Grid computing is a collection of computing resources that are collected from multiple administrative domains to reach a common goal. It enables users to solve complex problems by using distributed heterogeneous resources. It provides access to computing resources such as processors, storage, data and applications as needed with little or no knowledge of the location of those resources or the underlying technologies. Grids can be constructed with general-purpose grid middleware software libraries. A computational grid aggregates the processing power from a distributed collection of resources.

Grid scheduling is a thrust area of research since it affects the overall performance of the grid system. In order to perform the scheduling process, the grid scheduler has to be implemented. Grid scheduler is a mediate resource manager which acts as an interface between the consumers and the underlying resources.

Load balancing is a workload management system for distributing workloads across multiple computers or computer clusters, network links and other resources. This feature must always be integrated into all grid systems in order to minimize processing delays and over commitment of resources. It can be built in connection with schedulers. Since grid resources are extremely heterogeneous from multiple administrative domains, there are possibilities
for failures. Fault tolerance is the feature of a system that continues service consistent with its specifications even in the event of failure.

The main objective of this research is to develop an efficient scheduling algorithm which minimizes the makespan and waiting time of the jobs and improves resource utilization. The algorithms are designed (i) to provide better user satisfaction by considering the user deadline of jobs (ii) to achieve better resource utilization by considering the load factor of the resources (iii) to improve the reliability of resources by considering their failure rate that makes scheduling more efficient (iv) to reduce makespan by reducing waiting time of jobs.

To improve user satisfaction, a new prioritized user demand algorithm is proposed which mainly focuses on the deadlines of the jobs as expected by the users. It takes user’s deadlines into account and makes the job to be executed within the expected deadline by assigning it to the most suitable resource. User demand aware scheduling algorithm for data intensive applications is also proposed which considers the data transfer requirements along with user deadline of the jobs. This algorithm includes data transfer time required for the jobs while calculating expected completion time of a job on a resource. It allocates jobs to the resources based on user deadline and expected completion time.

The resources in grid are dynamic in nature. So the availability time of the resources needs to be considered for effective utilization of the resources. To reduce scheduling overhead and communication overhead and
improves resource utilization, a grouping based user demand aware job scheduling algorithm is proposed in which jobs are grouped based on capacity and availability time in order to utilize the resources effectively. This reduces communication overhead and improves the utilization of resources.

Hierarchical load balancing approach is proposed which considers load of each resource and performs load balancing. It considers dynamicity of the resource i.e. the availability time of the resources. Since load is shared effectively among all the resources in the grid, it minimizes the response time of jobs and improves the utilization of resources in grid environment.

Resource failures may frequently occur in grid systems with an adverse effect on jobs. To handle resource failure, a fault tolerant load balancing scheduling algorithm is proposed. It considers failure rate of the resources to predict job failures before scheduling. The algorithm also considers load of each resource while scheduling the jobs. Since it considers load and fault rate of the resources, it increases resource utilization, system throughput and minimizes the completion time of the jobs. Performance of the various proposed algorithms is compared with existing methods and the experimental results show that the proposed algorithms have better results over the other algorithms.