**Lab Assessment 1**

Software Process Model & Software Project Management document of the project

Mobile Fog–Based Intelligent Monitoring System.

Slot: L9 + L10

Course Code and Title - BCSE301P, Software Engineering Lab.

Team Members and their roles -

**Pushkal Gupta (23BCT0253)**–Team Coordinator/System Architect & Fog

**Adrivid Mishra (23BCT0264)**–Hardware & Embedded Systems.

**Vaibhav Jain (23BAI0033)**–AI & Data Intelligence.

**Shagnik Paul (23BCT0266)**–UI, Backend & Visualization.

1. **Project Description.**

**1.1 Background.**

Modern cyber‑physical and vehicular systems generate high‑frequency sensor data that is often sent directly to the cloud for analysis. This cloud‑centric approach introduces latency, bandwidth overhead, and loss of functionality during network outages. In safety‑critical environments (vehicles, industrial motors, smart infrastructure), even milliseconds of delay can cause irreversible damage.

**1.2 Problem Definition.**

Existing monitoring systems: - Depend heavily on cloud availability - React slowly to real‑time failures - Transmit large volumes of redundant raw data - Fail during poor or zero network connectivity. Hence, there is a need for a low‑latency, resilient, fog‑based monitoring architecture capable of

autonomous local decision‑making.

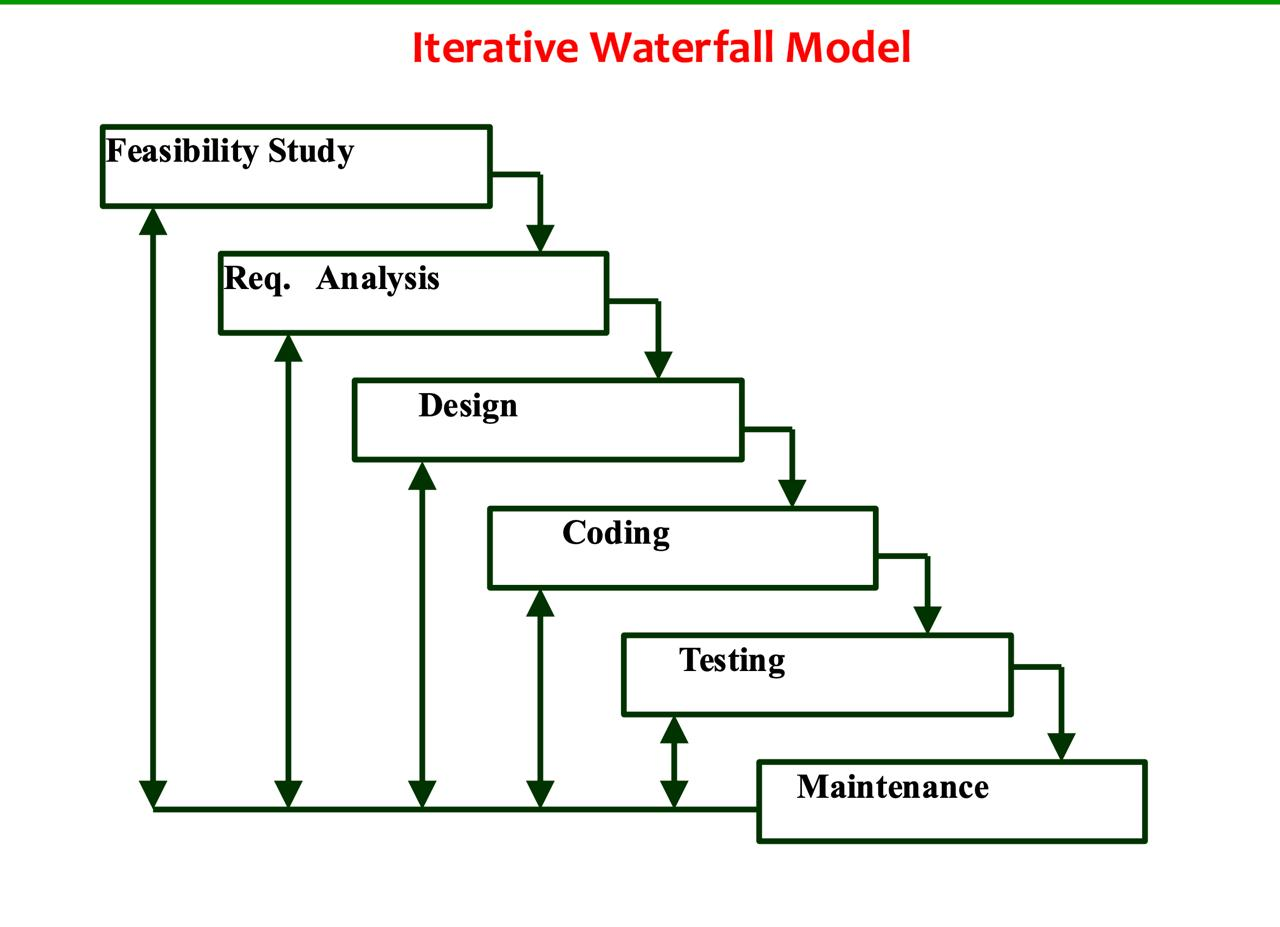
**1.3 Five major objectives of this project.**

1. **Non‑Intrusive Energy Monitoring** – Monitor subsystem health using electrical signatures instead of multiple physical sensors.
2. **Fog‑Level Autonomous Decision Making** – Enable real‑time safety actions at the smartphone fog node without cloud dependency.
3. **Edge Feature Extraction** – Perform R.M.S., F.F.T., and T.H.D. extraction locally to reduce data transmission.
4. **AI‑Based Predictive Health Assessment** – Predict component degradation using ML models.
5. **Real‑Time Visualization & Digital Twin** – Provide live system status to users via mobile and web dashboards.
6. **Process Model Selection.**

The **Iterative Waterfall** model has been chosen as the most appropriate model for our project. The following section gives a detailed explanation as to why it was selected.

**2.1 Justification.**

The **Iterative Waterfall Model** is selected for this project because the system **has a clearly defined scope**, **known hardware components**, and **well-understood functional requirements**, while still requiring **controlled feedback and refinement between development phases**.



This project is an **academic, system-level implementation** involving embedded hardware, fog computing, AI models, and user interfaces. Although the overall architecture is fixed at the beginning, defects and improvements are expected to be identified during later stages such as coding, testing, and integration. The Iterative Waterfall Model supports this by allowing **backtracking to earlier phases whenever errors are detected**, ensuring correctness before progressing further.

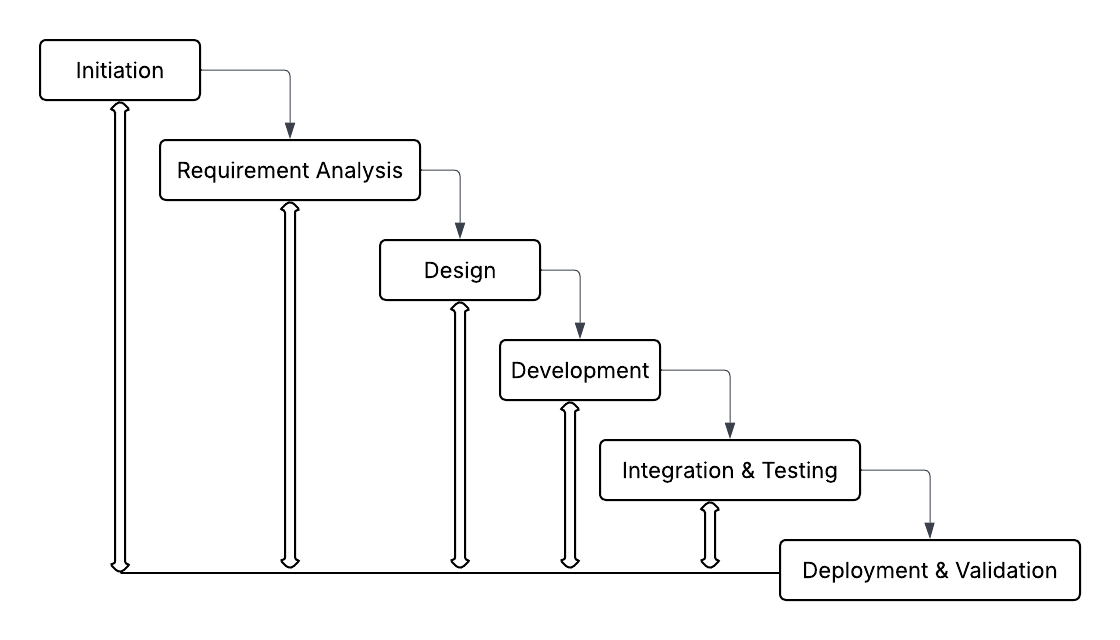
**2.2 Suitability to the Project.**

The model is suitable because: - Core requirements are identified early and documented clearly - Hardware and software interfaces are predefined - Phase-wise development is required for academic

evaluation - Reviews and corrections are expected at every stage

Since none of the listed limitations apply to this project—there is no requirement for incremental delivery, no phase overlap as defined in the Gantt chart, no direct customer involvement due to its patent-oriented nature, and no dependency on risk-driven iterations or code reuse—the constraints of other process models are not relevant. With a well-defined architecture and clearly separated phases, the **Iterative Waterfall model** aligns effectively with the project’s execution and final full-scale submission, making it the most appropriate development approach.

**2.3 Phase mapping to the Project.**

****

| **Iterative Waterfall Phase** | **Project Activities** |
| --- | --- |
| Feasibility Study | Initial project idea, constraints, and tool selection. |
| Requirement Analysis | Functional and non-functional requirements, SRS. |
| System Design | Architecture design, data flow, hardware and model selection. |
| Coding & Unit Testing | ESP32 firmware, fog logic, AI model development. |
| Integration & System Testing | Device–fog–cloud integration and validation. |
| Maintenance (Iteration) | Model tuning, threshold adjustment, logic refinement. |

1. **Tools and Technologies.**

### **Software**

* **ESP32 Firmware (C / C++)** – Handles real-time data acquisition from CT sensors, low-level signal processing, and wireless transmission to the fog layer.
* **Fog Layer (Python)** – Performs local feature extraction, safety logic execution, and autonomous decision-making at the fog node.
* **Messaging (MQTT – Eclipse Mosquitto)** – Enables lightweight, low-latency communication between edge devices, fog nodes, and backend services.
* **AI Framework (TensorFlow Lite)** – Runs optimized machine learning models for predictive health monitoring and fault detection on edge and fog devices.
* **Backend (REST APIs)** – Provides structured services for data ingestion, device management, and integration with visualization layers.
* **Database (MongoDB Atlas)** – Stores sensor data, extracted features, system logs, and prediction outputs in a scalable document-based repository.
* **UI (Android – Java/Kotlin, Web – React.js)** – Delivers real-time dashboards, alerts, and system status visualization across mobile and web platforms.
* **Project Management (ProjectLibre)** – Supports task scheduling, dependency tracking, and critical path analysis for effective project planning.

### **Hardware**

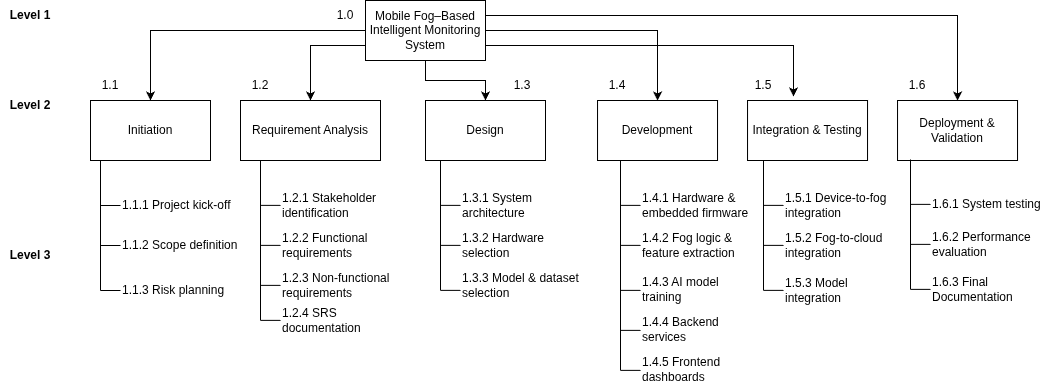
* **ESP32 Microcontroller** – Acts as the edge controller for sensor interfacing, data sampling, and communication with the fog node.
* **CT Sensors (Non-Intrusive)** – Monitor electrical current without direct contact, enabling safe and non-invasive subsystem health monitoring.
* **Burden Resistors & Signal Conditioning** – Convert sensor output into stable voltage signals and reduce noise for accurate measurements.
* **Electromechanical Relays** – Enable hardware-level control actions such as isolation or shutdown during fault conditions.
* **Smartphone (Fog Node)** – Functions as the fog computing unit for real-time analytics, AI inference, and local system control.

1. **Work Breakdown Structure (WBS).**

**1.0 Mobile Fog–Based Intelligent Monitoring System**

* **1.1 Initiation** 1.1.1 Project kick-off  
   1.1.2 Scope definition  
   1.1.3 Risk planning
* **1.2 Requirement Analysis** 1.2.1 Stakeholder identification  
   1.2.2 Functional requirements  
   1.2.3 Non-functional requirements  
   1.2.4 SRS documentation
* **1.3 Design** 1.3.1 System architecture  
   1.3.2 Hardware selection  
   1.3.3 Model & dataset selection
* **1.4 Development** 1.4.1 Hardware & embedded firmware  
   1.4.2 Fog logic & feature extraction  
   1.4.3 AI model training  
   1.4.4 Backend services  
   1.4.5 Frontend dashboards
* **1.5 Integration & Testing** 1.5.1 Device-to-fog integration  
   1.5.2 Fog-to-cloud integration  
   1.5.3 Model integration
* **1.6 Deployment & Validation** 1.6.1 System testing  
   1.6.2 Performance evaluation

1.6.3 Final Documentation



*Diagram 1 - Work breakdown structure.*

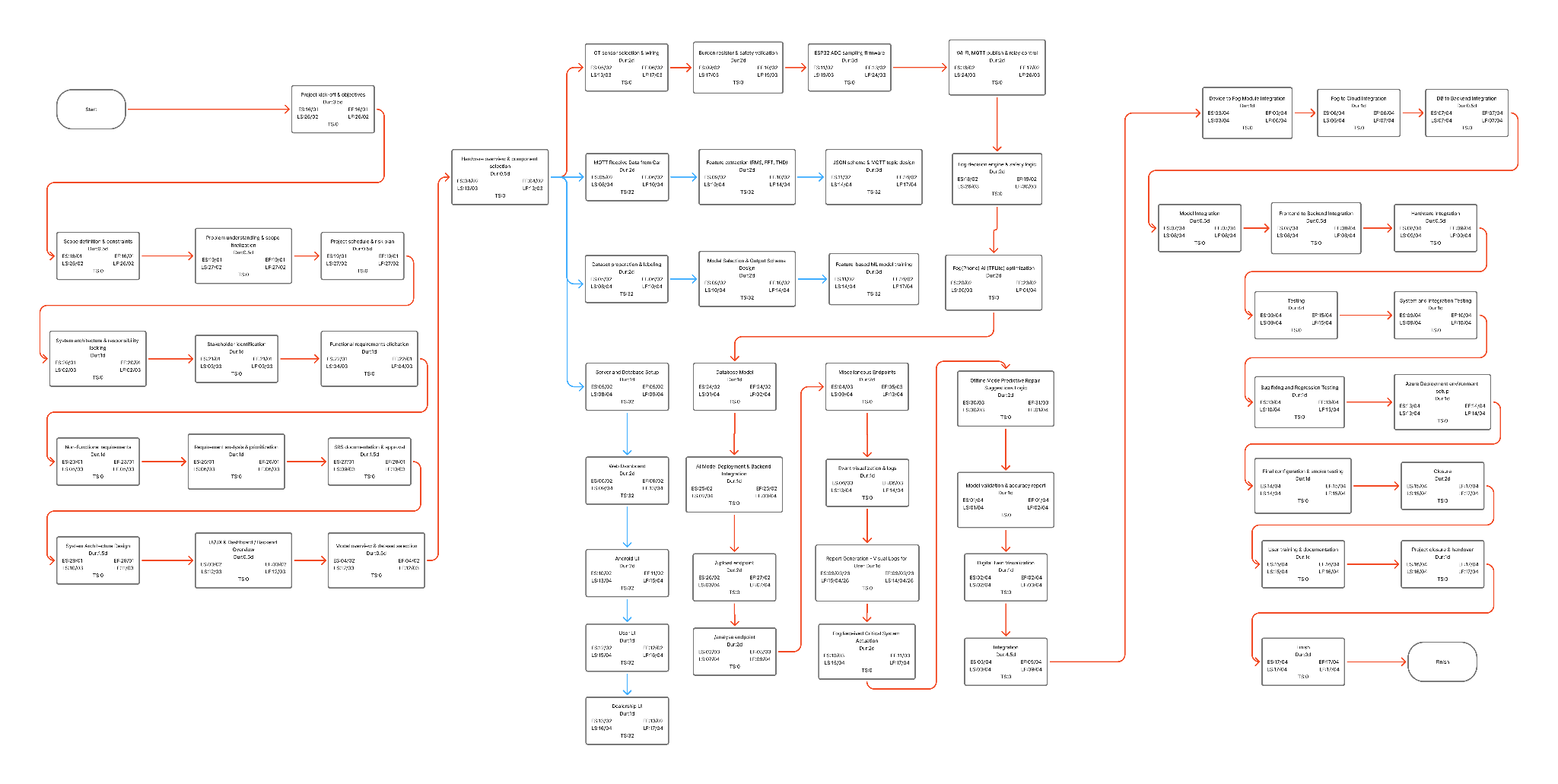
1. **P.E.R.T. Chart and Critical Path identification.**

In this project, **PERT analysis** is applied to map out key activities such as requirement analysis, system design, development, and integration. By analyzing task dependencies and durations, the PERT chart helps

identify critical tasks with zero slack that directly impact the overall timeline. This facilitates effective scheduling and resource allocation to

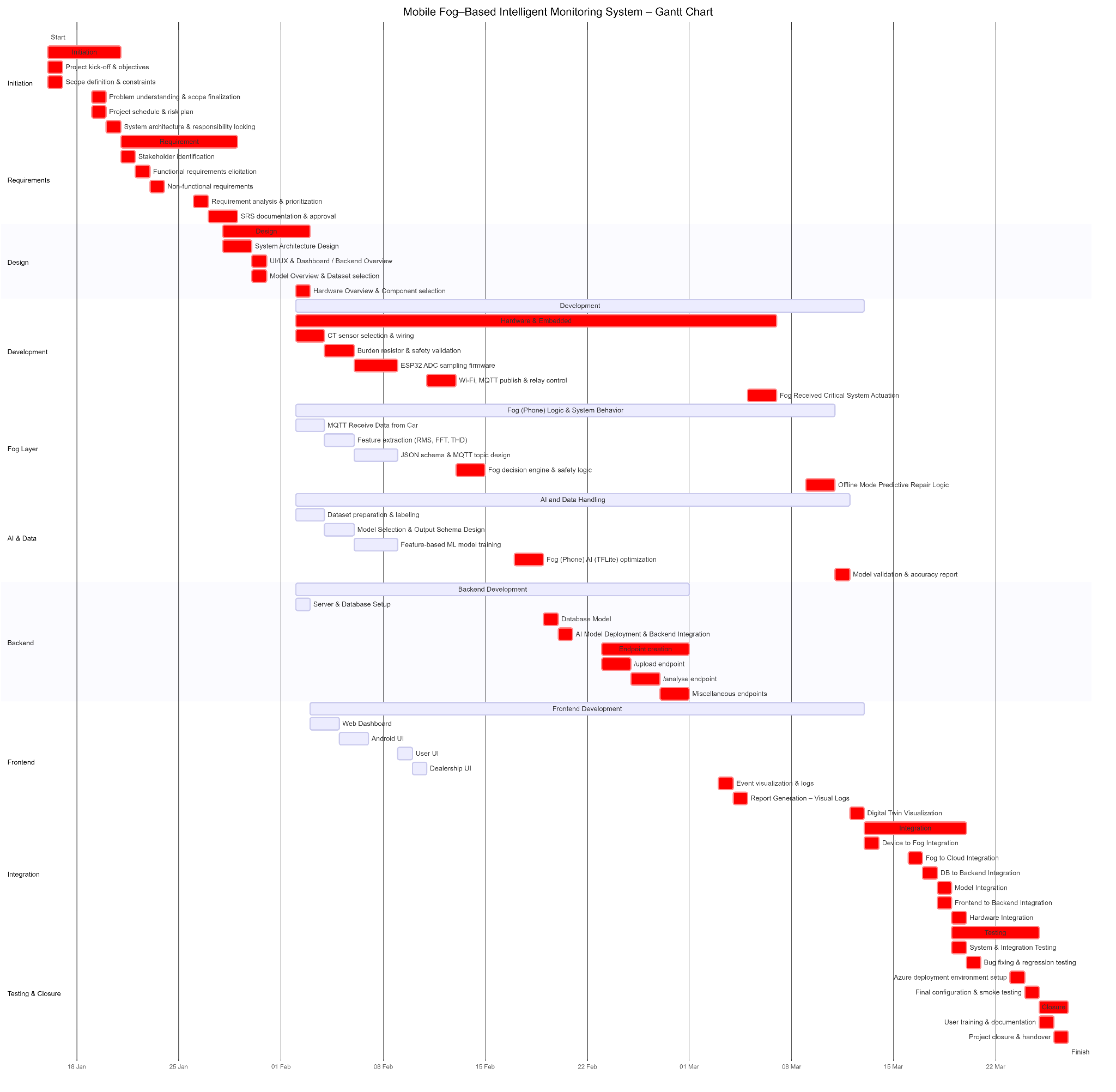
ensure timely project delivery.

**5.1 Manually drawn PERT Chart**



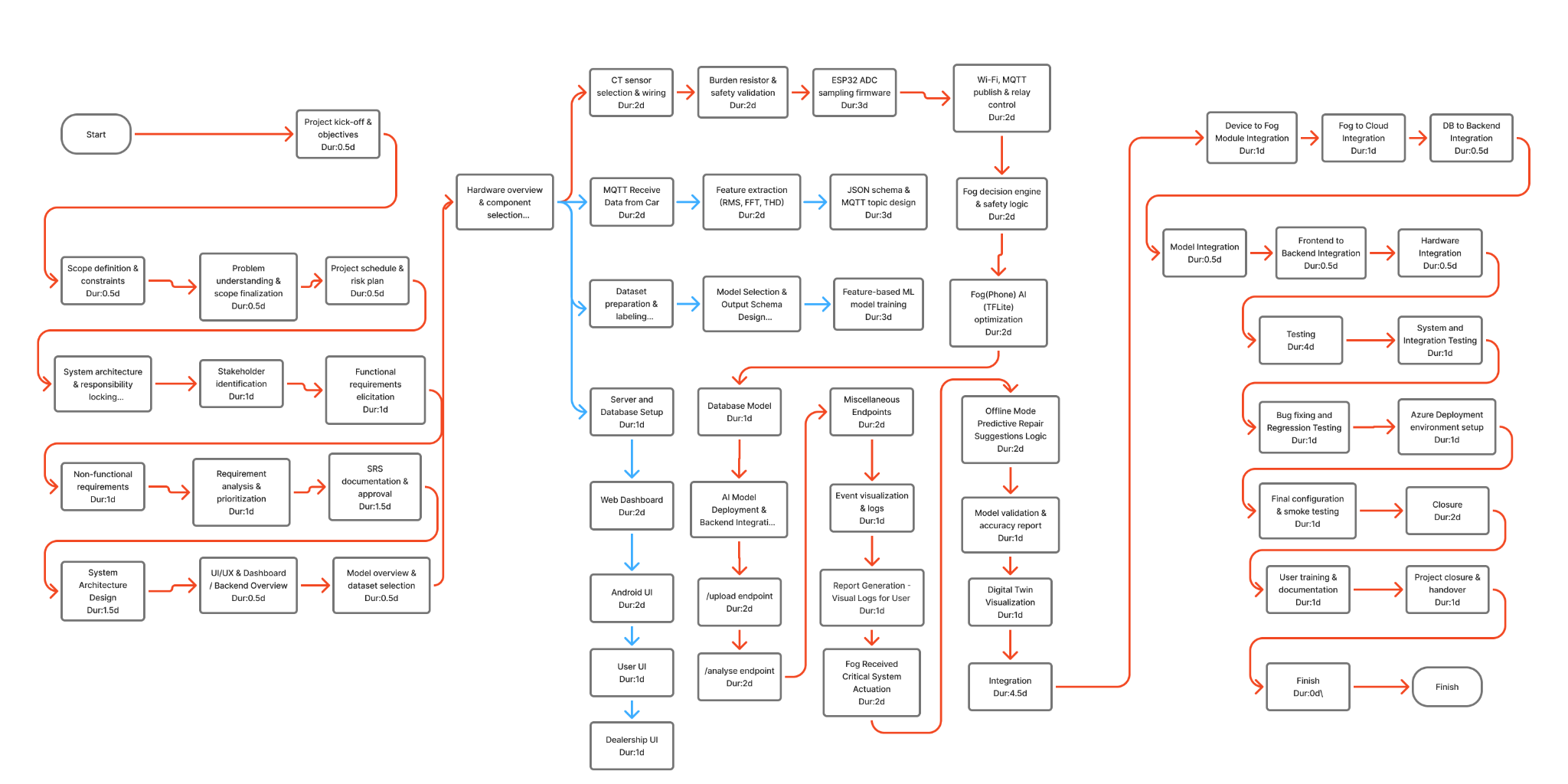
***PERT Chart*** *for the Project. Visit this* [*link*](https://raw.githubusercontent.com/Software-engineering-Project-VIT/SE_Project_docs/refs/heads/main/Complete%20PERT%20Chart%20%E2%80%93%20All%20Tasks%20Included%20(5).png) *for a higher fidelity diagram in case it is not readable.*

**5.2 Manually drawn Gantt Chart.**



***Gantt*** *for the Project. Visit this* [*link*](https://github.com/Software-engineering-Project-VIT/SE_Project_docs/blob/main/gnatt%20chart.png?raw=true) *for a higher fidelity diagram in case it is not readable.*

**5.3 Manually drawn Activity Network.**

***Activity Network Diagram*** *for the Project. Visit this* [*link*](https://github.com/Software-engineering-Project-VIT/SE_Project_docs/blob/main/activity_network.png?raw=true) *for a higher fidelity diagram in case it is not readable.*

**5.4.1 Deterministic Manual Calculations for ES, EF, LS, LF and Slack time for each task. (CPM)**

\*Slack Time = LS - ES (or) LF - EF

| **No.** | **Activity Name** | **Duration (days)** | **ES** | **EF** | **LS** | **LF** | **Slack** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Start | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | **Initiation** | **3** | **0** | **3** | **0** | **3** | **0** |
| 3 | Project kick-off & objectives | 0.5 | 0 | 0.5 | 0 | 0.5 | 0 |
| 4 | Scope definition & constraints | 0.5 | 0.5 | 1 | 0.5 | 1 | 0 |
| 5 | Problem understanding & scope finalization | 0.5 | 3 | 3.5 | 3 | 3.5 | 0 |
| 6 | Project schedule & risk plan | 0.5 | 3.5 | 4 | 3.5 | 4 | 0 |
| 7 | System architecture & responsibility locking | 1 | 4 | 5 | 4 | 5 | 0 |
| **8** | **Requirement** | **5.5** | **5** | **10.5** | **5** | **10.5** | **0** |
| 9 | Stakeholder identification | 1 | 5 | 6 | 5 | 6 | 0 |
| 10 | Functional requirements elicitation | 1 | 6 | 7 | 6 | 7 | 0 |
| 11 | Non-functional requirements | 1 | 7 | 8 | 7 | 8 | 0 |
| 12 | Requirement analysis & prioritization | 1 | 8 | 9 | 8 | 9 | 0 |
| 13 | SRS documentation & approval | 1.5 | 9 | 10.5 | 9 | 10.5 | 0 |
| **14** | **Design** | **3** | **10.5** | **13.5** | **10.5** | **13.5** | **0** |
| 15 | System Architecture Design | 1.5 | 10.5 | 12 | 10.5 | 12 | 0 |
| 16 | UI/UX & Dashboard / Backend Overview | 0.5 | 12 | 12.5 | 12 | 12.5 | 0 |
| 17 | Model Overview & Dataset selection | 0.5 | 12.5 | 13 | 12.5 | 13 | 0 |
| 18 | Hardware Overview & Component selection | 0.5 | 13 | 13.5 | 13 | 13.5 | 0 |
| **19** | **Development** | **29** | **13.5** | **42.5** | **23.5** | **52.5** | **10** |
| **20** | **Hardware and Embedded** | **25** | **13.5** | **38.5** | **13.5** | **38.5** | **0** |
| 21 | CT sensor selection & wiring | 2 | 13.5 | 15.5 | 13.5 | 15.5 | 0 |
| 22 | Burden resistor & safety validation | 2 | 15.5 | 17.5 | 15.5 | 17.5 | 0 |
| 23 | ESP32 ADC sampling firmware | 3 | 17.5 | 20.5 | 17.5 | 20.5 | 0 |
| 24 | Wi-Fi, MQTT publish & relay control | 2 | 20.5 | 22.5 | 20.5 | 22.5 | 0 |
| 25 | Fog Received Critical System Actuation | 2 | 38.5 | 40.5 | 38.5 | 40.5 | 0 |
| **26** | **Fog(Phone) Logic & System Behavior** | **27** | **13.5** | **40.5** | **25.5** | **52.5** | **12** |
| 27 | MQTT Receive Data from Car | 2 | 13.5 | 15.5 | 45.5 | 47.5 | 32 |
| 28 | Feature extraction (RMS, FFT, THD) | 2 | 15.5 | 17.5 | 47.5 | 49.5 | 32 |
| 29 | JSON schema & MQTT topic design | 3 | 17.5 | 20.5 | 49.5 | 52.5 | 32 |
| 30 | Fog decision engine & safety logic | 2 | 22.5 | 24.5 | 22.5 | 24.5 | 0 |
| 31 | Offline Mode Predictive Repair Suggestions Logic | 2 | 40.5 | 42.5 | 40.5 | 42.5 | 0 |
| **32** | **AI and Data Handling** | **28** | **13.5** | **41.5** | **24.5** | **52.5** | **11** |
| 33 | Dataset preparation & labeling | 2 | 13.5 | 15.5 | 45.5 | 47.5 | 32 |
| 34 | Model Selection and Output Schema Design | 2 | 15.5 | 17.5 | 47.5 | 49.5 | 32 |
| 35 | Feature-based ML model training | 3 | 17.5 | 20.5 | 49.5 | 52.5 | 32 |
| 36 | Fog(Phone) AI (TFLite) optimization | 2 | 24.5 | 26.5 | 24.5 | 26.5 | 0 |
| 37 | Model validation & accuracy report | 1 | 42.5 | 43.5 | 42.5 | 43.5 | 0 |
| **38** | **Backend Development** | **21** | **13.5** | **34.5** | **25.5** | **46.5** | **12** |
| 39 | Server and Database Setup | 1 | 13.5 | 14.5 | 45.5 | 46.5 | 32 |
| 40 | Database Model | 1 | 26.5 | 27.5 | 26.5 | 27.5 | 0 |
| 41 | AI Model Deployment and Backend Integration | 1 | 27.5 | 28.5 | 27.5 | 28.5 | 0 |
| **42** | **Endpoint creation** | **6** | **28.5** | **34.5** | **28.5** | **34.5** | **0** |
| 43 | /upload endpoint | 2 | 28.5 | 30.5 | 28.5 | 30.5 | 0 |
| 44 | /analyse endpoint | 2 | 30.5 | 32.5 | 30.5 | 32.5 | 0 |
| 45 | Miscellaneous Endpoints | 2 | 32.5 | 34.5 | 32.5 | 34.5 | 0 |
| **46** | **Frontend Development** | **28** | **14.5** | **42.5** | **32.5** | **60.5** | **10** |
| 47 | Web Dashboard | 2 | 14.5 | 16.5 | 46.5 | 48.5 | 32 |
| 48 | Android UI | 2 | 16.5 | 18.5 | 48.5 | 50.5 | 32 |
| 49 | User UI | 1 | 18.5 | 19.5 | 50.5 | 51.5 | 32 |
| 50 | Dealership UI | 1 | 19.5 | 20.5 | 51.5 | 52.5 | 32 |
| 51 | Event visualization & logs | 1 | 34.5 | 35.5 | 34.5 | 35.5 | 0 |
| 52 | Report Generation - Visual Logs | 1 | 35.5 | 36.5 | 35.5 | 36.5 | 0 |
| 53 | Digital Twin Visualization | 1 | 43.5 | 44.5 | 43.5 | 44.5 | 0 |
| **54** | **Integration** | **4** | **44.5** | **48.5** | **44.5** | **48.5** | **0** |
| 55 | Device to Fog Module Integration | 1 | 44.5 | 45.5 | 44.5 | 45.5 | 0 |
| 56 | Fog to Cloud Integration | 1 | 45.5 | 46.5 | 45.5 | 46.5 | 0 |
| 57 | DB to Backend Integration | 0.5 | 46.5 | 47 | 46.5 | 47 | 0 |
| 58 | Model Integration | 0.5 | 47 | 47.5 | 47 | 47.5 | 0 |
| 59 | Frontend to Backend Integration | 0.5 | 47.5 | 48 | 47.5 | 48 | 0 |
| 60 | Hardware Integration | 0.5 | 48 | 48.5 | 48 | 48.5 | 0 |
| **61** | **Testing** | **4** | **48.5** | **52.5** | **48.5** | **52.5** | **0** |
| 62 | System and Integration Testing | 1 | 48.5 | 49.5 | 48.5 | 49.5 | 0 |
| 63 | Bug fixing and Regression Testing | 1 | 49.5 | 50.5 | 49.5 | 50.5 | 0 |
| 64 | Azure Deployment environment setup | 1 | 50.5 | 51.5 | 50.5 | 51.5 | 0 |
| 65 | Final configuration & smoke testing | 1 | 51.5 | 52.5 | 51.5 | 52.5 | 0 |
| **66** | **Closure** | **2** | **52.5** | **54.5** | **52.5** | **54.5** | **0** |
| 67 | User training & documentation | 1 | 52.5 | 53.5 | 52.5 | 53.5 | 0 |
| 68 | Project closure & handover | 1 | 53.5 | 54.5 | 53.5 | 54.5 | 0 |
| 69 | Finish | 0 | 54.5 | 54.5 | 54.5 | 54.5 | 0 |

**5.4.2 Probabilistic Manual Calculations for ES, EF, LS, LF and Slack time for each task.(PERT)**

\*Slack Time = LS - ES (or) LF - EF

Since the CPM analysis of the project is carried out in days, the PERT (probabilistic scheduling) analysis is also represented in days.  
  
 PERT uses three time estimates for each activity:  
 a = Optimistic time (hours)  
 m = Most likely time (hours)  
 b = Pessimistic time (hours)  
  
 Expected time:  
 tₑ = (a + 4m + b) / 6

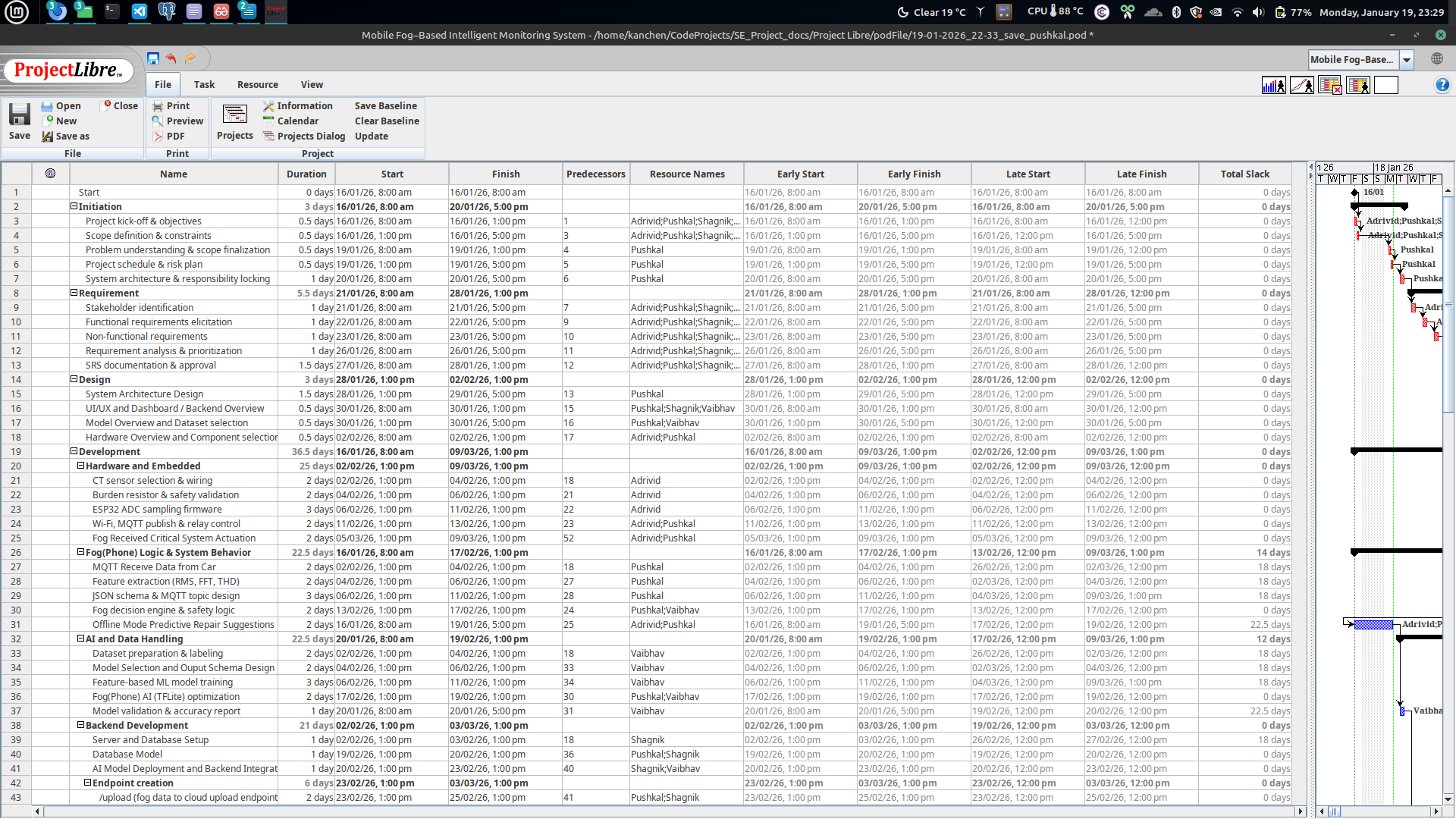
| **No.** | **Activity Name** | **A**  **(days)** | **M**  **(days)** | **B**  **(days)** | **Duration (days)** | **ES** | **EF** | **LS** | **LF** | **Slack** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | Start | 0.0 | 0.0 | 0.0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | **Initiation** | 2.7 | 3.0 | 3.3 | **3** | **0** | **3** | **0** | **3** | **0** |
| 3 | Project kick-off & objectives | 0.45 | 0.5 | 0.55 | 0.5 | 0 | 0.5 | 0 | 0.5 | 0 |
| 4 | Scope definition & constraints | 0.45 | 0.5 | 0.55 | 0.5 | 0.5 | 1 | 0.5 | 1 | 0 |
| 5 | Problem understanding & scope finalization | 0.45 | 0.5 | 0.55 | 0.5 | 3 | 3.5 | 3 | 3.5 | 0 |
| 6 | Project schedule & risk plan | 0.45 | 0.5 | 0.55 | 0.5 | 3.5 | 4 | 3.5 | 4 | 0 |
| 7 | System architecture & responsibility locking | 0.9 | 1.0 | 1.1 | 1 | 4 | 5 | 4 | 5 | 0 |
| **8** | **Requirement** | **4.95** | **5.5** | **6.05** | **5.5** | **5** | **10.5** | **5** | **10.5** | **0** |
| 9 | Stakeholder identification | 0.9 | 1.0 | 1.1 | 1 | 5 | 6 | 5 | 6 | 0 |
| 10 | Functional requirements elicitation | 0.9 | 1.0 | 1.1 | 1 | 6 | 7 | 6 | 7 | 0 |
| 11 | Non-functional requirements | 0.9 | 1.0 | 1.1 | 1 | 7 | 8 | 7 | 8 | 0 |
| 12 | Requirement analysis & prioritization | 0.9 | 1.0 | 1.1 | 1 | 8 | 9 | 8 | 9 | 0 |
| 13 | SRS documentation & approval | 1.35 | 1.5 | 1.65 | 1.5 | 9 | 10.5 | 9 | 10.5 | 0 |
| **14** | **Design** | **2.7** | **3.0** | **3.3** | **3** | **10.5** | **13.5** | **10.5** | **13.5** | **0** |
| 15 | System Architecture Design | 1.35 | 1.5 | 1.65 | 1.5 | 10.5 | 12 | 10.5 | 12 | 0 |
| 16 | UI/UX & Dashboard / Backend Overview | 0.45 | 0.5 | 0.55 | 0.5 | 12 | 12.5 | 12 | 12.5 | 0 |
| 17 | Model Overview & Dataset selection | 0.45 | 0.5 | 0.55 | 0.5 | 12.5 | 13 | 12.5 | 13 | 0 |
| 18 | Hardware Overview & Component selection | 0.45 | 0.5 | 0.55 | 0.5 | 13 | 13.5 | 13 | 13.5 | 0 |
| **19** | **Development** | **26.1** | **29.0** | **31.9** | **29** | **13.5** | **42.5** | **23.5** | **52.5** | **10** |
| **20** | **Hardware and Embedded** | **22.5** | **25.0** | **27.5** | **25** | **13.5** | **38.5** | **13.5** | **38.5** | **0** |
| 21 | CT sensor selection & wiring | 1.8 | 2.0 | 2.2 | 2 | 13.5 | 15.5 | 13.5 | 15.5 | 0 |
| 22 | Burden resistor & safety validation | 1.8 | 2.0 | 2.2 | 2 | 15.5 | 17.5 | 15.5 | 17.5 | 0 |
| 23 | ESP32 ADC sampling firmware | 2.7 | 3.0 | 3.3 | 3 | 17.5 | 20.5 | 17.5 | 20.5 | 0 |
| 24 | Wi-Fi, MQTT publish & relay control | 1.8 | 2.0 | 2.2 | 2 | 20.5 | 22.5 | 20.5 | 22.5 | 0 |
| 25 | Fog Received Critical System Actuation | 1.8 | 2.0 | 2.2 | 2 | 38.5 | 40.5 | 38.5 | 40.5 | 0 |
| **26** | **Fog(Phone) Logic & System Behavior** | **24.3** | **27.0** | **29.7** | **27** | **13.5** | **40.5** | **25.5** | **52.5** | **12** |
| 27 | MQTT Receive Data from Car | 1.8 | 2.0 | 2.2 | 2 | 13.5 | 15.5 | 45.5 | 47.5 | 32 |
| 28 | Feature extraction (RMS, FFT, THD) | 1.8 | 2.0 | 2.2 | 2 | 15.5 | 17.5 | 47.5 | 49.5 | 32 |
| 29 | JSON schema & MQTT topic design | 2.7 | 3.0 | 3.3 | 3 | 17.5 | 20.5 | 49.5 | 52.5 | 32 |
| 30 | Fog decision engine & safety logic | 1.8 | 2.0 | 2.2 | 2 | 22.5 | 24.5 | 22.5 | 24.5 | 0 |
| 31 | Offline Mode Predictive Repair Suggestions Logic | 1.8 | 2.0 | 2.2 | 2 | 40.5 | 42.5 | 40.5 | 42.5 | 0 |
| **32** | **AI and Data Handling** | **25.2** | **28.0** | **30.8** | **28** | **13.5** | **41.5** | **24.5** | **52.5** | **11** |
| 33 | Dataset preparation & labeling | 1.8 | 2.0 | 2.2 | 2 | 13.5 | 15.5 | 45.5 | 47.5 | 32 |
| 34 | Model Selection and Output Schema Design | 1.8 | 2.0 | 2.2 | 2 | 15.5 | 17.5 | 47.5 | 49.5 | 32 |
| 35 | Feature-based ML model training | 2.7 | 3.0 | 3.3 | 3 | 17.5 | 20.5 | 49.5 | 52.5 | 32 |
| 36 | Fog(Phone) AI (TFLite) optimization | 1.8 | 2.0 | 2.2 | 2 | 24.5 | 26.5 | 24.5 | 26.5 | 0 |
| 37 | Model validation & accuracy report | 0.9 | 1.0 | 1.1 | 1 | 42.5 | 43.5 | 42.5 | 43.5 | 0 |
| **38** | **Backend Development** | **18.9** | **21.0** | **23.1** | **21** | **13.5** | **34.5** | **25.5** | **46.5** | **12** |
| 39 | Server and Database Setup | 0.9 | 1.0 | 1.1 | 1 | 13.5 | 14.5 | 45.5 | 46.5 | 32 |
| 40 | Database Model | 0.9 | 1.0 | 1.1 | 1 | 26.5 | 27.5 | 26.5 | 27.5 | 0 |
| 41 | AI Model Deployment and Backend Integration | 0.9 | 1.0 | 1.1 | 1 | 27.5 | 28.5 | 27.5 | 28.5 | 0 |
| **42** | **Endpoint creation** | **5.4** | **6.0** | **6.6** | **6** | **28.5** | **34.5** | **28.5** | **34.5** | **0** |
| 43 | /upload endpoint | 1.8 | 2.0 | 2.2 | 2 | 28.5 | 30.5 | 28.5 | 30.5 | 0 |
| 44 | /analyse endpoint | 1.8 | 2.0 | 2.2 | 2 | 30.5 | 32.5 | 30.5 | 32.5 | 0 |
| 45 | Miscellaneous Endpoints | 1.8 | 2.0 | 2.2 | 2 | 32.5 | 34.5 | 32.5 | 34.5 | 0 |
| **46** | **Frontend Development** | **25.2** | **28.0** | **30.8** | **28** | **14.5** | **42.5** | **32.5** | **60.5** | **10** |
| 47 | Web Dashboard | 1.8 | 2.0 | 2.2 | 2 | 14.5 | 16.5 | 46.5 | 48.5 | 32 |
| 48 | Android UI | 1.8 | 2.0 | 2.2 | 2 | 16.5 | 18.5 | 48.5 | 50.5 | 32 |
| 49 | User UI | 0.9 | 1.0 | 1.1 | 1 | 18.5 | 19.5 | 50.5 | 51.5 | 32 |
| 50 | Dealership UI | 0.9 | 1.0 | 1.1 | 1 | 19.5 | 20.5 | 51.5 | 52.5 | 32 |
| 51 | Event visualization & logs | 0.9 | 1.0 | 1.1 | 1 | 34.5 | 35.5 | 34.5 | 35.5 | 0 |
| 52 | Report Generation - Visual Logs | 0.9 | 1.0 | 1.1 | 1 | 35.5 | 36.5 | 35.5 | 36.5 | 0 |
| 53 | Digital Twin Visualization | 0.9 | 1.0 | 1.1 | 1 | 43.5 | 44.5 | 43.5 | 44.5 | 0 |
| **54** | **Integration** | **3.6** | **4.0** | **4.4** | **4** | **44.5** | **48.5** | **44.5** | **48.5** | **0** |
| 55 | Device to Fog Module Integration | 0.9 | 1.0 | 1.1 | 1 | 44.5 | 45.5 | 44.5 | 45.5 | 0 |
| 56 | Fog to Cloud Integration | 0.9 | 1.0 | 1.1 | 1 | 45.5 | 46.5 | 45.5 | 46.5 | 0 |
| 57 | DB to Backend Integration | 0.45 | 0.5 | 0.55 | 0.5 | 46.5 | 47 | 46.5 | 47 | 0 |
| 58 | Model Integration | 0.45 | 0.5 | 0.55 | 0.5 | 47 | 47.5 | 47 | 47.5 | 0 |
| 59 | Frontend to Backend Integration | 0.45 | 0.5 | 0.55 | 0.5 | 47.5 | 48 | 47.5 | 48 | 0 |
| 60 | Hardware Integration | 0.45 | 0.5 | 0.55 | 0.5 | 48 | 48.5 | 48 | 48.5 | 0 |
| **61** | **Testing** | **3.6** | **4.0** | **4.4** | **4** | **48.5** | **52.5** | **48.5** | **52.5** | **0** |
| 62 | System and Integration Testing | 0.9 | 1.0 | 1.1 | 1 | 48.5 | 49.5 | 48.5 | 49.5 | 0 |
| 63 | Bug fixing and Regression Testing | 0.9 | 1.0 | 1.1 | 1 | 49.5 | 50.5 | 49.5 | 50.5 | 0 |
| 64 | Azure Deployment environment setup | 0.9 | 1.0 | 1.1 | 1 | 50.5 | 51.5 | 50.5 | 51.5 | 0 |
| 65 | Final configuration & smoke testing | 0.9 | 1.0 | 1.1 | 1 | 51.5 | 52.5 | 51.5 | 52.5 | 0 |
| **66** | **Closure** | **1.8** | **2.0** | **2.2** | **2** | **52.5** | **54.5** | **52.5** | **54.5** | **0** |
| 67 | User training & documentation | 0.9 | 1.0 | 1.1 | 1 | 52.5 | 53.5 | 52.5 | 53.5 | 0 |
| 68 | Project closure & handover | 0.9 | 1.0 | 1.1 | 1 | 53.5 | 54.5 | 53.5 | 54.5 | 0 |
| 69 | Finish | 0.0 | 0.0 | 0.0 | 0 | 54.5 | 54.5 | 54.5 | 54.5 | 0 |

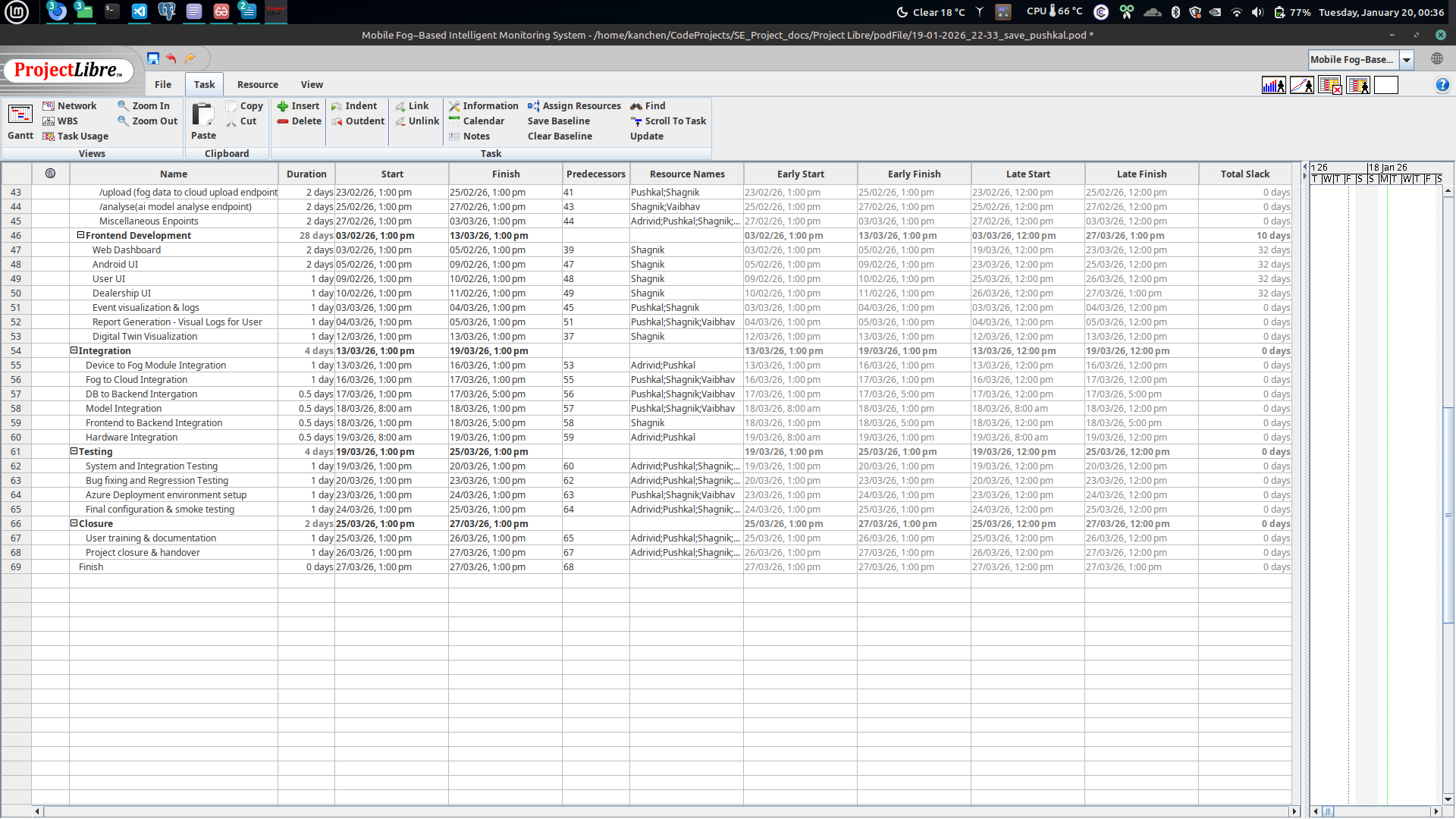
**5.5 Critical Activities Identified from manual calculation:**

* Start
* Initiation
* Project kick-off & objectives
* Scope definition & constraints
* Problem understanding & scope finalization
* Project schedule & risk plan
* System architecture & responsibility locking
* Requirement (summary)
* Stakeholder identification
* Functional requirements elicitation
* Non-functional requirements
* Requirement analysis & prioritization
* SRS documentation & approval
* Design (summary)
* System Architecture Design
* UI/UX and Dashboard / Backend Overview
* Model Overview and Dataset selection
* Hardware Overview and Component selection
* Hardware and Embedded
* CT sensor selection & wiring
* Burden resistor & safety validation
* ESP32 ADC sampling firmware
* Wi-Fi, MQTT publish & relay control
* Fog Received Critical System Actuation
* Fog decision engine & safety logic
* Offline Mode Predictive Repair Suggestions Logic
* Fog(Phone) AI (TFLite) optimization
* Model validation & accuracy report
* Database Model
* AI Model Deployment and Backend Integration
* Endpoint creation
* /upload endpoint
* /analyse endpoint
* Miscellaneous Endpoints
* Event visualization & logs
* Report Generation – Visual Logs for User
* Digital Twin Visualization
* Integration
* Device to Fog Module Integration
* Fog to Cloud Integration
* DB to Backend Integration
* Model Integration
* Frontend to Backend Integration
* Hardware Integration
* Testing
* System and Integration Testing
* Bug fixing and Regression Testing
* Azure Deployment environment setup
* Final configuration & smoke testing
* Closure
* User training & documentation
* Project closure & handover
* Finish

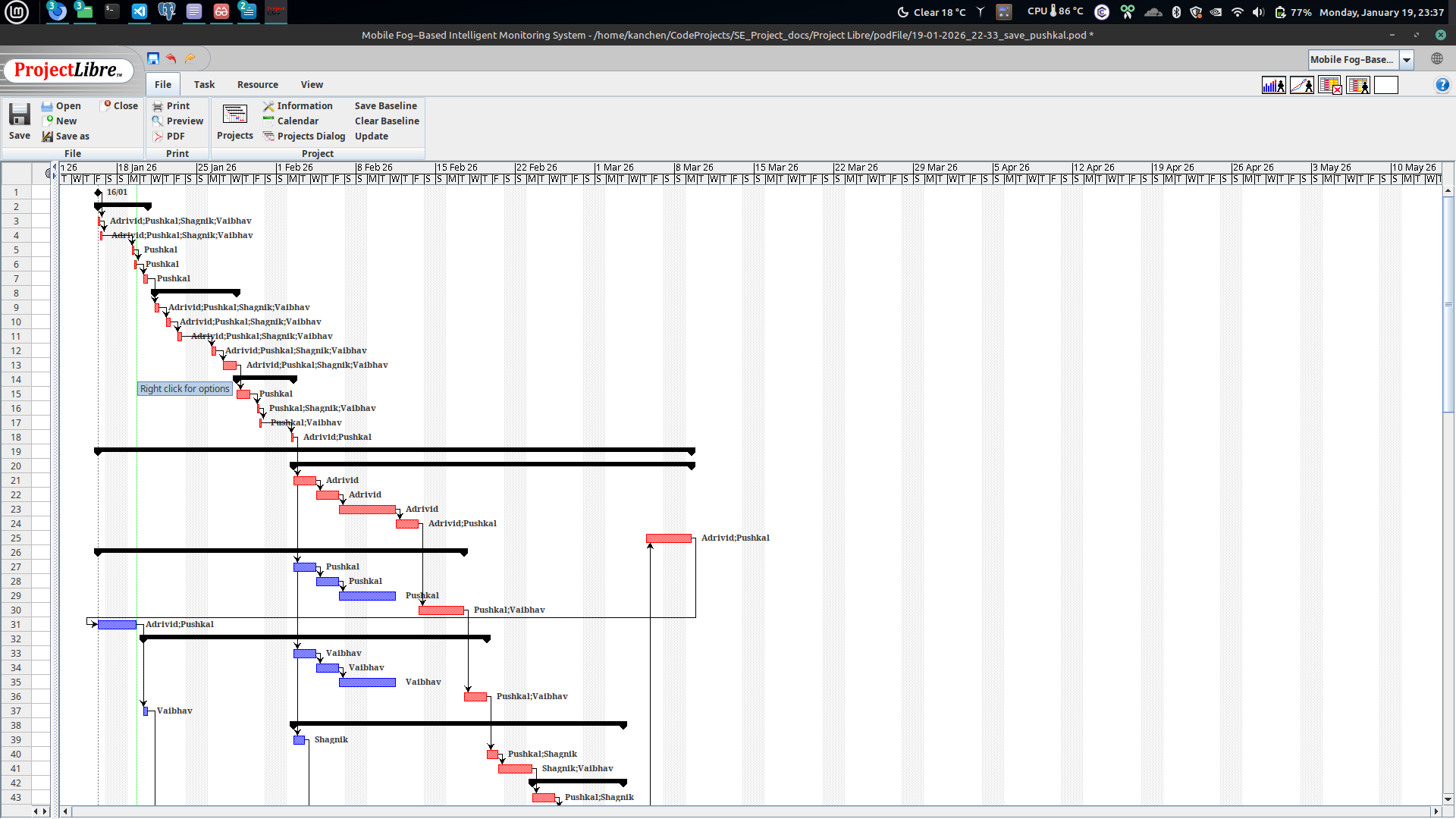
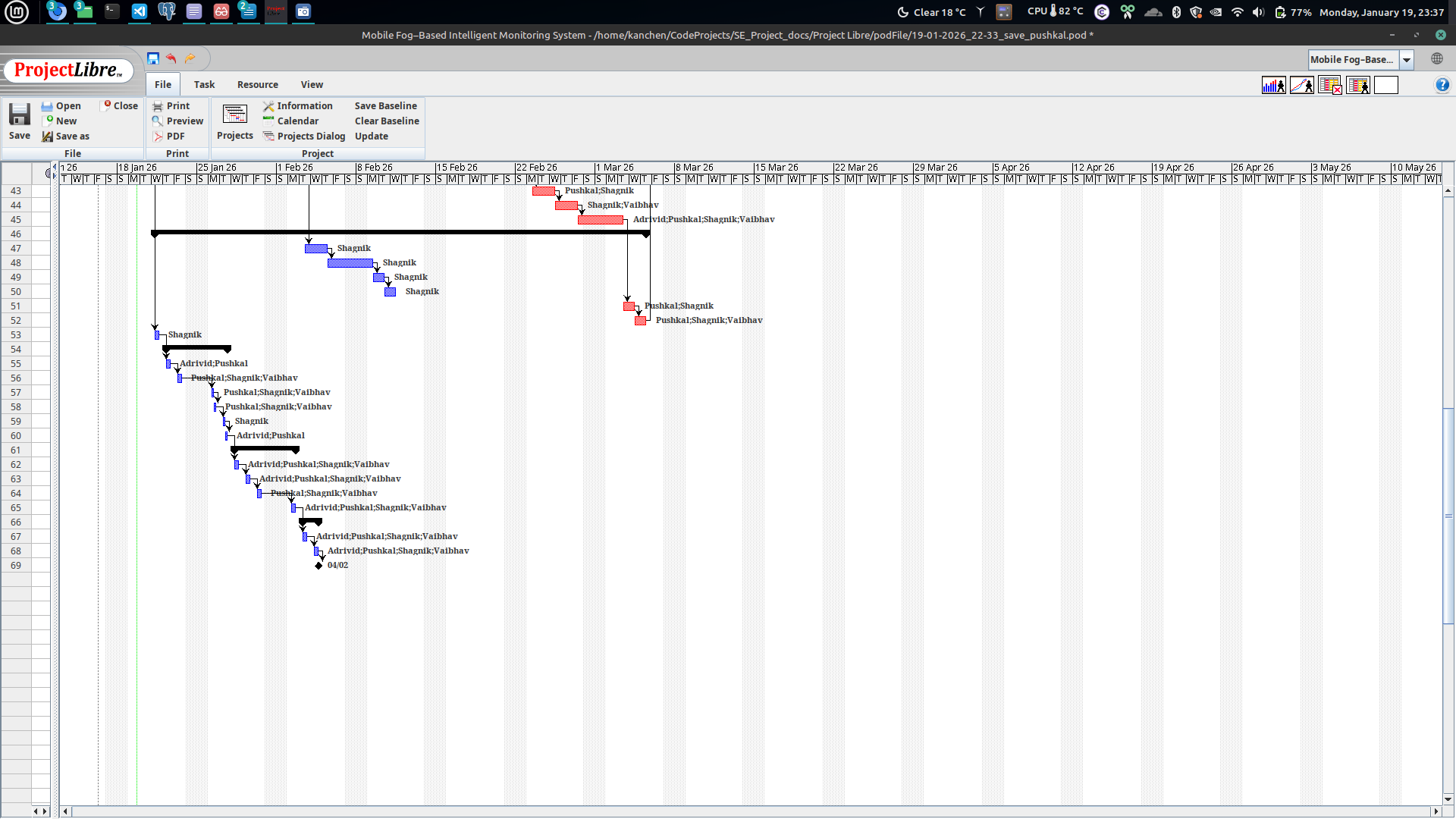
1. **Project-Libre Screenshots.**

**6.1 Activity List Screenshots.**

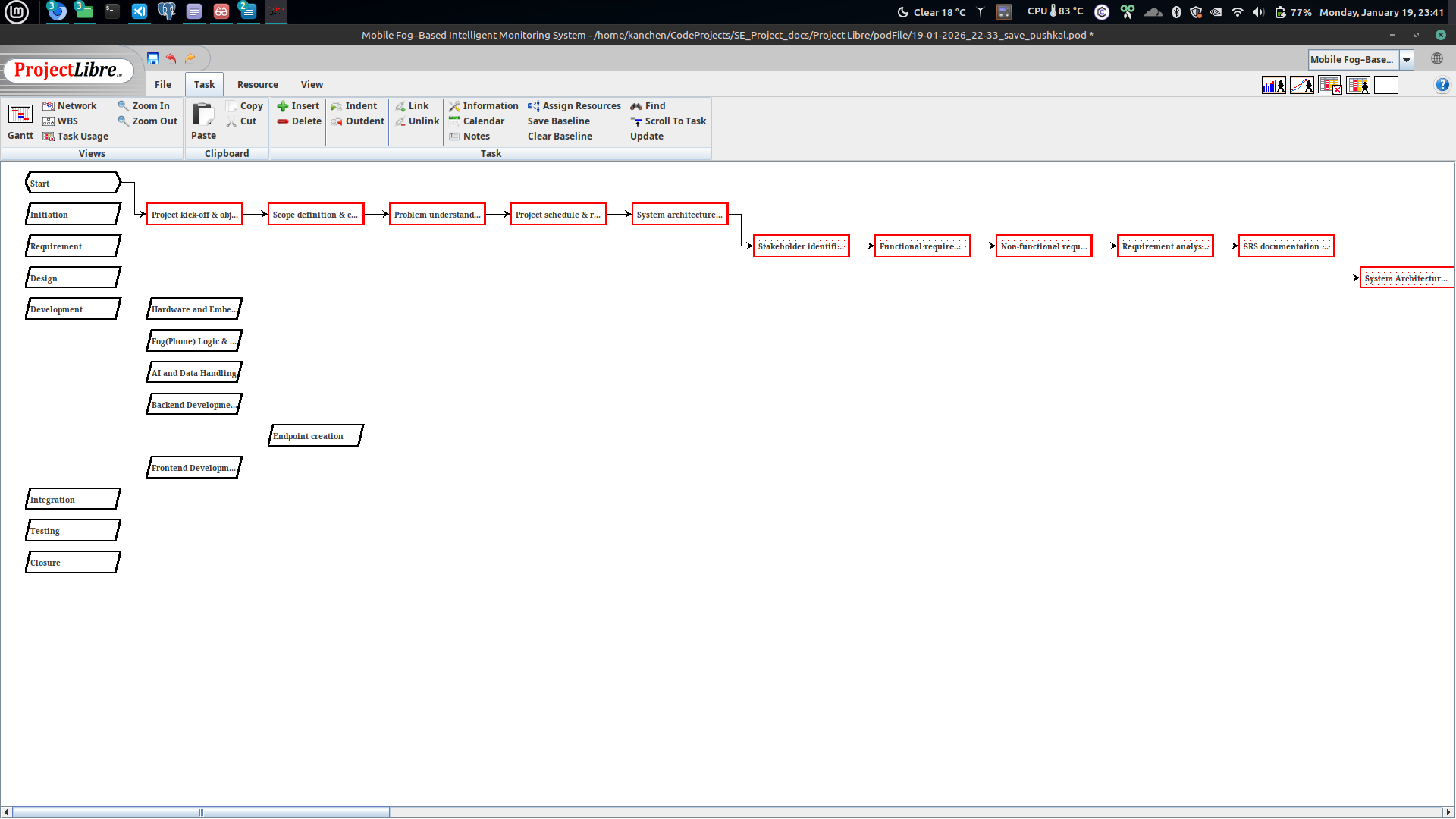


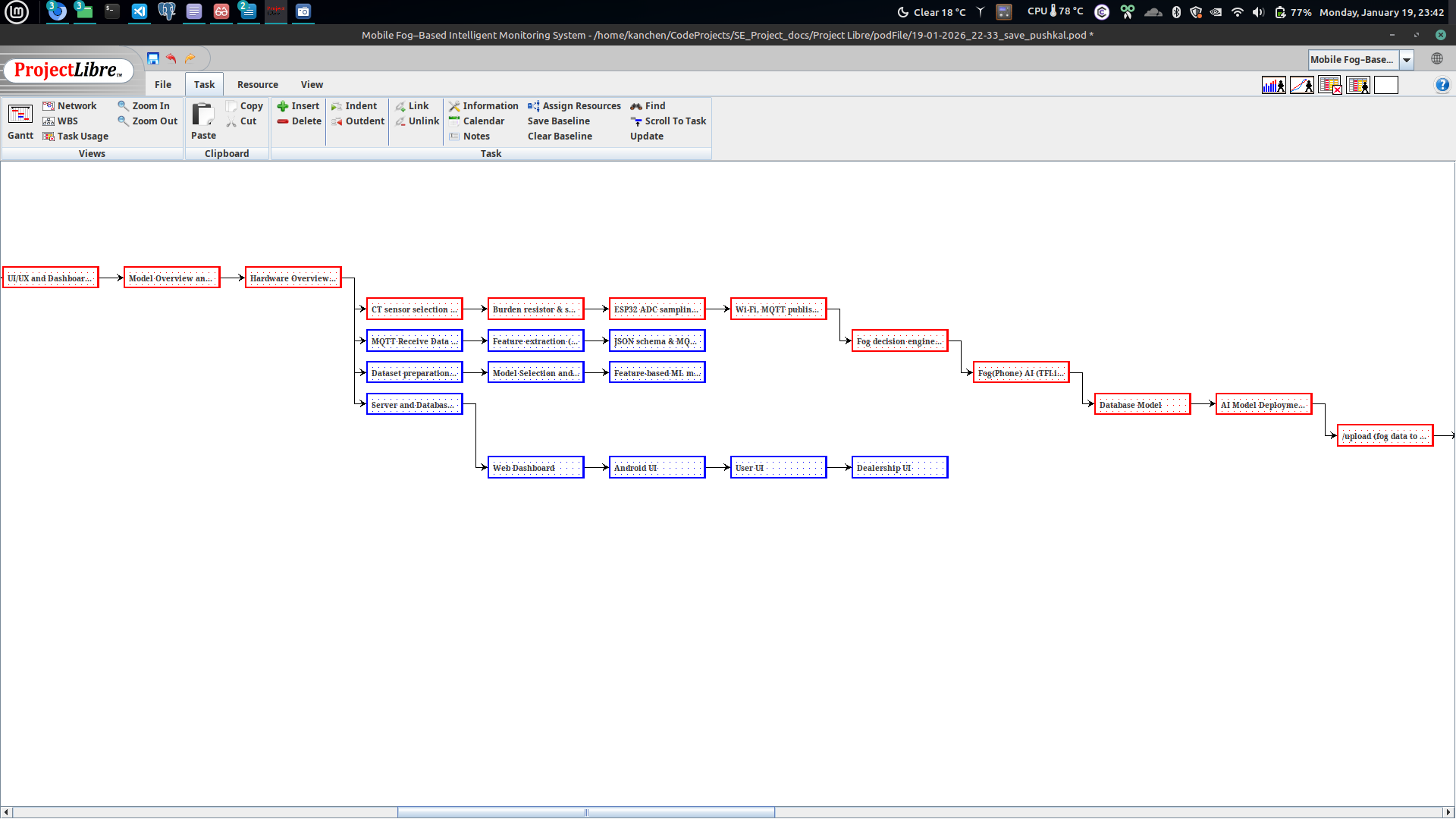


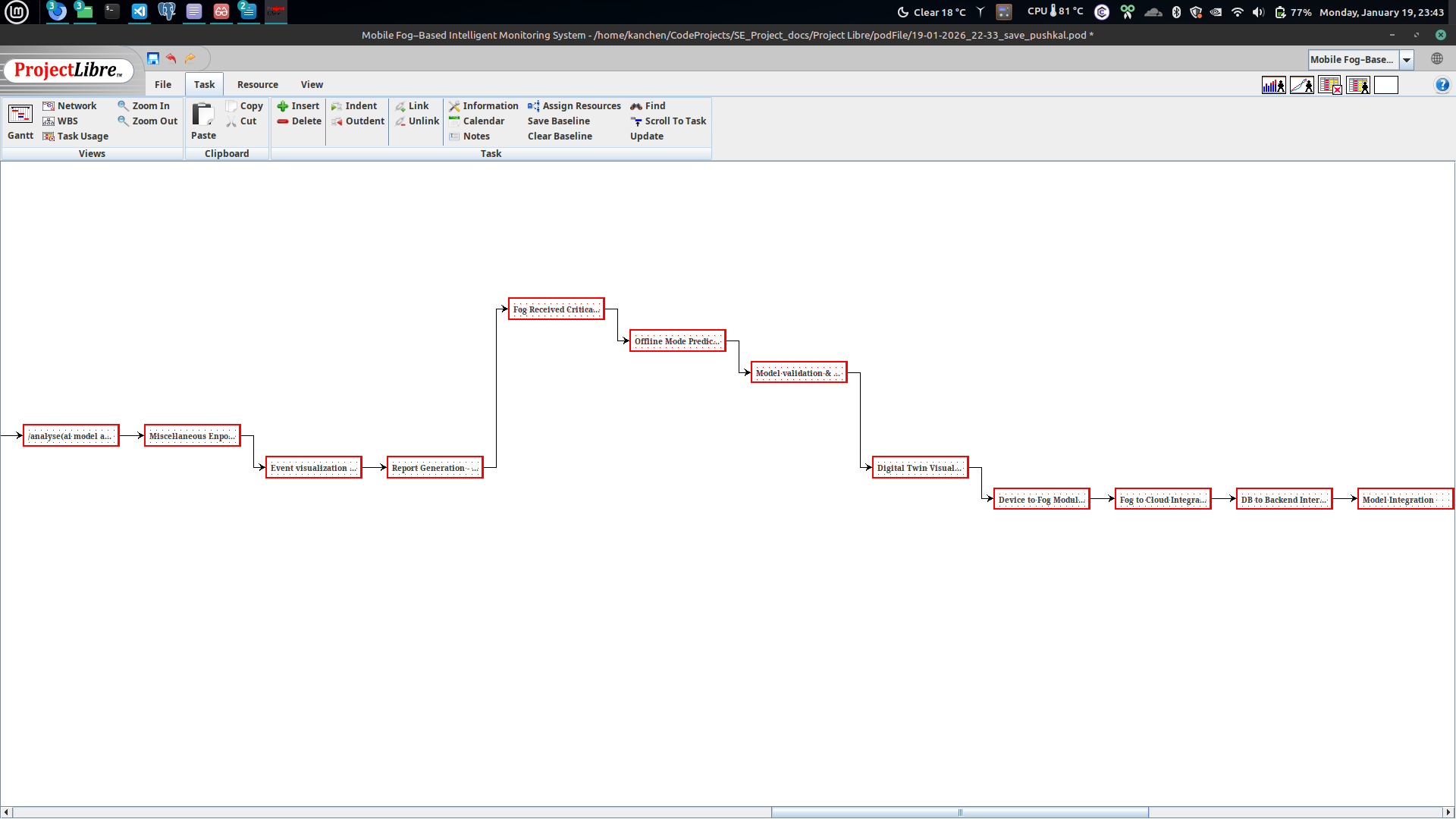
**6.2 Gantt Chart.**

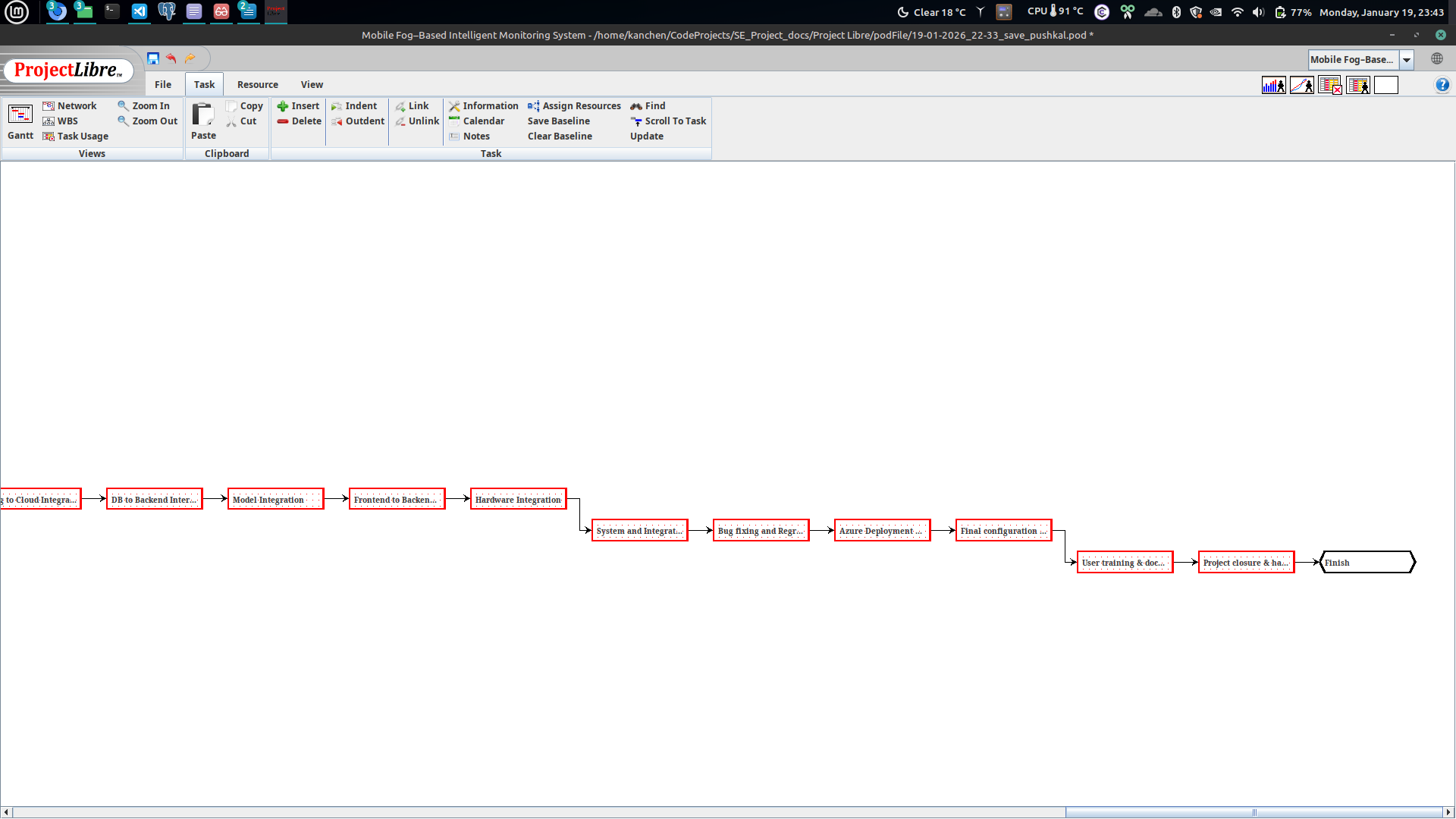
  


**6.3 Activity Network.**

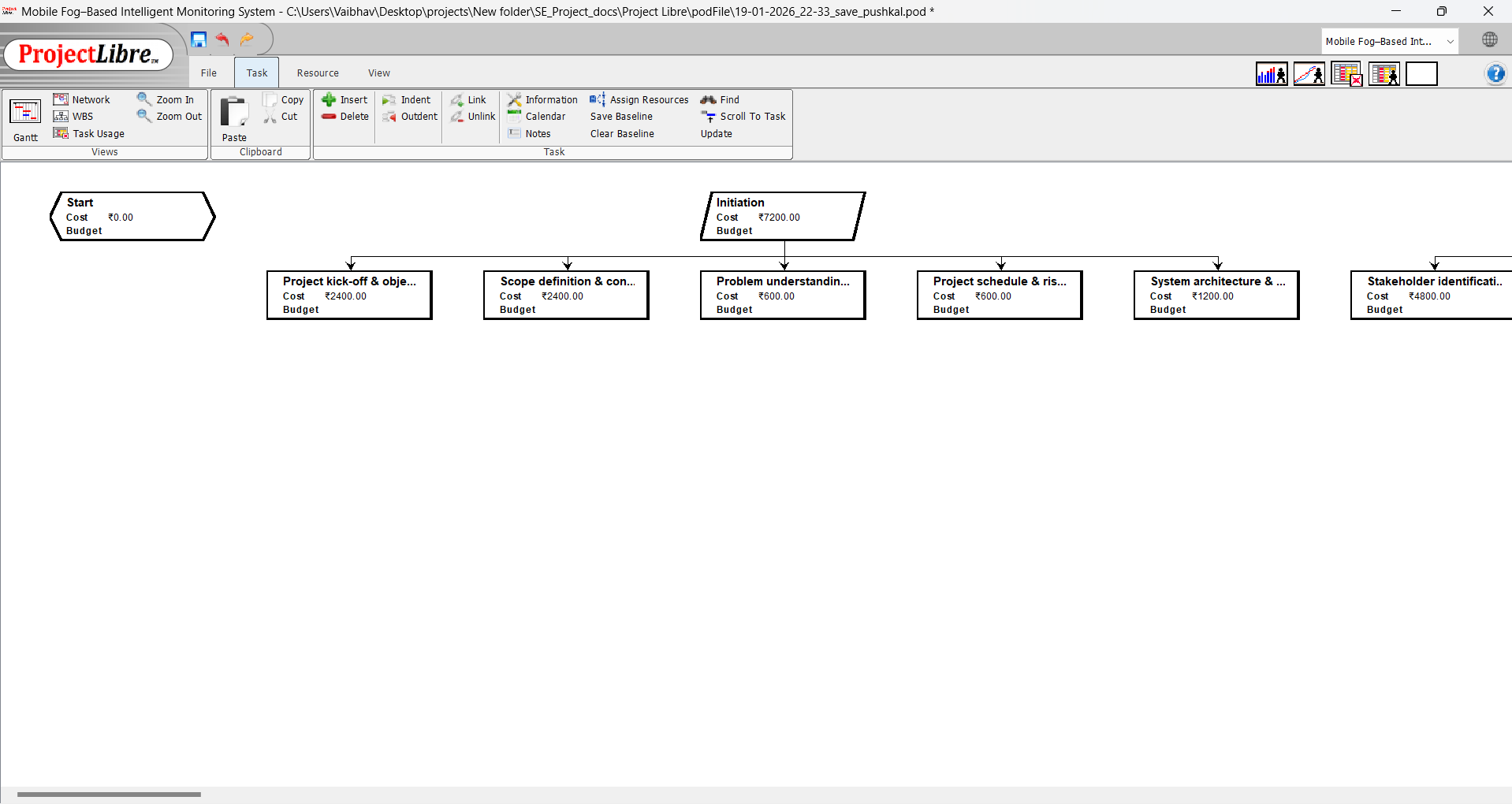


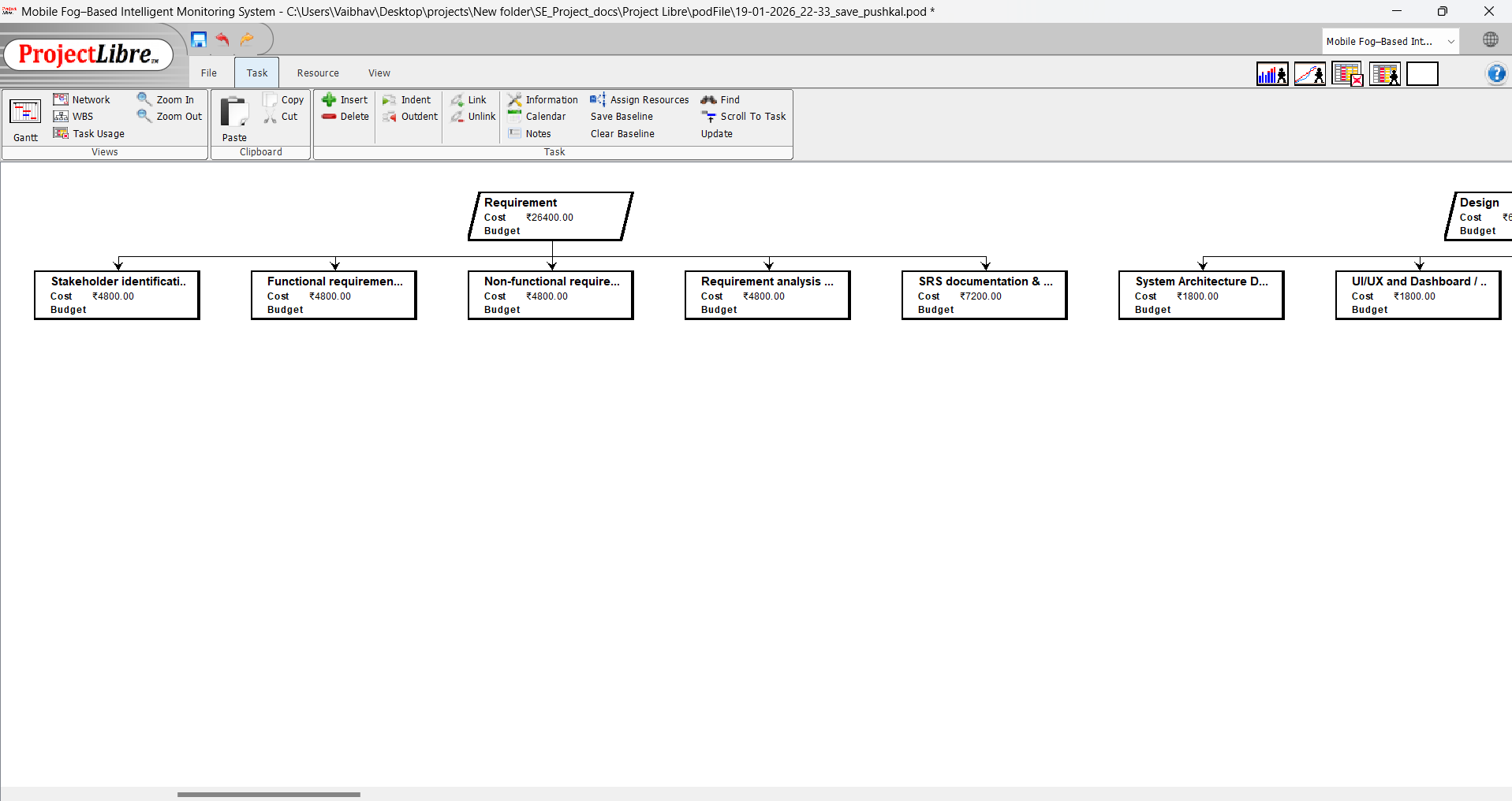


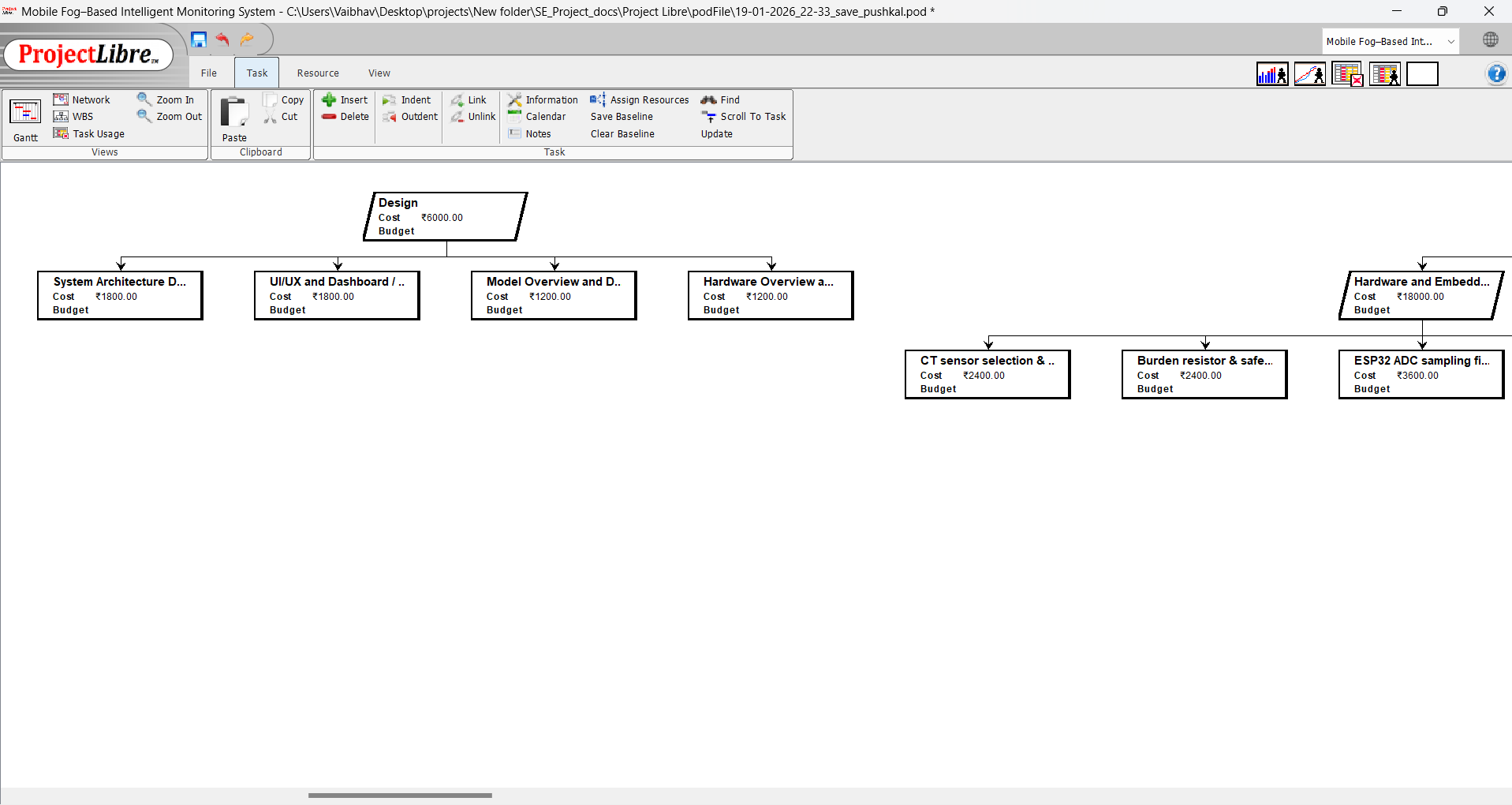




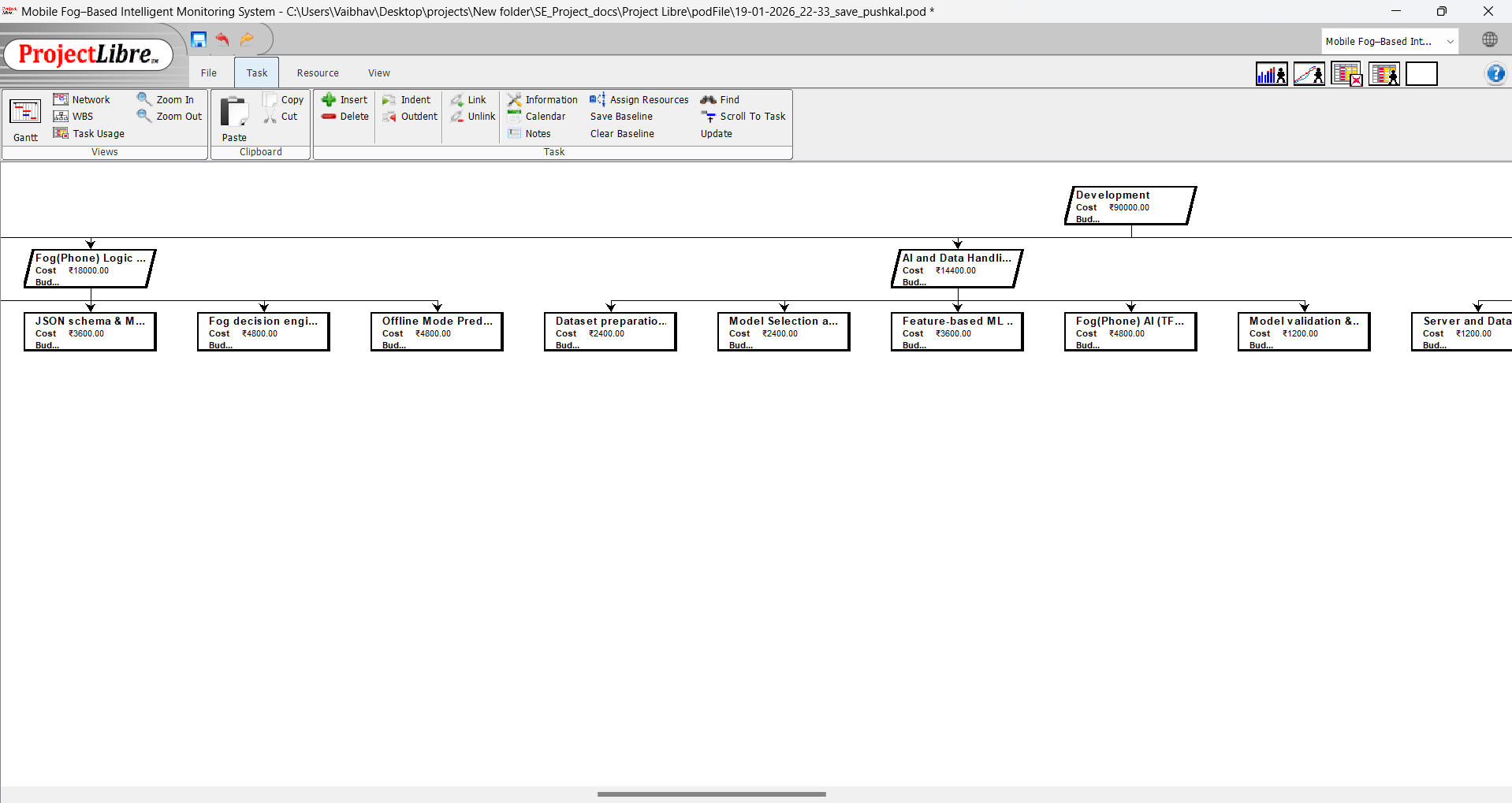
**6.4 WBS**

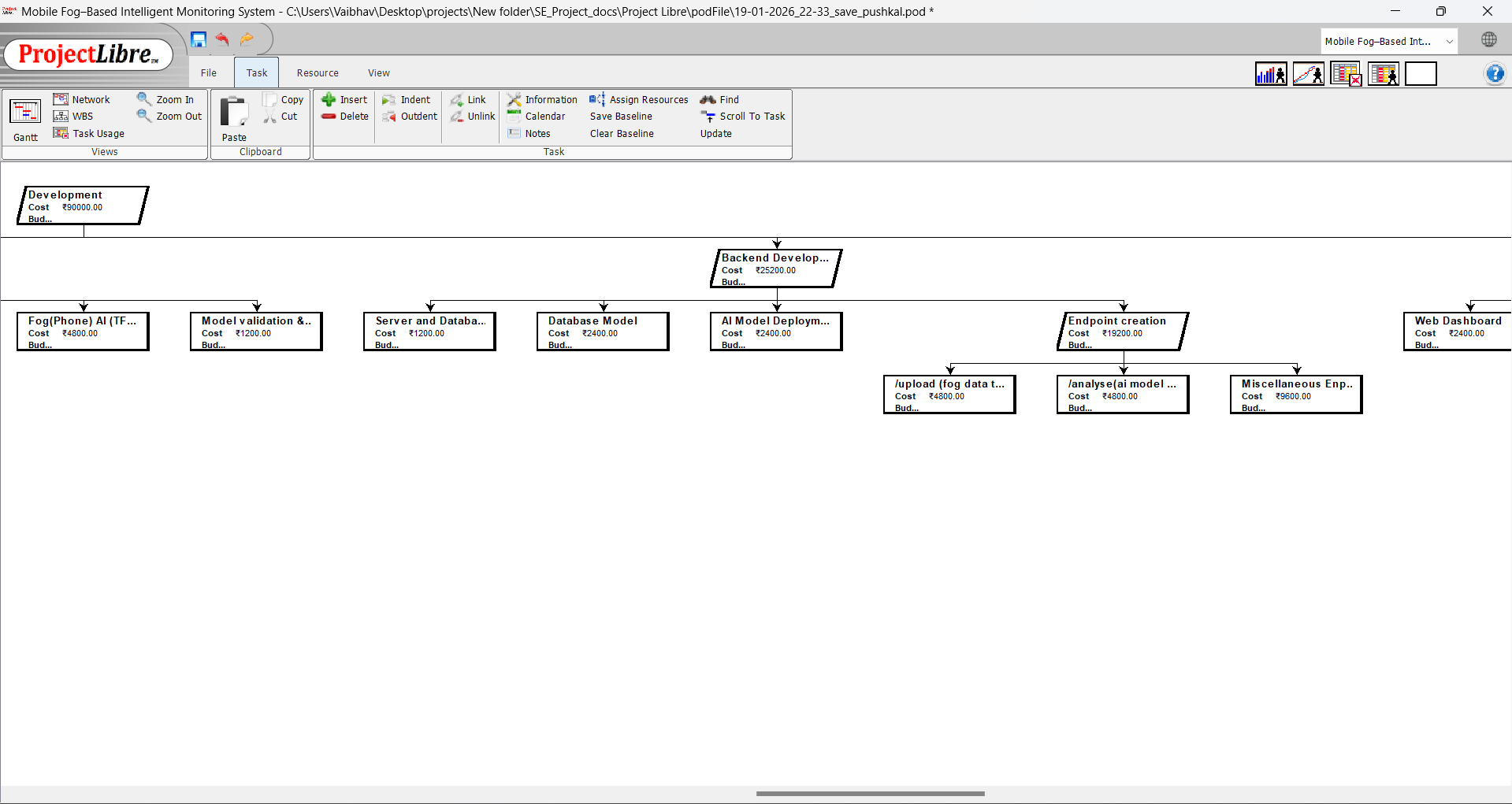


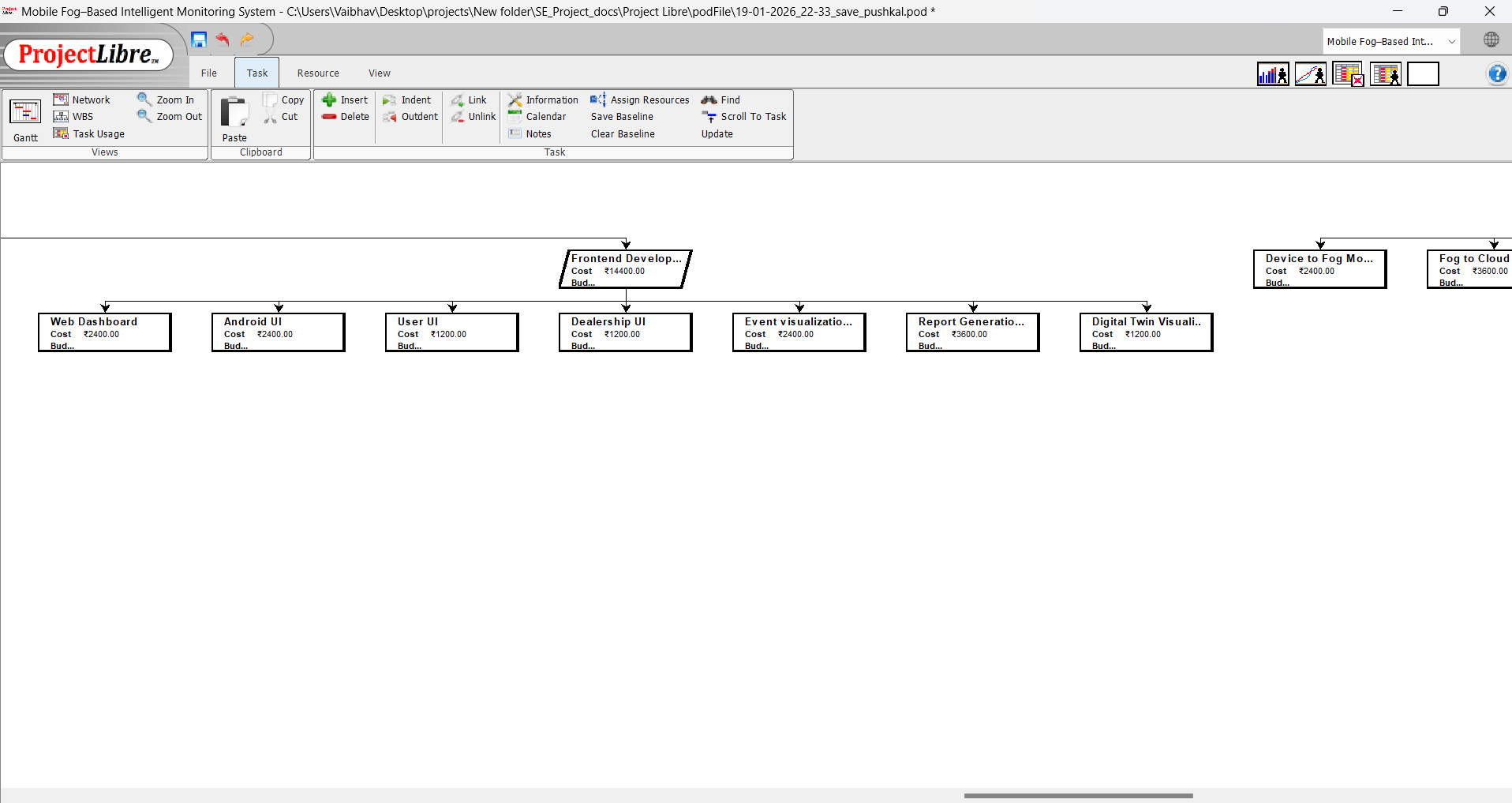


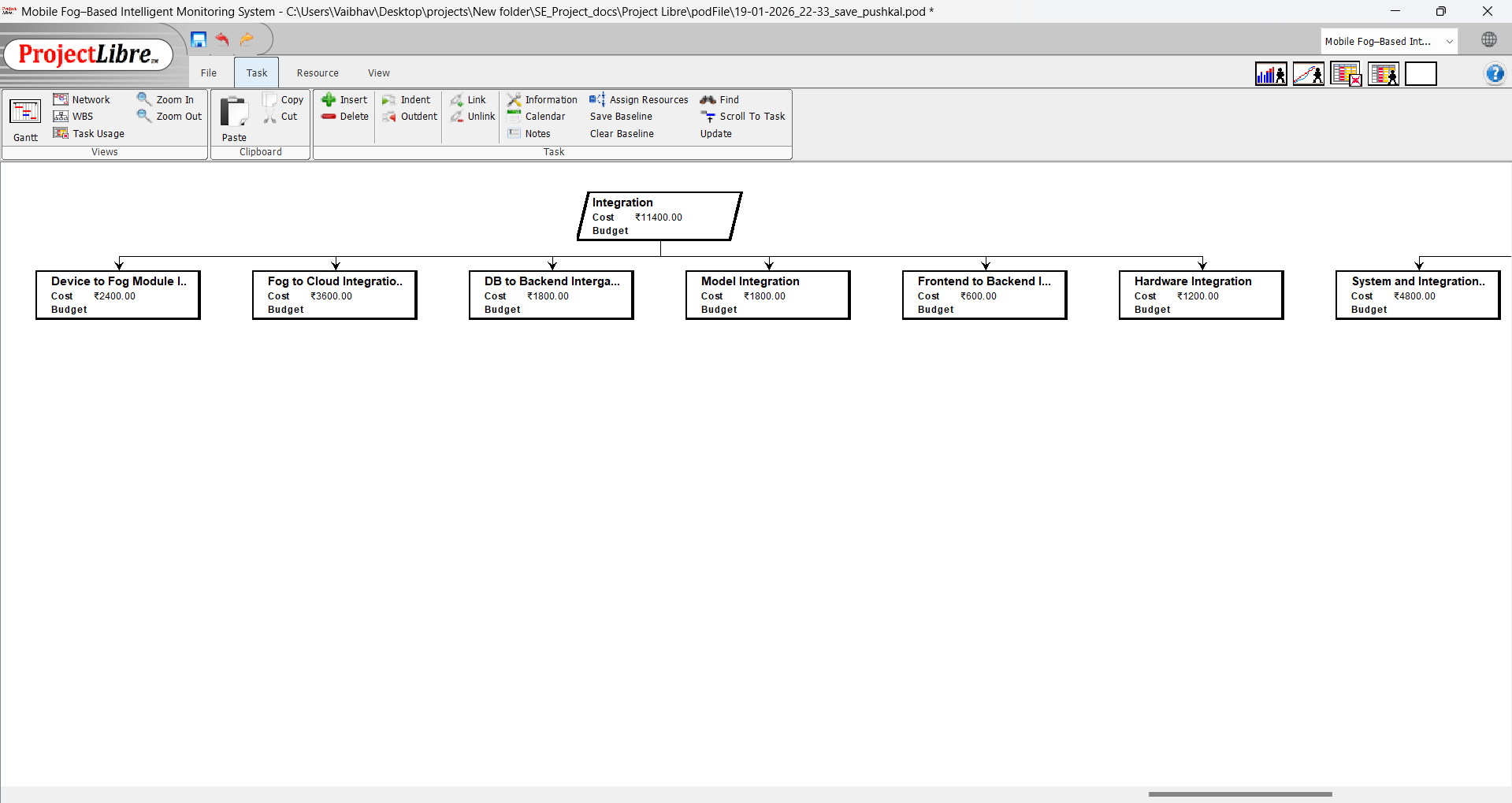


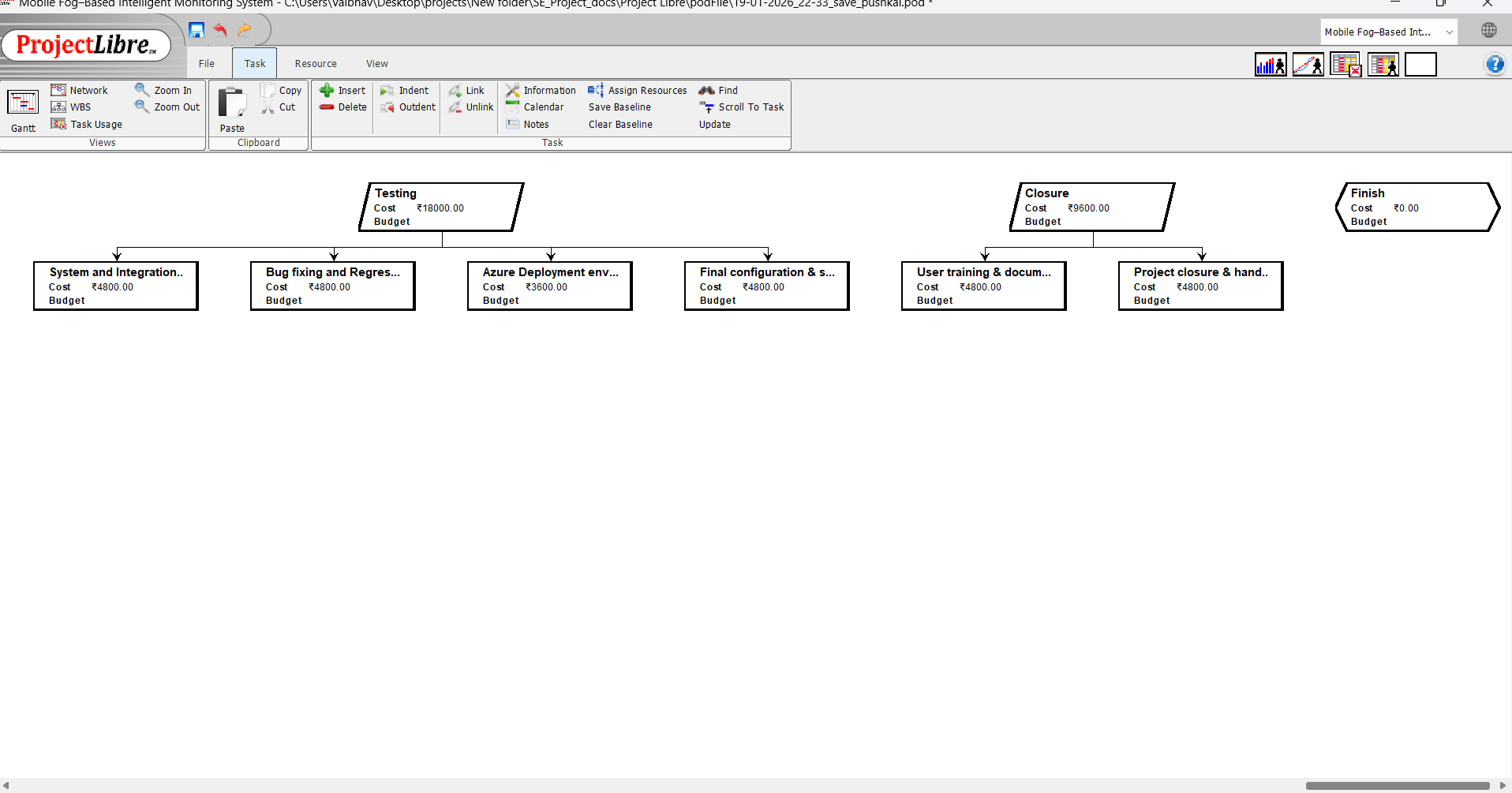








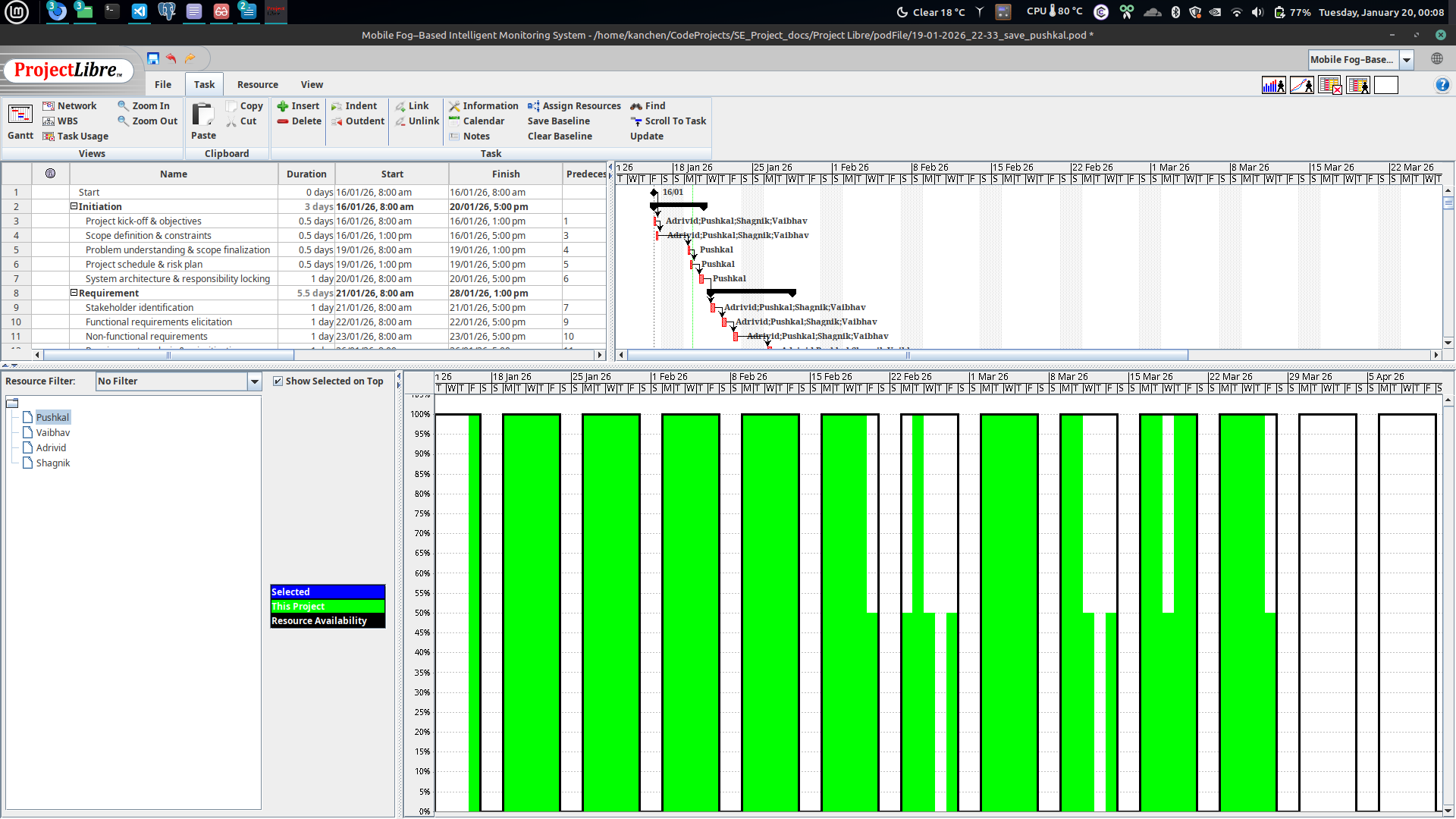


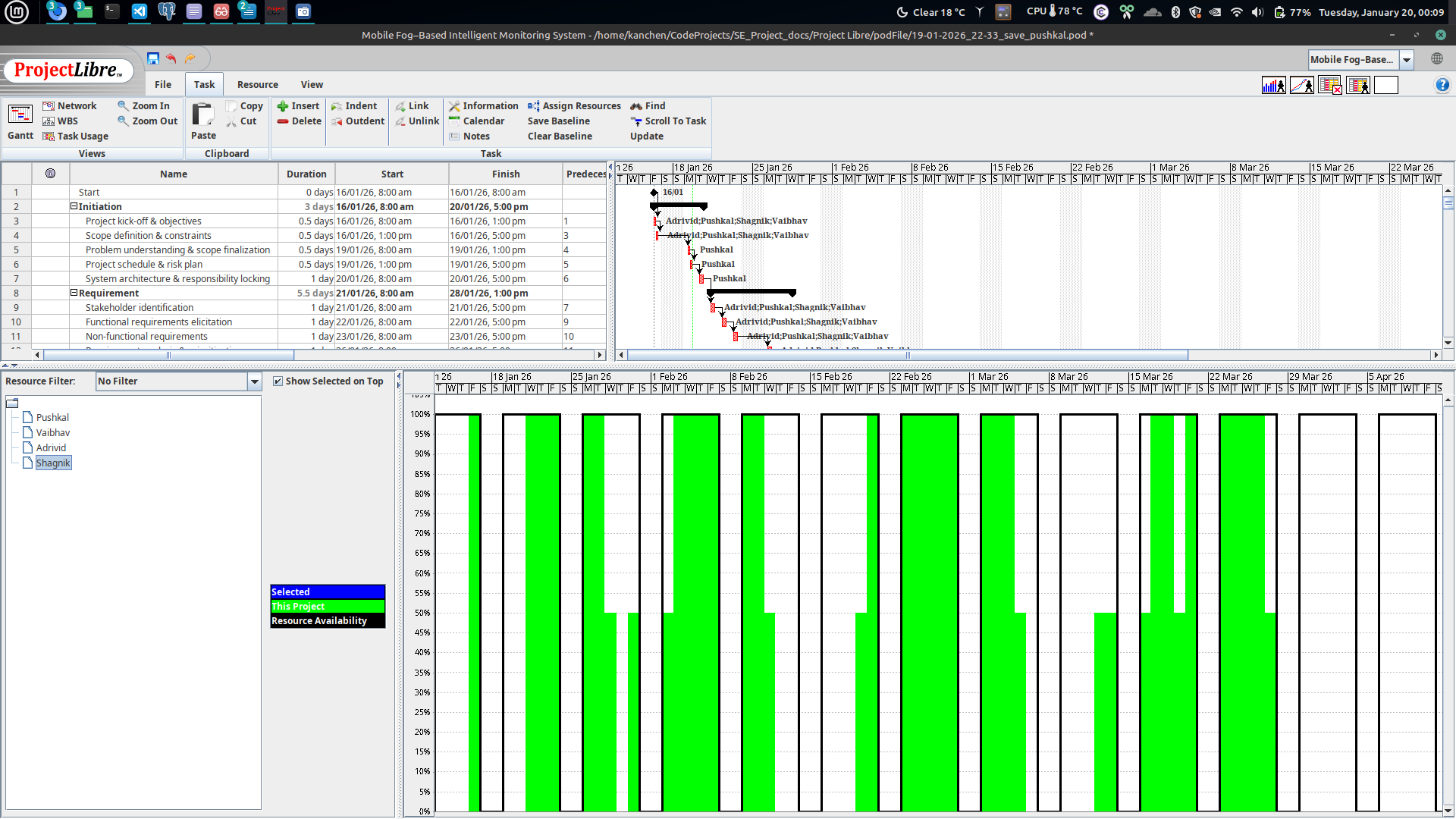


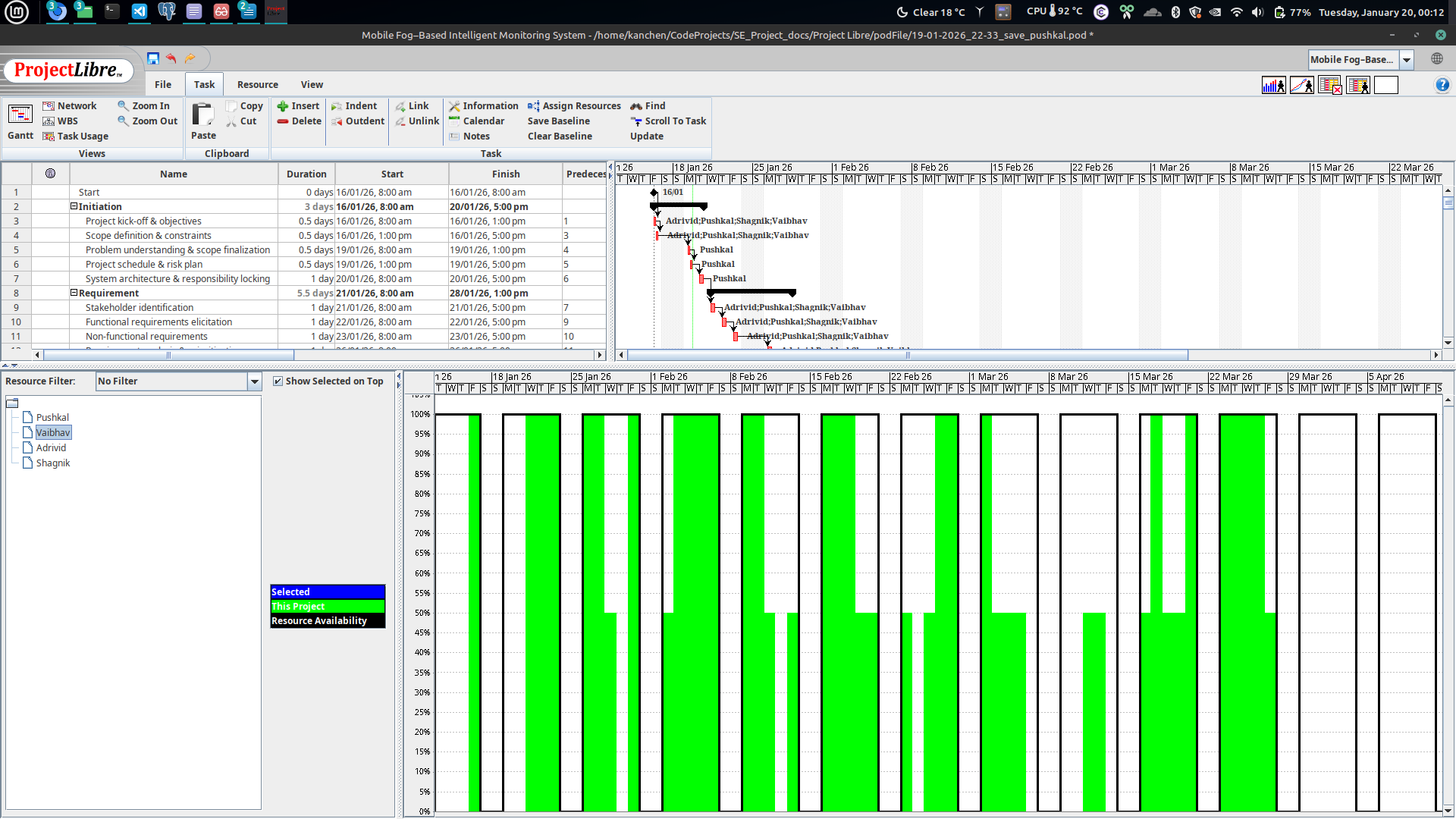
**6.5 Resources List**

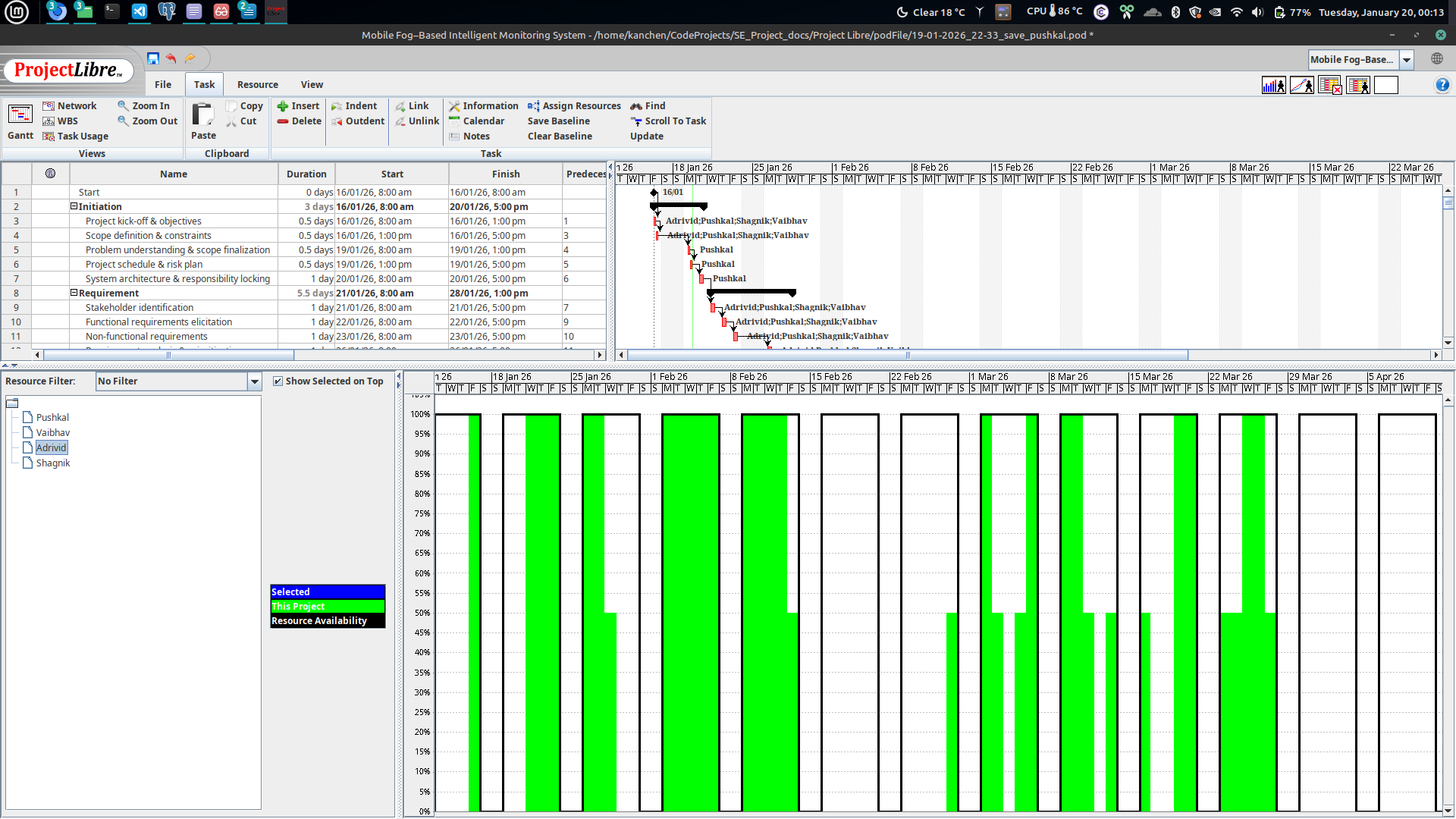


**6.6 Resource Allocation Histogram**

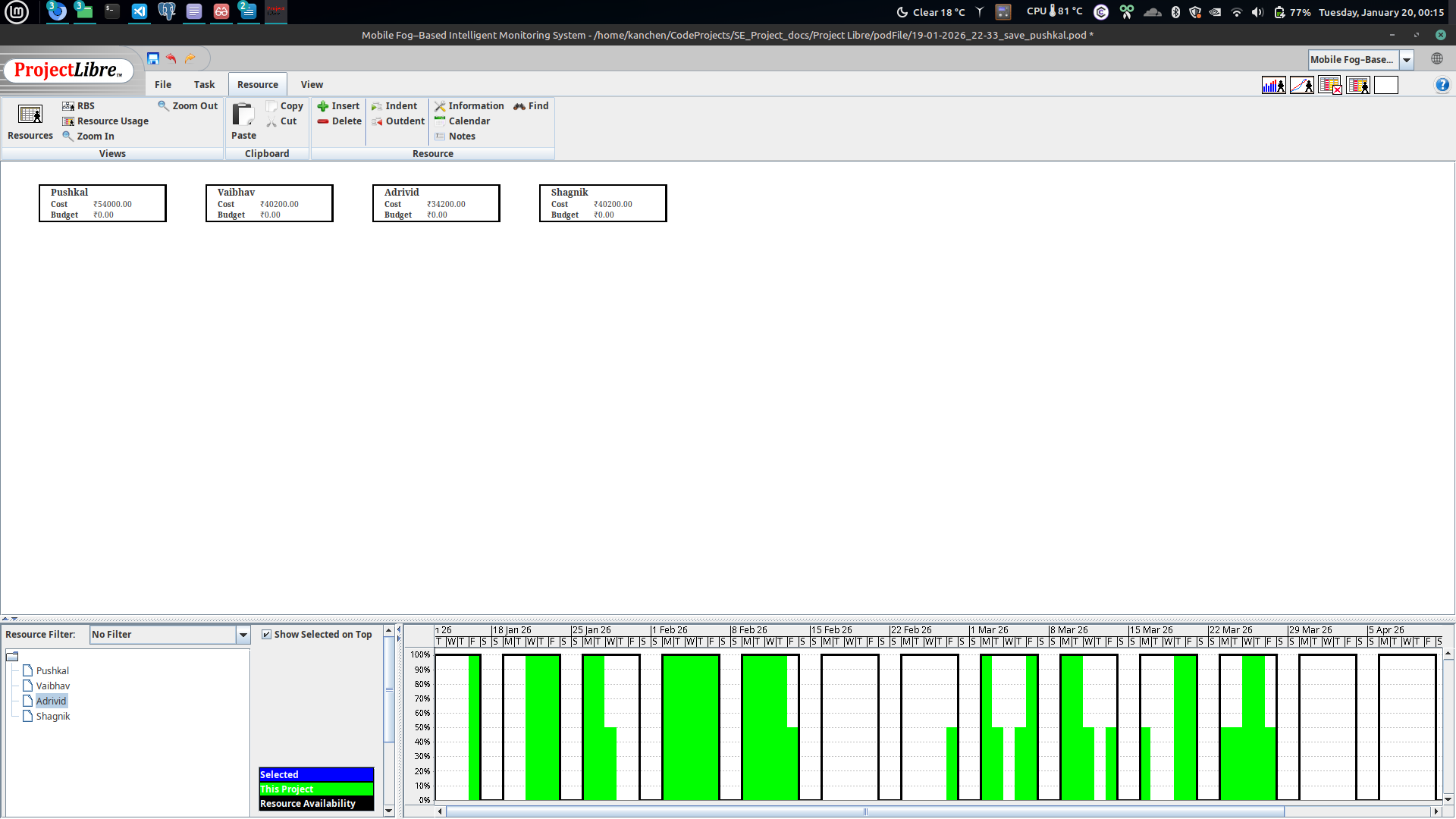
Pushkal Gupta

Shagnik Paul

Vaibhav Jain

Adrivid Mishra 

**6.7 R.B.S.**



**6.8 Project Information (Statistics)**

