**A MAJOR PROJECT REPORT**

**ON**

**A MULTI OBJECTIVE OPTIMIZATION SCHEME FOR JOB SCHEDULING CLOUD DATA CENTERS.**

*project report submitted*

In partial fulfillment of the requirement for the award of the degree

**MASTER OF COMPUTER APPLICATIONS**



Done By

GANDHAM PRIYANKA (4321124)

*Under the valuable guidance of*

**A. SHIVA KUMAR**, M.C.A

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE

**SRI PADMAVATHI COLLEGE OF COMPUTER**

**SCIENCES & TECHNOLOGY**

TIRUPATI-517502

**DEPARTMENT OF COMPUTER SCIENCE**

**SRI PADMAVATHI COLLEGE OF**

**COMPUTER SCIENCES & TECHNOLOGY**



**CERTIFICATE**

This is to certify that this project entitled **“A MULTI OBJECTIVE OPTIMIZATION SCHEME FOR JOB SCHEDULING CLOUD DATA CENTERS”** is a bonafied work done and submitted by G.PRIYANKA(4321124) In partial fulfillment of the requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS** in **SRI PADMAVATHI COLLEGE OF COMPUTER SCINCES &TECHNOLOGY,** TIRUCHANOOR, of **S.V University** Tirupati, during the academic year 2021-2023.

**PROJECT GUIDE HEAD OF THE DEPARTMENT**

**A. SHIVA KUMAR, M.C.A, (DR. T.RAGHU TRIVEDI)**

**l l**

Attended and submitted for the University Major Project Work.

Signature (s) of the Examiners: 1.

2.

3.

**DECLARATION**

We hereby declare that this is entitled “**A MULTI OBJECTIVE OPTIMIZATION SCHEME FOR JOB SCHEDULING CLOUD DATA CENTERS**” under the valuable guidance and supervision of Assistant Professor **A.SHIVA KUMAR**, **M.C.A**, Department of computer science, Sri Padmavathi College of Computer Sciences and technology , is submitted in partial fulfillment of the degree of Master of Computer Applications to and this Sri Padmavathi College of computer sciences and technology Major project is the result of our own effort and has been submitted earlier for the award of **MASTER OF COMPUTER APPLICATIONS** degree.

Place: Tirupati G. PRIYANKA (4321124)

Date:

**ACKNOWLEDGEMENT**

An endeavor over a long period can be successful only with the advice and support of many well-wishers. we take this opportunity to express our gratitude and appreciation to all of them.

We extend our thanks to our chairperson smt. **P. SULOCHANA** Garu, **SRI PADMAVATHI COLLEGE OF COMPUTER SCIENCE AND TECHNOLOGY** for her encouragement.

We would like to thank our principal **Dr. T RAGHU TRIVEDI** Garu, principal of, **SRI PADMAVATHI COLLEGE OF COMPUTER SCIENCE AND TECHNOLOGY** for his encouragement

We have taken pleasure in thanking **A. SHIVA KUMAR**, **M.C.A** and Project Guide for providing me with an opportunity to undergo the project internship at his esteemed organization. This opportunity has given an explain the software development area besides enriching me with an understanding with the business requirement and the organizational work place.

I am thankful to my entire lecturers in **MCA Department** Who helped me to complete my course in great success.

G. PRIYANKA (4321124)

**INDEX**

**S. NO: CONTENTS**

1. **INTRODUCTION**

**2. PROBLEM DEFINATION**

2.1 EXISTING SYSTEM

2.1.1 DISADVANTAGE OF EXISTING SYSTEM

2.2 PROPOSED SYSTEM

2.2.1 ADVANTAGES OF PROPOSED SYSTEM

2.3 MODULES

* + 1. **3. SYSTEM STUDY**

3.1 FEASIBILITY STUDY

* + 1. 3.2 ECONOMICAL FEASIBILITY

3.3 TECHNICAL FEASIBILITY

* 1. SOCIAL FEASIBILITY

**4**.  **SYSTEM REQURIMENTS**

4.1 HARDWARE REQURIMENTS

4.2 SOFTWARE REQURIMENTS

* + 1. **5. SYSTEM ANALYSIS**
    2. 5.1 JDBC GOALS
    3. 5.2 NETWORKING
    4. 5.3 IP DIAGRAMS

* + 1. **6. SYSTEM TESTING**
    2. 6.1 UNIT TESTING
    3. 6.2 INTEGRATION TESTING
    4. 6.3 ACCEPTANCE TESTING
    5. **7. SYSTEM DESIGN**
    6. 7.1 INPUT DESIGN
    7. 7.2 OUTPUT DESIGN
    8. **8. REQUREMENTS SPECIFICATION**
    9. 8.1 FUNCITIONAL REQUREMENTS
    10. 8.2 USER INTERFACE REQUREMENTS
    11. **9. CONCLUSION**
    12. 9.1 CONCLUSION
    13. 9.2 BIBLOGRAPHY
    14. 9.3 OUTPUT SCREENS

**ABSTRACT**

For a number of years, due to an exponential increase in the demand for an eco-friendly environment, there has been a rapid increase in the green city revolution across the globe. Subsequently, load shifting of major energy consumers from conventional power grids to renewable energy sources (RES) has become inevitable. Towards this end, cloud data centers (DCs) have emerged as significant consumers of energy that solely rely on power grids to fuel their day-to-day operations. Nevertheless, their energy consumption has increased significantly which in turn has substantially raised the global carbon footprint rate. These challenges can be best addressed by the judicious utilization of RES which have well established advantages like reduced operational costs and carbon emissions. Keeping in view of the above facts, the ultimate goal of the proposed work is to design a comprehensive workload classification; and job scheduling and Virtual machine placement architecture for cloud DCs powered by RES and power grids. For this, a multi-objective optimization scheme is proposed which operates in two phases. In phase I, a random forest-based wrapper scheme known as Boruta, is used for relevant feature set selection for the incoming workload. This is followed by classification of the workload using a locality sensitive hashing-based support vector machines approach. In phase II, a multi-objective optimization problem for job scheduling and VM placement is formulated with respect to parameters such as service level agreement (SLA), energy cost, carbon footprint rate (CFR), and availability of RES. It is further solved using an enhanced heuristic approach based on a greedy strategy. Our experimental evaluations show an average improvement of approximately 31% in energy utilization, 28% in energy cost, and 36% in CFR, with a slight degradation in SLA assurance (about 2%) compared with the existing scheme

**INTRODUCTION**

Cloud computing (CC) is one of the most powerful technologies of the modern era for providing on-demand services to end users as per their demands. In order to cater to the service requirements of the intelligent end user domain, CC has evolved as a ubiquitous platform by virtue of virtualization. But, adherence to the service-level agreement (SLA) is a challenging task while providing parallel and distributed resources to end users. In CC, virtual resources are sliced over physical servers to handle incoming workloads [1]. For this purpose, massive data centers (DCs) are deployed by different service providers across the globe. With a rapid growth in the number of smart users, the need for different types of cloud resources has increased exponentially [2]. In order to handle millions of job requests, DCs deployed across the globe have witnessed an exponential growth in terms of number, size, and functionalities. For example, to provide cloud resources for efficient provision of on-line activities such as-email, social media, and e-commerce across the US, nearly 3 million DCs are deployed [3]. However, with the advent of the latest technologies such as industrial internet of things, smart healthcare, and intelligent transportation; the existing computing capabilities of DCs would soon be insufficient. According to [4] (2014), only 8.5% of global DCs had enough resources to cater for the increasing end user demands beyond 2015 and the years to come. It has been projected that 75% of DCs deployed by various cloud service providers (CSPs) are expected to expand both vertically and horizontally. By 2020, such an expansion is expected to double the size of DCs relative to 2010.

**Existing System:**

In CC, virtual resources are sliced over physical servers to handle incoming workloads. For this purpose, massive data centers (DCs) are deployed by different service providers across the globe. With a rapid growth in the number of smart users, the need for different types of cloud resources has increased exponentially.

In order to handle millions of job requests, DCs deployed across the globe have witnessed an exponential growth in terms of number, size, and functionalities. For example, to provide cloud resources for efficient provision of on-line activities such as-email, social media, and e-commerce across the US, nearly 3 million DCs are deployed.

However, with the advent of the latest technologies such as industrial internet of things, smart healthcare, and intelligent transportation; the existing computing capabilities of DCs would soon be insufficient. According to (2014), only 8.5% of global DCs had enough resources to cater for the increasing end user demands beyond 2015 and the years to come.

It has been projected that 75% of DCs deployed by various cloud service providers (CSPs) are expected to expand both vertically and horizontally. By 2020, such an expansion is expected to double the size of DCs relative to 2010.

**Disadvantages:**

Multi-objective optimization aims to find a set of solutions that optimize multiple conflicting objectives simultaneously. In the context of job scheduling in cloud data centers, these objectives may include minimizing energy consumption, reducing carbon footprint, maximizing resource utilization, and minimizing job completion time.

However, solving multi-objective optimization problems requires considering a larger solution space and evaluating trade-offs among multiple objectives. This can significantly increase the computational complexity and time required to find an optimal or near-optimal solution. As a result, the job scheduling process may become slower and more resource-intensive.

Moreover, the introduction of a multi-objective optimization scheme may require sophisticated algorithms, such as evolutionary algorithms or swarm intelligence techniques, which can further increase the computational burden. Implementing and maintaining these complex algorithms can be challenging and may require additional expertise or resources.

Additionally, the multi-objective optimization scheme may introduce more uncertainty and variability in the scheduling process. Different combinations of objectives and their weights can lead to different scheduling decisions and outcomes. This variability may make it difficult to predict or guarantee specific performance levels or make consistent scheduling decisions.

**Proposed System:**

The overall working architecture of the proposed scheme is depicted. In the proposed system architecture, n geo-distributed DCs powered by RES and grid (through a battery energy storage system) have been considered. The provisioning of the incoming workload to these servers is achieved using virtualization across two phases.

In the first phase, the incoming workload (W) is classified into different queues using the proposed LSH-based SVM classifier. The classification is initially made by the selection of the relevant feature set using Boruta. In the next phase, the classified workload is scheduled among DCs by a cloud broker which comprises of job scheduler, VM placement manager, etc.

The job scheduling and VM placement approach adopted in this phase maintains the trade-off with the competing functions using MOOP. The formulated MOOP is solved using a heuristic approach based on a greedy strategy keeping in mind the complexity of the underlying system. The working of the above mentioned phases is elaborated in the upcoming sections.

**Advantage:**

1. Enhanced Sustainability: By incorporating sustainability objectives into the job scheduling process, the scheme enables the data center to operate in a more environmentally friendly manner. It allows for optimization of energy consumption, carbon footprint, and resource utilization, leading to reduced environmental impact and improved sustainability.
2. Improved Resource Utilization: The scheme optimizes resource allocation and scheduling decisions, considering multiple objectives simultaneously. This leads to better utilization of available resources within the data center, such as servers, storage, and network bandwidth. By maximizing resource utilization, the scheme helps to minimize waste and increase overall efficiency.
3. Increased Energy Efficiency: Energy consumption is a critical concern for cloud data centers. By integrating energy-related objectives, such as minimizing power consumption or maximizing energy efficiency, the optimization scheme can help reduce the overall energy requirements of the data center. This leads to cost savings and a reduced carbon footprint.
4. Better Performance and Throughput: The multi-objective optimization scheme aims to optimize job scheduling based on various performance metrics, such as job completion time, response time, or throughput. By considering these objectives, the scheme can improve overall system performance and increase the throughput of the data center, resulting in better user experiences and higher customer satisfaction.
5. Flexibility and Adaptability: The scheme allows for flexibility and adaptability in the scheduling process. Different combinations of objectives and their weights can be easily adjusted to meet changing requirements and priorities. This flexibility enables data center administrators to fine-tune the scheduling process based on evolving needs and goals.

**Software Requirements:**

**Hardware Requirements:**

* System : Pentium IV 2.4 GHz.
* Hard Disk : 40 GB.
* Floppy Drive : 1.44 Mb.
* Monitor : 15 VGA Colour.
* Mouse : Logitech.
* RAM : 256 Mb.

**Software Requirements:**

* Operating system :Windows 7.
* Front End : JAVA or JSP
* Database : MySQL
* Tools : Eclipse IDE

**Module description:**

1. Objective Definition: This component involves defining the objectives that will guide the job scheduling process. The objectives may include minimizing energy consumption, reducing carbon footprint, maximizing resource utilization, minimizing job completion time, or any other relevant performance metrics. The module allows for customization of objectives based on the specific requirements and priorities of the data center.
2. Optimization Algorithms: The module incorporates advanced optimization algorithms suitable for multi-objective problems, such as evolutionary algorithms, genetic algorithms, or swarm intelligence techniques. These algorithms explore the solution space, evaluate trade-offs between objectives, and generate a set of Pareto-optimal solutions representing the best trade-offs between conflicting objectives.
3. Sustainability Considerations: This component focuses on integrating sustainability principles into the job scheduling process. It considers energy-efficient scheduling strategies, renewable energy utilization, load balancing techniques, and dynamic resource allocation to minimize environmental impact and maximize sustainability. The module may also incorporate predictive models and data analytics to forecast energy consumption patterns and optimize scheduling decisions accordingly.
4. Resource Management: This component addresses the efficient allocation and utilization of resources within the data center. It considers factors such as server availability, network bandwidth, storage capacity, and computational requirements of jobs. The module employs optimization algorithms to allocate resources optimally, minimize resource wastage, and ensure efficient utilization across different jobs and tasks.
5. Performance Metrics and Evaluation: The module includes mechanisms for evaluating the performance of the scheduling scheme. It provides metrics and analytics to assess the achieved objectives, such as energy consumption reduction, carbon footprint reduction, resource utilization improvement, and job completion time. These evaluations allow data center administrators to understand the impact of the scheduling scheme and make data-driven decisions for further optimization.
6. Flexibility and Customization: The module offers flexibility and customization options to adapt the scheduling scheme to specific data center requirements and constraints. It allows administrators to adjust objective weights, define constraints, and incorporate specific preferences or policies into the optimization process. This flexibility ensures that the module can be tailored to different data center environments and evolving needs.
7. Multi-cloud Optimization: Extend the optimization scheme to consider multi-cloud environments, where jobs can be distributed across multiple cloud providers. Develop algorithms to optimize job placement and resource allocation across different clouds, considering factors such as cost, performance, energy consumption, and sustainability. This can enable efficient utilization of resources across multiple clouds while meeting diverse objectives.

**Literature Survey:**

1. Title: "Multi-objective job scheduling in cloud computing environments: Trends and research challenges" Authors: Zhang, Qi, et al. Published: Future Generation Computer Systems, 2016

This paper provides an overview of multi-objective job scheduling in cloud computing environments. It discusses various optimization techniques, objectives, and challenges in scheduling jobs. It also highlights the importance of sustainability considerations in job scheduling and identifies future research directions.

1. Title: "A Multi-objective Genetic Algorithm for Energy-Aware Scheduling in Cloud Data Centers" Authors: Lin, Xuemin, et al. Published: IEEE Transactions on Parallel and Distributed Systems, 2013

The paper proposes a multi-objective genetic algorithm for energy-aware job scheduling in cloud data centers. It focuses on optimizing energy consumption, performance, and resource utilization. The algorithm utilizes a Pareto-based approach to generate a set of trade-off solutions and provides insights into the energy-performance trade-offs in cloud data centers.

1. Title: "Multi-objective scheduling of workflow applications on sustainable cloud data centers" Authors: Wang, Lizhe, et al. Published: Future Generation Computer Systems, 2014

This study presents a multi-objective scheduling approach for workflow applications in sustainable cloud data centers. It considers objectives such as energy efficiency, resource utilization, and job completion time. The paper proposes an algorithm based on particle swarm optimization (PSO) to find near-optimal solutions and conducts experiments to evaluate its performance.

1. Title: "Multi-objective optimization of job scheduling in cloud computing using Non-dominated Sorting Genetic Algorithm-II" Authors: Prakash, R., et al. Published: Journal of Ambient Intelligence and Humanized Computing, 2017

This research work applies the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) to solve the multi-objective job scheduling problem in cloud computing. It considers objectives such as makespan, energy consumption, and monetary cost. The paper evaluates the proposed approach through simulation experiments and provides insights into the trade-offs among different objectives.

1. Title: "Multi-objective job scheduling for energy-efficient virtualized data centers" Authors: Naderi, Babak, et al. Published: Sustainable Computing: Informatics and Systems, 2017

The paper presents a multi-objective job scheduling framework for energy-efficient virtualized data centers. It considers objectives including energy consumption, carbon emissions, and job execution time. The study introduces a multi-objective evolutionary algorithm to optimize scheduling decisions and conducts experiments to validate its effectiveness in reducing energy consumption and carbon emissions.

1. Title: "Multi-objective optimization for scheduling in cloud computing environments: A survey" Authors: Hsu, Ching-Hsien, et al. Published: Journal of Network and Computer Applications, 2018

This survey paper provides a comprehensive review of multi-objective optimization for scheduling in cloud computing environments. It covers various optimization techniques, objectives, and challenges. The paper discusses different scheduling approaches and algorithms, highlighting their strengths and limitations. It also identifies open research issues and future directions in the field.

**Methodology:**

1. Problem Formulation: Define the problem of job scheduling in sustainable cloud data centers, considering multiple objectives such as energy consumption, resource utilization, job completion time, and sustainability metrics. Identify the constraints and limitations of the system, including resource availability, workload variations, and environmental considerations.
2. Objective Identification: Determine the specific objectives to be optimized in the scheduling scheme based on the requirements and priorities of the data center. Common objectives include minimizing energy consumption, reducing carbon footprint, maximizing resource utilization, and minimizing job completion time. Assign weights or priorities to each objective to indicate their relative importance.
3. Data Collection: Gather relevant data about the cloud data center, including historical job information, resource availability, energy consumption patterns, and sustainability metrics. This data will be used for analysis, modeling, and optimization purposes.
4. Mathematical Modeling: Develop mathematical models that represent the job scheduling problem and capture the relationships between objectives, constraints, and decision variables. The models should consider the dynamic nature of the data center, the characteristics of jobs and resources, and the sustainability aspects.
5. Optimization Algorithms: Select appropriate multi-objective optimization algorithms to solve the formulated problem. Common algorithms include evolutionary algorithms (e.g., genetic algorithms, particle swarm optimization), simulated annealing, or ant colony optimization. These algorithms explore the solution space, evaluate different combinations of scheduling decisions, and generate a set of Pareto-optimal solutions.
6. Objective Trade-off Analysis: Perform a trade-off analysis of the Pareto-optimal solutions to evaluate the trade-offs between different objectives. Visualize and analyze the trade-off surface to identify the preferred solutions based on the decision-maker's preferences and priorities. Consider sensitivity analysis to understand the impact of changing objective weights or constraints on the scheduling decisions.
7. Sustainability Considerations: Integrate sustainability considerations into the optimization scheme. Develop energy models and carbon footprint estimation methods to assess the environmental impact of scheduling decisions. Incorporate renewable energy utilization, load balancing techniques, and dynamic resource allocation to enhance sustainability.
8. Performance Evaluation: Evaluate the performance of the optimization scheme by comparing it with existing scheduling algorithms or benchmarking datasets. Use performance metrics such as energy consumption reduction, carbon footprint reduction, resource utilization improvement, and job completion time. Conduct experiments or simulations to validate the effectiveness and efficiency of the proposed scheme.
9. Implementation and Validation: Implement the developed optimization scheme in a real cloud data center environment or a simulation framework. Validate the scheme's performance, scalability, and practicality by conducting experiments with realistic workloads and evaluating its impact on sustainability and resource utilization.
10. Future Research Directions: Identify potential areas for future research and improvement. This may include exploring advanced machine learning techniques, dynamic and real-time scheduling strategies, handling heterogeneous data centers, or considering multi-cloud optimization. Consider scalability, efficiency enhancements, and adaptability to evolving data center requirements.

**Algorithm: NSGA-II algorithm**

1. Initialization:
   * Generate an initial population of scheduling solutions (chromosomes) randomly or using heuristics.
   * Evaluate the fitness of each chromosome by calculating objective values such as energy consumption, resource utilization, job completion time, and sustainability metrics.
2. Non-dominated Sorting and Crowding Distance Calculation:
   * Perform non-dominated sorting to classify the chromosomes into different fronts based on their dominance relationship.
   * Calculate the crowding distance of each chromosome in each front to preserve diversity and favor solutions that are well-distributed across the Pareto front.
3. Selection:
   * Select individuals for reproduction using a combination of non-dominated sorting and crowding distance. Favor solutions in the first few fronts with higher crowding distance values to maintain diversity.
4. Genetic Operators:
   * Apply crossover and mutation operators to the selected individuals to create offspring.
   * Perform crossover by exchanging genetic information between parent chromosomes to generate new solutions.
   * Perform mutation by introducing random changes to the chromosomes to explore new regions of the search space.
5. Evaluation and Fitness Assignment:
   * Evaluate the fitness of the offspring chromosomes by calculating the objective values.
   * Assign fitness values based on the non-dominated sorting and crowding distance calculations.
6. Environmental Selection:
   * Merge the parent and offspring populations.
   * Perform non-dominated sorting and crowding distance calculation on the combined population.
   * Select the best solutions based on the non-dominated fronts and crowding distance values to form the next generation.
7. Termination:
   * Check for termination conditions, such as reaching a maximum number of iterations or satisfying a predefined stopping criterion.
   * If the termination condition is not met, go back to step 3 (Selection) and continue the algorithm.
   * If the termination condition is met, the algorithm terminates, and the final Pareto-optimal solutions represent the optimized job scheduling solutions.

**INTRODUCTION**

**DATA MINING**

**Data mining** is an interdisciplinary subfield of [computer science](https://en.wikipedia.org/wiki/Computer_science). It is the computational process of discovering patterns in large [data sets](https://en.wikipedia.org/wiki/Data_set) ("[big data](https://en.wikipedia.org/wiki/Big_data)") involving methods at the intersection of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence), [machine learning](https://en.wikipedia.org/wiki/Machine_learning), [statistics](https://en.wikipedia.org/wiki/Statistics), and [database systems](https://en.wikipedia.org/wiki/Database_system). The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and [data management](https://en.wikipedia.org/wiki/Data_management) aspects, [data pre-processing](https://en.wikipedia.org/wiki/Data_pre-processing), [model](https://en.wikipedia.org/wiki/Statistical_model) and [inference](https://en.wikipedia.org/wiki/Statistical_inference) considerations-interestingness-

metrics, [complexity](https://en.wikipedia.org/wiki/Computational_complexity_theory) considerations, post-processing of discovered structures, [visualization](https://en.wikipedia.org/wiki/Data_visualization), and [online updating](https://en.wikipedia.org/wiki/Online_algorithm).  Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD. The actual data mining task is the automatic or semi-automatic analysis of large quantities of data to extract previously unknown, interesting patterns such as groups of data records ([cluster analysis](https://en.wikipedia.org/wiki/Cluster_analysis)), unusual records ([anomaly detection](https://en.wikipedia.org/wiki/Anomaly_detection)), and dependencies ([association rule mining](https://en.wikipedia.org/wiki/Association_rule_mining)). This usually involves using database techniques such as [spatial indices](https://en.wikipedia.org/wiki/Spatial_index). These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics). For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a [decision support system](https://en.wikipedia.org/wiki/Decision_support_system). Neither the data collection, data preparation, nor result interpretation and reporting is part of the data mining step, but do belong to the overall KDD process as additional steps.

The related terms [data dredging](https://en.wikipedia.org/wiki/Data_dredging), data fishing, and data snooping refer to the use of data mining methods to sample parts of a larger population data set that are (or may be) too small for reliable statistical inferences to be made about the validity of any patterns discovered. These methods can, however, be used in creating new hypotheses to test against the larger data populations.

Big Data concern large-volume, complex, growing data sets with multiple, autonomous sources. With the fast development of networking, data storage, and the data collection capacity, Big Data are now rapidly expanding in all science and engineering domains, including physical, biological and biomedical sciences. This paper presents a HACE theorem that characterizes the features of the Big Data revolution, and proposes a Big Data processing model, from the data mining perspective. This data-driven model involves demand-driven aggregation of information sources, mining and analysis, user interest modeling, and security and privacy considerations. We analyze the challenging issues in the data-driven model and also in the Big Data revolution.

**BIG DATA**

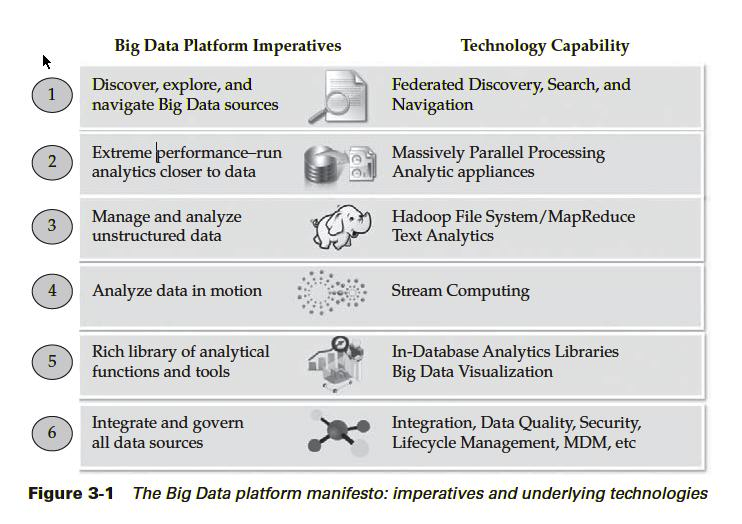
Big data is a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools. The challenges include capture, curation, storage, search, sharing, analysis, and visualization. The trend to larger data sets is due to the additional information derivable from analysis of a single large set of related data, as compared to separate smaller sets with the same total amount of data, allowing correlations to be found to "spot business trends, determine quality of research, prevent diseases, link legal citations, combat crime, and determine real-time roadway traffic conditions.

Put another way, big data is the realization of greater business intelligence by storing, processing, and analyzing data that was previously ignored due to the limitations of traditional data management technologies

**The four dimensions of Big Data**

* Volume: Large volumes of data
* Velocity: Quickly moving data
* Variety: structured, unstructured, images, etc.
* Veracity: Trust and integrity is a challenge and a must and is important for big data just as for traditional relational DBs
* Big Data is about better analytics!

**The Big Data platform Manifesto**



**Some concepts**

* No SQL (Not Only SQL): Databases that “move beyond” relational data models (i.e., no tables, limited or no use of SQL)
  + Focus on retrieval of data and appending new data (not necessarily tables)
  + Focus on key-value data stores that can be used to locate data objects
  + Focus on supporting storage of large quantities of unstructured data
  + SQL is not used for storage or retrieval of data
  + No ACID (atomicity, consistency, isolation, durability)

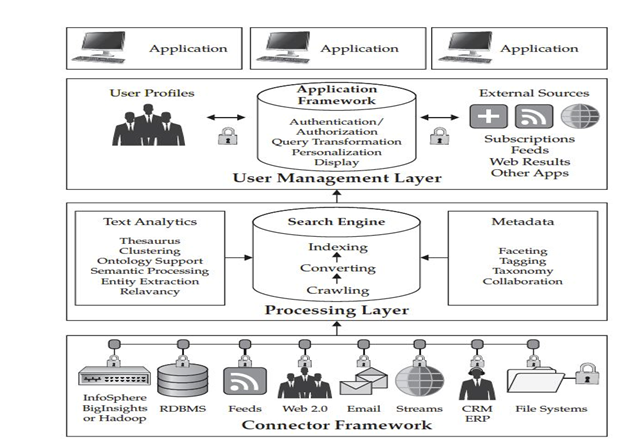
**Hadoop**

* Hadoop is a distributed file system and data processing engine that is designed to handle extremely high volumes of data in any structure.
* Hadoop has two components:
  + The Hadoop distributed file system (HDFS), which supports data in structured relational form, in unstructured form, and in any form in between
  + The MapReduce programming paradigm for managing applications on multiple distributed servers
* The focus is on supporting redundancy, distributed architectures, and parallel processing

**Some Hadoop Related Names to Know**

* **Apache Avro**: designed for communication between Hadoop nodes through data serialization
* **Cassandra and Hbase**: a non-relational database designed for use with Hadoop
* **Hive:** a query language similar to SQL (HiveQL) but compatible with Hadoop
* **Mahout**: an AI tool designed for machine learning; that is, to assist with filtering data for analysis and exploration
* **Pig Latin**: A data-flow language and execution framework for parallel computation
* **Zoo Keeper**: Keeps all the parts coordinated and working togethe

**What to do with the data**



The Knowledge Discovery in Databases (KDD) process is commonly defined with the stages:

(1) Selection

(2) Pre-processing

(3) Transformation

(4) Data Mining

(5) Interpretation/Evaluation.

It exists, however, in many variations on this theme, such as the [Cross Industry Standard Process for Data Mining](https://en.wikipedia.org/wiki/Cross_Industry_Standard_Process_for_Data_Mining) (CRISP-DM) which defines six phases:

(1) Business Understanding

(2) Data Understanding

(3) Data Preparation

(4) Modeling

(5) Evaluation

(6) Deployment

or a simplified process such as (1) pre-processing, (2) data mining, and (3) results validation.

**TensorFlow** is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [software library](https://en.wikipedia.org/wiki/Library_(computing)) for [machine learning](https://en.wikipedia.org/wiki/Machine_learning) and [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence). It can be used across a range of tasks but has a particular focus on [training](https://en.wikipedia.org/wiki/Types_of_artificial_neural_networks#Training) and [inference](https://en.wikipedia.org/wiki/Statistical_inference) of [deep neural networks](https://en.wikipedia.org/wiki/Deep_neural_networks).

**Usage and extensions**

TensorFlow serves as the core platform and library for machine learning. TensorFlow’s APIs use Keras to allow users to make their own machine learning models. In addition to building and training their model, TensorFlow can also help load the data to train the model, and deploy it using TensorFlow Serving.

**Integrations**

**Numpy**

Numpy is one of the most popular Python data libraries, and TensorFlow offers integration and compatibility with its data structures. Numpy NDarrays, the library’s native datatype, are automatically converted to TensorFlow Tensors in TF operations; the same is also true vice-versa. This allows for the two libraries to work in unison without requiring the user to write explicit data conversions. Moreover, the integration extends to memory optimization by having TF Tensors share the underlying memory representations of Numpy NDarrays whenever possible.

**Extensions**TensorFlow also offers a variety of libraries and extensions to advance and extend the models and methods used.For example, TensorFlow Recommenders and TensorFlow Graphics are libraries for their respective functionalities in recommendation systems and graphics, TensorFlow Federated provides a framework for decentralized data, and TensorFlow Cloud allows users to directly interact with Google Cloud to integrate their local code to Google Cloud. Other add-ons, libraries, and frameworks include TensorFlow Model Optimization, TensorFlow Probability, TensorFlow Quantum, and TensorFlow Decision Forests.

**INPUT DESIGN**

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

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

**OBJECTIVES**

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

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

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**OUTPUT DESIGN**

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

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

2.Select methods for presenting information.

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

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

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

**1 FEASIBILITY STUDY:**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**5.1.1 ECONOMICAL FEASIBILITY:**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

**5.1.2 TECHNICAL FEASIBILITY:**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**5.1.3 SOCIAL FEASIBILITY:**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**Introduction**

Systems design

**Introduction: Systems design** is the process or art of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development. There is some overlap and synergy with the disciplines of systems analysis, systems architecture and systems engineering.

**System Design Document**

Overview

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces*.*

1. **INTRODUCTION**

**Purpose and Scope**

This section provides a brief description of the Systems Design Document’s purpose and scope.

**Project Executive Summary**

This section provides a description of the project from a management perspective and an overview of the framework within which the conceptual system design was prepared. If appropriate, include the information discussed in the subsequent sections in the summary.

System Overview

This section describes the system in narrative form using non-technical terms. It should provide a high-level system architecture diagram showing a subsystem breakout of the system, if applicable. The high-level system architecture or subsystem diagrams should, if applicable, show interfaces to external systems. Supply a high-level context diagram for the system and subsystems, if applicable. Refer to the requirements trace ability matrix (RTM) in the Functional Requirements Document (FRD), to identify the allocation of the functional requirements into this design document.

Design Constraints

This section describes any constraints in the system design (reference any trade-off analyses conducted such, as resource use versus productivity, or conflicts with other systems) and includes any assumptions made by the project team in developing the system design.

Future Contingencies

This section describes any contingencies that might arise in the design of the system that may change the development direction. Possibilities include lack of interface agreements with outside agencies or unstable architectures at the time this document is produced. Address any possible workarounds or alternative plans.

**Document Organization**

This section describes the organization of the Systems Design Document.

**Points of Contact**

This section provides the organization code and title of the key points of contact (and alternates if appropriate) for the information system development effort. These points of contact should include the Project Manager, System Proponent, User Organization, Quality Assurance (QA) Manager, Security Manager, and Configuration Manager, as appropriate.

**Project References**

This section provides a bibliography of key project references and deliverables that have been produced before this point.

**Glossary**

Supply a glossary of all terms and abbreviations used in this document. If the glossary is several pages in length, it may be included as an appendix.

**SYSTEM ARCHITECTURE**

In this section, describe the system and/or subsystem(s) architecture for the project. References to external entities should be minimal, as they will be described in detail in Section 6, External Interfaces.

**System Hardware Architecture**

In this section, describe the overall system hardware and organization. Include a list of hardware components (with a brief description of each item) and diagrams showing the connectivity between the components. If appropriate, use subsections to address each subsystem.

**System Software Architecture**

In this section, describe the overall system software and organization. Include a list of software modules (this could include functions, subroutines, or classes), computer languages, and programming computer-aided software engineering tools (with a brief description of the function of each item). Use structured organization diagrams/object-oriented diagrams that show the various segmentation levels down to the lowest level. All features on the diagrams should have reference numbers and names. Include a narrative that expands on and enhances the understanding of the functional breakdown. If appropriate, use subsections to address each module.

**Note:** The diagrams should map to the FRD data flow diagrams, providing the physical process and data flow related to the FRD logical process and data flow.

**Internal Communications Architecture**

In this section, describe the overall communications within the system; for example, LANs, buses, etc. Include the communications architecture(s) being implemented, such as X.25*,* Token Ring, etc. Provide a diagram depicting the communications path(s) between the system and subsystem modules. If appropriate, use subsections to address each architecture being employed.

**Note:** The diagrams should map to the FRD context diagrams.

**FILE AND DATABASE DESIGN**

Interact with the Database Administrator (DBA) when preparing this section. The section should reveal the final design of all database management system (DBMS) files and the non-DBMS files associated with the system under development. Additional information may add as required for the particular project. Provide a comprehensive data dictionary showing data element name, type, length, source, validation rules, maintenance (create, read, update, delete (CRUD) capability), data stores, outputs, aliases, and description. Can be included as an appendix.

**Database Management System Files**

This section reveals the final design of the DBMS files and includes the following information, as appropriate (refer to the data dictionary):

* Refined logical model; provide normalized table layouts, entity relationship diagrams, and other logical design information
* A physical description of the DBMS schemas, sub-schemas, records, sets, tables, storage page sizes, etc.
* Access methods (such as indexed, via set, sequential, random access, sorted pointer array, etc.)
* Estimate of the DBMS file size or volume of data within the file, and data pages, including overhead resulting from access methods and free space
* Definition of the update frequency of the database tables, views, files, areas, records, sets, and data pages; estimate the number of transactions if the database is an online transaction-based system

**Non-Database Management System Files**

In this section, provide the detailed description of all non-DBMS files and include a narrative description of the usage of each file—including if the file is used for input, output, or both; if this file is a temporary file; an indication of which modules read and write the file, etc.; and file structures (refer to the data dictionary). As appropriate, the file structure information should:

* Identify record structures, record keys or indexes, and reference data elements within the records
* Define record length (fixed or maximum variable length) and blocking factors
* Define file access method—for example, index sequential, virtual sequential, random access, etc.
* Estimate the file size or volume of data within the file, including overhead resulting from file access methods
* Define the update frequency of the file; if the file is part of an online transaction-based system, provide the estimated number of transactions per unit time, and the statistical mean, mode, and distribution of those transactions

**HUMAN-MACHINE INTERFACE**

This section provides the detailed design of the system and subsystem inputs and outputs relative to the user/operator. Any additional information may be added to this section and may be organized according to whatever structure best presents the operator input and output designs. Depending on the particular nature of the project, it may be appropriate to repeat these sections at both the subsystem and design module levels. Additional information may be added to the subsections if the suggested lists are inadequate to describe the project inputs and outputs.

**Inputs**

This section is a description of the input media used by the operator for providing information to the system; show a mapping to the high-level data flows described in Section 1 .2.1, System Overview. For example, data entry screens, optical character readers, bar scanners, etc. If appropriate, the input record types, file structures, and database structures provided in Section 3, File and Database Design, may be referenced. Include data element definitions, or refer to the data dictionary.

Provide the layout of all input data screens or graphical user interfaces (GUTs) (for example, windows). Provide a graphic representation of each interface. Define all data elements associated with each screen or GUI, or reference the data dictionary.

This section should contain edit criteria for the data elements, including specific values, range of values, mandatory/optional, alphanumeric values, and length. Also address data entry controls to prevent edit bypassing.

Discuss the miscellaneous messages associated with operator inputs, including the following:

* Copies of form(s) if the input data are keyed or scanned for data entry from printed forms
* Description of any access restrictions or security considerations
* Each transaction name, code, and definition, if the system is a transaction-based processing system

**Outputs**

This section describes of the system output design relative to the user/operator; show a mapping to the high-level data flows described in Section 1.2.1. System outputs include reports, data display screens and GUIs, query results, etc. The output files are described in Section 3 and may be referenced in this section. The following should be provided, if appropriate:

* Identification of codes and names for reports and data display screens
* Description of report and screen contents (provide a graphic representation of each layout and define all data elements associated with the layout or reference the data dictionary)
* Description of the purpose of the output, including identification of the primary users
* Report distribution requirements, if any (include frequency for periodic reports)
* Description of any access restrictions or security considerations

**DETAILED DESIGN**

This section provides the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product. Additionally, this section addresses the detailed procedures for combining separate COTS packages into a single system. Every detailed requirement should map back to the FRD, and the mapping should be presented in an update to the RTM and include the RTM as an appendix to this design document.

**Hardware Detailed Design**

A hardware component is the lowest level of design granularity in the system. Depending on the design requirements, there may be one or more components per system. This section should provide enough detailed information about individual component requirements to correctly build and/or procure all the hardware for the system (or integrate COTS items).

If there are many components or if the component documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each component and its functions, adequately. Industry-standard component specification practices should be followed. For COTS procurements, if a specific vendor has been identified, include appropriate item names. Include the following information in the detailed component designs (as applicable):

* Power input requirements for each component
* Signal impedances and logic states
* Connector specifications (serial/parallel, 11-pin, male/female, etc.)
* Memory and/or storage space requirements
* Processor requirements (speed and functionality)
* Graphical representation depicting the number of hardware items (for example, monitors, printers, servers, I/O devices), and the relative positioning of the components to each other
* Cable type(s) and length(s)
* User interfaces (buttons, toggle switches, etc.)
* Hard drive/floppy drive/CD-ROM requirements
* Monitor resolution

**Software Detailed Design**

A software module is the lowest level of design granularity in the system. Depending on the software development approach, there may be one or more modules per system. This section should provide enough detailed information about logic and data necessary to completely write source code for all modules in the system (and/or integrate COTS software programs).

If there are many modules or if the module documentation is extensive, place it in an appendix or reference a separate document. Add additional diagrams and information, if necessary, to describe each module, its functionality, and its hierarchy. Industry-standard module specification practices should be followed. Include the following information in the detailed module designs:

* A narrative description of each module, its function(s), the conditions under which it is used (called or scheduled for execution), its overall processing, logic, interfaces to other modules, interfaces to external systems, security requirements, etc.; explain any algorithms used by the module in detail
* For COTS packages, specify any call routines or bridging programs to integrate the package with the system and/or other COTS packages (for example, Dynamic Link Libraries)
* Data elements, record structures, and file structures associated with module input and output
* Graphical representation of the module processing, logic, flow of control, and algorithms, using an accepted diagramming approach (for example, structure charts, action diagrams, flowcharts, etc.)
* Data entry and data output graphics; define or reference associated data elements; if the project is large and complex or if the detailed module designs will be incorporated into a separate document, then it may be appropriate to repeat the screen information in this section
* Report layout

**Internal Communications Detailed Design**

If the system includes more than one component there may be a requirement for internal communications to exchange information, provide commands, or support input/output functions. This section should provide enough detailed information about the communication requirements to correctly build and/or procure the communications components for the system. Include the following information in the detailed designs (as appropriate):

* The number of servers and clients to be included on each area network
* Specifications for bus timing requirements and bus control
* Format(s) for data being exchanged between components
* Graphical representation of the connectivity between components, showing the direction of data flow (if applicable), and approximate distances between components; information should provide enough detail to support the procurement of hardware to complete the installation at a given location
* LAN topology

**EXTERNAL INTERFACES**

External systems are any systems that are not within the scope of the system under development, regardless whether the other systems are managed by the State or another agency. In this section, describe the electronic interface(s) between this system and each of the other systems and/or subsystem(s), emphasizing the point of view of the system being developed.

**Interface Architecture**

In this section, describe the interface(s) between the system being developed and other systems; for example, batch transfers, queries, etc. Include the interface architecture(s) being implemented, such as wide area networks, gateways, etc. Provide a diagram depicting the communications path(s) between this system and each of the other systems, which should map to the context diagrams in Section 1.2.1. If appropriate, use subsections to address each interface being implemented.

**Interface Detailed Design**

For each system that provides information exchange with the system under development, there is a requirement for rules governing the interface. This section should provide enough detailed information about the interface requirements to correctly format, transmit, and/or receive data across the interface. Include the following information in the detailed design for each interface (as appropriate):

* The data format requirements; if there is a need to reformat data before they are transmitted or after incoming data is received, tools and/or methods for the reformat process should be defined
* Specifications for hand-shaking protocols between the two systems; include the content and format of the information to be included in the hand-shake messages, the timing for exchanging these messages, and the steps to be taken when errors are identified
* Format(s) for error reports exchanged between the systems; should address the disposition of error reports; for example, retained in a file, sent to a printer, flag/alarm sent to the operator, etc.
* Graphical representation of the connectivity between systems, showing the direction of data flow
* Query and response descriptions

If a formal Interface Control Document (ICD) exists for a given interface, the information can be copied, or the ICD can be referenced in this section.

**SYSTEM INTEGRITY CONTROLS**

Sensitive systems use information for which the loss, misuse, modification of, or unauthorized access to that information could affect the conduct of State programs, or the privacy to which individuals are entitled.

Developers of sensitive State systems are required to develop specifications for the following minimum levels of control:

* Internal security to restrict access of critical data items to only those access types required by users
* Audit procedures to meet control, reporting, and retention period requirements for operational and management reports
* Application audit trails to dynamically audit retrieval access to designated critical data
* Standard Tables to be used or requested for validating data fields
* Verification processes for additions, deletions, or updates of critical data

Ability to identify all audit information by user identification, network terminal identification, date, time, and data accessed or changed.

**SYSTEM TESTING**

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

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

**TYPES OF TESTS**

**Unit testing**

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

**Integration testing**

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

**Functional test**

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

Functional testing is centered on the following items:

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

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

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

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

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

**System Test**

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

**White Box Testing**

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

**Black Box Testing**

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

**6.1 Unit Testing:**

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

**Test strategy and approach**

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

**Test objectives**

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

**Features to be tested**

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

**6.2 Integration Testing**

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

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

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

**6.3 Acceptance Testing**

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

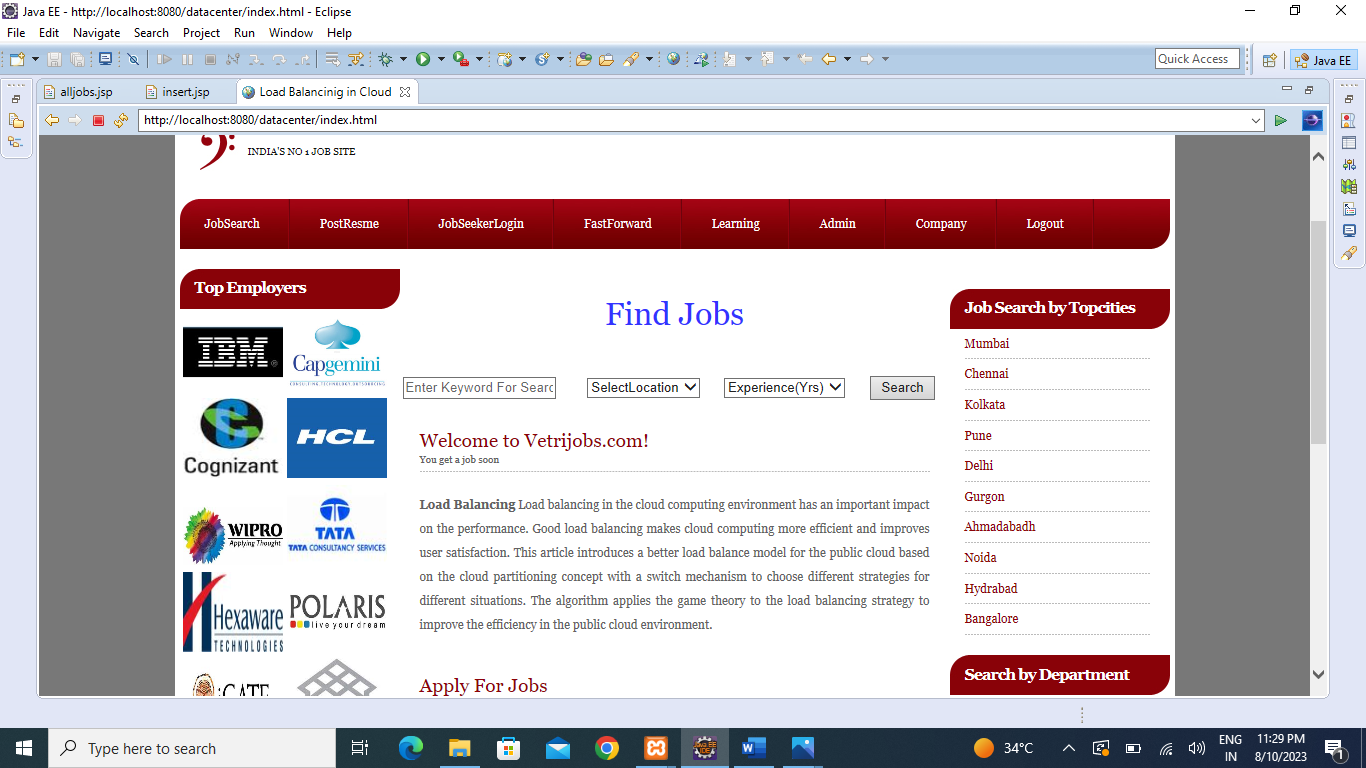
**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

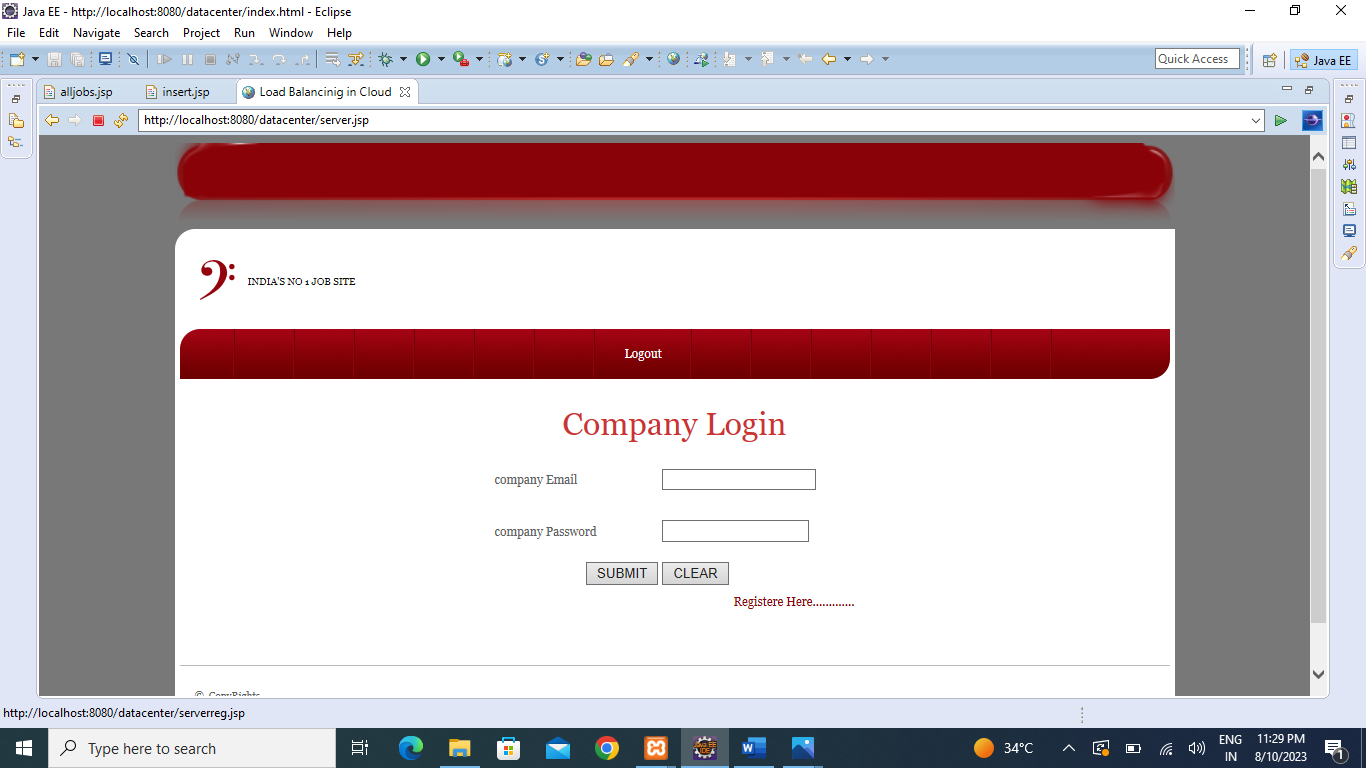
**Future Work:**

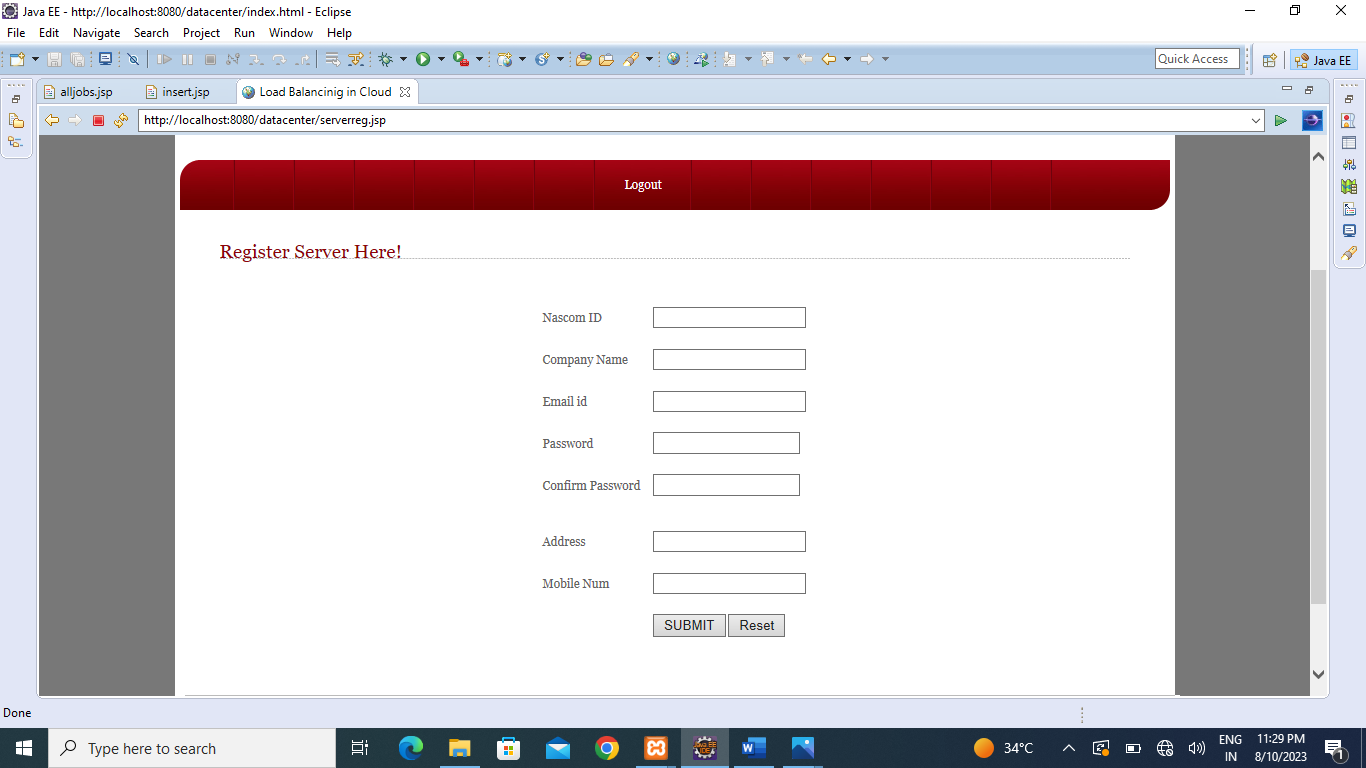
1. Integration of Machine Learning: Explore the integration of machine learning techniques to improve the performance and adaptability of the optimization scheme. Machine learning algorithms can learn from historical data and make predictions or recommendations for scheduling decisions. This can enhance the accuracy of resource utilization predictions, energy consumption estimations, and job performance forecasts.
2. Dynamic and Real-time Scheduling: Develop algorithms and methodologies for dynamic and real-time job scheduling. This involves continuously monitoring the data center's workload, resource availability, and energy consumption patterns and making scheduling decisions in real-time to adapt to changing conditions. Dynamic scheduling algorithms can optimize resource allocation and job placement on the fly, taking into account fluctuating workloads and varying resource demands.
3. Consideration of Heterogeneous Data Centers: Extend the optimization scheme to handle heterogeneous data centers that consist of a mix of different hardware architectures, such as CPUs, GPUs, and specialized accelerators. Develop scheduling algorithms that consider the specific characteristics and capabilities of different hardware resources to optimize job placement and resource allocation in a heterogeneous environment.
4. Multi-objective Trade-off Analysis: Improve the trade-off analysis capabilities of the optimization scheme by incorporating advanced decision-making techniques. This could involve developing methods for decision support, visualization, and interactive exploration of the Pareto-optimal solutions. Administrators should be able to understand and compare different trade-offs between objectives, explore sensitivity analysis, and make informed decisions based on their preferences and priorities.Energy-Aware
5. Task Mapping: Investigate task mapping strategies that consider energy-awareness at the task level. This involves mapping individual tasks of a job to specific computing resources based on their energy efficiency characteristics. By assigning tasks to energy-efficient resources, overall energy consumption can be reduced, leading to improved sustainability.

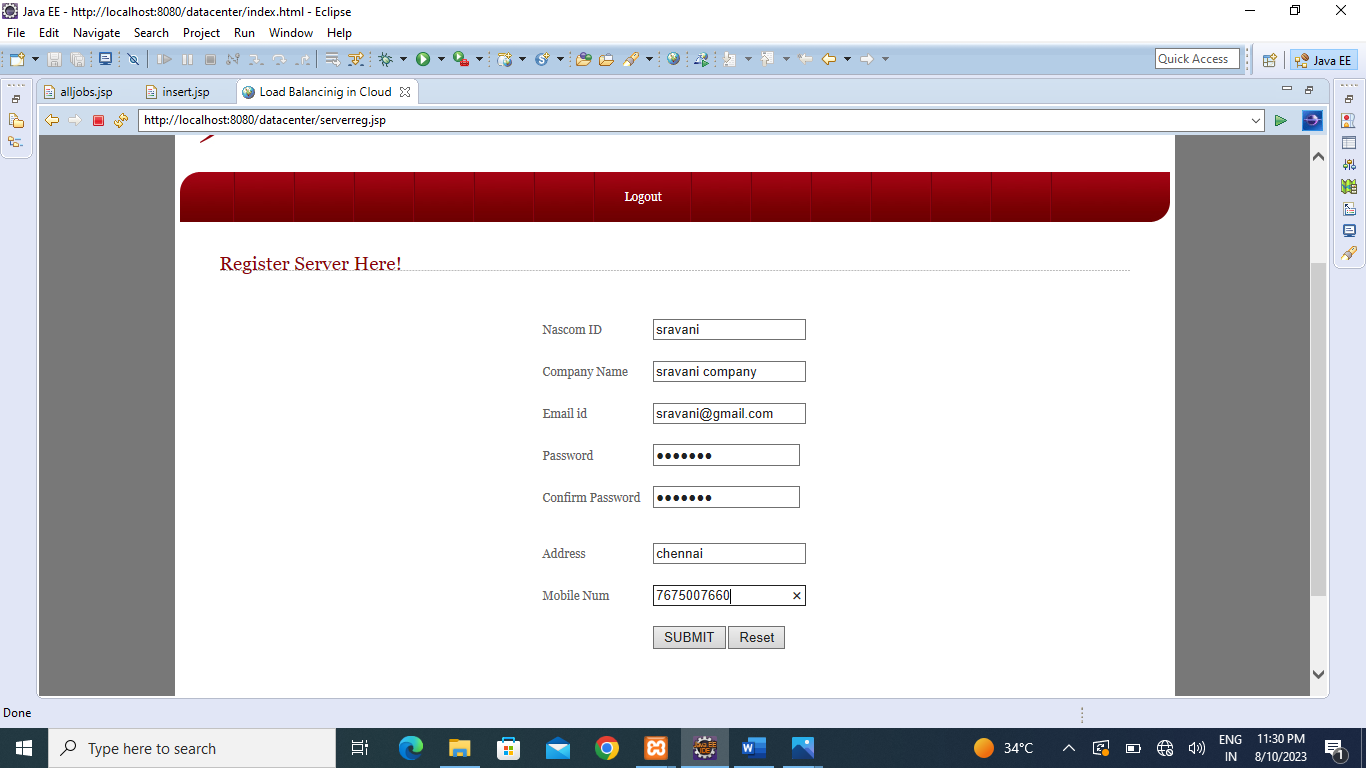
**SCREEN SHORTS**

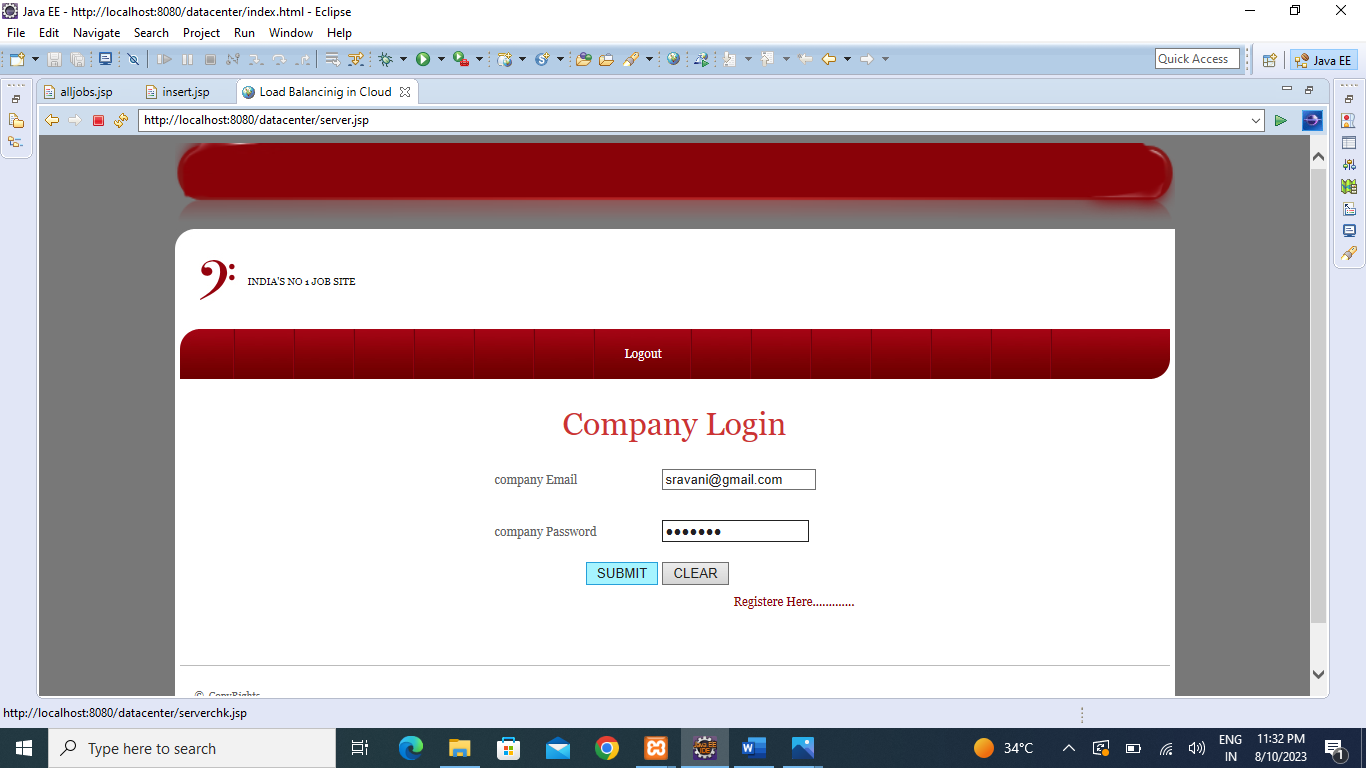


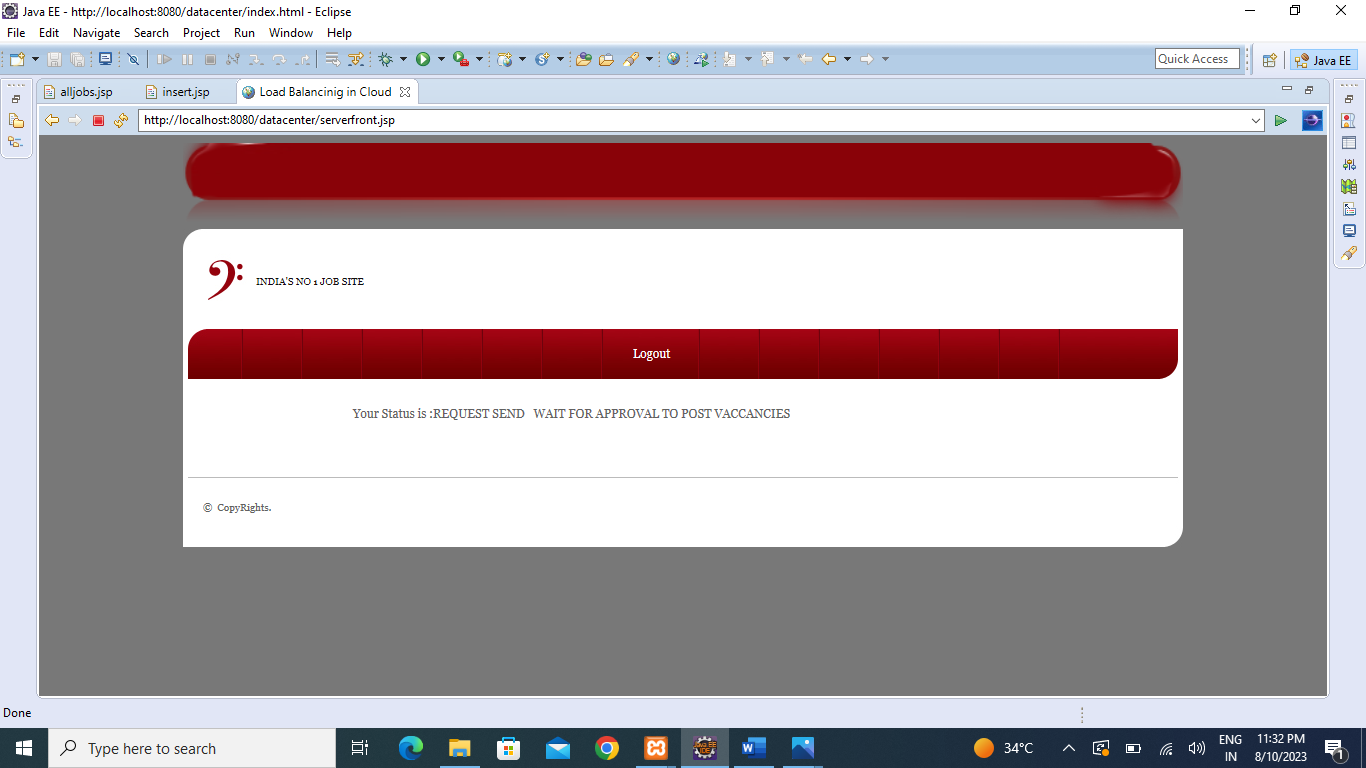


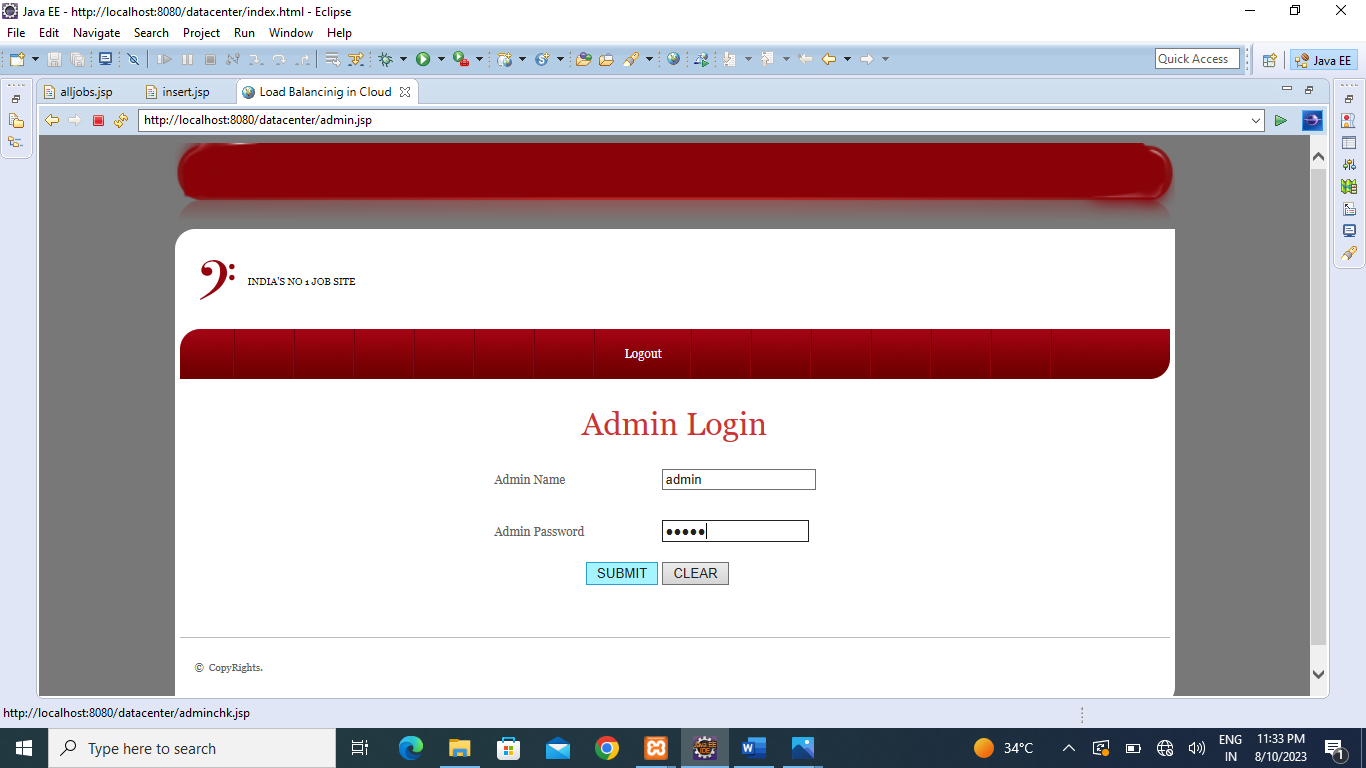


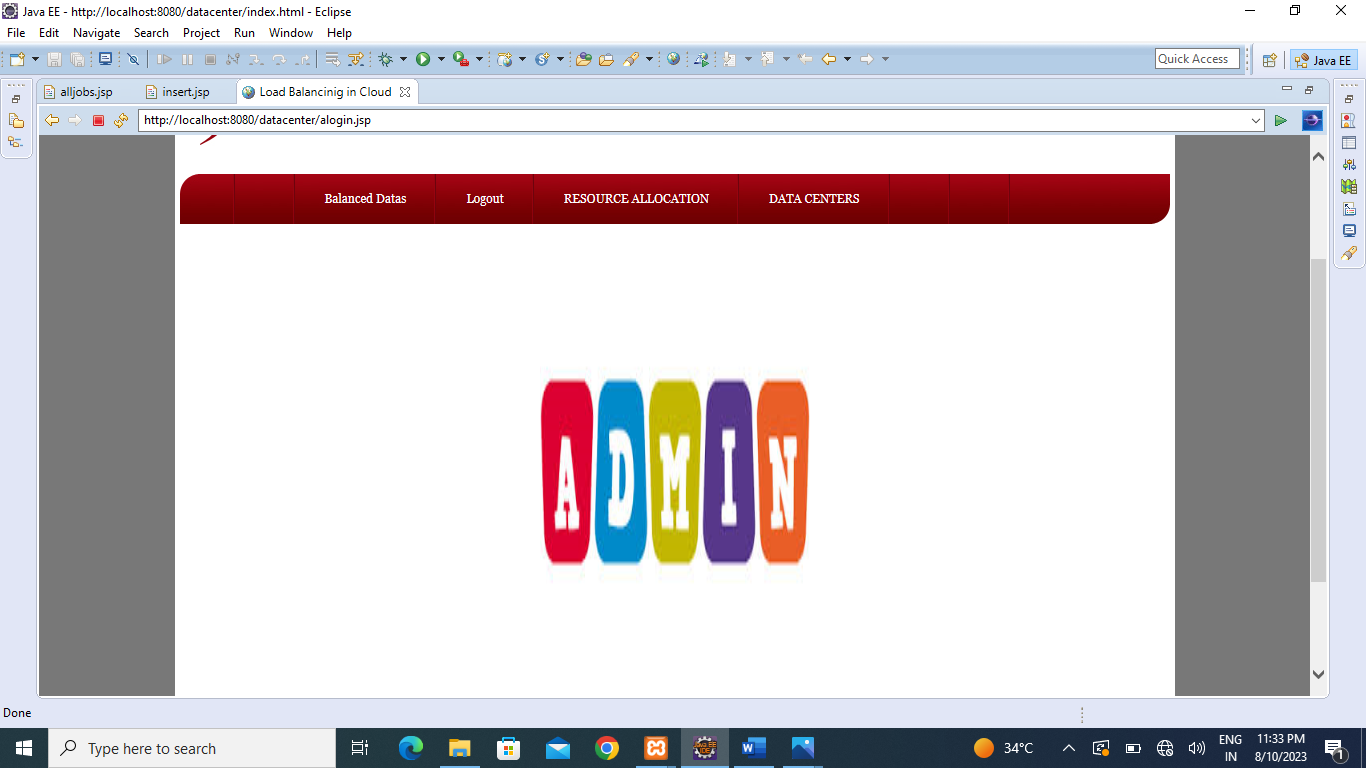




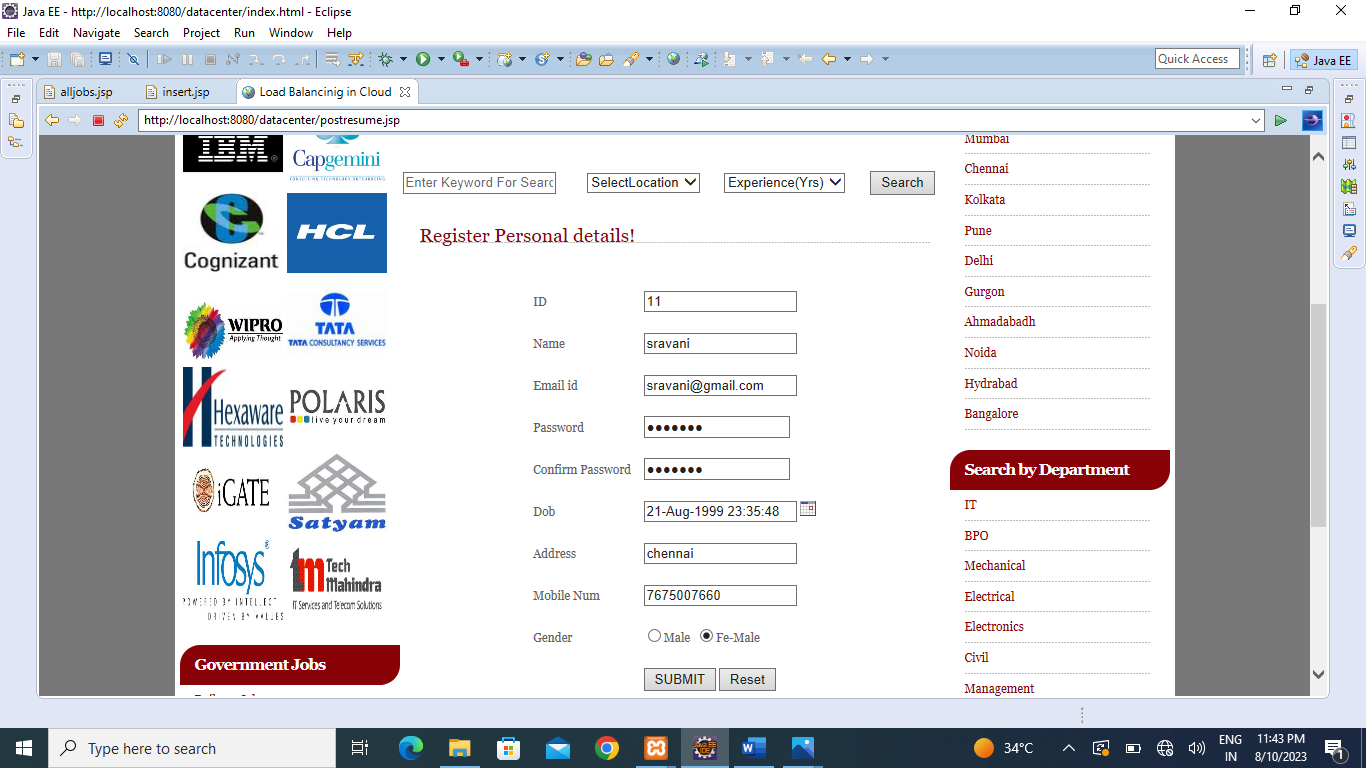


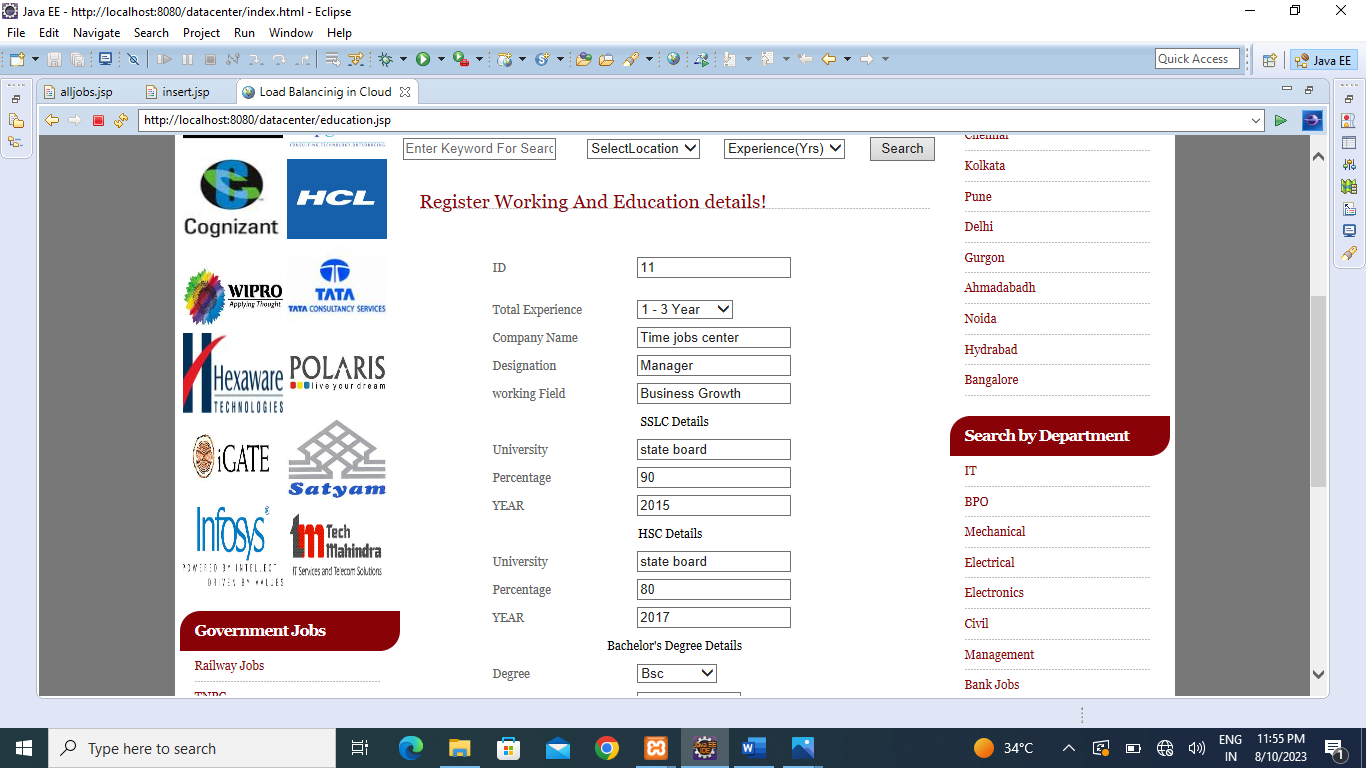


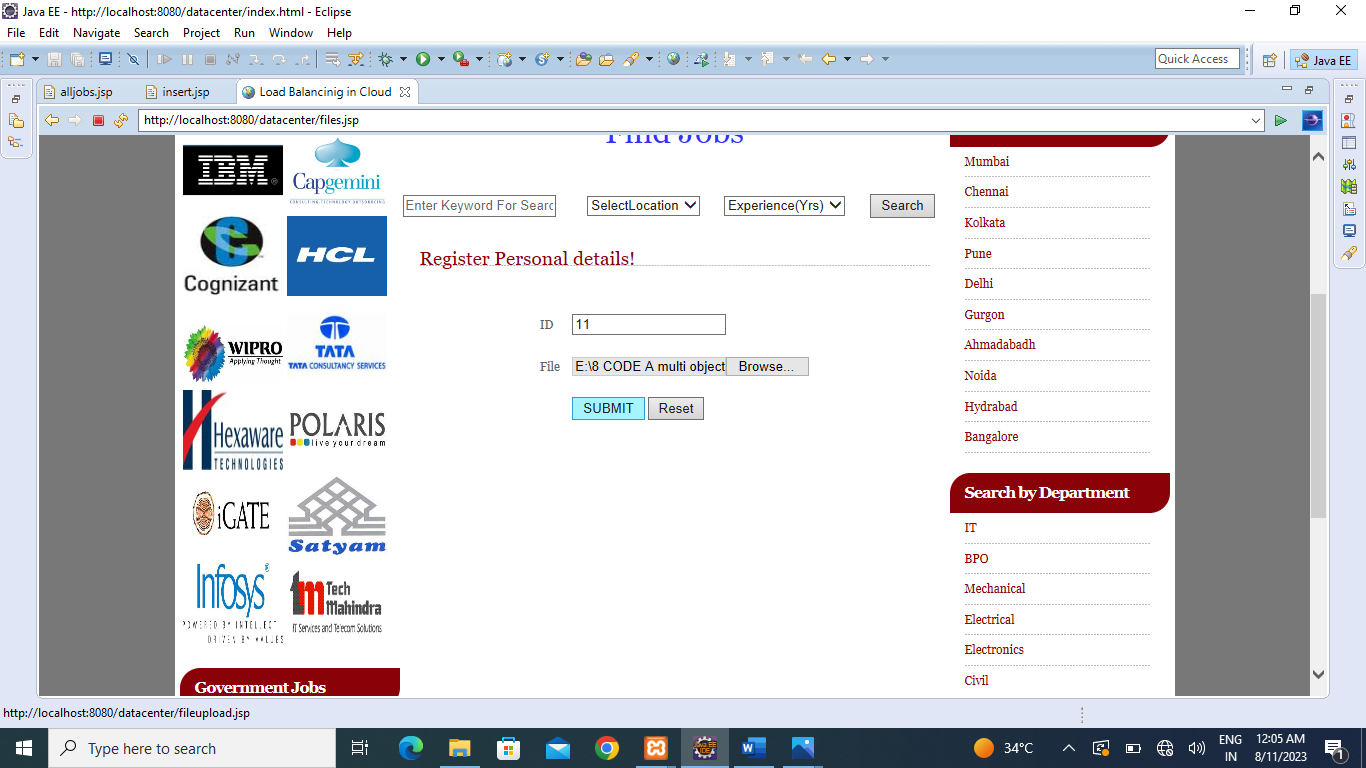


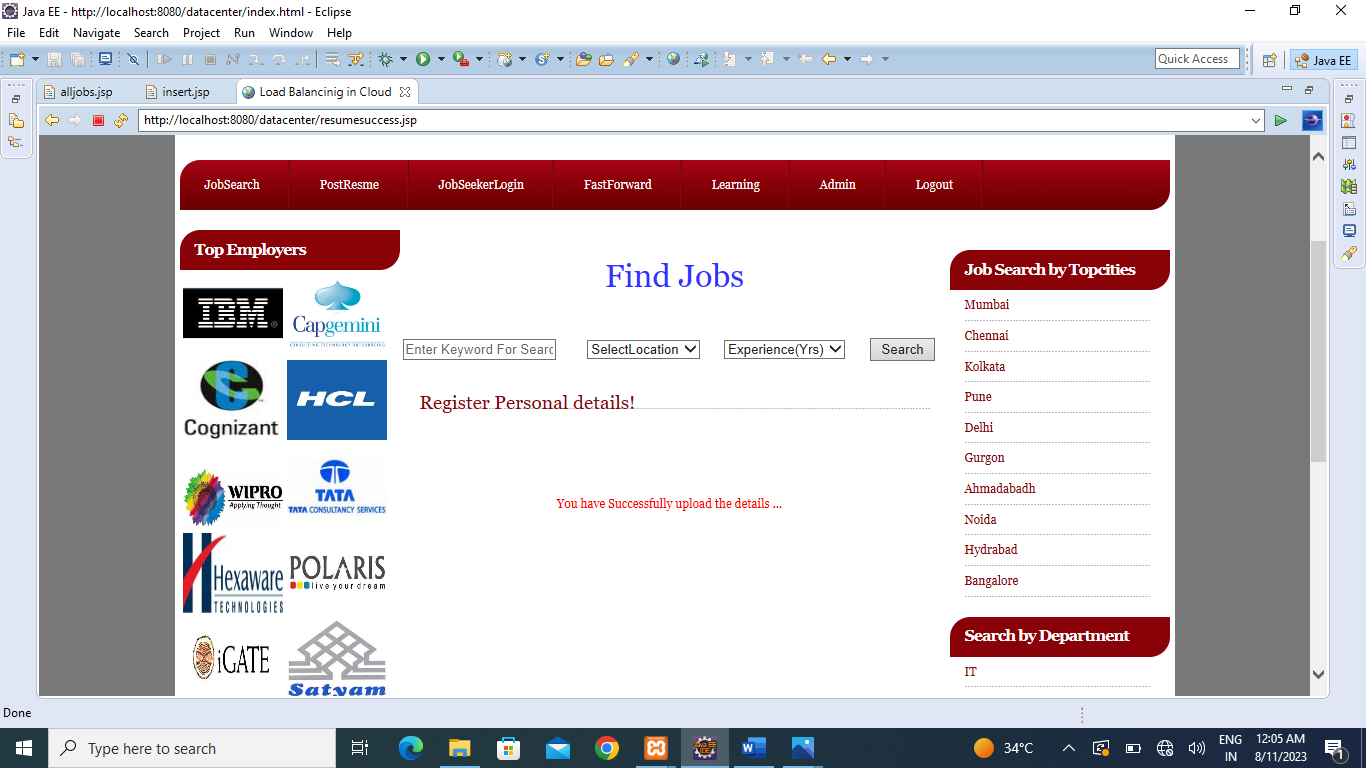


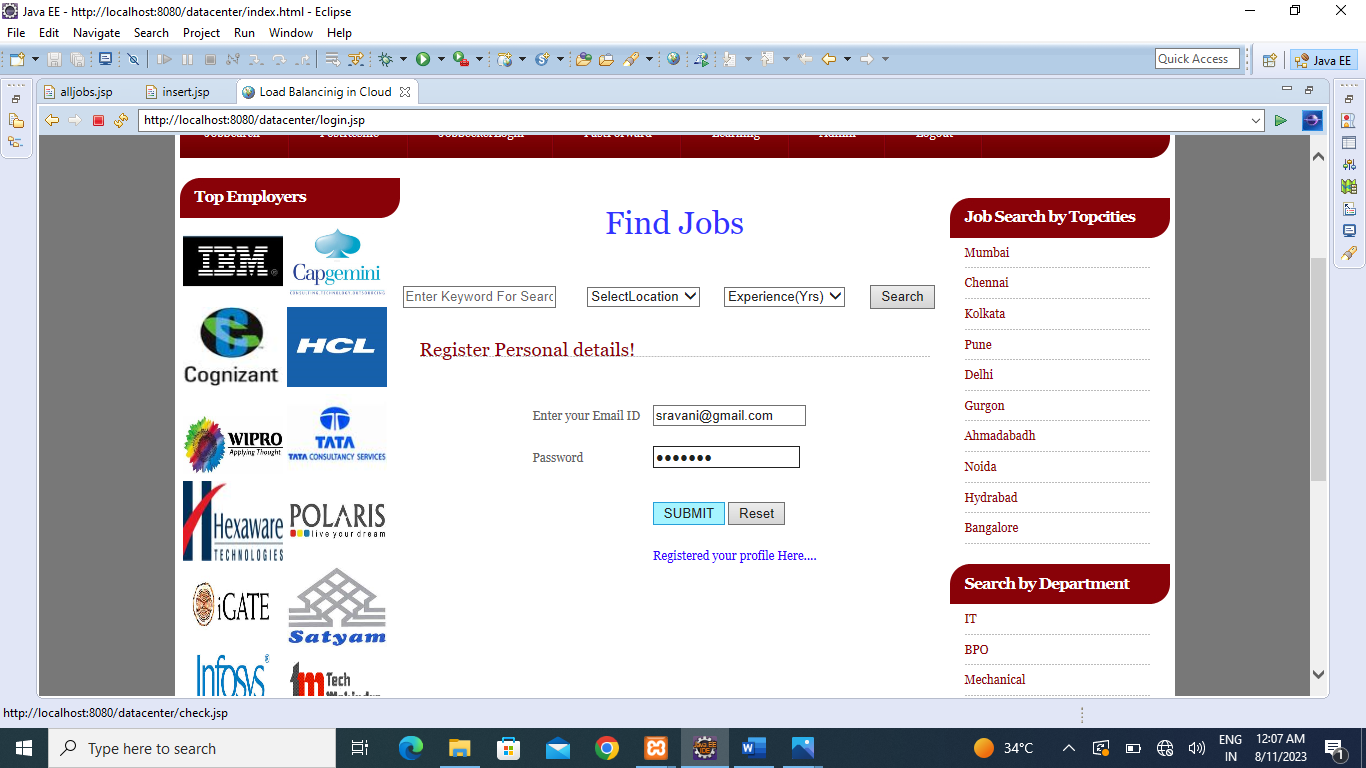


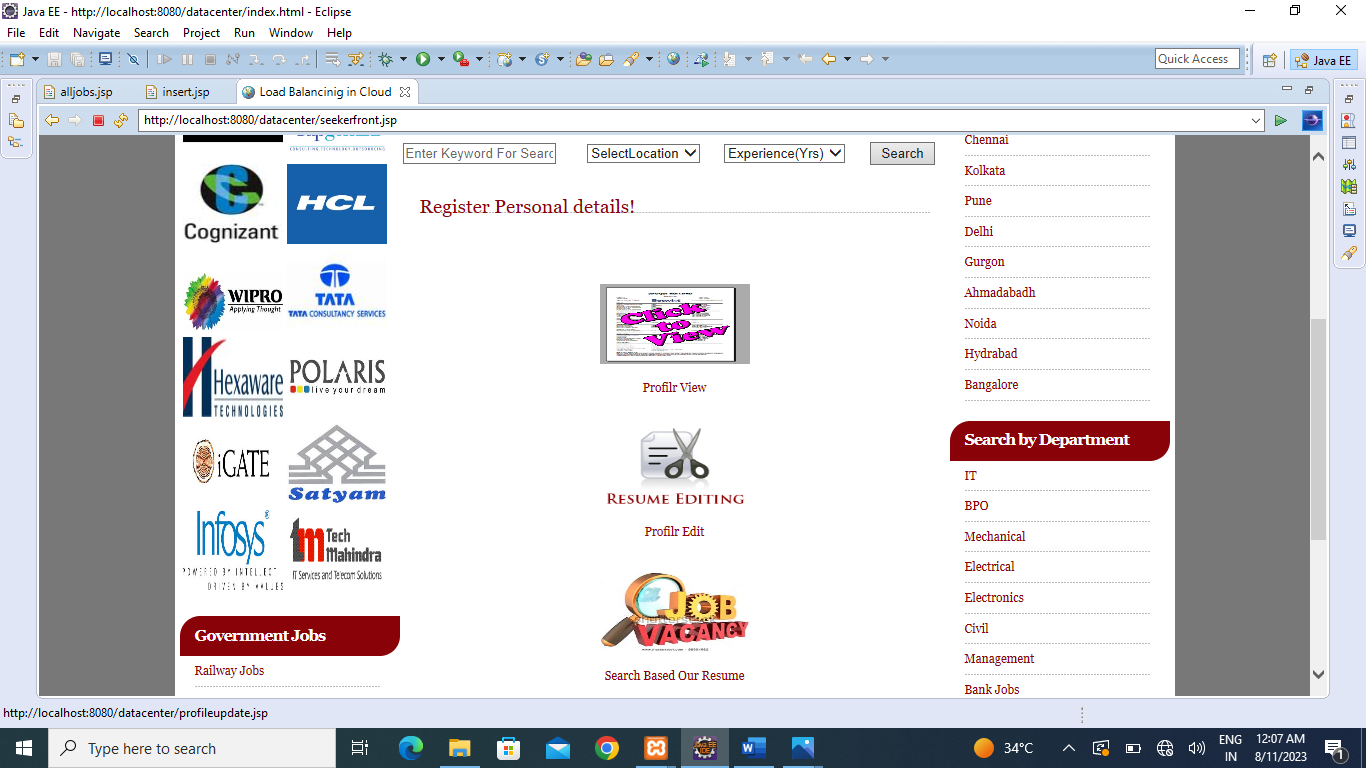


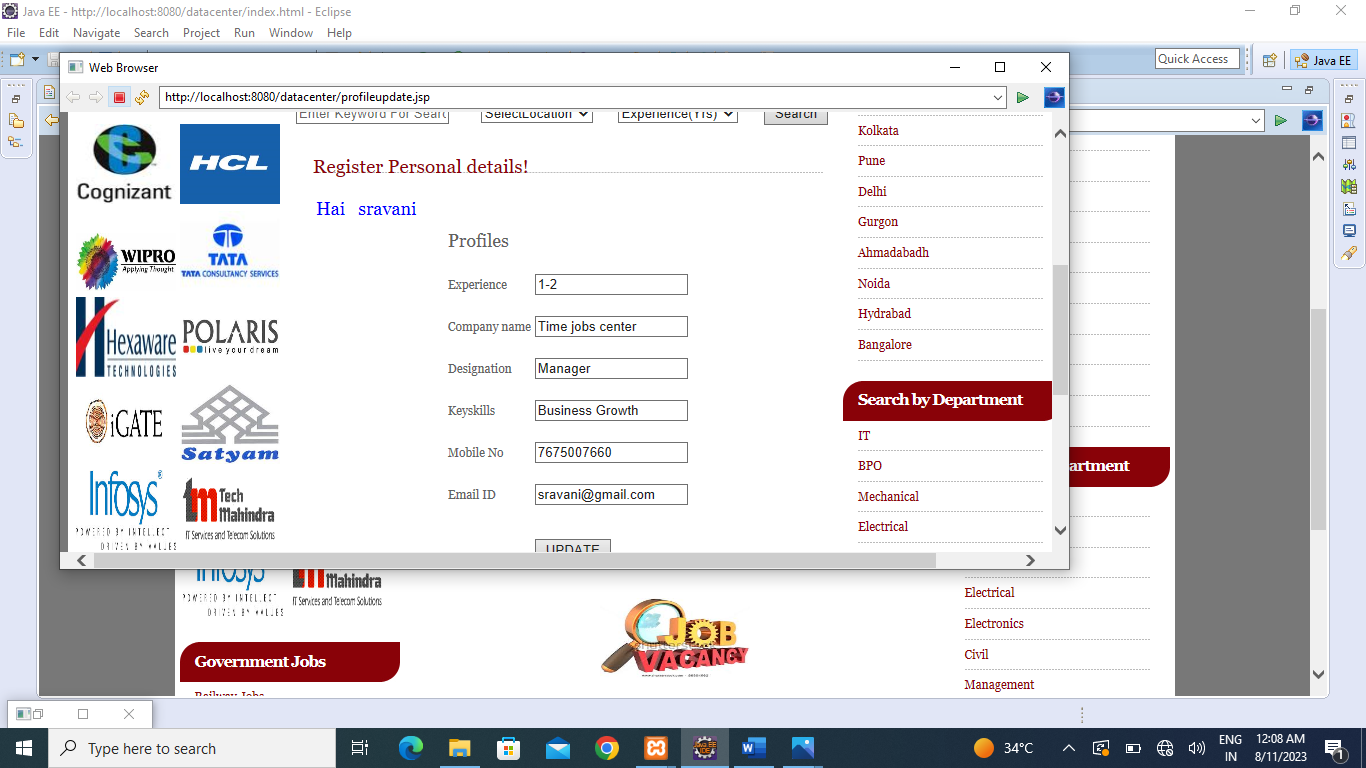


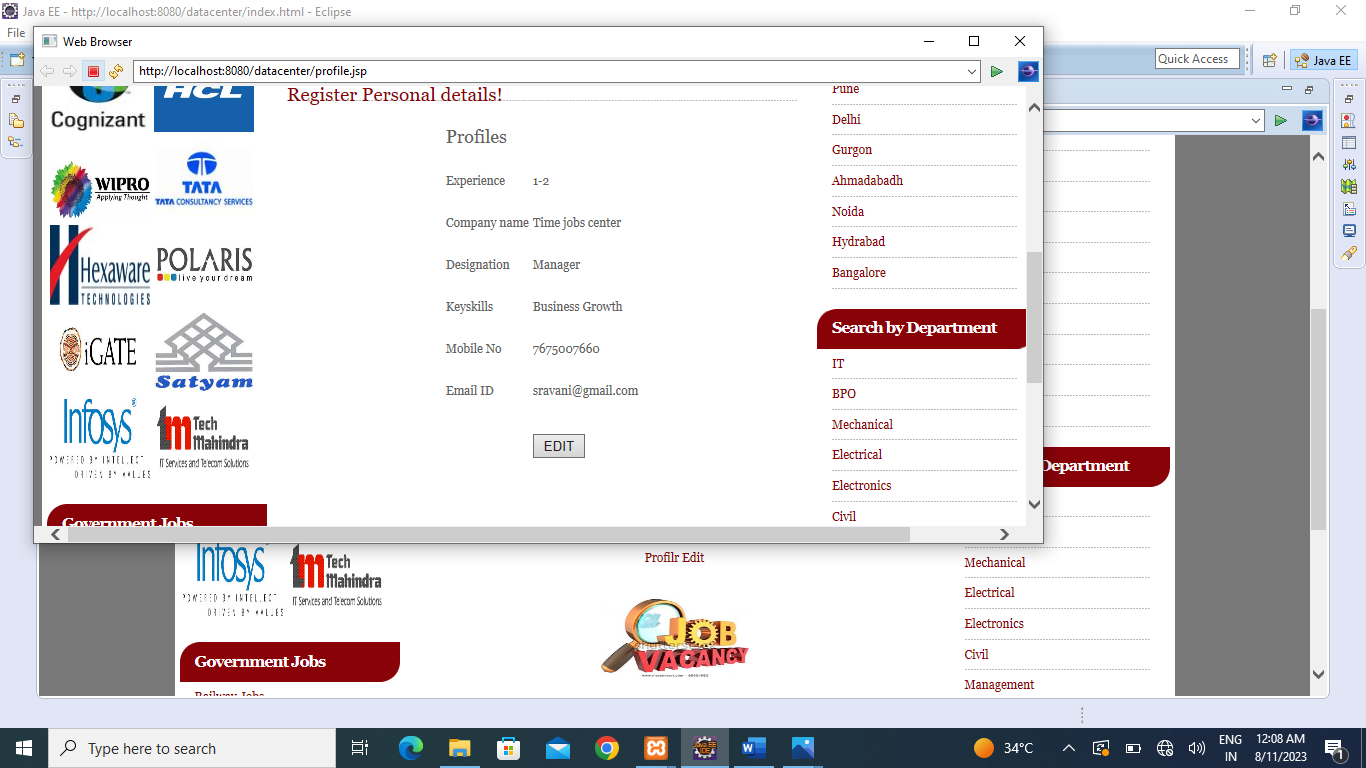


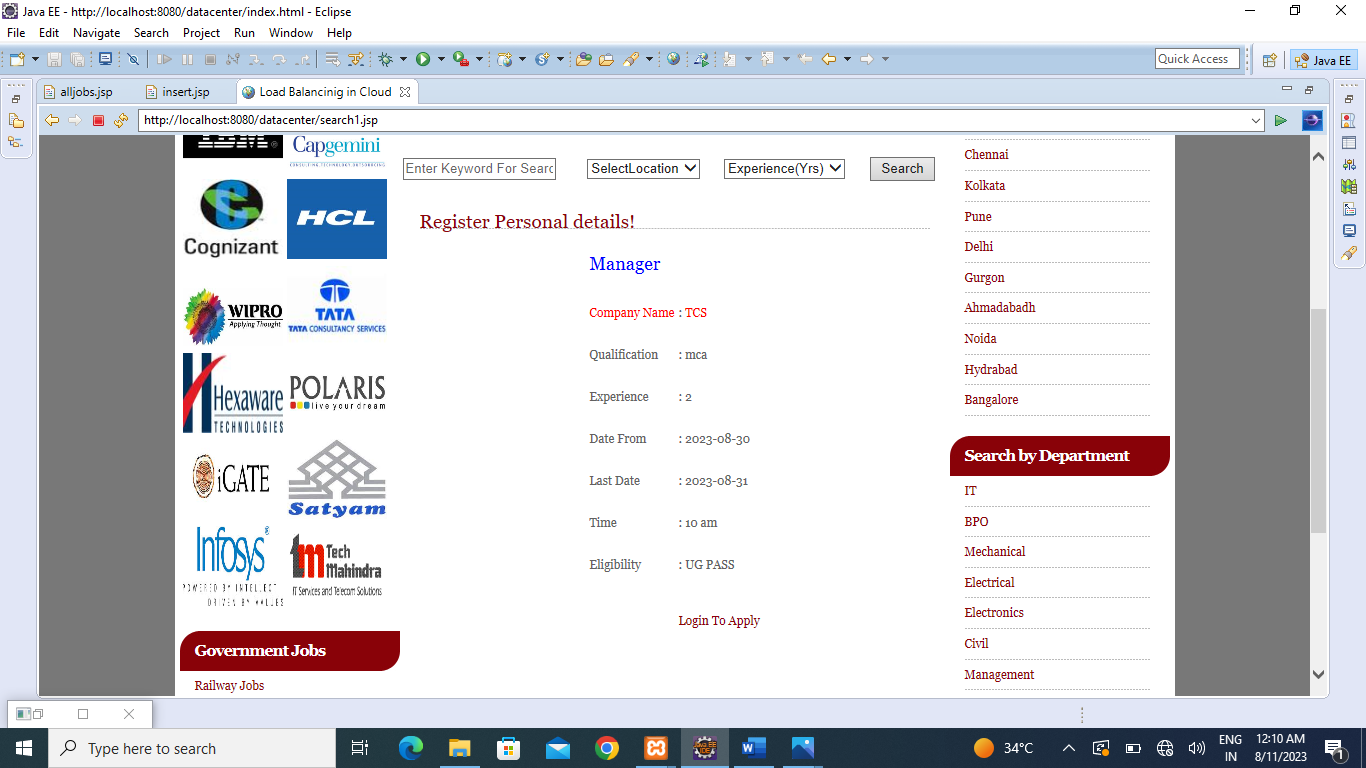


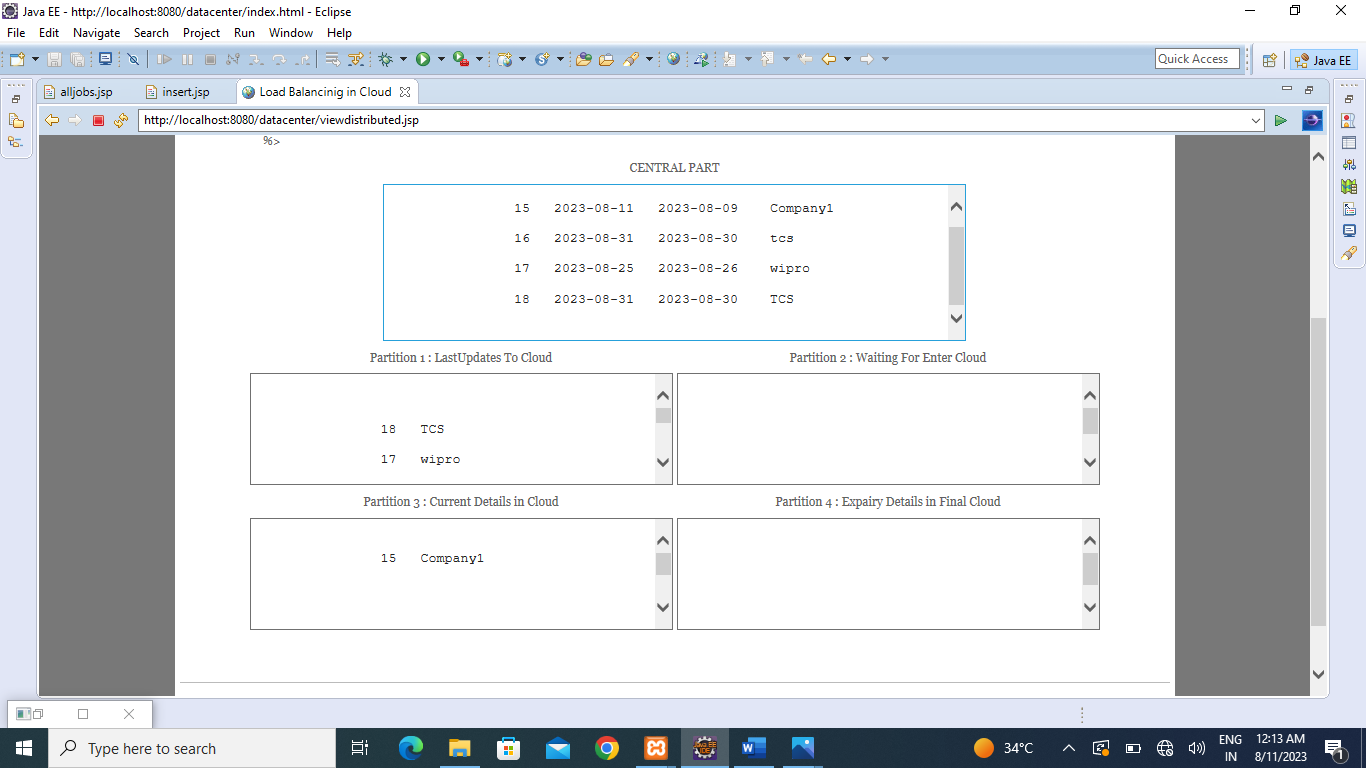


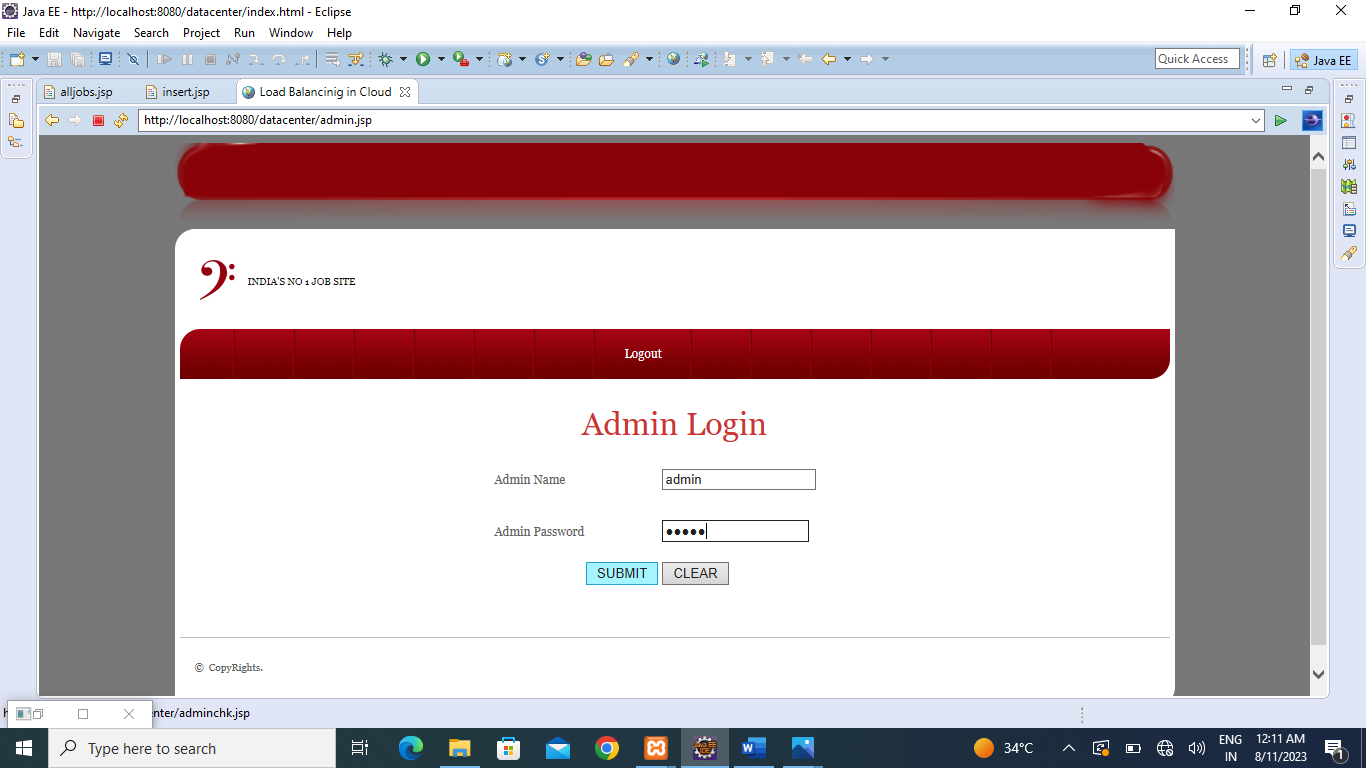
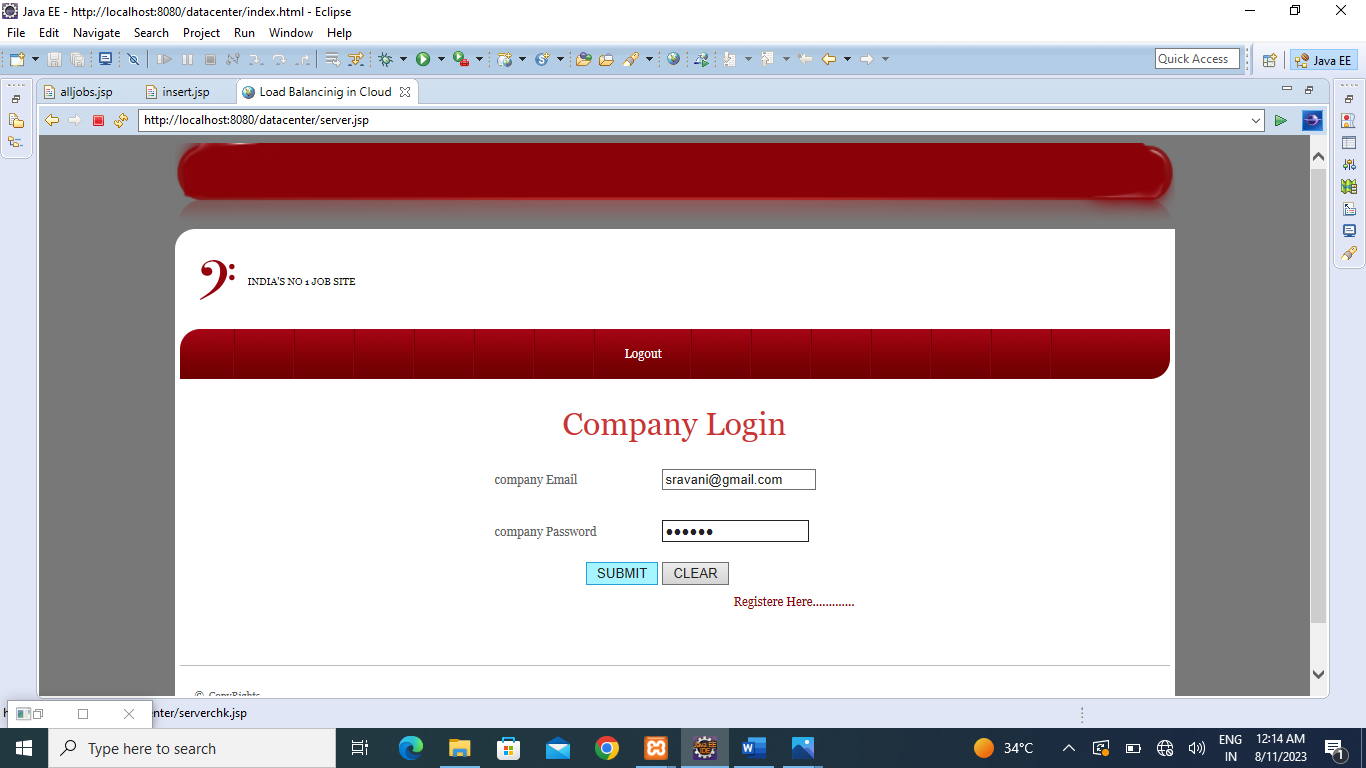


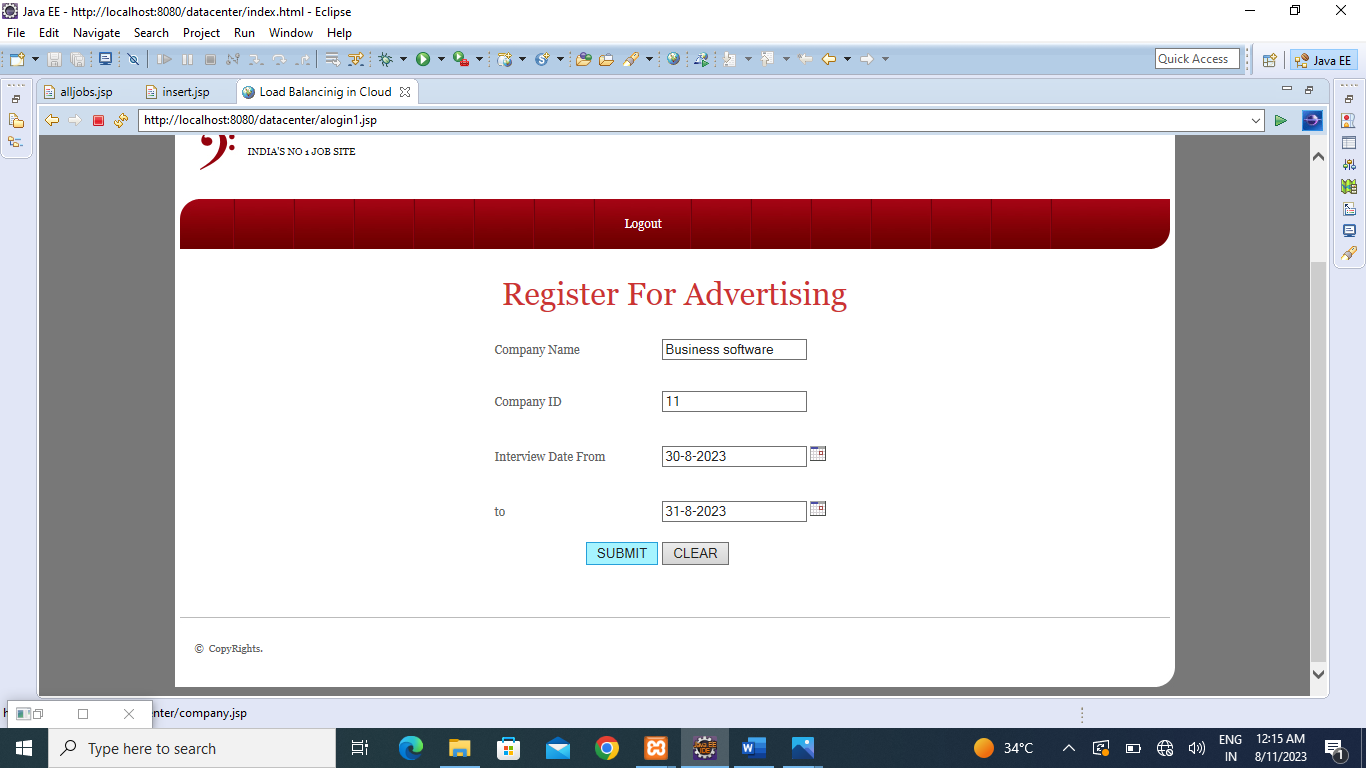
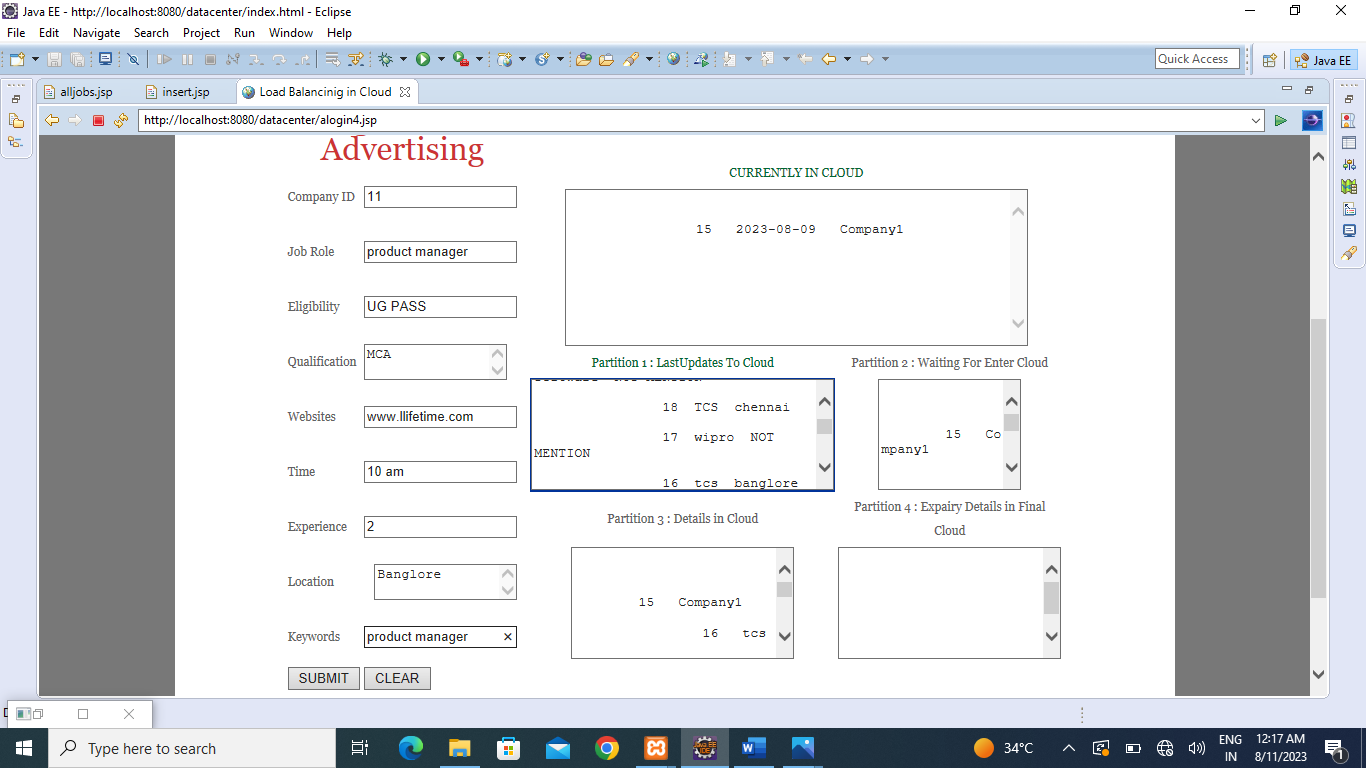


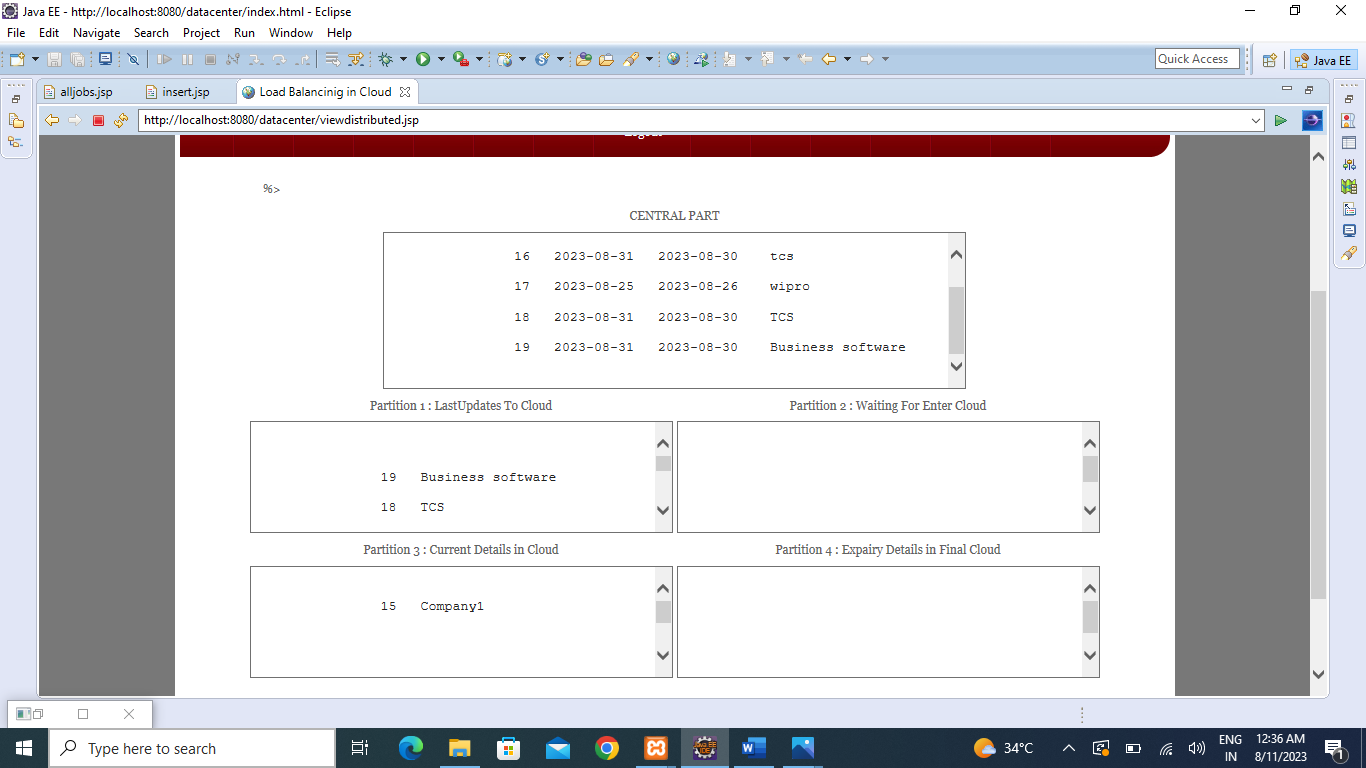


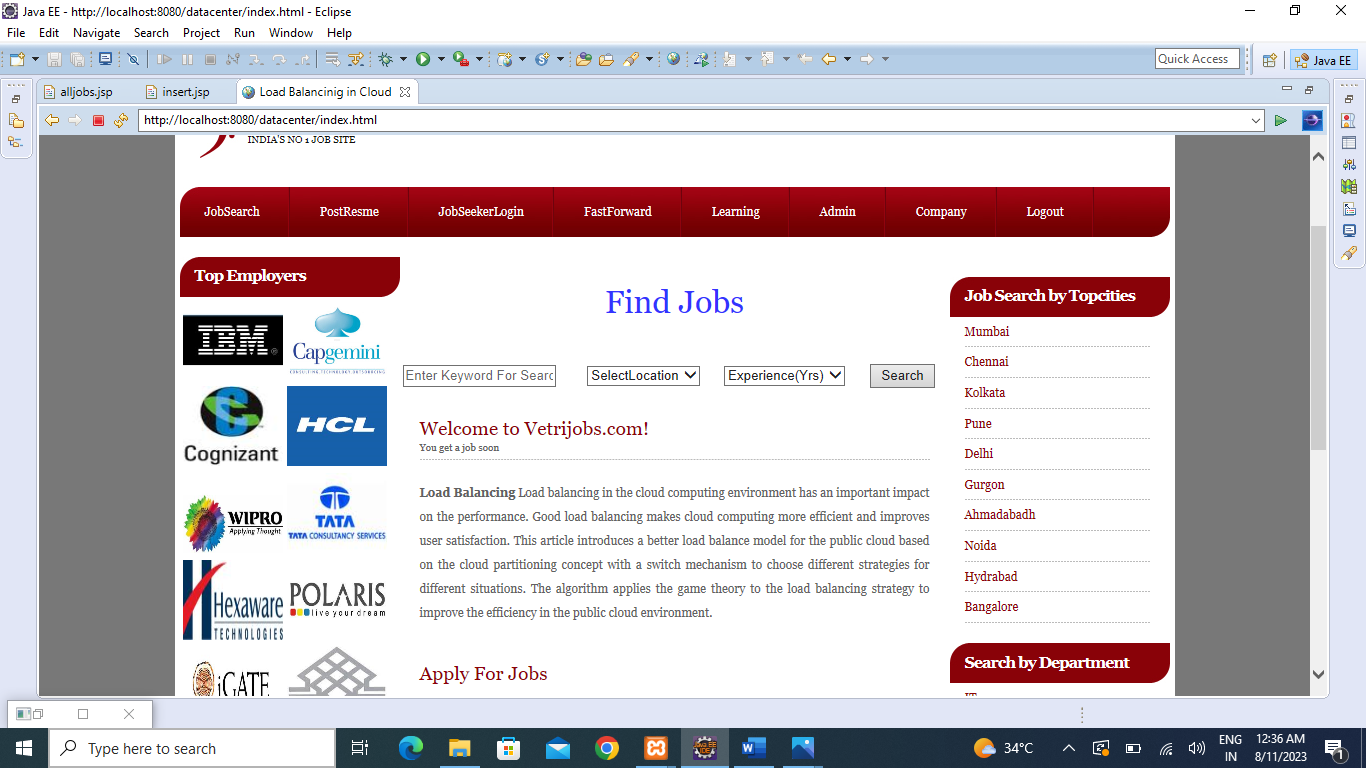












**Conclusion**

In conclusion, a multi-objective optimization scheme for job scheduling in sustainable cloud data centers offers significant advantages for improving resource efficiency, energy consumption, job performance, and overall sustainability. By considering multiple objectives simultaneously, such as minimizing energy consumption, maximizing resource utilization, reducing job completion time, and minimizing carbon footprint, the scheme enables decision-makers to find trade-offs and make informed scheduling decisions that align with their specific priorities and requirements.

The scheme utilizes mathematical modeling and optimization algorithms, such as the Non-dominated Sorting Genetic Algorithm-II (NSGA-II), to explore the solution space and generate a set of Pareto-optimal solutions..

By integrating sustainability considerations into the optimization scheme, such as renewable energy utilization, load balancing techniques, and dynamic resource allocation, the scheme contributes to reducing the environmental impact of cloud data centers. It promotes the efficient use of resources, decreases energy consumption, and lowers carbon emissions, thereby advancing the goal of achieving sustainable cloud computing.

The scheme's effectiveness and efficiency can be evaluated through performance metrics, simulations, or experiments using real-world data centers. Through validation and benchmarking, researchers and practitioners can assess the scheme's performance, scalability, and practicality in different scenarios and compare it with existing scheduling algorithms.

Future research directions may include integrating machine learning techniques, dynamic and real-time scheduling strategies, handling heterogeneous data centers, considering multi-cloud optimization, and enhancing scalability and efficiency. By addressing these areas, the multi-objective optimization scheme can continue to evolve and adapt to the evolving needs of cloud data centers, leading to improved sustainability, resource utilization, and performance.

In summary, a multi-objective optimization scheme for job scheduling in sustainable cloud data centers offers a systematic and effective approach to address the complex challenges of resource management and sustainability. It enables data center operators to optimize job scheduling decisions, reduce energy consumption, and enhance overall efficiency while considering multiple objectives and sustainability goals.

**BIBLIOGRAPH**

Good Teachers are worth more than thousand books, we have them in Our Department **References Made From:**

1. User Interfaces in C#: Windows Forms and Custom Controls by Matthew MacDonald.

2. Applied Microsoft® .NET Framework Programming (Pro-Developer) by Jeffrey Richter.

3. Practical .Net2 and C#2: Harness the Platform, the Language, and the Framework by Patrick Smacchia.

**REFERENCE**

[1] C. Chang, S. N. Srirama, and R. Buyya, “Mobile cloud business process management system for the internet of things: A survey,” ACM Comput. Surv., vol. 49, no. 4, pp. 70:1–70:42, 2017.

[2] M. Graiet, A. Mammar, S. Boubaker, and W. Gaaloul, “Towards correct cloud resource allocation in business processes,” IEEE Trans. Services Computing, vol. 10, no. 1, pp. 23–36, 2017.

[3] S. Schulte, C. Janiesch, S. Venugopal, I. Weber, and P. Hoenisch, “Elastic business process management: State of the art and open challenges for BPM in the cloud,” Future Generation Comp. Syst., vol. 46, pp. 36–50, 2015.

[4] U. Riemann, “Benefits and challenges for business process management in the cloud,” International Journal of Organizational & Collective Intelligence, vol. 5, no. 2, pp. 80–104, 2015.

[5] W. Song and H.-A. Jacobsen, “Static and dynamic process change,” IEEE Trans. Services Computing, vol. 11, no. 1, pp. 215–231, 2018.