Sri Lanka Institute of Information Technology

B.Sc. Honours Degree in Information Technology Field of specialization: Information Systems Engineering

Automated Man Power Allocation By Performance Analysis and Project Categorization For Construction Projects

Project Management Tool Document

Group ID: 24-25J-018

Student ID	Name
IT21270956	Munagama M.K.H
IT21276750	Isuranga K.M.S
IT21087660	Perera K.M
IT21069840	Devashika R.P.P.A

Table of Contents

List of Tables

List of Figures

1. Executive Summary

In construction industry, there two types of manpower used as, employees and labor. Currently in the construction industry, this manpower is allocated to the construction projects.

1.1 Objectives

Our objective is to develop a web application,

1. To categorize the projects according to the entered project details as “High”,” Low”, “Medium”.
2. To Generate a KPI value for the recruited employees to the company by analysing the performance and CV details such as skills and experience.
3. To allocate the employees by optimizing matching the KPI values with the project categorization results.
4. To predict the labor count required for the project based on the project details.

1.2 Scope

Our client to this project is MAGA Engineering Pvt Ltd. In MAGA there are different types of construction projects going on.

1. Buildings
2. Highways & Bridges
3. Water & Wastewater
4. Irrigation

In this project we are considering about the **building** construction projects only.

1.3 Tools and Technologies Used

- Frontend: React.js
- Backend: Node.js
- Database: MySQL
- ML Model: Trained with Python (Scikit-learn, TensorFlow)
- Version Control: GitHub
- Project Management: MS Planner (via MS Teams)
- API Testing: POSTMAN

2. Project Metrics and Tasks Summary

2.1 Gantt Chart

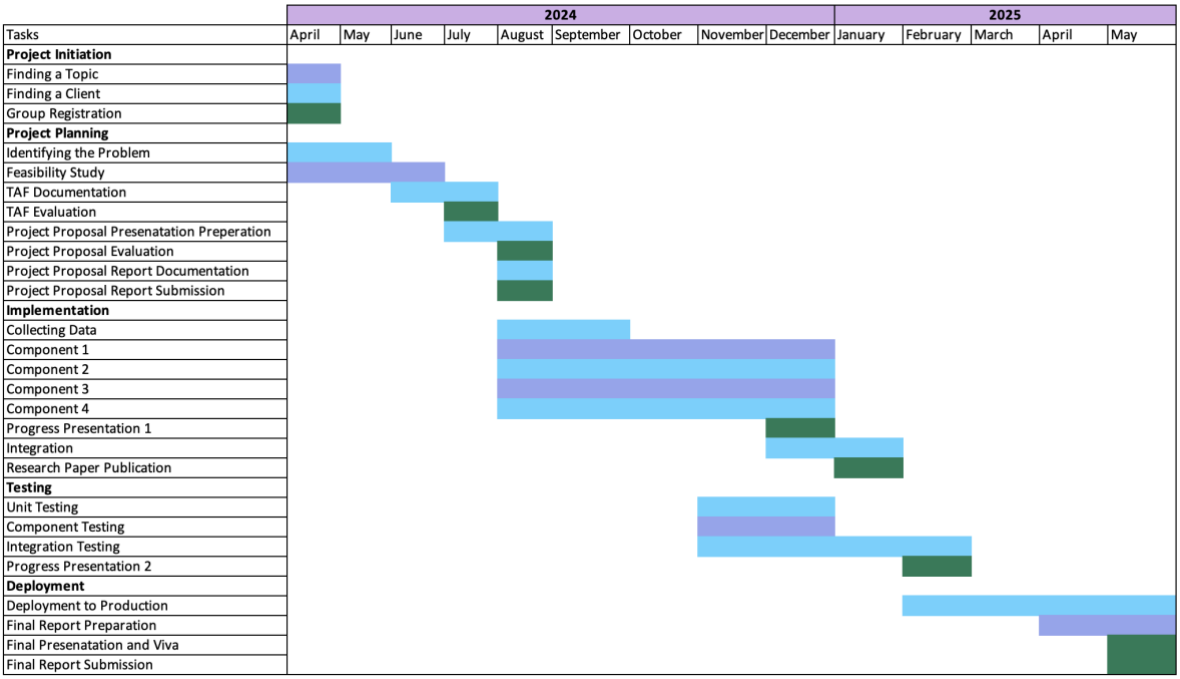

Figure 1: Gantt Chart

2.2 Use of Project Management Tool

We have used MS Planner via MS teams to manage our tasks throughout the project so far.

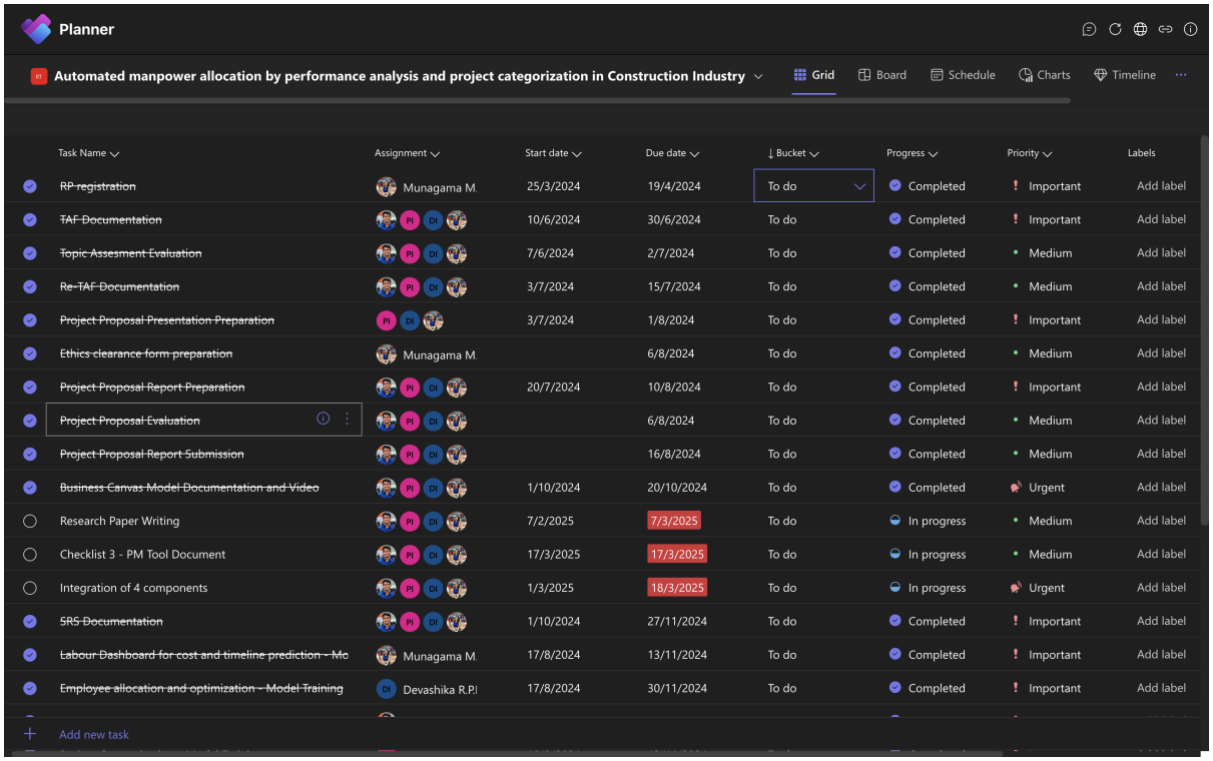

Figure 2 : MS Planner Proof 1

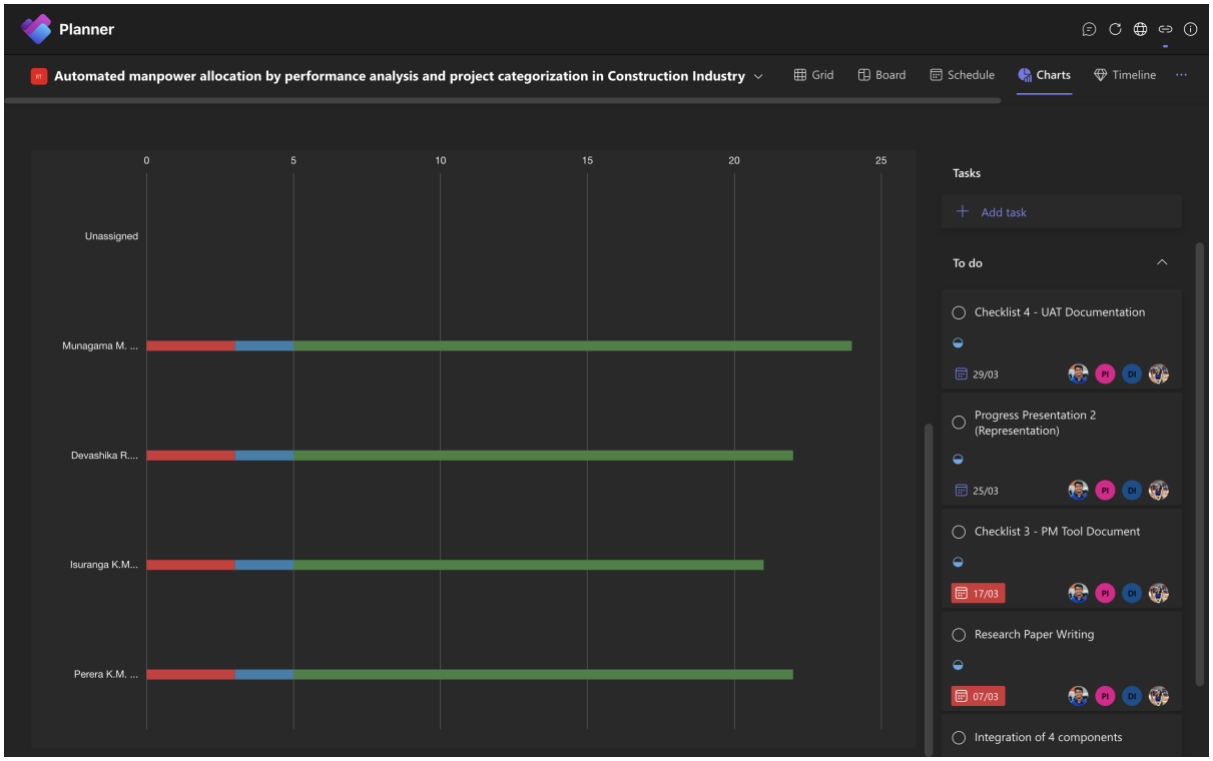

Figure 3: MS Planner Proof 2

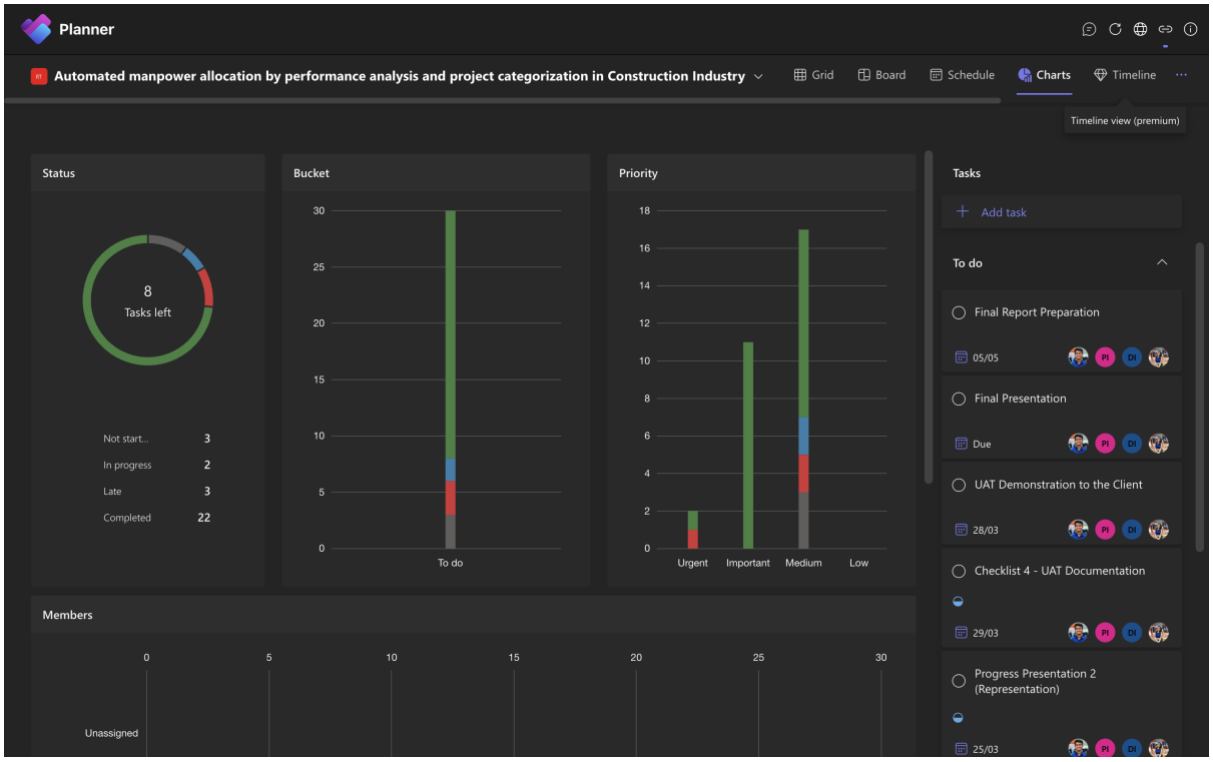

Figure 4: MS Planner Proof 3

3. Resources and Work Management

Table 1: Resource and work allocation

Name	Allocated Work
Munagama M.K.H - IT21270956	Labor, Cost and Timeline Prediction
Isuranga K.M.S - IT21276750	KPI generation by CV and performance analysis
Perera K.M - IT21087660	Project Categorization
Devashika R.P.P.A - IT21069840	Employee Allocation and Optimization

4. Lessons Learned

4.1 Successes

1. Strong Team Coordination

Using MS Teams and MS Planner really helped streamline our workflow. Everyone knew their roles, tasks were clearly tracked, and we had real-time visibility into project progress. That structure kept us on the same page throughout.

2. Meeting Deadlines

Thanks to early planning and clear task delegation, we managed to hit nearly all our milestones on time—some even ahead of schedule. This gave us breathing room during more complex stages of development.

3. Smooth Integration of Machine Learning Models

We successfully embedded both the KPI prediction and project categorization models into the system via REST APIs. Despite the technical depth, integration went smoothly without major deployment hiccups.

4. Client Collaboration

Keeping the lines of communication open with MAGA Engineering from the start paid off. Their feedback shaped key features and ensured the system aligned closely with the realities of construction project management.

4.2 Challenges

1. **Data Limitations for Model Training**
   Finding suitable real-world data—especially related to employee performance and CVs—was tough. Privacy concerns and the lack of digitized records slowed down our progress during model development.
2. **Managing Conflicting Commitments**
   Some teammates had to juggle academic responsibilities alongside project work. At times, that overlap impacted turnaround on shared tasks and collaborative sessions.
3. **Cross-Technology Integration Hurdles**
   Bringing together Python-based ML models with a Node.js backend wasn't straightforward. We hit several roadblocks that required extra research and plenty of trial and error to overcome compatibility issues.

4.3 Improvements

1. **Flag Risks Earlier**
   For future projects, we plan to introduce a formal risk management process early on. That way, we can identify potential blockers before they slow us down.

2. **Better Documentation**

   While our code was functional, we noticed gaps in inline comments and user-facing guides. Improving both would help with long-term maintainability and onboarding new contributors.

3. **Start Automated Testing Sooner**

   Manual testing served its purpose, but it lacked depth. Incorporating automated tools like Jest and Postman scripts earlier would've boosted test coverage and confidence in deployment.

4. **Structure Client Feedback Loops**

   Although we had good communication with the client, setting up more structured and earlier feedback cycles could've helped refine requirements and catch misalignments faster.