40.014 Engineering Systems Architecture
&
40.012 Manufacturing and Service Operations

2D Project Report 2
Group 7

Names	Student IDs
Toh Kin Yong	1006875
Khoo Teng Jin	1007104
Ernest Ong	1006346
Luvena Liethanti	1007105
Julianna Sherine	1007035
Phoebe Kuek	1007065
Asher Yeo	1006950

Table of Contents

1. Summary of Report 1

The project aims to design a Material Requirements Planning (MRP) system using a top-down approach, focusing on optimising processes for efficient and effective scheduling. However, given the project's constraints, we decided to skip the iteration process from our project to exclude unnecessary iterations if the system functions well.

Stakeholders of our project include the group members, professors guiding the project, and manufacturing companies whose operational needs shape the programme's requirements. The primary users are the project management team and project managers of manufacturing companies, who rely on the programme to enhance productivity and efficiency of their companies.

The project emphasises a user-friendly interface, efficient and reliable scheduling, high performance, robust notification and stability features, adaptability, and scalability. The context involved project managers seeking optimal manufacturing schedules through an interface that inputs data into an optimisation program, which processes the data to provide feasible schedules.

Customer feedback, collected online from manufacturing companies who use MRP systems, highlighted the need for a visually appealing interface, flexibility, minimal essential features, and error pinpointing capabilities.

Primary use cases of high priorities include users using, interacting with the interface, and inputting and retrieving data.

For functional requirements, the system shall accept a CSV file as the input, provide a feasible solution within 5% of the optimal solution, display results as a Gantt chart, and ensure a user-friendly navigation.

2. Optimising our Design Choices

In Report 1, we explored the alternative choices on how we were going to build the system. In this section, we will select the most optimal choice for each area of focus.

<u>Mixed-Integer Linear Programme (MILP) Solvers</u>

Attribute Name	Weights	Mosek Tools Optimiser		GLPK Optimiser		Gurobi Optimiser		(IBM) CPLEX Organiser	
		Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Speed	30%	3	0.90	2	0.60	5	1.50	4	1.20
Ease of Integration with R	30%	3	0.90	4	1.20	4	1.20	4	1.20
User-friendliness	25%	4	1.00	3	0.75	4	1.00	4	1.00
Support and Documentation	15%	3	0.45	4	0.60	4	0.60	4	0.60

Total Score			3.25		3.15		4.30		4.00
Rank			3		4		1		2
Continue?			No		No		Develop		No

Gurobi and CPLEX are widely recognized as the fastest solvers available, making them ideal choices for our scheduling optimiser program. Extensive research often compares these two optimisers to determine which performs better for various problem types. Given that our project requires mixed-integer linear programming, we chose Gurobi as it consistently delivers faster and more optimal results than CPLEX. Additionally, Gurobi is compatible with R, the language we are using for our application, thus allowing for seamless integration between the front-end and back-end code.

Additionally, the Gurobi optimiser was recommended by one of our stakeholders, Prof. Rakesh, to use for solving the mixed-integer linear program.

Area of Focus	**Possible Choices**
Mixed Integer Linear Programme Formulations	**Formulation 1 (Agrawal, A., Harhalakis, G., Minis, I., & Nagi, R. (1996)) (selected choice):**

$$\text{Maximize}: \text{cost}(\mathcal{S}) = \min\{S_i:\ i = 1, 2, \ldots, n\} \quad (2.1)$$

subject to:

$$S_{d(i)} \geq C_i \qquad i = 1, 2, \ldots, n; \quad (2.2)$$

$$C_i = S_i + t_i \qquad i = 1, 2, \ldots, n; \quad (2.3)$$

$$C_e \leq D_e \qquad e = 1, 2, \ldots, n_f,$$

D_e = due date of eth final assembly;

C_e = completion time of the last of eth final assembly; (2.4)

$$\delta_{ij} + \delta_{ji} = 1 \qquad i, j \in I_{\mathcal{K}},\ i \neq j;\ \mathcal{K} = 1, 2, \ldots, m; \quad (2.5)$$

$$S_i - C_j \geq M\left(\delta_{ij} + \Delta_{ik} + \Delta_{jk} - 3\right)$$
$$i, j \in I_{\mathcal{K}},\ i \neq j;\ \mathcal{K} = 1, 2, \ldots, m;\ k = 1, \ldots, f_{\mathcal{K}}; \quad (2.6)$$

$$\sum_{s=1}^{f_k} \Delta_{is} = 1 \qquad i \in I_{\mathcal{K}};\ \mathcal{K} = 1, 2, \ldots, m; \quad (2.7)$$

$$S_i \geq 0 \qquad i = 1, 2, \ldots, n; \quad (2.8)$$

$$\delta_{ij} \in \{0, 1\} \qquad i, j \in I_{\mathcal{K}};\ i \neq j;\ \mathcal{K} = 1, 2, \ldots, m; \quad (2.9)$$

$$\Delta_{ik} \in \{0, 1\} \qquad i \in I_k; \mathcal{K} = 1, 2, \ldots, m; k = 1, 2, \ldots, f_{\mathcal{K}}. \quad (2.10)$$

Formulation 2 (Xu, J., & Nagi, R. (2013)):

$$\text{Minimize}: \ Z$$

$$\text{Subject to}:$$

$$S_j \geq F_i, \quad \forall i,j \in N, \chi_{ij} = 1 \tag{1}$$

$$S_j \geq ES_j, \quad \forall j \in N \tag{2}$$

$$F_j \leq Z, \quad \forall j \in N \tag{3}$$

$$F_j = S_j + t_j, \quad \forall j \in N \tag{4}$$

$$\psi_{ij} + \psi_{ji} = 1, \quad \forall i,j \in I_Y, i \neq j, Y \in \{1,2,\ldots,w\} \tag{5}$$

$$S_i - F_j \geq (\psi_{ij} + \phi_{iy} + \phi_{jy} - 3)M, \quad \forall i,j \in I_Y, i \neq j, \\ y \in \{1,2,\ldots,f_Y\}, \quad Y \in \{1,2,\ldots,w\} \tag{6}$$

$$\sum_{y=1}^{f_Y} (\phi_{jy}) = 1, \quad \forall j \in I_Y, Y \in \{1,2,\ldots,w\} \tag{7}$$

$$\psi_{ij} \in \{0,1\}, \quad \forall i,j \in I_Y, i \neq j, \\ Y \in \{1,2,\ldots,w\} \tag{8}$$

$$\phi_{jy} \in \{0,1\}, \quad \forall j \in I_Y, \\ y \in \{1,\ldots,f_Y\}, Y \in \{1,2,\ldots,w\}. \tag{9}$$

We have selected Formulation 1 for our mixed-integer linear program as we found this formulation to be easier to understand than Formulation 2. Formulation 2 employs a Lagrangian relaxation approach for solving scheduling problems, which is an approach that we are unfamiliar with. Whereas for formulation 1, we are also able to obtain the desired output that we require and can better program in a way such that it can be implemented into our application.

Programming Languages for Interface

Attribute Name	Weights	HTML, CSS, JavaScript		Python, PyQt, PyTkinter		Swift, SwiftUI		Kotlin, Jetpack Compose		R	
		Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Ease of Learning	30%	4	1.20	4	1.20	4	1.20	3	0.90	4	1.20
Performance	35%	3	1.05	3	1.05	4	1.40	3	1.05	4	1.05
Integration with Existing Systems	20%	4	0.80	4	0.80	3	0.60	2	0.40	5	1.00
Flexibility and Features	15%	4	0.60	5	0.75	4	0.75	4	0.75	4	0.75
Total Score			3.65		3.80		3.95		3.10		4.00

Rank			4		3		2		5		1
Continue?			No		No		No		No		Develop

We have selected R as our programming language as it supports both our interface design and our selected solver. This reduces the number of processes and lines of code needed, thus ensuring that our application runs more smoothly.

Programs for Interface Design

Attribute Name	Weights	Figma		React		R Shiny		Invision		Adobe XD	
		Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Ease of Learning	30%	4	1.20	3	0.90	4	1.20	4	1.20	4	1.20
Design Capabilities and Flexibility	30%	4	1.20	4	1.20	4	1.20	3	0.90	4	1.20
Integration with Development Workflow	25%	4	1.00	3	0.75	5	1.25	3	0.75	3	0.75
Collaboration features	15%	4	0.60	3	0.45	3	0.45	4	0.60	4	0.60
Total Score			4.00		3.30		4.10		3.45		3.75
Rank			2		5		1		4		3
Continue?			No		No		Develop		No		No

We have chosen R Shiny for developing our user interface. Since we are more familiar with R, using it for both the front-end and back-end will make the code easier to understand and implement. Additionally, R Shiny allows us to integrate R packages and functions seamlessly, while also supporting other script files such as CSS, JavaScript, and HTML. This flexibility enhances our ability to design a creative and functional interface.

Programming Languages for Running the MILP

Attribute Name	Weights	Julia/JuMP		Matlab		R		Octave		Python	
		Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Ease of Use	30%	3	0.90	3	0.90	4	1.20	2	0.60	4	1.20
Performance	40%	3	1.20	3	1.20	3	1.20	3	1.20	4	1.60
Integration with Existing Systems	30%	3	0.90	2	0.60	5	1.50	2	0.60	3	0.90
Total Score			3.00		2.70		3.90		2.40		3.70
Rank			3		4		1		5		2
Continue?			No		No		Develop		No		No

We have chosen R as our backend programming language as it is the same programming language used for our interface design. This allows for more seamless integration between the two without the need for an external programming translator to convert the codes to be readable for each other.

3. Develop the Architecture

Functional Requirements	**Function Name**
The system shall take in a CSV file (that contains the Bill-of-Materials, Operation Network, and Routings) as the input.	CSV input
The system shall provide a feasible solution that is within 5% of the optimal solution.	Feasible solution
The system shall provide a schedule of operations as the output.	Output operations schedule
The system shall provide a schedule that is formatted via a Gantt chart for readability.	Schedule as Gantt chart
The system shall allow users to learn how to navigate.	Learn to navigate
The system shall deliver the output.	Deliver Output

Functional Interrelationship Matrix:

Function Name	CSV input	Feasible solution	Output operations schedule	Schedule as Gantt chart	Learn to navigate	Deliver Output
CSV input		triggers				precedes
Feasible solution			triggers			
Output operations schedule				triggers		precedes
Schedule as Gantt chart						triggers
Learn to navigate	precedes					
Deliver Output						

State Change Matrix:

State	System off	System on and idle	System on and solving	System off
System off		“Turn on system”		
System on and idle			“Receive input”	“Turn off system”
System on and solving		“Output derived”		“System crash”
System off				

Interface Matrix:

(Events) Relates subsystem to	Operator (Scheduler)	User Interface System	Optimisation Program System	Data
Operator (Scheduler)		“Interacts”		“Retrieves”
User Interface System			“Delivers input”	
Optimisation Program System		“Delivers output”		“Solves”
Data				

Design Structure Matrix:

(Events) Relates subsystem to	User Interface System	Optimisation Program System
User Interface System		“Delivers input”
Optimisation Program System	“Delivers output”	

4. Validate the Design

To ensure the robustness and reliability of the MRP programme, we need to identify the potential risks and develop corresponding actions to reduce their likelihood and severity.

- Data Inaccuracy & User Error
    - Risk: Users might misinterpret the scheduling outputs or input incorrect data which can lead to flawed scheduling, resulting in delays, resource shortages, or overproduction.
    - Actions: Develop an intuitive, user-friendly interface to decrease chances of data entry mistakes and misinterpretation of outputs, implement robust data validation checks, use automated data collection systems to minimise manual entry errors, provide comprehensive user training.
- System Downtime
    - Risk: System failures or downtime can disrupt scheduling operations which may cause delays and inefficiencies.
    - Actions: Implement failover systems (e.g., button that terminates the system once an error is detected), schedule regular maintenance and updates.
- Security Breaches
    - Risk: Unauthorised access to the system can result in data theft or manipulation which can disrupt operations.
    - Actions: Implement strong authentication and authorization protocols.
- Unexpected Changes in Demand
    - Risk: Sudden changes in customer demand can render the schedule ineffective which may lead to overproduction or stockouts.
    - Actions: Integrate demand forecasting tools to predict changes.
- Multiple Interface Instances
    - Risk: Rapid or repeated clicks by the user can create multiple instances of the interface, potentially leading to overlapping processes and negatively impacting functionality.
    - Actions: Temporarily disable buttons after a click, create a queue system for sequential processing of click events, provide immediate visual feedback to the user to indicate that the click has been registered and operations are ongoing.

To ensure the reliability and efficiency of the scheduling system, we prioritised implementing a user-friendly interface and a failover system button. These critical features address immediate risks and enhance user experience, while other strategies are deferred for future consideration.

- User-Friendly Interface: Essential for reducing errors by providing clear instructions and real-time feedback. This feature is feasible and directly impacts data accuracy as well as scheduling reliability.

- Failover System Button: Crucial for minimising downtime by allowing for quick resolution of system errors. It is vital for maintaining operational efficiency and preventing extended disruptions.

5. Execute the Design

As of Week 11, our group has successfully completed and submitted the project proposal and Report 1. The upcoming submissions include the proposed interface for the scheduling optimiser program and Report 2. The interface development consists of both the front-end and back-end components. The building of the user interface (front-end) was completed in Week 10. Currently, we are simultaneously drafting the second report and coding the backend optimiser.

For the next steps, we aim to complete the programme interface by Week 11, including Report 2 and the Poster that would be showcased on Week 13. Following these submissions, we would be wrapping up the project by presenting our project process and proposal on Week 13.

To ensure timely completion, we divided our team into 3 sub-teams, each focusing on different overlapping tasks. These tasks include the building of the user interface, coding of the backend optimiser, and drafting of Report 2. Given the extensive time required for each of these tasks, splitting the work among different teams would allow us to manage the workload efficiently and meet our deadlines. This approach ensured that each team could focus on its specific task without experiencing excessive workload.

The Gantt chart below illustrates our project timeline:

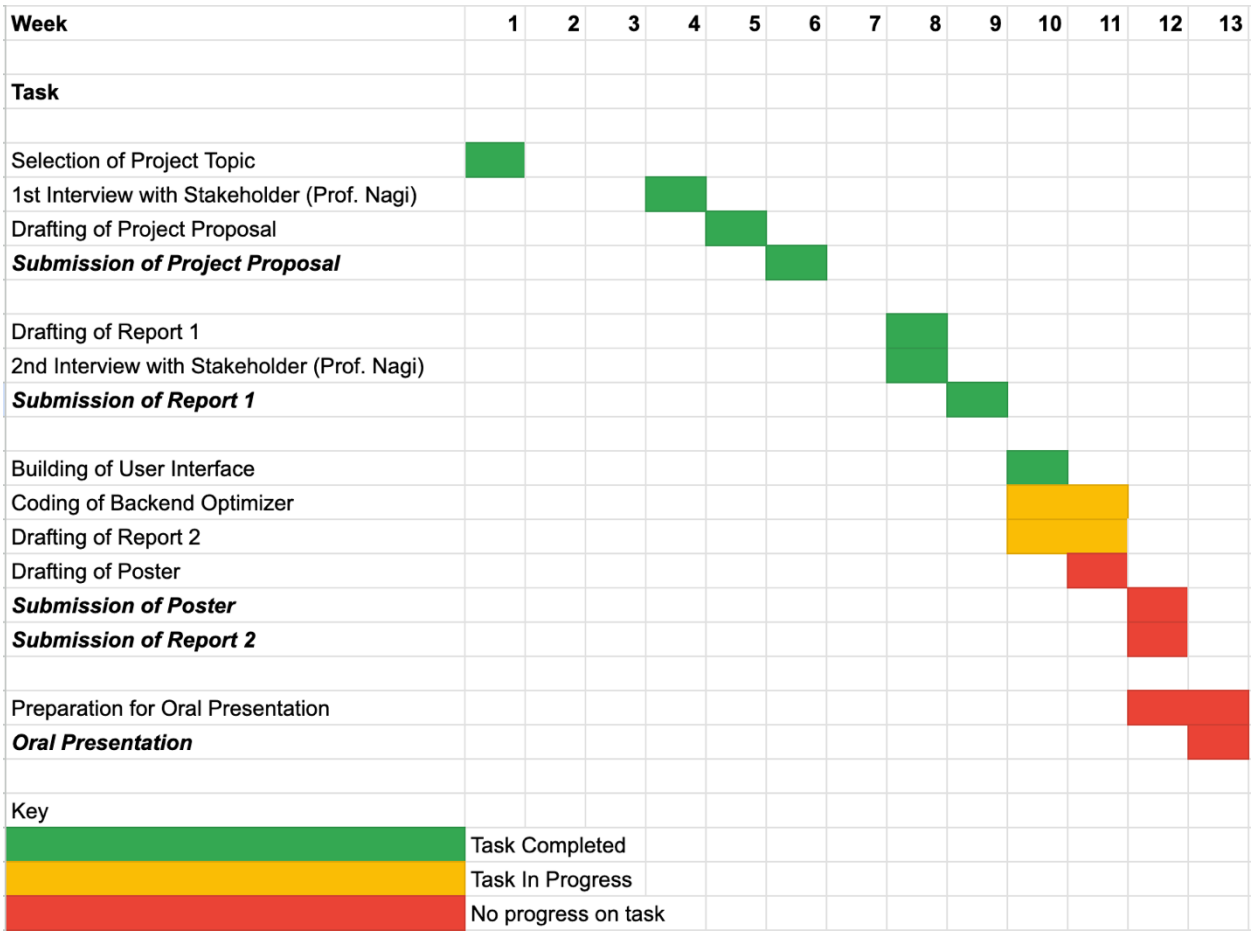

Figure 1. Project Process Gantt Chart

6. Understanding of MSO Materials

The required inputs for our optimization program are the Bill-of-Materials (BOM), the routings and processing times, as well as the operation network. An explanation for each type of input is given below.

1. Bill-of-Materials (BOM)

A BOM refers to the complete list of raw materials, parts, and tools needed to manufacture a particular product. BOMs also include the components and subcomponents that make up the product, as well as the quantities required for each (Mecalux, 2021).

There are two common types of BOMs, namely the single-level and multi-level BOMs. The type of BOM chosen depends mainly on the product's complexity and level-of-detail:

- The single-level BOM is usually used for less complex products. It lists the components, and the corresponding quantity required for each component to produce the final product. An example of a product which can be represented using a single-level BOM is a table, as it consists of four legs, a tabletop, nuts, and bolts (Mecalux, 2021).
- The multi-level BOM, on the other hand, is usually used for more complex products. Besides listing the quantity of materials needed to produce the final product, the multi-

level BOM also includes multiple levels, and showcases the relationships between the materials on different levels (Mecalux, 2021). An example of a multi-level BOM for a bicycle is shown below:

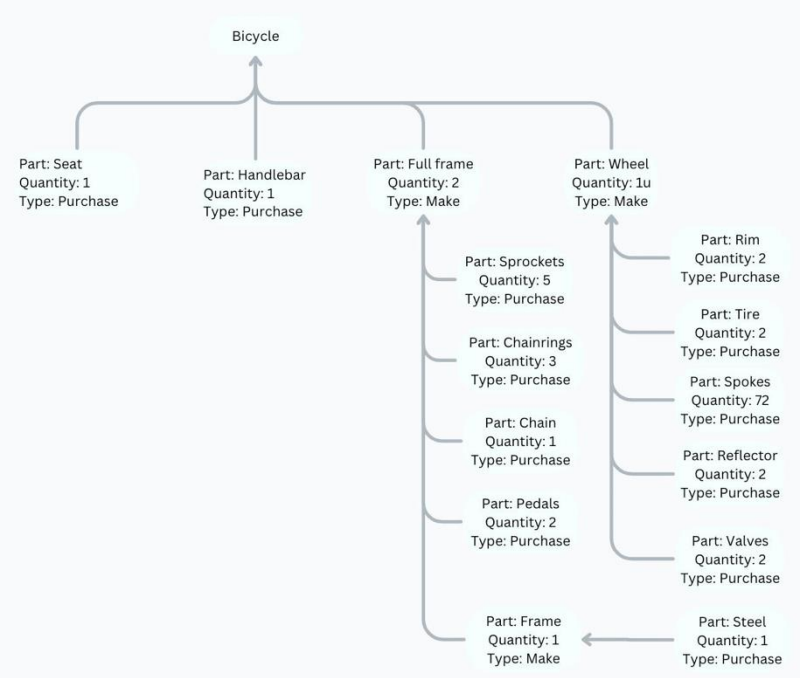

Figure 2. Multi-tiered Bicycle Manufacturing Bill-of-Material. (Mecalux, 2021)

In the figure above, we can see that the first level consists of four components and each of their required quantities, namely the seat, handlebar, full frame, and wheel. In turn, the full frame consists of multiple subcomponents such as the pedals and chains, while the wheel consists of multiple subcomponents such as the rim and the valves.

2. Routings and Processing Times

Routing refers to the detailed sequence of operations needed for each make/manufactured part. It can also contain other information such as the processing time required for each operation. The table below showcases the routing for a bicycle, where the routings and processing times are documented to understand the workflow and dependencies.

Routings of Make Parts

Part	Operation	Components Required	Processing Time (minutes)	Work Center
Wheel	Tire Assembly	Rim, Tire, Spokes, Reflector, Valves	8.47	1
Skeleton	Cut	Metal	0.27	1
	Conformed	Metal	2.48	1
	Surface Treatment	Metal	10	2
	Painting	Metal	7.25	2
	Baking/Drying	Metal	120.0	2
Frame	Full Frame and Pitchfork Assembly	Skeleton, Sprockets, Chainrings, Chain, Pedals	3.78	3
Bicycle	Assemble	Wheel, Seat, Handlebar, Frame	12.8	3

Figure 3. An example of a routing for a bicycle. (Botto-Tobar et al.,2021, pg.116)

By integrating the routing information into the MRP programme, we ensured that each task was scheduled appropriately, by considering the required processing times and the sequence of operations.

3. Operation Network

Operation networks describe the interconnected processes and sequences in manufacturing. In the context of the MRP programme, we used the manufacturing system of a bicycle as a case study. The operation network outlines how each part is related to its predecessors and successors. It provides a structured sequence in which processes should be executed to ensure an efficient production flow. By understanding the operation network, manufacturers can better coordinate the production stages, minimise delays, and ensure timely delivery of the final product.

In the bicycle manufacturing case study, we collected detailed information on the different components, categorised them as either manufactured in-house or purchased externally, and gathered data on the manufacturing times. The operation network below was mapped out to show the sequence and connections of each part, ensuring all processes are correctly sequenced:

Figure 4. Bicycle Manufacturing Operation Network

We then created the operation network for the MRP system ensuring that key aspects such as sequence of processes, interconnected components, and optimization goals are integrated properly.

- Sequence of Processes: Defines the order in which steps are performed
- Interconnected Tasks: Shows how tasks relate to each other, which is critical for system integration
- Optimization Goals: Aims to minimise development time and meet deadlines effectively

4. MILP Constraints

MILP is used to optimise operation scheduling in systems by solving for the best possible schedule that meets various constraints. These constraints ensure that the solution is feasible and efficient. In the MRP system, MILP helped to develop the optimal operation scheduling, ensuring efficient use of resources and meeting all necessary deadlines.

$$\text{Maximize} : \text{cost}(\mathcal{S}) = \min\{S_i:\ i = 1, 2, \ldots, n\} \quad (2.1)$$

subject to:

$$S_{d(i)} \geq C_i \qquad i = 1, 2, \ldots, n; \quad (2.2)$$

$$C_i = S_i + t_i \qquad i = 1, 2, \ldots, n; \quad (2.3)$$

$$C_e \leq D_e \qquad e = 1, 2, \ldots, n_f,$$

D_e = due date of eth final assembly;

C_e = completion time of the last of eth final assembly; (2.4)

$$\delta_{ij} + \delta_{ji} = 1 \qquad i, j \in I_{\mathcal{K}},\ i \neq j;\ \mathcal{K} = 1, 2, \ldots, m; \quad (2.5)$$

$$S_i - C_j \geq M\left(\delta_{ij} + \Delta_{ik} + \Delta_{jk} - 3\right)$$
$$i, j \in I_{\mathcal{K}},\ i \neq j;\ \mathcal{K} = 1, 2, \ldots, m;\ k = 1, \ldots, f_{\mathcal{K}}; \quad (2.6)$$

$$\sum_{s=1}^{f_k} \Delta_{is} = 1 \qquad i \in I_{\mathcal{K}};\ \mathcal{K} = 1, 2, \ldots, m; \quad (2.7)$$

$$S_i \geq 0 \qquad i = 1, 2, \ldots, n; \quad (2.8)$$

$$\delta_{ij} \in \{0, 1\} \qquad i, j \in I_{\mathcal{K}};\ i \neq j;\ \mathcal{K} = 1, 2, \ldots, m; \quad (2.9)$$

$$\Delta_{ik} \in \{0, 1\} \qquad i \in I_k; \mathcal{K} = 1, 2, \ldots, m; k = 1, 2, \ldots, f_{\mathcal{K}}. \quad (2.10)$$

- Capacity Constraints: Ensure that development does not exceed available resources
- Precedence Constraints: Maintain the correct order of tasks as specified in the operation network
- Time Constraints: Adhere to the development start times and deadlines to minimise delays
- Resource Constraints: Manage the allocation of resources to prevent shortages or excess

These constraints helped to shape the backend interface, ensuring that the overall programme runs smoothly and efficiently.

5. Link for Video Demonstration of Interface

Video link: https://youtu.be/KzIMtEYBvjs?si=QG3l2WqkikZ1PoQA

7. References

1. Agrawal, A., Harhalakis, G., Minis, I., & Nagi, R. (1996). 'Just-In-Time' Production of Large Assemblies. IIE Transactions. 10.1080/15458830.1996.11770710
2. Xu, J., & Nagi, R. (2013). Solving assembly scheduling problems with tree-structure precedence constraints: A Lagrangian relaxation approach. IEEE Transactions on Automation Science and Engineering. 10.1109/TASE.2013.2259816
3. Mecalux. (2021, March 19). Bill of materials (BOM): your top supply chain ally. https://www.mecalux.com/blog/bill-of-materials-bom
4. Botto-Tobar, M., Montes León, S., Camacho, O., Torres-Carrión, P., & Zambrano Vizuete, M. (2021). Applied Technologies: Second international conference. Springer.
5. Guasque, A., Balbastre, P., Evaluation and Comparison of Integer Programming Solvers for Hard Real-Time Scheduling, IEICE Transactions on Information and Systems, 2022, Volume E105.D, Issue 10, Pages 1726-1733, Released on J-STAGE October 01, 2022, Online ISSN 1745-1361, Print ISSN 0916-8532, https://doi.org/10.1587/transinf.2022EDP7073