrate the feasibility and applicability of the approach with a prototypical implementation and a number of experiments.

Suhrid Satyal, Ingo Weber, Len Bass, Min Fu [2] imparted Scalable Rollback for Cloud Operations using AI Planning. Human-induced faults play a large role in systems reliability. In cloud platforms, system administrators may inadvertently make catastrophic mistakes, like deleting a virtual disk with important data. Providing rollback for cloud operations can reduce the severity and impact of such mistakes by allowing to revert back to a known, good state. In our previous work, we provided a system that augments cloud APIs and provides rollback operation using an AI planner. In this paper, we divide and parallelize rollback plan generation, based on characteristics unique to the rollback scenario. Through experimental evaluation, we show that this approach scales better than the previous, na¨ıve approach, and effectively avoids the exponential behavior.

Mohammad H. Alshayeji, Mohammad Al-Rousan , Eman Yossef, Hanem Ellethy [3] presented A Study on Fault Tolerance Mechanisms in Cloud Computing. Cloud computing is widely popular due to its elasticity, economics, reliability and much more. Cloud computing offers a scalable service without any initial investment in servers, storages, or networks. Fault Tolerance (FT) is the ability of any system to continue performing its function regardless of any unexpected hardware or software failures. Fault Tolerance in Cloud Computing (FTCC) is an important area of research due to its complexity. Based on the study, a comparison on the main fault tolerance techniques is presented considering the cost, overhead, failure types, performance, and the tools used**.** Moreover, we study and compare the models that enhance the performance of checkpoint and replication based techniques.

Ioannis Giannakopoulos, Ioannis Konstantinou, Dimitrios Tsoumakos and Nectarios Koziris[4] imparted Cloud application deployment with transient failure recovery (2018). Application deployment is a crucial operation for modern cloud providers. The ability to dynamically allocate resources and deploy a new application instance based on a user-provided description in a fully automated manner is of great importance for the cloud users as it facilitates the generation of fully reproducible application environments with minimum effort. In order to tackle this challenge, AURA formulates the application deployment as a Directed Acyclic Graph. Whenever a transient failure occurs, it traverses the graph, identifies the parts of it that failed and re-executes the respective scripts, based on the fact that when the transient failure disappears the script execution will succeed. Moreover, in order to guarantee that each script execution is idempotent, AURA adopts a lightweight file system snapshot mechanism that aims at canceling the side effects of the failed scripts. introducing a minimal time overhead, proportional to the failure probability of the deployment scripts.

T. Bylander [5] imparted The computational complexity of propositional strips planning, Artiﬁcial Intelligence, vol. 69, pp. 165–204, 1994. Many operations in cloud APIs have a dual: an operation that apparently does the opposite. In order to make them reversible, we apply the pseudo-delete technique. The undo system offers a wrapper for cloud APIs. After a checkpoint, when the user asks to delete a resource, the wrapper sets a delete flag, indicating that the resource is logically deleted. Subsequent API calls to the resource are altered by the wrapper, e.g., by returning a “not found” error or filtering the resource out from a query result. When a delete operation needs to be reversed, the wrapper simply removes this flag; the user can request this by issuing a rollback. Only when a commit is issued, all resources with a delete flag are physically deleted.

M.Helmert [6] presented The Fast Downward Planning System Fast Downward is a classical planning system based on heuristic search. It can deal with general deterministic planning problems encoded in the propositional fragment of PDDL2.2, including advanced features like ADL conditions and effects and derived predicates (axioms). Like other well-known planners such as HSP and FF, Fast Downward is a progression planner, searching the space of world states of a planning task in the forward direction. However, unlike other PDDL planning systems, Fast Downward does not use the propositional PDDL representation of a planning task directly. Instead, the input is first translated into an alternative representation called multi-valued planning tasks, which makes many of the implicit constraints of a propositional planning task explicit.

Kaustubh R. Joshi, Matti A. Hiltunen, William H. Sanders, Richard D. Schlichting [7] imparted Automatic Model-Driven Recovery in Distributed Systems. Automatic system monitoring and recovery has the potential to provide a low-cost solution for high availability. However, automating recovery is difﬁcult in practice because of the challenge of accurate fault diagnosis in the presence of low coverage, poor localization ability, and false positives that are inherent in many widely used monitoring techniques. In this paper, we present a holistic model-based approach that overcomes these challenges and enables automatic recovery in distributed systems. To do so, it uses theoretically sound techniques including Bayesian estimation and Markov decision theory to provide controllers that choose good, if not optimal, recovery actions according to a user-deﬁned optimization criteria.

C. E. da Silvaand R. de Lemos [8], presented A framework for automatic generation of processes for self-adaptive software systems, Informatica, vol. 35, pp. 3–13, 2011, publisher: Slovenian Society Informatika. The self-adaptation of software systems is a complex process that depends on several factors that can change during the system operational lifetime. Hence, it is necessary to define mechanisms for providing a self-adaptive system the capability of generating during run-time the process that controls its adaptation. This paper presents a framework for the automatic generation of processes for self-adaptive software systems based on the use of workflows, model-based and artificial intelligence planning techniques.

J. Hoffmann, I. Weber, and F. M. Kraft [9], imparted Exploiting a software-engineering model for planning in business process management . Planning is concerned with the automated solution of action sequencing problems described in declarative languages giving the action preconditions and effects. One important application area for such technology is the creation of new processes in Business Process Management (BPM), which is essential in an ever more dynamic business environment. A major obstacle for the application of Planning in this area lies in the modeling. Obtaining a suitable model to plan with – ideally a description in PDDL, the most commonly used planning language – is often prohibitively complicated and/or costly. Our core observation in this work is that this problem can be ameliorated by leveraging synergies with model-based software development.

J. Hoffmann and B. Nebel [10] presented The FF planning system: Fast plan generation through heuristic search. We describe and evaluate the algorithmic techniques that are used in the FF planning system. Like the HSP system, FF relies on forward state space search, using a heuristic that estimates goal distances by ignoring delete lists. Unlike HSP's heuristic, our method does not assume facts to be independent. We introduce a novel search strategy that combines hill-climbing with systematic search, and we show how other powerful heuristic information can be extracted and used to prune the search space. FF was the most successful automatic planner at the recent AIPS-2000 planning competition. We review the results of the competition, give data for other benchmark domains, and investigate the reasons for the runtime performance of FF compared to HSP.

**CHAPTER III**

**SYSTEM ANALYSIS**

**3.1 Existing System:**

* However, deletion operations are not generally reversible, because creation would not revive the deleted resource in its state at the time of deletion.
* It is possible to invoke a “detaching a volume” operation any time, however, doing so could cause a serious failure such as disk inconsistency.
* It does not scale well when the number of operations increases further.

**3.1.1 Limitations:**

* For instance, inadvertently deleting a virtual disk results in the complete loss of the contained data.
* Execution times or cost of operations would be increasing.
* Repeats the edge addition process.

**3.2 Proposed System:**

* The undo system offers a wrapper for cloud APIs checkpoint, when the user asks to delete a resource, the wrapper sets a delete flag, indicating that the resource is logically deleted.
* It propose strategies for dividing and parallelizing rollback plan generation based on the intermediate checkpoints, divide the planning task into smaller, independent tasks, parallelize planning process, and assemble the resulting partial plans into a comprehensive rollback plan.
* One approach to facilitate scalable rollback on an API controlled cloud platform is to use intermediate checkpoints.
* AI planner to discover appropriate sequences of recovery actions for each planning task, and aggregate them into a complete rollback plan.

**Advantages:**

* To measure solution quality over time
* The cluster communication between master and workers is minimal.
* Checkpoint creation time can improve usability of the system.
* ICR approach finds plans more quickly

**Algorithm Description:**

**Algorithm 1: Depth First edge exploration**

Step 1: Fn depth\_first()

Step 2: skip level ← 1;

Step 3: max feasible hops ← number of checkpoints - 3 ;

Step 4: while skip level ≤ max feasible hops do

Step 5: foreach Vertex v in Rollback Graph do

Step 6: candidate vertex ← vertex at skip levelth hop to the left ;

Step 7: add edge from v to candidate vertex;

**Algorithm 2: Breadth First edge exploration**

Step 1: Fn breadth\_first()

Step 2: skip level ← 1;

Step 3: max feasible hops ← number of checkpoints - 3 ;

Step 4: foreach Vertex v in Rollback Graph do

Step 5: add vertices within max feasible hops from v to candidate vertices;

Step 6: foreach Vertex candidate vertex in candidate vertices do

Step 7: add edge from v to candidate vertex;

**Algorithm 3: Divide and Conquer edge exploration Data: MC = manual checkpoint, CC = current checkpoint**

Step 1: Fn divide\_and\_conquer()

Step 2: n ← size(sorted checkpoints);

Step 3: m ← n/2;

Step 4: candidate vertex ← mth elem in sorted checkpoints;

Step 5: add edge from candidate vertex to MC;

Step 6: add edge from CC to candidate vertex;

Step 7: left ← m; right ← m;

Step 8: while true do

Step 9: if lef t ≤ 0 and right ≥ n − 1 then

Step 10: return;

Step 11: lef t ← lef t − 1; right ← right + 1;

Step 12: if left > 0 then

Step 13: add edge from lef tth elem of sorted checkpoints to MC;

Step 14: add edge from CC to lef tth elem of sorted checkpoints;

Step 15: if right < n − 1 then

Step 16: add edge from rightth elem of sorted checkpoints to MC;

Step 17: add edge from CC to rightth elem of sorted checkpoints;

**CHAPTER 4**

**SYSTEM DeSIGN**

**4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modeling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**4.1 System Architecture:**

**Registration**

**Third Party Auditor**

**Check Point**

**Data Storage Analysis**

**Roll back for recovery of Data**

**AI Planning for Data Deletion**

**Client**

**Cloud Service Provider**

**4.2 DATA FLOW DIAGRAM:**

**DFD Level 0:**

User

Login

Registration Registration

Database

**DFD LEVEL-1**

User Login

Store data

File Upload

Database

**DFD Level-2:**

User Login

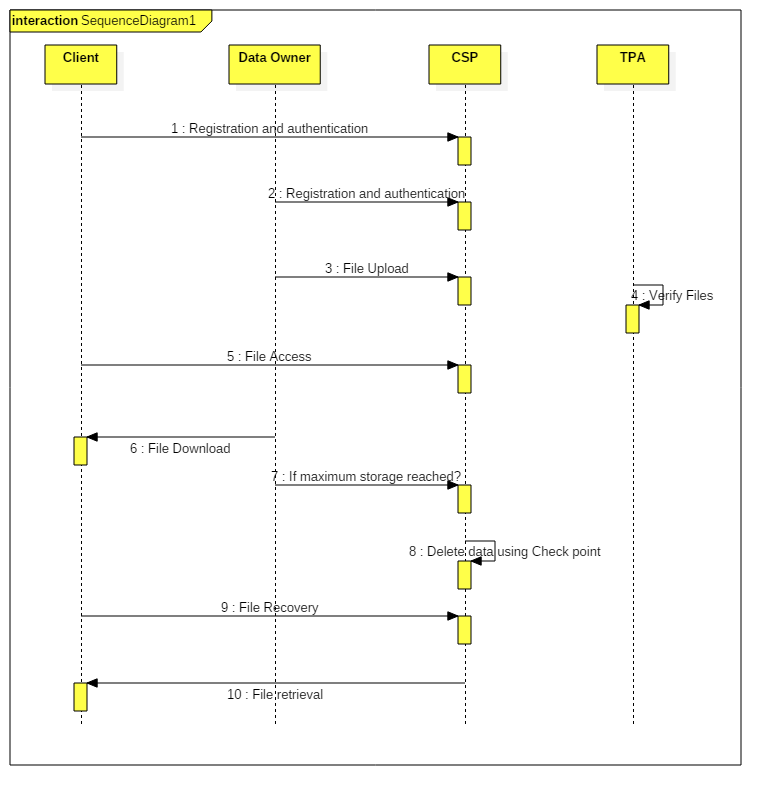
Check Point

AI planning for Rollback Data

Database

**4.3 UML DIAGRAM**

**4.3.1 SEQUENCE DIAGRAM**

****A sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a message sequence chart. Sequence diagrams are sometimes called Event-trace diagrams, event scenarios, and timing diagrams.

**CHAPTER 5**

**SYSTEM SPECIFICATIONS**

**5.1 GENERAL**

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application.

**5.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

* Processor : Intel Core i7-8700k
* Operating system : Windows Family
* RAM : 2 GB.
* Hard disk : 20 GB.
* Monitor : LCD (800x600)
* Mouse : Logitech
* Keyboard : 115 keys(minimum)

**5.3 SOFTWARE REQUIREMENTS**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

* Operating system   : Windows Family
* Front End : Java
* Back end : MySQL

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**6.1 Module List**

* System Administration
* AI Planning
* Checkpoints
  + Manual Checkpoint
  + Intermediate Checkpoint
* Scalable Rollback
* Performance Measurement

**6.2 Module Descriptions**

**System Administration**

When administrators want to have the ability to rollback, they create a checkpoint before making changes to resources on the cloud. The undo system then gathers information about state and relationships of cloud resources, and saves it to persistent storage.

**AI Planning**

AI planner generates a list of rollback actions based on a problem specification and a domain model file. The problem specification is created using two checkpoints: the earlier checkpoint comprises the goal state, and the checkpoint created later comprises the initial state.

**Checkpoints**

A checkpoint is a reference that identifies state of resources on the cloud at a point in time.

**Manual Checkpoint**

A manual checkpoint represents a consistent state to which a system can be rolled back. This is a checkpoint created by the system administrator manually before making changes to resources on cloud. The manual checkpoint information is stored persistently.

**Intermediate Checkpoint**

This is a checkpoint created by the undo system after a certain number of change commands, i.e., commands which change state of the system, have been executed. Intermediate checkpoints are stored persistently. They are used to improve rollback plan generation time.

**Scalable Rollback**

It propose strategies for dividing and parallelizing rollback plan generation: based on the intermediate checkpoints, divide the planning task into smaller, independent tasks, parallelize planning process, and assemble the resulting partial plans into a comprehensive rollback plan.

**Performance Measurement**

That planning tasks are not executed on the application servers or virtual machines that are subject to configuration changes. Therefore, in our experiments, planning is not affected by application workload. We cannot guarantee that similar plan generation times can be achieved if application servers are used.

**CHAPTER 7**

**SOFTWARE SPECIFICATION**

**7.1. GENERAL** This chapter is about the software language and the tools used in the development of the project. The platform used here is J2EE. The Primary languages are JAVA and MYSQL.

**7.2 .THE JAVA PLATFORM**

### 7.2.1 Java

Java acts as the front end, which drives its syntax from C and object-oriented features from C++. The main feature is platform independent. Java is popular among Internet programmers. It expends the universe of objects that can move about freely in cyberspace. Java can be used to create two types of programs, application and applets. An application is a program that runs on the computer, under the operation system of that computer. An applet is a tiny java program, dynamically downloaded across the network.

##### **7.2.2The Byte code**

The output of java compiler is not executable code, it is byte code. It is a set of instructions to be executed by java run-time system called java virtual machine (JVM) it is an interpreter for byte code. Some of the java Buzzwords.

* **Secure**
* **Portable**
* **Object-oriented**
* **Robust**
* **Multithreaded**
* **Architecture-neutral**
* **Interpreted**
* **Distributed**
* **Dynamic**

##### **7.2.3Simple**

It inherits the C syntax and many of the object oriented features of C++. It is the consolidated form of both.

##### **7.2.4 Portable**

Programs to be dynamically downloaded to all various types of platforms connected to the Internet, some means of generating portable executable code is needed.

##### **7.2.5 Secure**

With a help of java-compatible web browser, you can safely download java applets without fear. It is possible with the help of java execution environment and not allowing it access to other parts of computer.

##### **7.2.6 Object-Oriented**

It organizes a program around its data and a set of well-defined interfaces to that data. Some of its concepts are as follows:

* **Abstraction**
* **Encapsulation**
* **Inheritance.**
* **Polymorphism**
* **Exception Handling**

**CHAPTER 8**

**TESTING**

**8.1 GENERAL**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**8.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used.

The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**8.3 TYPES OF TESTING**

**8.3.1. Unit testing**

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

**8.3.2. Functional testing**

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

**8.3.3. System Testing**

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

**8.3.4. Performance Testing**

The Performance test ensures that the output be produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**8.3.5. Integration Testing**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**8.3.6. Acceptance Testing**

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

**Acceptance testing for Data Synchronization:**

* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updating process

**8.3.7. Black Box and White Box Testing**

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

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

**Build the test plan**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**8.3.8. VALIDATION**

At the culmination of the integration testing, Software is completely assembled as a package. Interfacing errors have been uncovered and corrected and a final series of software test begin in validation testing. Validation testing can be defined in many ways, but a simple definition is that the validation succeeds when the software functions in a manner that is expected by the customer. After validation test has been conducted, one of the three possible conditions exists.

1. The function or performance characteristics confirm to specification and are accepted.
2. A deviation from specification is uncovered and a deficiency lists is created.
3. Proposed system under consideration has been tested by using validation test and found to be working satisfactory.

**Require field Validation:** User input the all require field for data validate.

**Compare Validation:** Compare the two fields of the input data.

**Custom Validation:** We can apply here the own validation.

**Regular Expression Validation:**  We can use for control to validate the input class. You can use regular Expression to restrict the range of valid characters, to strip unwanted characters, and to perform length and format checks. We can constrain the input format by defining patterns the input must match.

**CHAPTER 9**

**CONCLUSION AND FUTURE WORK**

**9.1 Conclusion**

It describes our support for undo ability, building on two approaches: (i) an undo ability checker analyses to what degree operations can be rolled back; and (ii) an undo system that automatically generates rollback workflows, when the need arises. The latter can essentially provide transactional atomicity over API controlled environments, like cloud management. It evaluated the approaches through the prototypes we developed, with performance experiments and by applying them to real-world examples of cloud management APIs and best practices. An intrinsic limitation of our approaches is that they operate on a manually created model of the available operations. While it took care to assess that the model truthfully captures the implementation, this cannot be formally guaranteed without access to AWS’s API implementation, deployment, and operation of the cloud platform. Further, the model is not going to be aware of future changes to the API and its implementation.

**9.2 Future Work:**

In future work, we plan to extend the undoability checker with an approach to find projections leading to full undoability automatically, such that the removed properties are minimal. For the undo system, it plan an extension to capture the internal state of resources when checkpointing and to restore the internal state on rollback. For example, the content of a disk volume can be captured by taking a snapshot. Finally, the undo system will be extended to handle multiple checkpoints and manage them by their names, where administrators can then choose to rollback to checkpoint P1 or commit all changes up to checkpoint P2.

**APPENDIX-I**

**SAMPLE CODE**

**//FileUpload.jsp**

<%@page import="java.util.Random"%>

<%@page import="java.util.Calendar"%>

<%@page import="org.apache.commons.fileupload.FileUpload"%>

<%@page import="org.apache.commons.fileupload.FileItem"%>

<%@page import="org.apache.commons.fileupload.DiskFileUpload"%>

<%@page import="java.io.File"%>

<%@page import="java.util.Iterator"%>

<%@page import="java.util.List"%>

<%@ include file="dbconnect.jsp"%>

<%

String uname = (String) session.getAttribute("userid");

String email = (String) session.getAttribute("email");

%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="en-US">

<head profile="http://gmpg.org/xfn/11">

<title>Cloud</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link rel="stylesheet" href="style.css" type="text/css" media="screen" />

<!--[if IE]><link rel="stylesheet" type="text/css" href="style\_ie.css" /><![endif]-->

<!--[if lt IE 7]><link rel="stylesheet" type="text/css" href="style\_ie6.css" /><![endif]-->

<style type="text/css">

<!--

.style1 {color: #FFFFFF}

#Layer1 {

position:absolute;

width:200px;

height:52px;

z-index:1;

left: 63px;

top: 108px;

}

.style2 {

color: #FFFFFF;

font-size: 36px;

font-weight: bold;

}

#Layer2 {

position:absolute;

width:568px;

height:31px;

z-index:1;

left: 265px;

top: 803px;

}

.style13 {color: #FFFFFF; font-weight: bold; font-size: 100%; }

.style3 {

color: #FF0000;

font-weight: bold;

}

.style14 {color: #FFFFFF; font-weight: bold; font-size: 110%; }

-->

</style>

</head>

<%

String msg = "";

String act = request.getParameter("act");

if (act != null) {

try {

String fname = "", filetype = "", description = "", filepath = "", AbsFile = "";

boolean isMultipart = FileUpload.isMultipartContent(request);

if (!isMultipart) {

request.setAttribute("msg", "Request was not multipart!");

request.getRequestDispatcher("Fileupload.jsp").forward(request, response);

return;

}

DiskFileUpload upload = new DiskFileUpload();

List items = upload.parseRequest(request);

Iterator itr = items.iterator();

while (itr.hasNext()) {

FileItem item = (FileItem) itr.next();

if (item.isFormField()) {

String fieldName = item.getFieldName();

if (fieldName.equals("fname")) {

fname = item.getString();

}

if (fieldName.equals("description")) {

description = item.getString();

}

} else {

File f = new File(config.getServletContext().getRealPath("/") + "Files/");

if (f.exists() == false) {

f.mkdir();

}

File ff = new File(config.getServletContext().getRealPath("/") + "Files/" + uname + "\\");

if (!ff.exists()) {

ff.mkdirs();

}

File fullFile = new File(item.getName());

File savedFile = new File(getServletContext().getRealPath("/") + "Files/" + uname + "/", fullFile.getName());

item.write(savedFile);

filepath = request.getContextPath() + "/" + "Files" + "/" + uname + "/" + fullFile.getName().trim();

AbsFile = fullFile.getName();

}

}

String fpath = filepath;

ResultSet rs = st.executeQuery("select \* from filedetails where fname='" + fname + "' and uname='" + uname + "'");

if (rs.next()) {

msg = "This data is Already Available in that Cloud...";

} else {

rs = st.executeQuery("select max(ID) as cnt from serverfiles1");

int id = 0;

if (rs.next()) {

id = rs.getInt("cnt");

}

id = id + 1;

ResultSet rs1 = st.executeQuery("select \* from serverfiles1");

int size = 0;

if (rs1.next()) {

size=rs1.getRow();

}

if (id >= 5) {

%><script language="">alert("Your Request has been sent to Altered Server...");</script><%

Calendar cal = Calendar.getInstance();

String caldate = cal.get(Calendar.DATE) + "-" + cal.get(Calendar.MONTH) + "-" + cal.get(Calendar.YEAR);

rs = st.executeQuery("select max(ID) as cnt from serverfiles2");

id = 0;

if (rs.next()) {

id = rs.getInt("cnt");

}

id = id + 1;

int n = st.executeUpdate("insert into serverfiles2 values('" + uname + "','" + email + "'," + id + ",'" + caldate + "','" + fname + "','" + description + "','" + AbsFile + "','" + filepath + "','')");

if (n == 1) {

st.executeUpdate("insert into convertedfiles values('" + uname + "','" + email + "'," + id + ",'" + caldate + "','" + fname + "','" + description + "','" + AbsFile + "','" + filepath + "')");

msg = "Your Request has been Sent";

}

} else {

Calendar cal = Calendar.getInstance();

String caldate = cal.get(Calendar.DATE) + "-" + cal.get(Calendar.MONTH) + "-" + cal.get(Calendar.YEAR);

int n = st.executeUpdate("insert into serverfiles1 values('" + uname + "','" + email + "'," + id + ",'" + caldate + "','" + fname + "','" + description + "','" + AbsFile + "','" + filepath + "','')");

if (n == 1) {

st.executeUpdate("insert into convertedfiles values('" + uname + "','" + email + "'," + id + ",'" + caldate + "','" + fname + "','" + description + "','" + AbsFile + "','" + filepath + "')");

msg = "Your Request has been Sent";

}

}

}

} catch (Exception ex) {

out.println(ex.getMessage());

}

}

%>

<body>

<div id="page">

<div id="page-top">

<div id="page-bottom">

<div id="header">

<div id="header-cats">

<ul>

<li class="cat-item"><a href="UserPage.jsp">Home Page</a> </li>

<li class="cat-item"><a href="Fileupload.jsp">File Conversion</a> </li>

<li class="cat-item"><a href="Viewfile.jsp">View File</a> </li>

<li class="cat-item"><a href="index.jsp">Logout</a> </li>

</ul>

</div>

<div class="style2" id="Layer1">Rollback Mechanism </div>

</div>

<div id="main">

<p>&nbsp;</p>

<p align="center"><strong>PDF File Conversion Request</strong></p>

<form action="Fileupload.jsp?act=UploadFile" enctype="multipart/form-data" method="post">

<table width="540" border="0" align="center" bgcolor="#336699" class="brdr">

<tr>

<td height="40" class="style14">File Name </td>

<td class="style14"><input name="fname" type="text" class="txt" title="Enter File Name" required="required"/> </td>

</tr>

<tr>

<td width="195" height="47"><span class="style14">Browse File </span></td>

<td width="333" class="style14"><input type="file" name="file" title="Browse File" required="required"/> </td>

</tr>

<tr>

<td height="88"><span class="style14">Description</span></td>

<td class="style14"><textarea name="description" cols="45" rows="4" title="Enter description" required="required"></textarea> </td>

</tr>

<tr>

<td height="65" colspan="2" align="center"><label>

<input name="Submit" type="submit" class="btn" value="File Upolad" />

</label></td>

</tr>

</table>

</form>

<p>&nbsp;</p>

<p align="center" class="style3"><%if (msg != null) {

out.println(msg);

}%></p>

<p align="center" class="style3"></p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p align="center" class="style1">&nbsp;</p>

<div id="top-panel">

<div id="top-panel-top"></div>

</div>

</div>

</div>

</div>

</div>

</body>

</html>

**//FileConversion.jsp**

<%@page import="org.apache.poi.xwpf.converter.pdf.PdfConverter"%>

<%@page import="org.apache.poi.xwpf.converter.pdf.PdfOptions"%>

<%@page import="org.apache.poi.xwpf.usermodel.XWPFDocument"%>

<%@page import="java.io.InputStream"%>

<%@page import="java.io.OutputStream"%>

<%@page import="java.io.FileOutputStream"%>

<%@page import="java.io.FileInputStream"%>

<%@page import="java.io.File"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="en-US">

<head profile="http://gmpg.org/xfn/11">

<title>Cloud</title>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

<link rel="stylesheet" href="style.css" type="text/css" media="screen" />

<!--[if IE]><link rel="stylesheet" type="text/css" href="style\_ie.css" /><![endif]-->

<!--[if lt IE 7]><link rel="stylesheet" type="text/css" href="style\_ie6.css" /><![endif]-->

<style type="text/css">

<!--

.style1 {color: #FFFFFF}

#Layer1 {

position:absolute;

width:200px;

height:52px;

z-index:1;

left: 63px;

top: 108px;

}

.style2 {

color: #FFFFFF;

font-size: 36px;

font-weight: bold;

}

#Layer2 {

position:absolute;

width:568px;

height:31px;

z-index:1;

left: 265px;

top: 803px;

}

.style13 {color: #FFFFFF; font-weight: bold; font-size: 110%; }

.style3 {

color: #FF0000;

font-weight: bold;

}

-->

</style>

</head>

<%

String uname="",fname = "", absfile = "", fpath = "";

uname = request.getParameter("uname");

fname = request.getParameter("fname");

absfile = request.getParameter("absfile");

fpath = request.getParameter("fpath");

%>

<%@include file="dbconnect.jsp"%>

<%

String b1, msg = null;

b1 = request.getParameter("Submit");

if (b1 != null) {

try {

//long start = System.currentTimeMillis();

// 1) Load DOCX into XWPFDocument

InputStream is = new FileInputStream(new File(getServletContext().getRealPath("/")+ "Files\\" +uname+ "\\"+absfile));

out.println(new File(getServletContext().getRealPath("/")+ "Files\\" +uname+ "\\"+absfile));

String filesplit[]=absfile.split(".");

out.println("klsadlkasm"+filesplit[0]);

XWPFDocument document = new XWPFDocument(is);

// 2) Prepare Pdf options

PdfOptions options = PdfOptions.create();

// 3) Convert XWPFDocument to Pdf

OutputStream out1 = new FileOutputStream(new File(getServletContext().getRealPath("/")+ "Files\\" +uname+ "\\"+absfile));

PdfConverter.getInstance().convert(document, out1, options);

//System.out.println("Generate pdf/Helloworld.pdf with "

//+ (System.currentTimeMillis() - start) + "ms");

} catch (Throwable e) {

e.printStackTrace();

}

}

%>

<body>

<div id="page">

<div id="page-top">

<div id="page-bottom">

<div id="header">

<div id="header-cats">

<ul>

<li class="cat-item"><a href="FileConversion.jsp">Close</a> </li>

</ul>

</div>

<div class="style2" id="Layer1">Rollback Mechanism</div>

</div>

<div id="main">

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p align="center"><strong>PDF File Conversion</strong></p>

<form method="post" action="">

<table width="386" border="0" align="center" bgcolor="#336699" class="brdr">

<tr>

<td width="127" height="52"><span class="style13">File Name </span></td>

<td width="215"><span class="style13"><%=fname%> </span></td>

</tr>

<tr>

<td height="60"><span class="style13">Document Name</span></td>

<td><span class="style13"><%=absfile%></span></td>

</tr>

<tr>

<td height="60" colspan="2" align="center"><label>

<input name="Submit" type="submit" class="btn" value="Convert" />

</label></td>

</tr>

</table>

</form>

<p align="center" class="style3"></p>

<p align="center" class="style3"><%if (msg != null) {

out.println(msg);

}%></p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p align="center" class="style1">&nbsp;</p>

</div>

</div>

</div>

</div>

</body>

</html>

**//Rollback.jsp**

<%@page import="java.sql.ResultSet"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>CloudSim-Server</title>

<meta name="keywords" content="free web template, business website, CSS, HTML" />

<meta name="description" content="free CSS HTML template for professional business websites" />

<link href="style.css" rel="stylesheet" type="text/css" />

</head>

<%@include file="dbconnect.jsp" %>

<body>

<div id="templatemo\_container">

<div id="templatemo\_header"><div id="templatemo\_logo"><img src="images/logo.gif" alt="Logo" />

<div id="templatemo\_sitetitle">Server</div>

</div>

</div>

<div id="templatemo\_menu">

<ul>

<li><a href="AdminHome.jsp" class="current">Home</a></li>

<li><a href="Rollback.jsp">Rollback</a></li>

<li><a href="index.jsp">Logout</a></li>

</ul>

</div>

<div id="templatemo\_banner">

<h1>&nbsp;</h1>

</div>

<p>&nbsp;</p>

<p><strong><center>

Rollback Mechanism

</center></strong></p>

<form method="post" id="form1">

<p>&nbsp;</p>

<p>&nbsp;</p>

<table width="553" border="0" align="center" bgcolor="#66CCFF">

<tr>

<td width="183" height="34"><strong>Enter Date </strong></td>

<td width="229"><label>

<input name="txt1" type="text" class="txt" />

</label></td>

<td width="127"><label>

<input name="Submit" type="submit" class="btn" value="Submit" />

</label></td>

</tr>

</table>

<p>&nbsp;</p>

<% String act = request.getParameter("Submit");

if (act != null) {

String cdate = request.getParameter("txt1");

%>

<table width="847" border="0" align="center" bgcolor="#66CCFF">

<tr>

<td width="39" height="40"><strong>S.No</strong></td>

<td width="96"><strong>User Name</strong></td>

<td width="203"><strong>File Name </strong></td>

<td width="276"><strong>Description </strong></td>

<td width="211"><strong>Action</strong></td>

</tr>

<%

int id = 0;

ResultSet rs = st.executeQuery("select \* from convertedfiles where caldate='"+cdate+"'");

while (rs.next()) {

id += 1;

%>

<tr>

<td height="38" bgcolor="#FFFFFF"><strong><%=id%></strong></td>

<td bgcolor="#FFFFFF"><strong><%=rs.getString("uname")%></strong></td>

<td bgcolor="#FFFFFF"><strong><%=rs.getString("fname")%></strong></td>

<td bgcolor="#FFFFFF"><strong><%=rs.getString("description")%></strong></td>

<td align="center" bgcolor="#FFFFFF"><a href="<%=rs.getString("filepath")%>">View </a> </td>

<%

}

}%>

</tr>

</table>

</form>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<p>&nbsp;</p>

<div id="templatemo\_light\_blue\_row"></div>

</div>

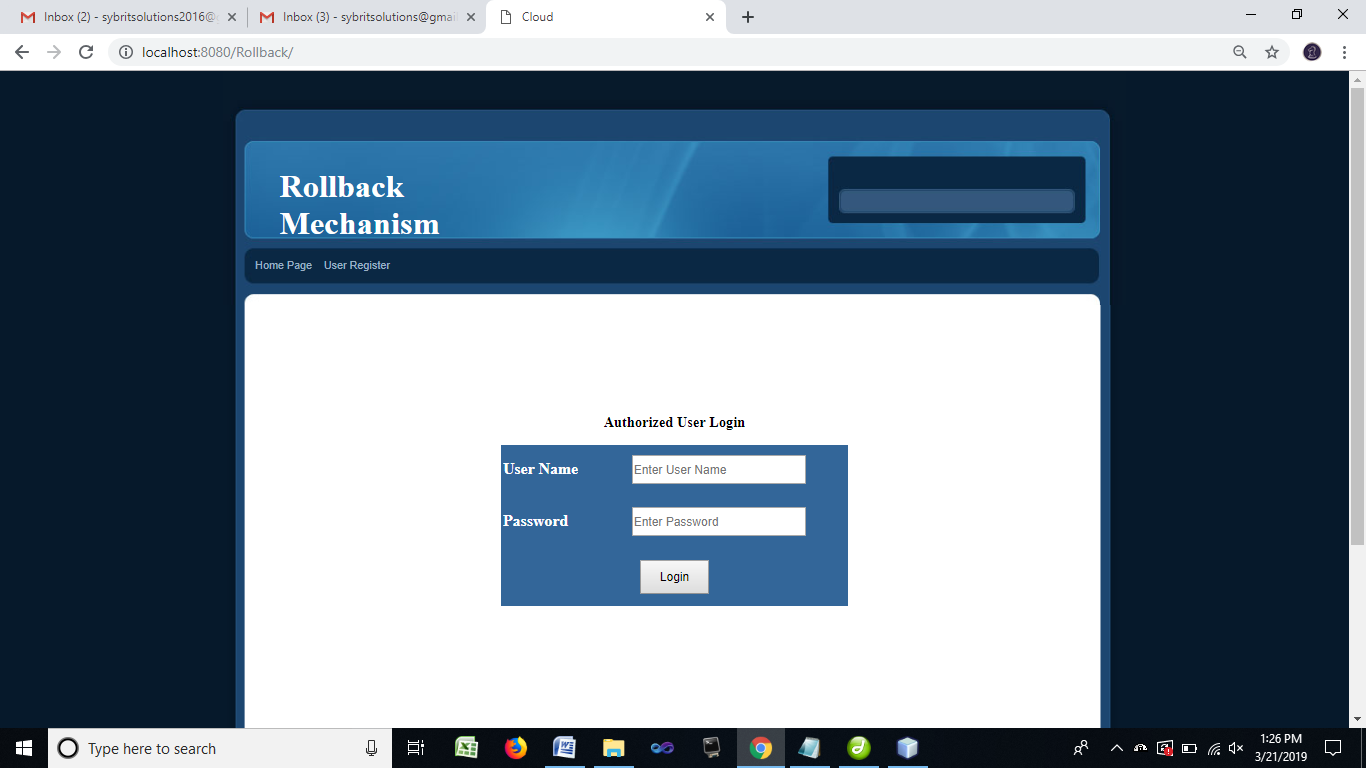
<div align=center></div>

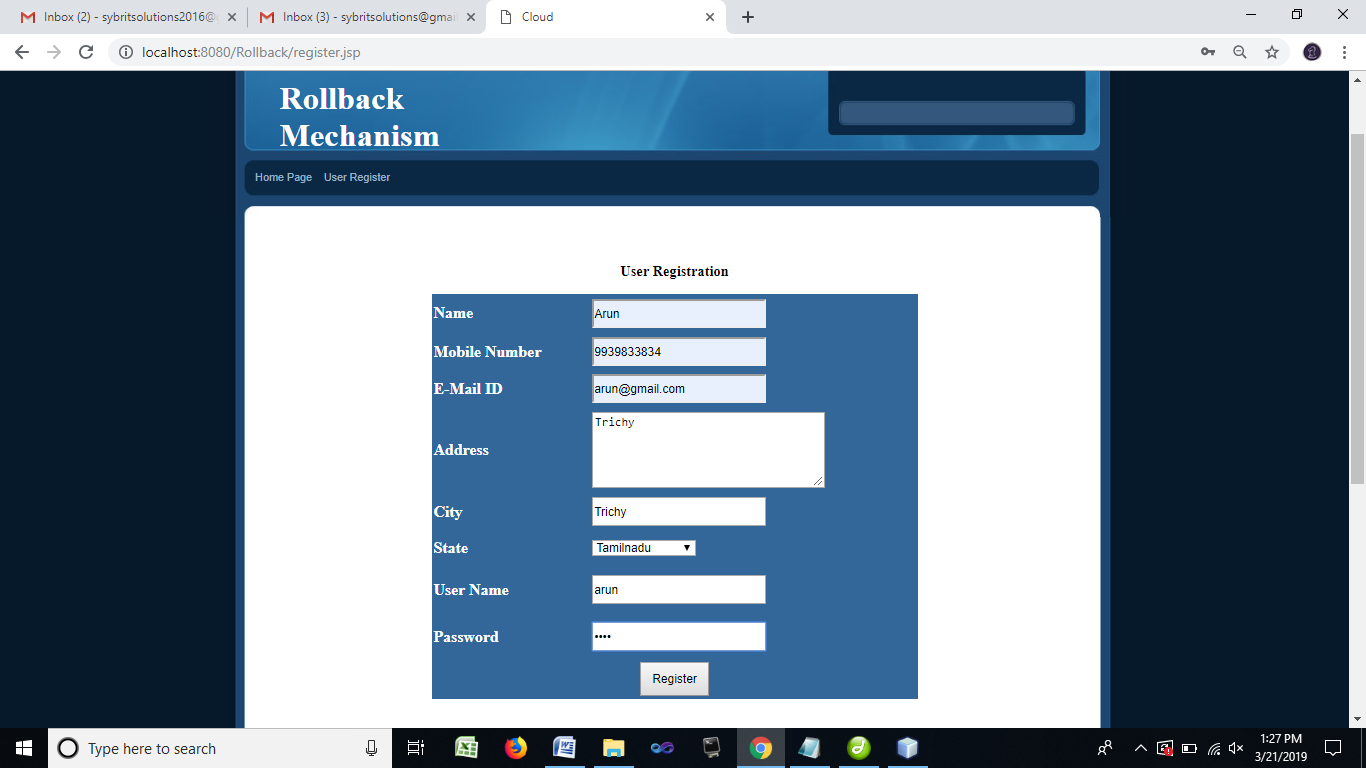
</body>

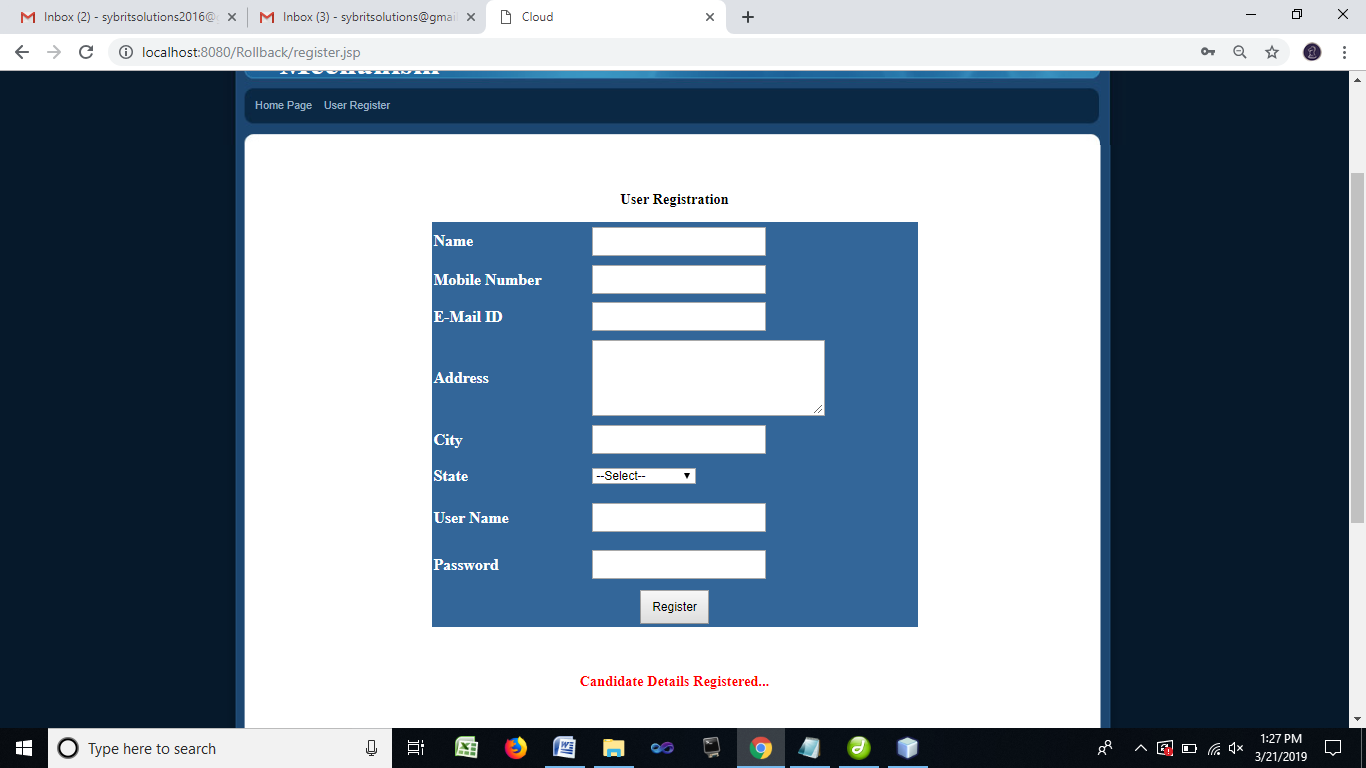
</html>

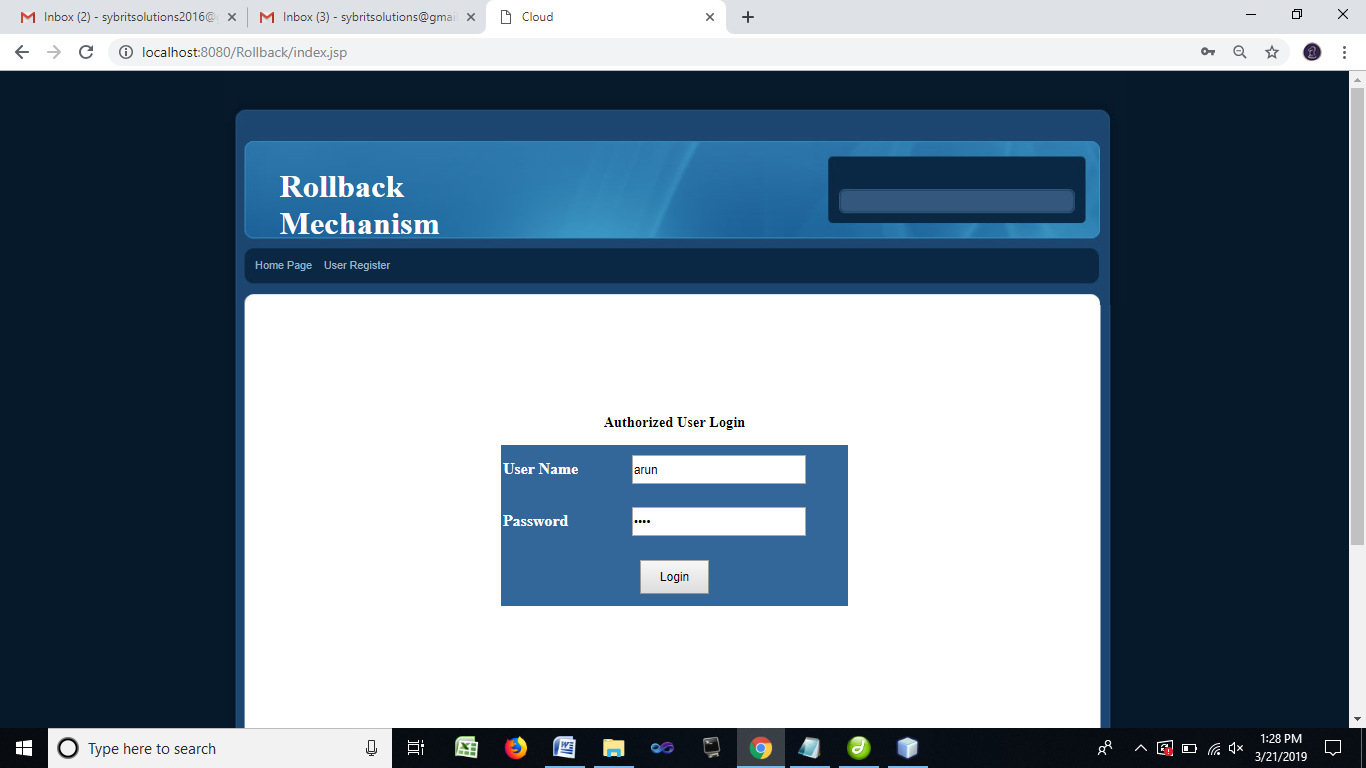
**APPENDIX-II**

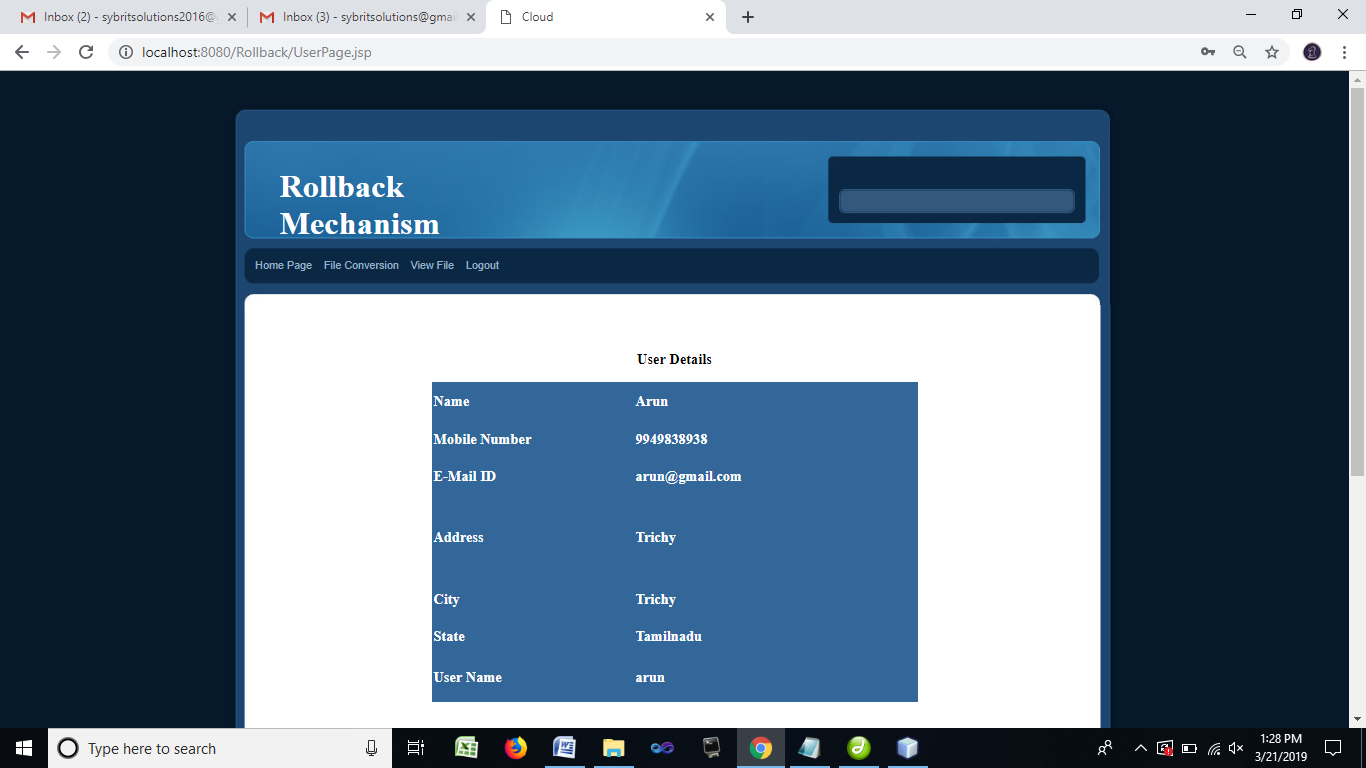
**SCREENSHOTS:**

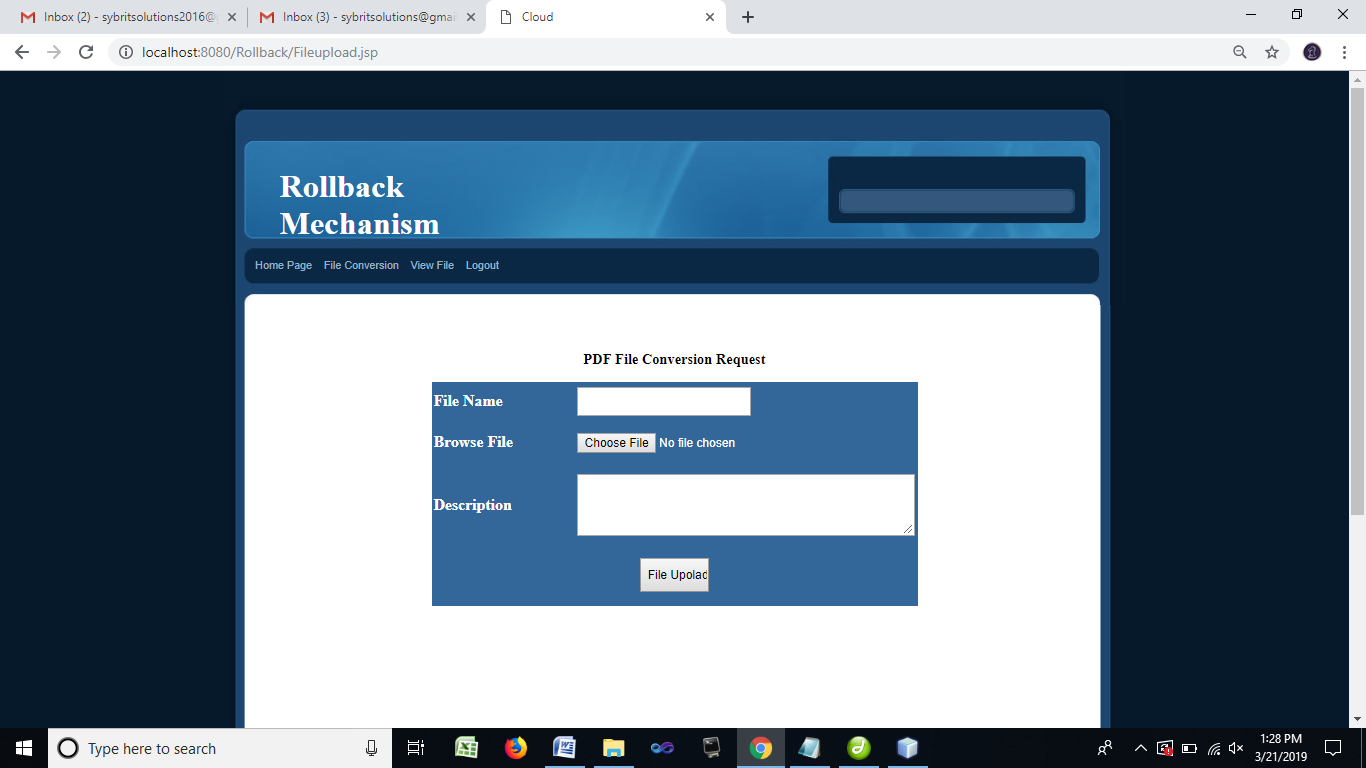


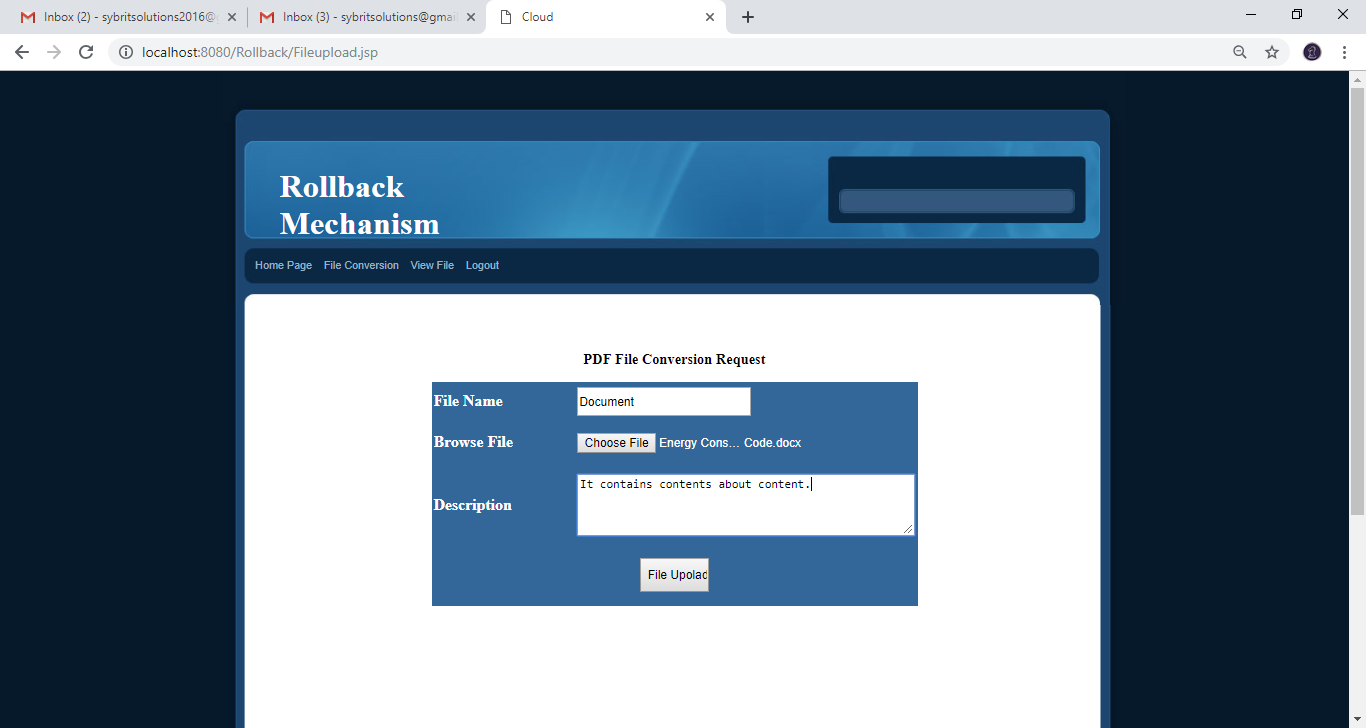


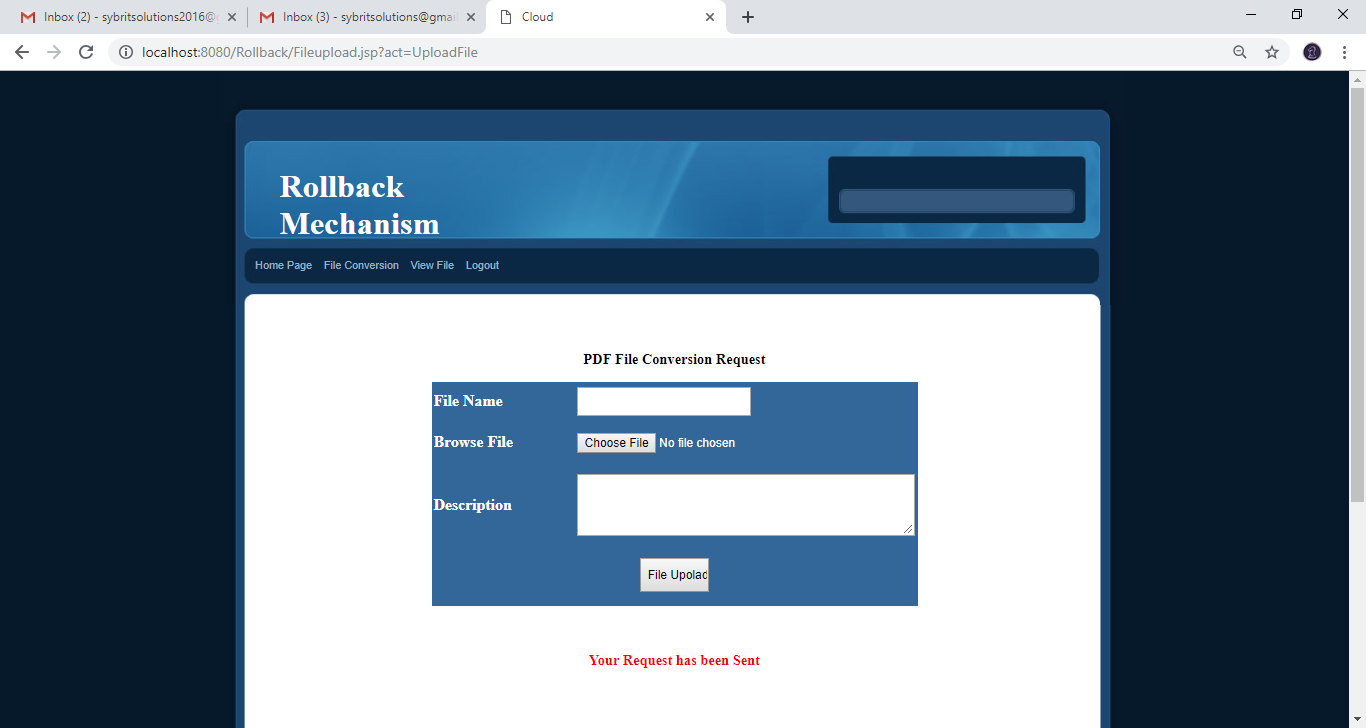


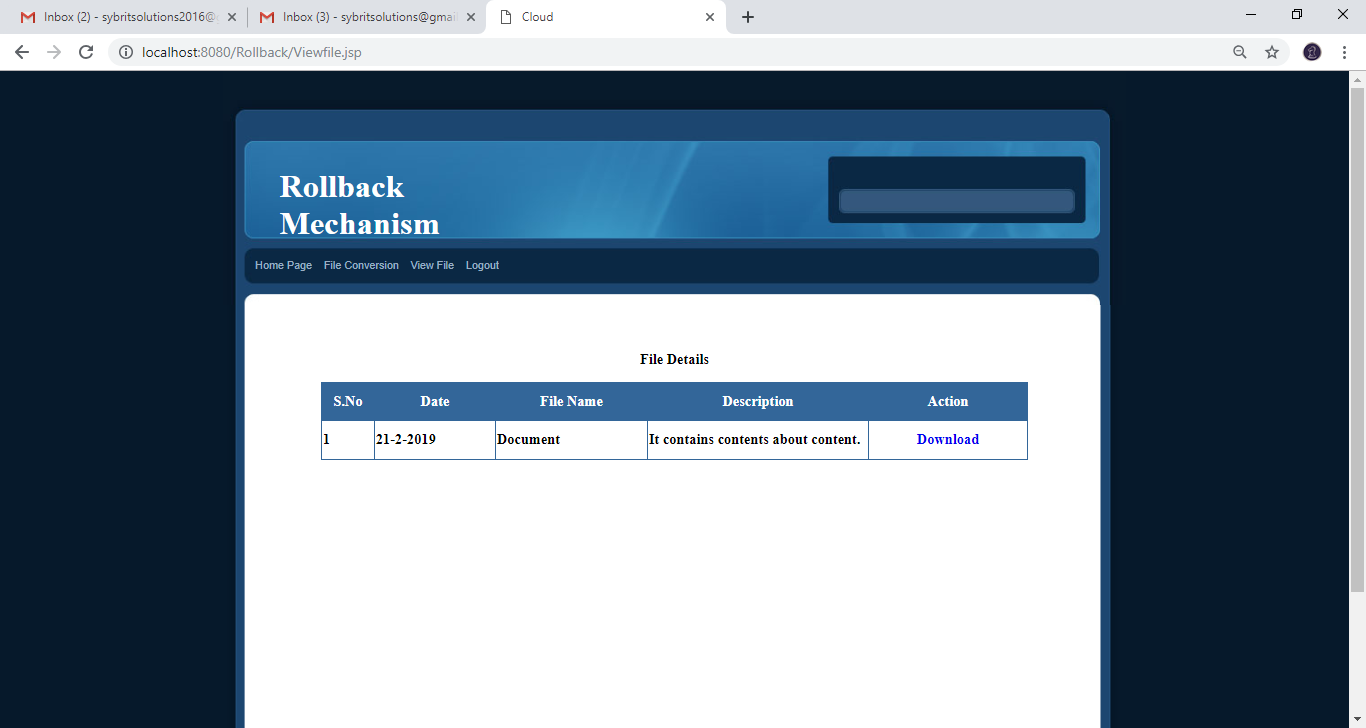


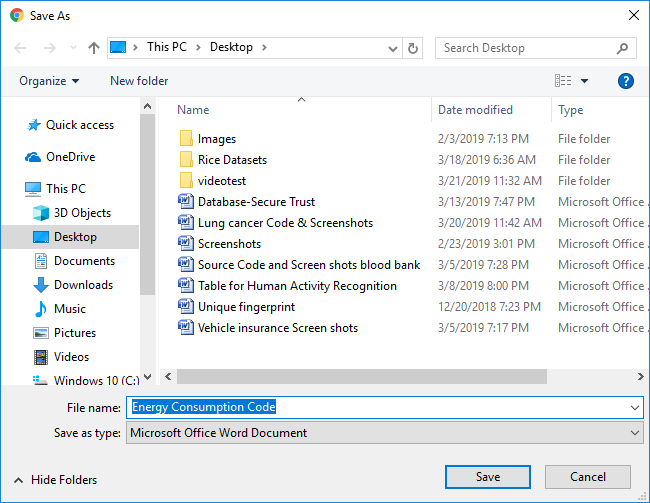


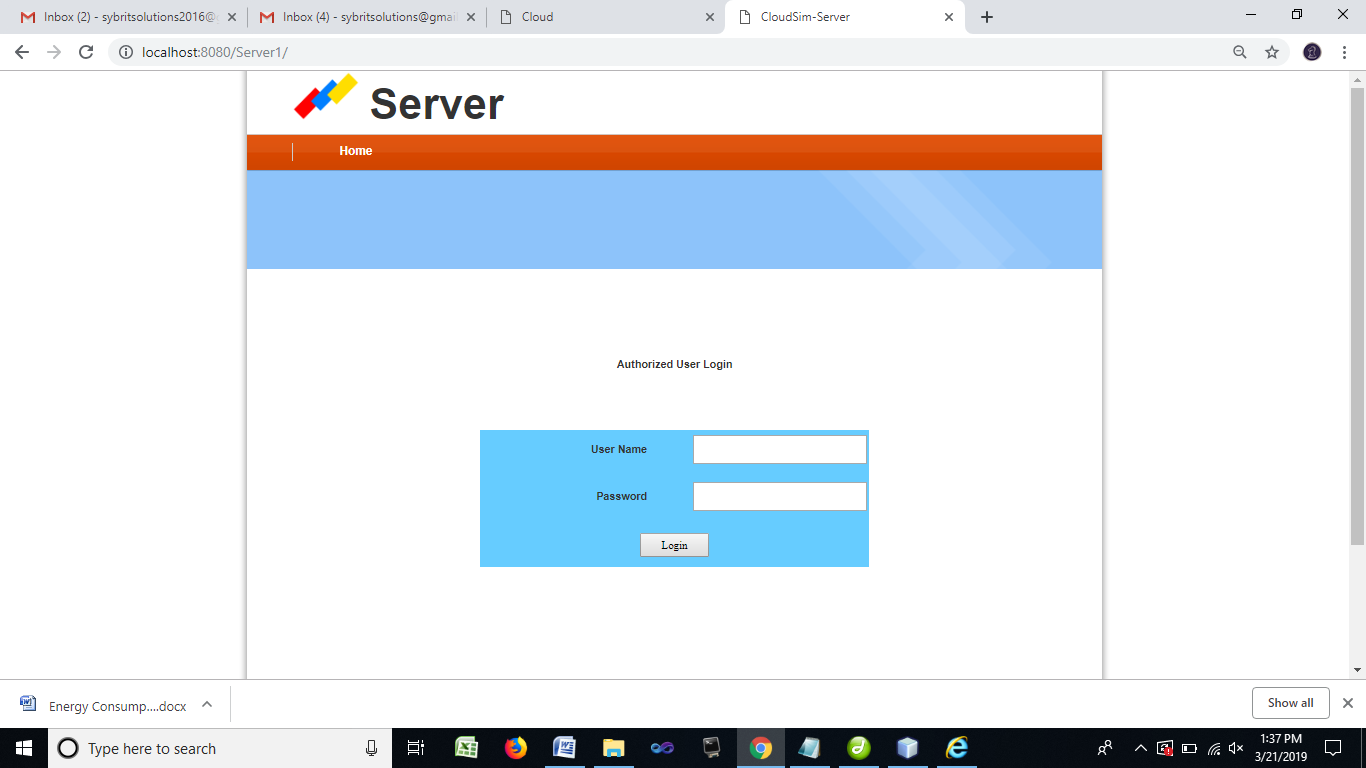


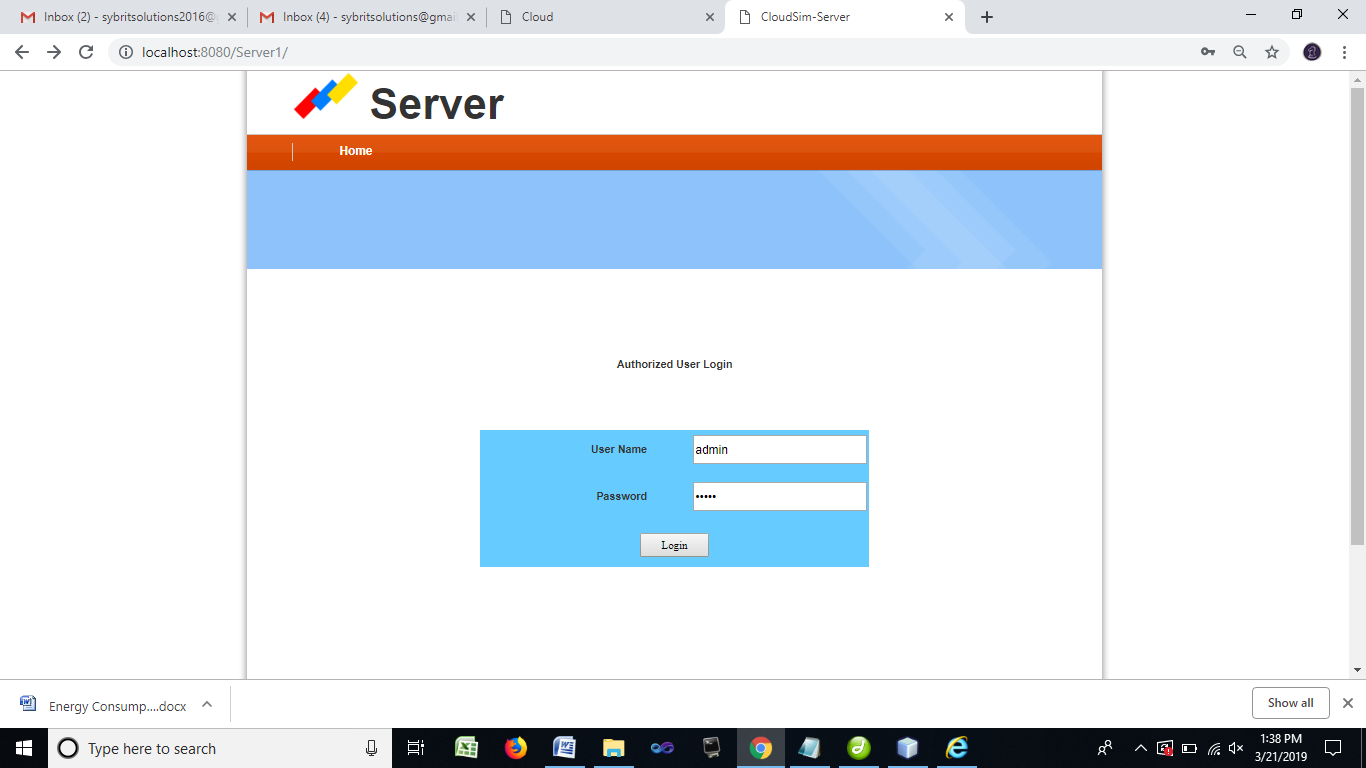


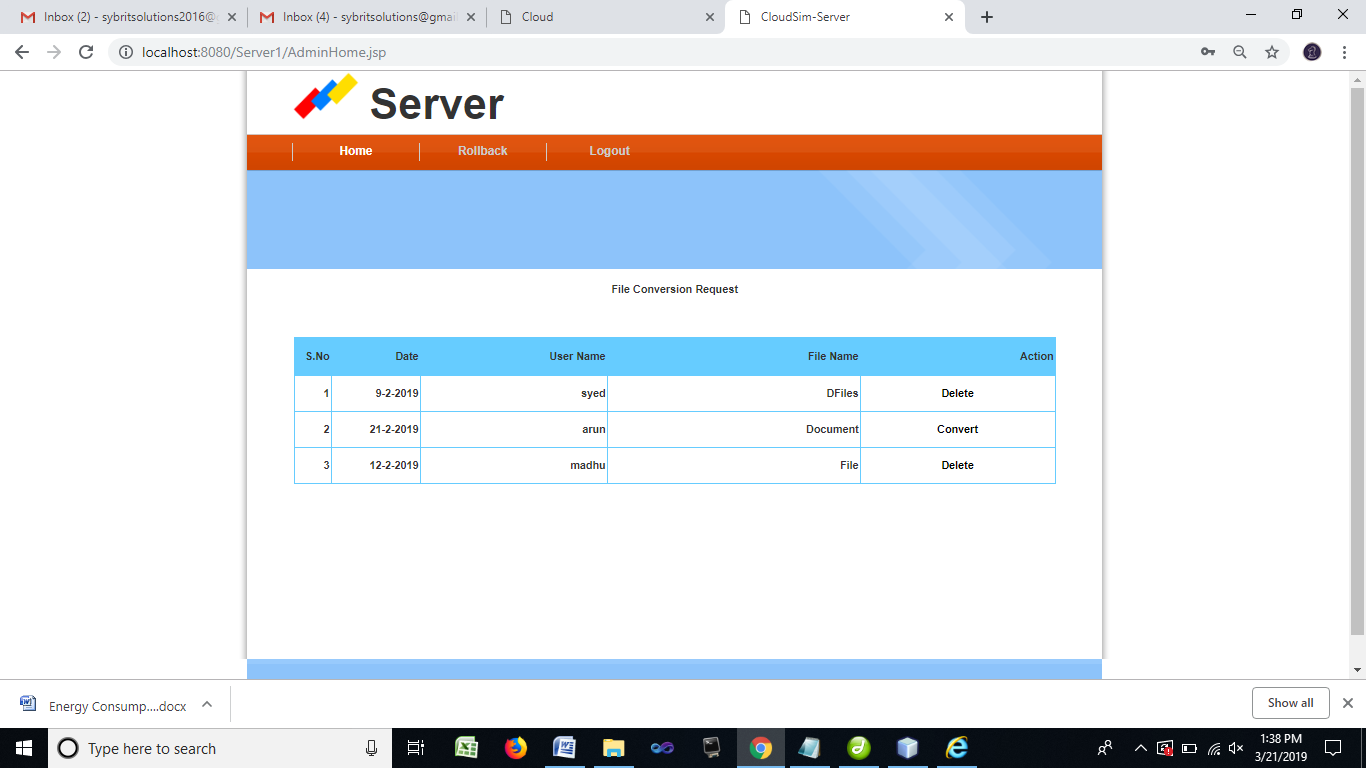


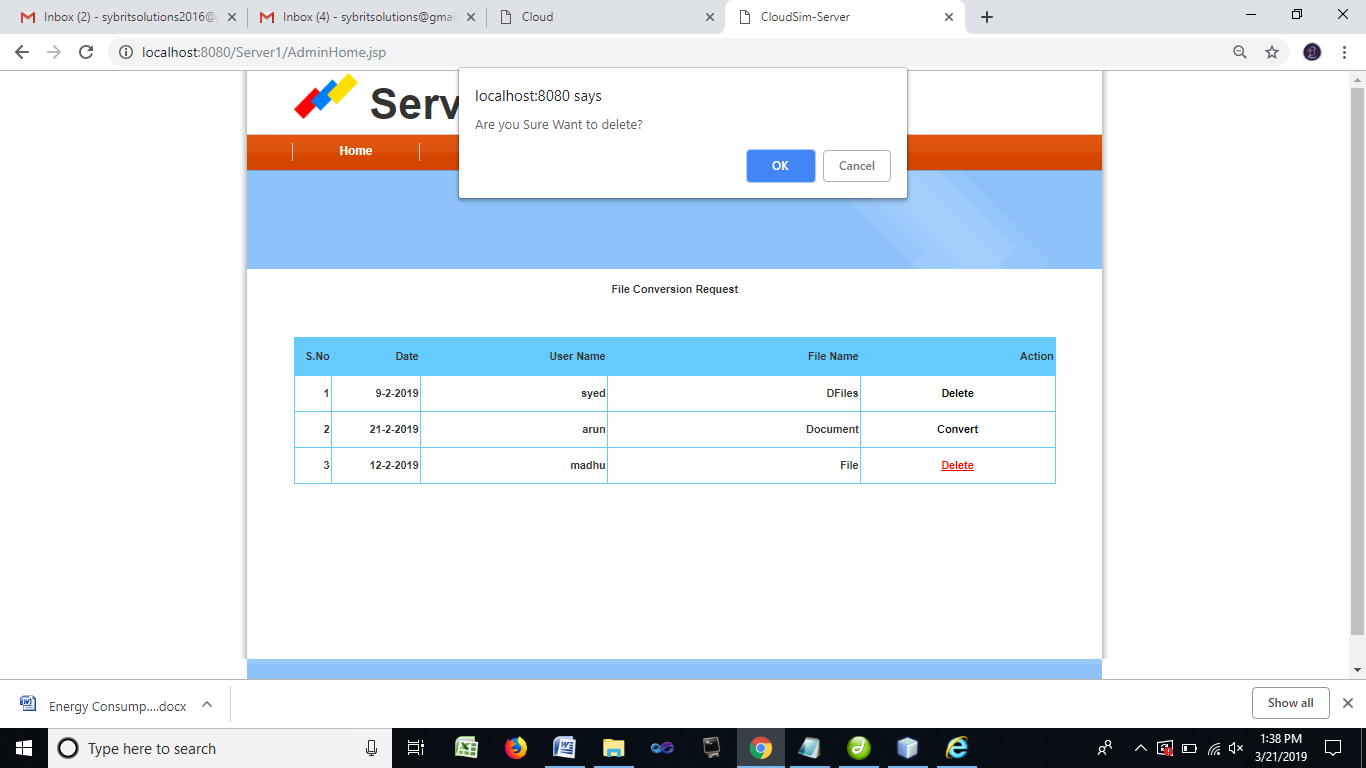


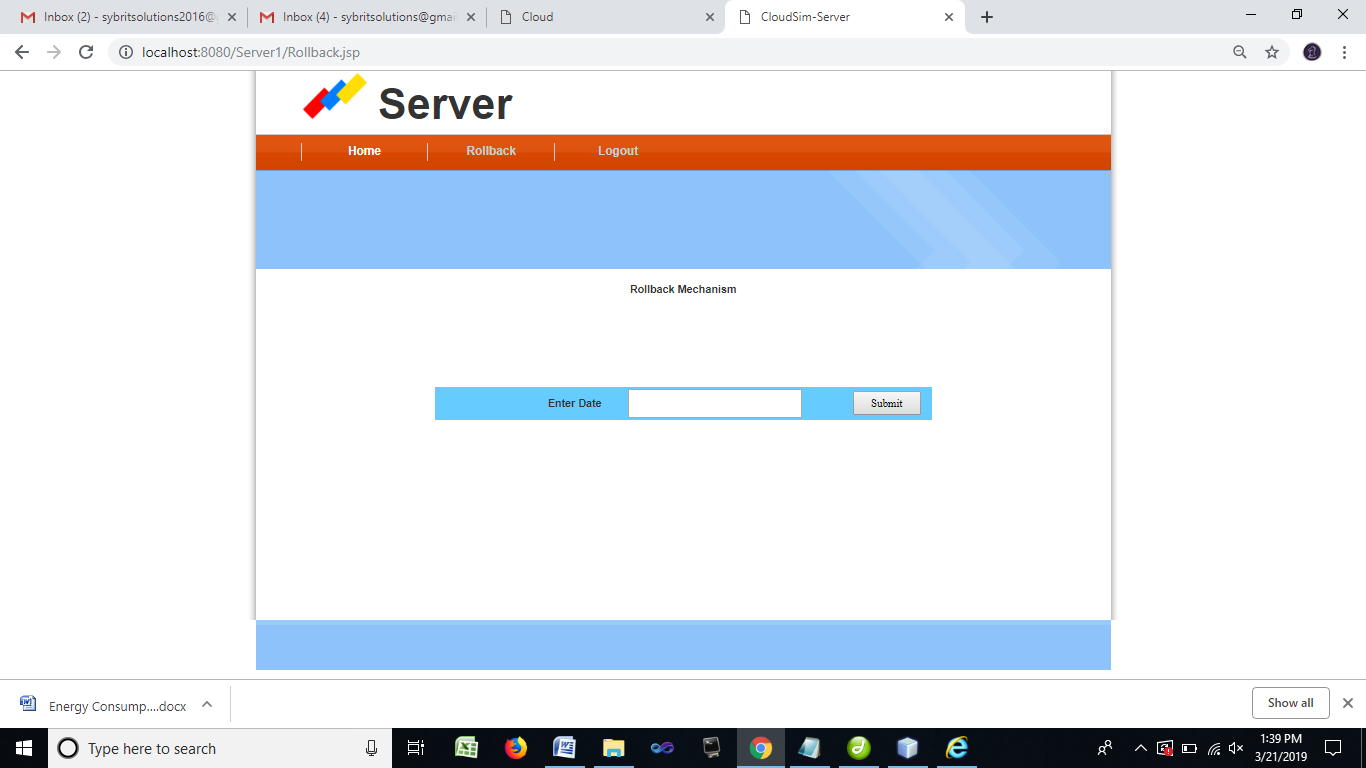


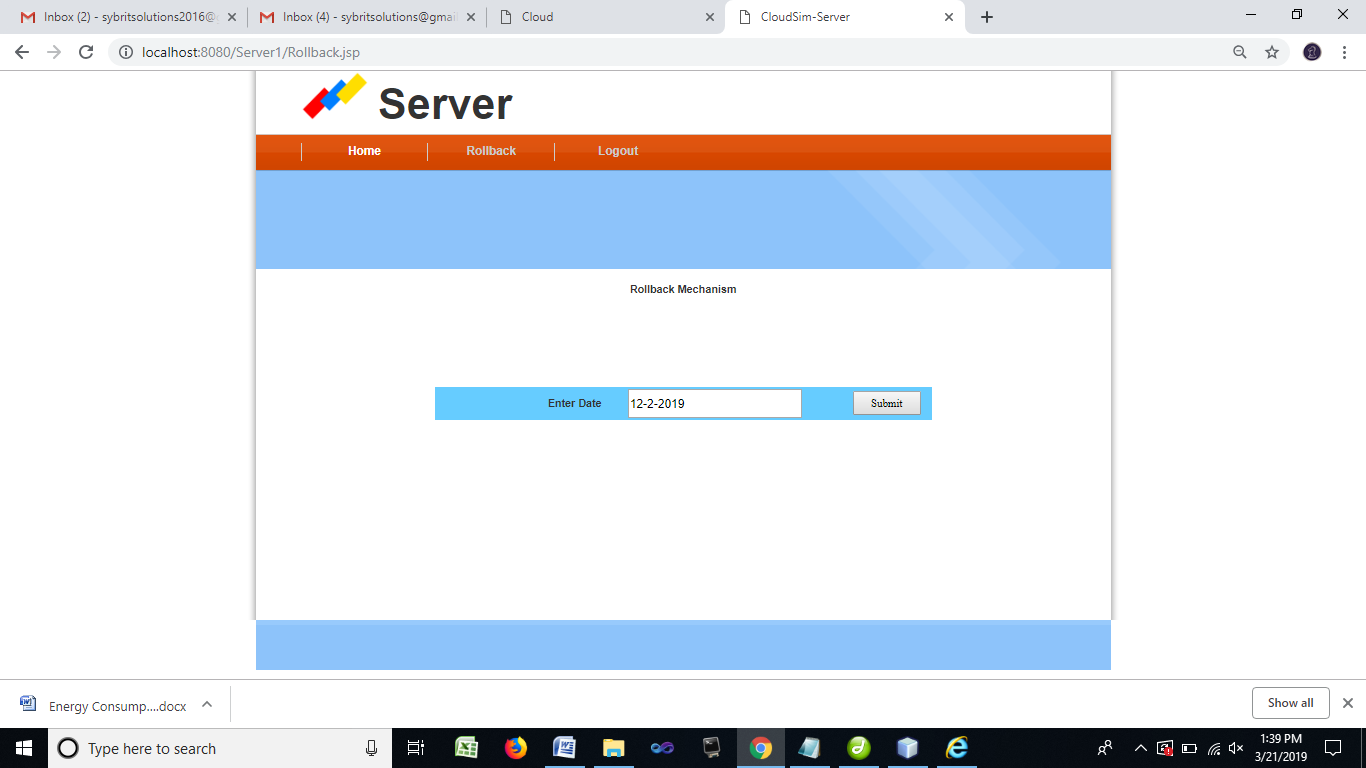


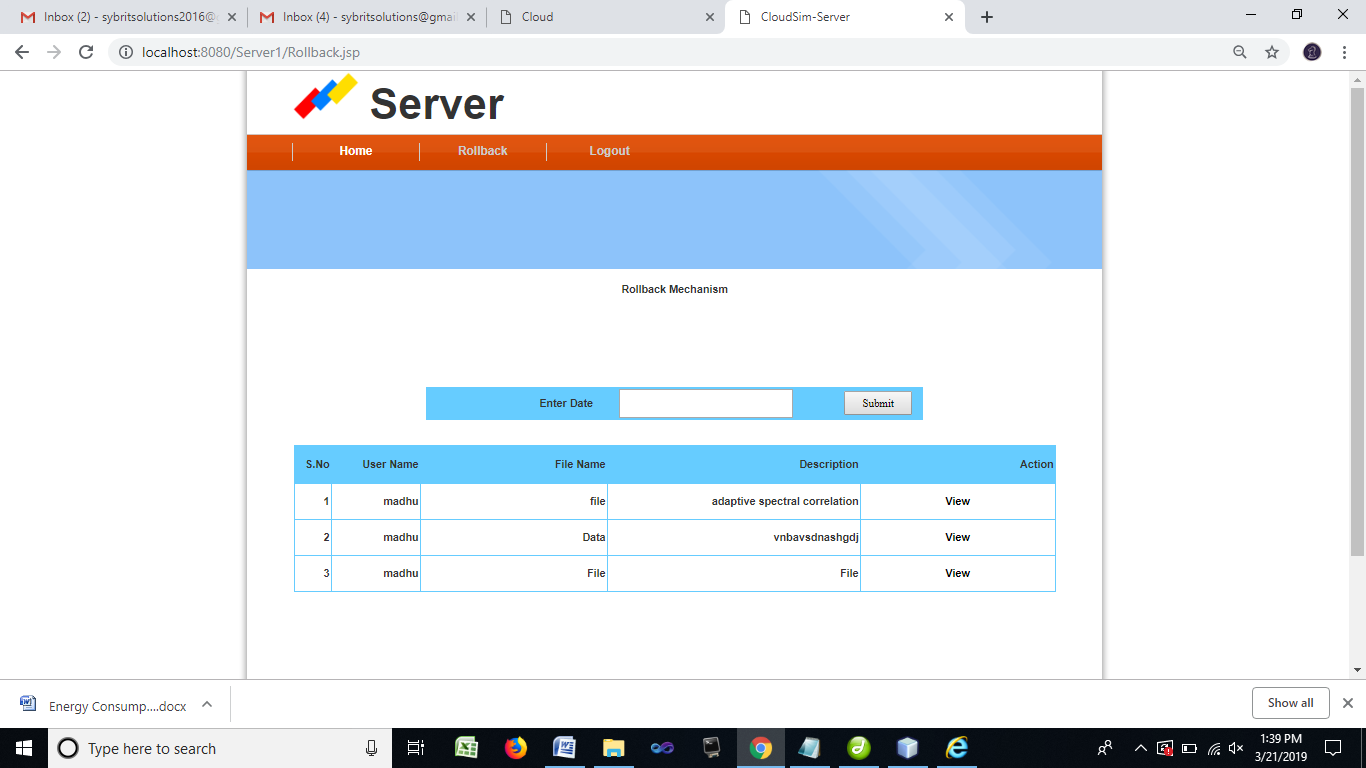












**REFERENCES**

[1]I. Weber, H. Wada, A. Fekete, A. Liu, and L. Bass, “Supporting undoability in systems operations,” in USENIX LISA’13: Large Installation System Administration Conference, Nov. 2013

[2] S. Satyal, I. Weber, L. Bass, and M. Fu, “Scalable rollback for cloud operations using AI planning,” in Software Engineering Conference (ASWEC), 2015 24th Australasian. IEEE, 2015, pp. 195–202.

[3] Mohammad H. Alshayeji, Mohammad Al-Rousan, Eman Yossef, Hanem Ellethy, “A Study on Fault Tolerance Mechanisms in Cloud Computing”, Computer Engineering Department, College of Computing Sciences and Engineering, Kuwait University P.O. Box No: 5969, Safat 13060, Kuwait.

[4]Ioannis Giannakopoulosl, Ioannis Konstantinoul, Dimitrios Tsoumakos and Nectarios Koziris, “Cloud application deployment with transient failure recovery”, Journal of Cloud Computing: Advances, Systems and Applications (2018) 7:11 <https://doi.org/10.1186/s13677-018-0112-9>

[5] T. Bylander, “The computational complexity of propositional strips planning,” Artificial Intelligence, vol. 69, pp. 165–204, 1994.

[6] M. Helmert, “The fast downward planning system,” CoRR, vol. abs/1109.6051, 2011. [Online]. Available: http://arxiv.org/abs/ 1109.6051

[7] K. R. Joshi, W. H. Sanders, M. A. Hiltunen, and R. D. Schlichting, “Automatic model-driven recovery in distributed systems,” in SRDS’05: IEEE Symposium on Reliable Distributed Systems, 2005.

[8] C.E.daSilvaandR.deLemos,“Aframeworkforautomaticgeneration of processes for self-adaptive software systems,” Informatica, vol. 35, pp. 3–13, 2011, publisher: Slovenian Society Informatika.

[9] J. Hoffmann, I. Weber, and F. M. Kraft, “SAP speaks PDDL: Exploiting a software-engineering model for planning in business process management,” Journal of Artificial Intelligence Research (JAIR), vol. 44, pp. 587–632, 2012.

[10] J. Hoffmann and B. Nebel, “The FF planning system: Fast plan generation through heuristic search,” J. Artif. Int. Res., vol. 14, no. 1, pp. 253–302, May 2001.

[11] D. McDermott et al., The PDDL Planning Domain Definition Language, The AIPS-98 Planning Competition Comitee, 1998.

[12] T. L. Dean and M. S. Boddy, “An analysis of time-dependent planning.” in AAAI, vol. 88, 1988, pp. 49–54.

[13] S. Zilberstein and S. Russell, “Approximate reasoning using anytime algorithms,” Kluwer International Series in Engineering and Computer Science, pp. 43–43, 1995.

[14] N. Gupta and D. S. Nau, “On the complexity of blocks-world planning,” Artificial Intelligence, vol. 56, no. 2, pp. 223–254, 1992.

[15] M. Helmert, “The fast downward planning system,” CoRR, vol. abs/1109.6051, 2011. [Online]. Available: <http://arxiv.org/abs/> 1109.6051

[16] J. Gray, “Why do computers stop and what can be done about it?” in Symposium on Reliability in Distributed Software and Database Systems, 1986, pp. 3–12.

[17] “AWS Elastic Compute Cloud (EC2),” <http://aws.amazon.com/> ec2/, accessed: 2015-06-02.

[18] “AWS CloudTrail,” http://aws.amazon.com/cloudtrail/, accessed: 2015-06-02.

[19] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, “Why do internet services fail, and what can be done about it?” in USITS’03: USENIX Symposium on Internet Technologies and Systems, 2003.

[20] B. Randell, “System structure for software fault tolerance,” IEEE Transactions On Software Engineering, vol. 1, no. 2, 1975.