MACHINE JOB SCHEDULING

High Level Design & Low Level Design

**Document Control :**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MACHINE JOB SCHEDULING | | | | | | | | |
| Guided by-**SANKAR SIR** |  |  | |  |  |  |  |  |
| **Date** | **Version** | **Author** | **Brief Description of Changes** | | | | **Approver Signature** | |
| December 07,2022 | 1 |  |  | | | |  | |
|  |  |  |  | | | |  | |
|  |  |  |  | | | |  | |

# 1.1 Introduction

* The introduction of the software requirement specification provides an overview of the entire software.
* The entire SRS with overview description purpose, scope, tools used and basic description.
* The aim of this document is to gather, analyze and give an in-depth insight into the complete
* Machine Job Scheduling application by defining the problem statement in detail.
* The detailed requirements of the Machine Job Scheduling application is provided in this document.

**1.2 Intended Audience:**

This document is intended to be read by, Client.

* 1. **Project Purpose:**

The purpose of this document is to show the requirements for the

“Machine Job Scheduling Application”, in which we will process jobs data and assign requested machines to the jobs then schedule for each machine is generated.

**2.1 Design Objective**

This project attempts to assure a seamless machine task scheduling procedure.

# When compared to previous ways, it enables the user to have effective management

# with little resources, delivering more efficiency in less time.

**2.2 Performance:**

The system will work on the admin terminal. The performance depends on the

hardware component of the admin’s system.

**2.3 Maintenance:**

* If maintenance demands consistently keep the Team from completing their Sprint Plan, stop planning for so much. In Sprint Planning, leave some headroom – an allowance for maintenance. Reduce the forecast for new feature work. The size of the allowance may be easy to determine from past Sprints or it may take experimentation. If the Team does not need all of the time budgeted in a Sprint, they can us it for more feature work or payment of technical debt.
* When an allowance is made for maintenance, we can take turns handing it. Team members can rotate in the role of “fixer” from Sprint to Sprint so that no one gets stuck with the cleanup work. The fixer can manage their own time between the maintenance and new work.
* This is the ultimate solution. Use proper development practices – pairing, test-first development, automated acceptance testing, continuous refactoring. Make the code better every day. Wrap the system in automated tests to make bug-hunting easier.

**2.4.1 Functional** **Overview :-**

**3.4.1 MJS\_01->:** void readFile() :- This function is used to read content of the file using function fgets().

Open a file using function fopen() and store reference of the file in a file pointer.

File close the file using the function fclose()

Fgets()—This function is used to read strings from the files.

**3.4.2 MJS\_02->** void findDisplayAndRemoveInavlidJobs() :- this function is used to find invalid jobs and write them in invalidJobs and display them.

**3.4.3 MJS\_03-> void scheduleJob()** :-This function is used to schedule jobs to machines based on JobList, Job count,

**3.4.4 MJS\_04-> void groupScheduleList() : -** This groupScheduleList function is based on machines by using scheduleList, scheduleCount,machineScheduleList, machineScheduleCount.

**3.4.5 MJS\_05-> void writeSchedule() :-** This function is used to write schedule from groupScheduleList.

**2. Design Overview:**

Instant Chatters comprises of the following modules:

|  |  |
| --- | --- |
| Name of the Module | Void readFile() |
| Handled by | Vyshnavi Kotala |
| Description | It will login the user at the designated position. |

|  |  |
| --- | --- |
| Name of the Module | findDisplayAndRemoveInvalidJobs() |
| Handled by | Swetha Kunduru |
| Description | This feature will load the details of the userstories and of the tasks in the respective files. |

|  |  |
| --- | --- |
| Name of the Module | Void scheduleJob() |
| Handled by | Ashwini Chavan |
| Description | This feature will update the user stories for its task data and other function will update the user stories data in the csv. |

|  |  |
| --- | --- |
| Name of the Module | Void groupScheduleList() |
| Handled by | Aakula Prathyusha |
| Description | It will insert a new user story. |

|  |  |
| --- | --- |
| Name of the Module | Void writeSchedule() |
| Handled by | Sadhana S Kumar |
| Description | It will list all the details of the user . |

**3. Environment Description:**

**3.1 User Desktop Requirements:**

a. 64-bit processor, 1 GHz or faster

b. At least 2 GB free hard drive space

c. At least 1 GB RAM

**3.2 Server-Side Requirements:**

a. 64-bit processor, 1 GHz or faster

b. At least 1 GB free hard drive space

c. At least 1GB RAM

**3.2.1 Deployment Considerations:**

a. Easy setup: no session storage daemon, use tmpfs and memory

caching to enhance performance.

b. Local storage is used.

c. No network latency to consider.

d. To scale buys a bigger CPU, more memory, larger hard drive, or

additional hardware.

**3.2.2 Application Server Disk Space:**

No such disk space is required as the program is fully functional on onlineIDE(s)

as well. The Local Operating System is required and one text file to store the records

of processes.

**3.2.3 Database Server Disk Space:**

No such disk space is required as the program is fully functional on online IDE(s) as well. The Local Operating System is required and one text file to store the records of processes.

**3.2.4 Integration Requirements:**

1. Language: C

2. Tools: Splint ,Valgrind, Makefile

3. Complier: GCC compiler

4. Linux Environment

**3.2.5 Jobs:**

We can establish connections between clients who are connected to the server. And we can search the chat history of the clients.

**3.2.6 Network:** End to End

**3.3 Configuration:**

**3.3.1: Operating System**: Linux environment