Flexi-Intelligent Resource and Scheduling Tool (FIRST) User Guide

# Overview

A Resource Management and Job Scheduling tool that aims to streamline a few work processes (which requires connection between multiple excel sheets) in a manufacturing environment.

These work processes are namely:

1. CRUD of Job and Employee details into a “database”
2. Allocating employee resource to a job (reduces human interface between planner and employee(technician) for job allocation
3. Intelligent Job scheduling
   1. Able to recommend alternative dates in scenario of job clashes
   2. Automatically reschedule jobs in the system in the event of employee attrition/removal
4. Employee resource update (Addition and removal) to account for new hiring/attritions
5. Searching of Job details by Date range or Job ID
6. Cost Summary report by Date Range (for management reporting)

The primary aims are as such:

1. An end-to-end tool for job scheduling, resource management and periodic cost reporting
2. A single source of reference for all accepted and scheduled jobs (eg. For jobs lined up in a certain month)
3. Automation of Job Scheduling and re-scheduling

**Submission Content:**

Submission contains 6 files (excluding folders) in total, as described below:

**Two .py scripts:**

1. main\_code\_base.py
2. class\_functions.py (script containing all the user defined functions and classes)

**Two .csv files\*:**

1. employee.csv (an eg. Of Employee details a company might have)
2. job.csv (an example of scheduled jobs that a company might have)

*\*Note: The .csv files should not be edited as the intent of these files is to “act” as an initial database to load data and assumes that the employee and jobs data are accurate and in the correct format.*

**One .txt file**

1. readme.txt

**One .pdf file:**

1. Scheduling Program User Guide

# Set up

Description of context:

Folder structure is as required in Project Deliverables.

The .csv files act as initial data for initialization of the Tool, with the following data columns described below:

The employee.csv contains information of each employee using the following columns:

* Employee ID – integers only
* First Name
* Last Name
* Hourly Rate – numerical values (SGD$)
* Total Hours per day – numerical values, describing the number of hours an employee has
* Competency – numerical values
* Craft – “Machinery”, “Metals”, or “Instrument/Electrical”

The job.csv database contains information of each job using the following columns:

* Job Name
* Start Date – must be in the format of dd/mm/yyyy, e.g. 15/2/2023
* Completion by Date – must be in the format of dd/mm/yyyy
* Resources required – in man-hours, numerical values only
* Total Cost – numerical values (SGD$)
* Craft - can only be “Machinery”, “Metals”, or “Instrument/Electrical”

These jobs are accurate in terms of the checks between Resources required and Total Hours Per Day in the employee.csv by craft and hence can be scheduled when the Tool is initialized.

Set-up steps:

1. On the Terminal or command line, run script “main\_code\_base”.py
2. Do these initialisation steps (in yellow) 🡪 Option 1: Upload Employee/Job Database [From .CSV only] 🡪 1: Employee database 🡪 Input “Y” to the prompt: Do you want to proceed with another action? Y/N
3. The Calendar Data structure has been initialised, **please proceed with Test Cases below**
4. **(Optional)** Once step 1- 3 is completed, initialising of job database is also available through these steps: 🡪 Option 1: Upload Employee/Job Database [From .CSV only] 🡪 2: Job database 🡪 Input “Y” to the prompt: Do you want to proceed with another action? Y/N

Once the above steps are done, resource and scheduling tool is ready for use and the tool works for the date range of 2022-12-31 to 2042-12-31.

Note: This date range can be extended further based on the datetime in-built python module.

# Classes

\*Classes are defined in the class\_functions.py file, there are a total of two classes

1.Job Class (Public Class)

Attributes:

* job\_id
* job\_name
* start\_date
* due\_date (original planned due date/deadline inputted by User **or** the recommended due date based on resource available after function recommendSchedule() is called)
* resources
* total\_cost
* employees (this is an association relationship with the employee class)
* craft
* scheduled\_end\_date (This is the actual end date of the job scheduled in the system, not necessarily the same as the user input due date/deadline)

Methods:

Nil

2.Employee Class (Private Class)

Attributes:

* emp\_id (Employee ID)
* first\_name
* last\_name
* hourly\_rate
* total\_hours\_per\_day
* competency
* craft

Methods:

* getter/setter methods for all attributes above (instance method)
* CurrentEmployeeList (static method)
* addEmployee (static method)
* removeEmployee (static method) [Contains auto-rescheduling algorithm]

# Functions

Overview of Functions defined in class\_functions.py:

1. createCalendarRange (start\_date, end\_date, calendar\_resource\_dict, list\_of\_employees)
   1. Takes in 4 parameters and creates a calendar data structure if calendar\_resource\_dict is an empty dictionary
   2. Calendar data structure is of three-dimensions of this format:

{start\_date: [{emp\_id: total\_work\_hours\_per\_day, “Craft”: Employee’s Craft}], start\_date +1 (day) : [{emp\_id: total\_work\_hours\_per\_day, “Craft” : Employee’s Craft }],end\_date: [{emp\_id: total\_work\_hours\_per\_day, “Craft” : Employee’s Craft } ] }

* 1. Calendar data structure runs from start\_date all the way till end\_date
  2. Returns Calendar data structure

1. scheduleJobCheck (job\_name, start\_date, due\_date, resources, total\_cost, craft, calendar\_resource\_dict)
   1. Takes in 7 parameters which describes the job requirements (start and end date and the craft and resources required)
   2. These requirements are checked against calendar\_resource\_dict to see if there are sufficient available resources to complete the job before or on the end date
   3. If sufficient available resource, prompt user to confirm schedule, returns True and the job dates if user confirms, else return False
   4. Else, prompt user if user would want a best available schedule recommended by the tool
   5. If user wants a best available schedule, call recommendSchedule function and prompts user if user wants to accept the new dates returned from recommendSchedule. Returns True and new dates if user agrees, else return False.
   6. Else, return False
2. recommendSchedule(resources, start\_date, due\_date, craft, calendar\_resource\_dict)
   1. Takes in 5 parameters containing job requirements of start\_date, due\_date and resources and craft required
   2. Compares these requirements with calendar\_resource\_dict and saves the start and end date where this job can be fulfilled completely
   3. Returns start date and new end date if the variables are different, else return new end date and None
3. scheduleJob(job\_name, start\_date, due\_date, resources, total\_cost, craft ,calendar\_resource\_dict, current\_job\_id, list\_of\_jobs)
   1. Takes in 9 parameters
   2. Generates a Job ID for current job being scheduled
   3. Creates new job instance and saves Job instance in list\_of\_jobs
   4. Allocate Employees in calendar\_resource\_dict to the job
   5. Creates an association between new job instance with the employees allocated to it
   6. Sets Job intance scheduled\_end\_date attribute to actual end\_date in the system (might be different from the due\_date originally given by user as some jobs can be completed earlier)
   7. Output Job details scheduled in the system for User, details shared are the start date and Job ID, as well as employees tagged to the job on the various dates

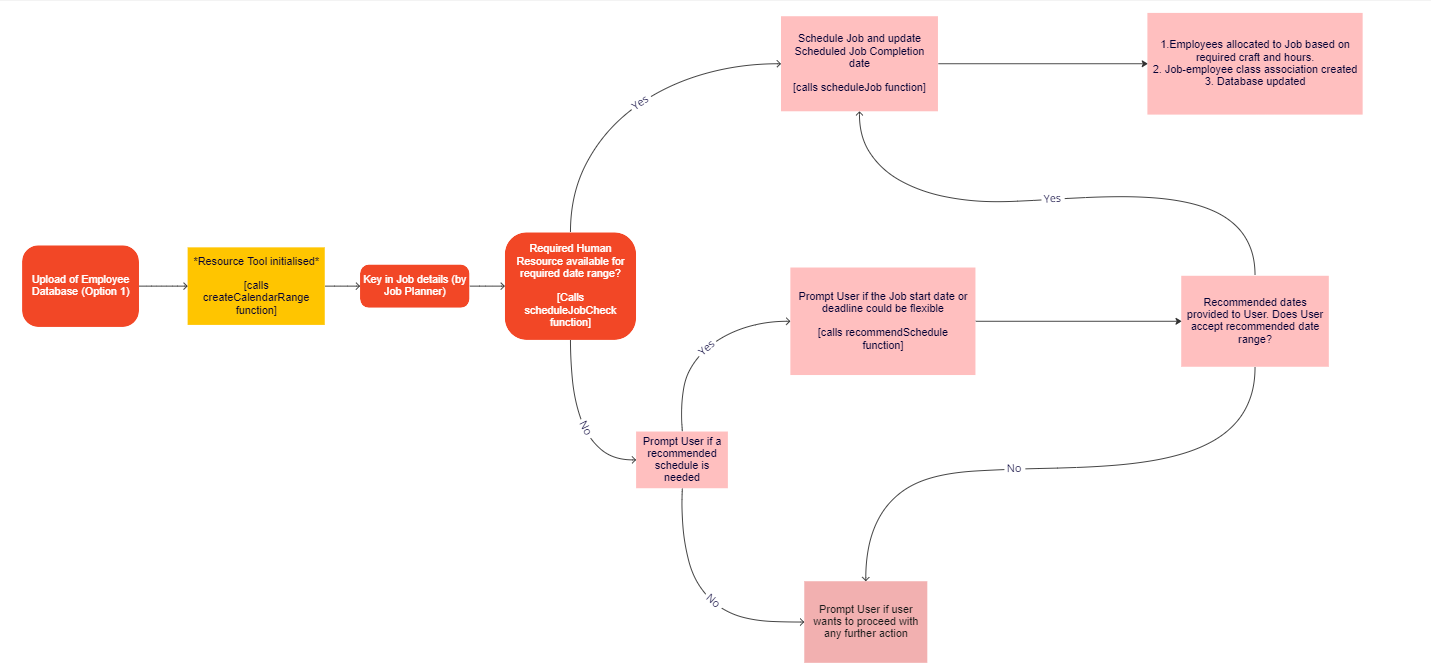


Figure 1 Flowchart on how a job can be scheduled in the Tool

# Use Cases

## Mass Upload of Employee and Job details

* To represent the case where a manufacturing company starts using the Tool
* Employee details and job details can be prepared in two separate .csv files to be uploaded in the Tool
* Numeric Option 1 🡪 1 leads to uploading of employee details
* Numeric Option 1 🡪 2 leads to uploading of job details (i.e Jobs already planned before using the Tool)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 2 Output when resource tool is successfully initialised

## Add/Remove Employees or Delete Scheduled Job

### Add an employee to database

User needs to input the following details, separated by commas: Employee ID, First Name, Last Name, Hourly Rate, Total Hours Per Day, Competency, Craft, Employee Start Date in yyyy-mm-dd.

Once the inputs are validated by the Tool, new employee will be added to the Tool and Employee Resource added to Calendar Data structure

|  |  |
| --- | --- |
| Sample input | Sample output\* |
| 25027188, shaun, lim, 20, 8, 5, metals, 2023-02-02 | “Employee 25027188 successfully added to database” |

\*Note: Although not shown in output, employee resource has been added to calendar data structure from Start-date onwards

Text

Description automatically generated

### Remove an employee from database

User needs to input Employee ID and Last Day of Work in yyyy-mm-dd with commas separating each input.

Once the inputs are validated, the employee will be removed from the Calendar data structure after his/her last day of work (i.e +1 day after Last Day of Work).

Any pending jobs attached to the employee past the Last Day of Work will be automatically rescheduled, if possible, else, an alert to the User will be provided.

Sample Input and Outputs given in Test cases section as it would be better explained with a full context with an Initial employee, Scheduled Job and followed by removal of employee.

### Delete a scheduled job

User needs to input the Job ID number. Tool will remove scheduled Job if it exists in the database. Resource allocated to the Job will be freed up for future use in calendar data structure.

|  |  |
| --- | --- |
| Sample input | Sample output |
| #1000 (Case when Job has not been added to database) | “Job does not exist in database” |
| 1)#1000 (Case when Job has been scheduled in database)  2)“Y” | “Job identified in database, Do you confirm Job deletion? Y/N”  “Job has been successfully deleted” |

## Schedule a job

User needs to input the following details, separated by commas: Job Name, Start Date in yyyy-mm-dd, Due Date in yyyy-mm-dd, Resources Required man-hours, Total cost (dollars), Craft Required

Refer to Figure 1. above for Flow Chart on Scheduling a Job.

|  |  |
| --- | --- |
| Sample input | Sample output |
| 1)Test1, 2023-02-01, 2023-02-05, 32, 800, instrument/electrical  2) “Y” | “Job can be scheduled, do you want to proceed? Y/N”  SUCCESS! Job 'Test1' has been scheduled with ID #1000 and Start date: 2023-02-01  Here are the allocated employee IDs and their allocated work hours for this job:  (i.e. Date: yyyy-mm-dd --> [{Employee ID: Work Hour(s)}])  ------------------------------------------------------------------------------  Date: 2023-02-01 --> [{25027504: 8.0}, {25027508: 8.0}]  ------------------------------------------------------------------------------  Date: 2023-02-02 --> [{25027504: 8.0}, {25027508: 8.0}]  ------------------------------------------------------------------------------ |
| 1. Test2, 2023-02-01, 2023-02-05, 32, 800, instrument/electrical (case when Test 1 is already scheduled) 2. “Y” | “Job can be scheduled, do you want to proceed? Y/N”  SUCCESS! Job 'Test2' has been scheduled with ID #1001 and Start date: 2023-02-01  Here are the allocated employee IDs and their allocated work hours for this job:  (i.e. Date: yyyy-mm-dd --> [{Employee ID: Work Hour(s)}])  --------------------------------------------------------------------------------  Date: 2023-02-03 --> [{25027504: 8.0}, {25027508: 8.0}]  --------------------------------------------------------------------------------  Date: 2023-02-04 --> [{25027504: 8.0}, {25027508: 8.0}]  -------------------------------------------------------------------------------- |
| 1. Test3, 2023-02-01, 2023-02-05, 32, 800, instrument/electrical (case when Test 1 & Test2 is already scheduled) 2. “Y” 3. “2” 4. “Y” | “Job cannot be scheduled due to unavailable resources, do you want to check for earliest available slot? Y/N”  Please input which of the job dates is flexible -  1 : Start Date, or  2 : Due Date  Job can be scheduled from Start date: 2023-02-05 --> End date: 2023-02-06, do you want to schedule it? Y/N  SUCCESS! Job 'Test3' has been scheduled with ID #1002 and Start date: 2023-02-05  Here are the allocated employee IDs and their allocated work hours for this job:  (i.e. Date: yyyy-mm-dd --> [{Employee ID: Work Hour(s)}])  --------------------------------------------------------------------------------  Date: 2023-02-05 --> [{25027504: 8.0}, {25027508: 8.0}]  --------------------------------------------------------------------------------  Date: 2023-02-06 --> [{25027504: 8.0}, {25027508: 8.0}]  -------------------------------------------------------------------------------- |

## Data persistence / Cost Summary Report

### Find all job details based on a specific date or by date range

User can search for jobs scheduled on a particular date or for a date range by keying in two same dates separated by a comma (for specific date) or two different dates separated by a comma (for date range).

|  |  |
| --- | --- |
| Sample input\* | Sample output |
| 1. 2023-02-02, 2023-02-02 2. “N” | These are the job(s) scheduled to happen on 2023-02-02  #1000: Test1 is scheduled with 2 employee(s), {Employee ID: Work Hours} --> [{25027504: 8.0}, {25027508: 8.0}]  Do you want to save Job details to .csv file? Y/N |
| 1. 2023-02-02, 2023-02-05 2. “N” | #1000: Test1 is scheduled with 2 employee(s) on 2023-02-02, {Employee ID: Work Hours} --> [{25027504: 8.0}, {25027508: 8.0}]  #1001: Test2 is scheduled with 2 employee(s) on 2023-02-03, {Employee ID: Work Hours} --> [{25027504: 8.0}, {25027508: 8.0}]  #1001: Test2 is scheduled with 2 employee(s) on 2023-02-04, {Employee ID: Work Hours} --> [{25027504: 8.0}, {25027508: 8.0}]  #1002: Test3 is scheduled with 2 employee(s) on 2023-02-05, {Employee ID: Work Hours} --> [{25027504: 8.0}, {25027508: 8.0}]  Do you want to save Job details to .csv file? Y/N |

\*Case when Test1, Test2 and Test3 Jobs have all been scheduled

### Find job details based on Job ID

User needs to input the Job ID number.

|  |  |
| --- | --- |
| Sample input\* | Sample output |
| #1000 | Job details are as follows: Job ID: #1000 , Job Name: Test1, Resources Required: 32.0, Total Cost SGD$: 800.0, Craft: instrument/electrical  These are the dates and employees and their hours working on the job:  2023-02-01 --> [{25027504: 8.0}, {25027508: 8.0}]  2023-02-02 --> [{25027504: 8.0}, {25027508: 8.0}] |
| #1002 | Please input Job ID:#1002  Job details are as follows: Job ID: #1002 , Job Name: Test3, Resources Required: 32.0, Total Cost SGD$: 800.0, Craft: instrument/electrical  These are the dates and employees and their hours working on the job:  2023-02-05 --> [{25027504: 8.0}, {25027508: 8.0}]  2023-02-06 --> [{25027504: 8.0}, {25027508: 8.0}] |

\*Case when Test1, Test2 and Test3 Jobs have all been scheduled

### Total Cost spent on Jobs for a specified Date Range

User can get a cost summary report on a particular date or for a date range by keying in two same dates separated by a comma (for specific date) or two different dates separated by a comma (for date range).

Cost is considered captured only on the last day of the job (Scheduled End Date). Deadline of the job is not necessarily equivalent to the Job scheduled end date as a job can finish earlier than the deadline.

|  |  |
| --- | --- |
| Sample input\* | Sample output |
| 1. 2023-02-02, 2023-02-02 2. “N” | These are the job(s) completed on this date: 2023-02-02, with their respective Total Cost(SGD)  {'#1000': 800.0}  Total Cost = SGD$800.0  Do you want to save these cost details to .csv file? Y/N |
| 1. 2023-02-01, 2023-02-01 | No cost incurred on this particular date: 2023-02-01 |

\*Case when Test1, Test2 and Test3 Jobs have all been scheduled

### Total Employee list for a specified Date (accounts for new hires and attrition)

User can get a list of employee(s) on a particular date by keying in specific date.

|  |  |
| --- | --- |
| Sample input\* | Sample output |
| 2023-01-01 | This is the list of employees on this date: 2023-01-01  {25027503: 'Shaun Lim', 'Craft': 'Metals'}  {25027504: 'Vishnu Kanth', 'Craft': 'Instrument/Electrical'}  {25027505: 'Zheng Yang', 'Craft': 'Machinery'}  {25027506: 'Albertus Alvin', 'Craft': 'Metals'}  {25027507: 'Ryan Phua', 'Craft': 'Metals'}  {25027508: 'Joel Lim', 'Craft': 'Instrument/Electrical'} |
| 1. Add Employee: 25027188, shaun, lim, 20, 8, 5, metals, 2023-02-02 2. 2023-02-02 | This is the list of employees on this date: 2023-02-02  {25027503: 'Shaun Lim', 'Craft': 'Metals'}  {25027504: 'Vishnu Kanth', 'Craft': 'Instrument/Electrical'}  {25027505: 'Zheng Yang', 'Craft': 'Machinery'}  {25027506: 'Albertus Alvin', 'Craft': 'Metals'}  {25027507: 'Ryan Phua', 'Craft': 'Metals'}  {25027508: 'Joel Lim', 'Craft': 'Instrument/Electrical'}  {25027188: 'shaun lim', 'Craft': 'metals'} |

\*Initialised Resource Tool with employee.csv (Option 1 – 1)

# Test Cases

Case 1: When no remaining employee fits the Job craft requirement

Inputs:

1. Mass upload Employee details through employee.csv (Option 1-1)
2. Mass Upload Pre-scheduled Job details through job.csv (Option 1-2)
3. Remove Sole Machinery Employee (ID 25027505) on 2023-01-31, which causes all Machinery Job in the schedule to be unable to complete (#1001, #1002)

Output:

A screenshot of a computer

Description automatically generated with medium confidence

Case 2: Job hours automatically rescheduled to another employee

Inputs:

1. Mass upload Employee details through employee.csv (Option 1-1)
2. Mass Upload Pre-scheduled Job details through job.csv (Option 1-2)
3. Remove one Metals Employee (ID 25027503) on 2023-01-31, which causes all metals Job to be rescheduled

Output:

Text

Description automatically generated

\*Note: The New scheduled end date is still before the deadline which helps to provide a check for the User that the new date is still within User’s previously stated deadline when scheduling the job.

An example of deadline vs scheduled end date can be seen in the Job “Test1” scheduled in Use Case 5.3:

|  |  |
| --- | --- |
| Sample input | Sample output |
| 1)Test1, 2023-02-01, 2023-02-05, 32, 800, instrument/electrical  2) “Y” | “Job can be scheduled, do you want to proceed? Y/N”  SUCCESS! Job 'Test1' has been scheduled with ID #1000 and Start date: 2023-02-01  Here are the allocated employee IDs and their allocated work hours for this job:  (i.e. Date: yyyy-mm-dd --> [{Employee ID: Work Hour(s)}])  ------------------------------------------------------------------------------  Date: 2023-02-01 --> [{25027504: 8.0}, {25027508: 8.0}]  ------------------------------------------------------------------------------  Date: 2023-02-02 --> [{25027504: 8.0}, {25027508: 8.0}]  ------------------------------------------------------------------------------ |

In the above example, deadline = 2023-02-05, however, scheduled end date is 2023-02-02, since the work can be completed earlier.