Automation Architecture

Contents

[Purpose of the document 3](#_Toc139896484)

[Problem Statement 3](#_Toc139896485)

[Approach 3](#_Toc139896486)

[Architecture Block Diagram 4](#_Toc139896487)

[Linux PC 4](#_Toc139896488)

[Frontend: HMI GUI 4](#_Toc139896489)

[Interface Module 5](#_Toc139896490)

[Device Status and Reconnection: 5](#_Toc139896491)

[Device Updation: 5](#_Toc139896492)

[Report Centre: 5](#_Toc139896493)

[Priority Queue: 5](#_Toc139896494)

[Security and Encryption: 5](#_Toc139896495)

[Device Diagnostics and Error log: 5](#_Toc139896496)

[Event and Notification Module: 6](#_Toc139896497)

[Result Handler 6](#_Toc139896498)

[Test Management Module: 6](#_Toc139896499)

[Device Command and Response Processing queue: 6](#_Toc139896500)

[User and Account Management: 6](#_Toc139896501)

[Recipe/Rule Engine and Management: 6](#_Toc139896502)

[Service Monitor: 7](#_Toc139896503)

[Device Management and Configuration: 7](#_Toc139896504)

[Dashboard Module: 7](#_Toc139896505)

[Database 7](#_Toc139896506)

[MES 7](#_Toc139896507)

[Configuration files 7](#_Toc139896508)

[Windows PC 7](#_Toc139896509)

[Autostart module 7](#_Toc139896510)

[IronPython 7](#_Toc139896511)

[Priority Queue 7](#_Toc139896512)

[Interface Module 7](#_Toc139896513)

[Pytest 7](#_Toc139896514)

[Failure Handling and Recovery 8](#_Toc139896515)

[Device Data attributes 9](#_Toc139896516)

[Brainbox 9](#_Toc139896517)

[Camera 9](#_Toc139896518)

[Janome 9](#_Toc139896519)

[Test Case Recipe Model 9](#_Toc139896520)

[For Linux 10](#_Toc139896521)

[For Windows 11](#_Toc139896522)

[Programming Languages Finalization 13](#_Toc139896523)

[BootUp Sequence 14](#_Toc139896524)

[Linux 14](#_Toc139896525)

[Windows 16](#_Toc139896526)

[API/ASCII References 17](#_Toc139896527)

[Project Management tools 17](#_Toc139896528)

[Feature Comparison of Project Management tools 18](#_Toc139896529)

[Price Comparison of Project Management tools 18](#_Toc139896530)

[Conclusion on Project Management Tools 19](#_Toc139896531)

[Version Control Tools Identification 19](#_Toc139896532)

[Brief Explanation of version control Tools 19](#_Toc139896533)

[Feature Comparison of Version Control tools 20](#_Toc139896534)

[Price Comparison of Version Control tools 21](#_Toc139896535)

[Conclusion on Version Control Tools 21](#_Toc139896536)

# Purpose of the document

This document provides a comprehensive and detailed description of the architecture and design that is to be employed in the Waymo Automation solution including hardware components, software modules and their interaction and the HMI. It also captures the rationale behind the design decisions such as selection of Operating Systems, hardware platforms, communication platforms and software frameworks.

# Problem Statement

The problem at hand entails managing a diverse range of hardware devices with varying driver compatibility. Some of these devices are exclusively supported by Linux drivers, while others rely solely on Windows drivers. The primary goal is to establish seamless communication with all these hardware devices and exercise control over them through a unified Human-Machine Interface (HMI) application software.

To facilitate the ease of test case creation, the HMI application software should provide drag and drop controls. These intuitive controls enable users to visually design and assemble test cases by arranging the devices and defining the desired inputs and outputs between them. By simplifying the test case construction process, users can efficiently configure complex interactions among the connected hardware devices.

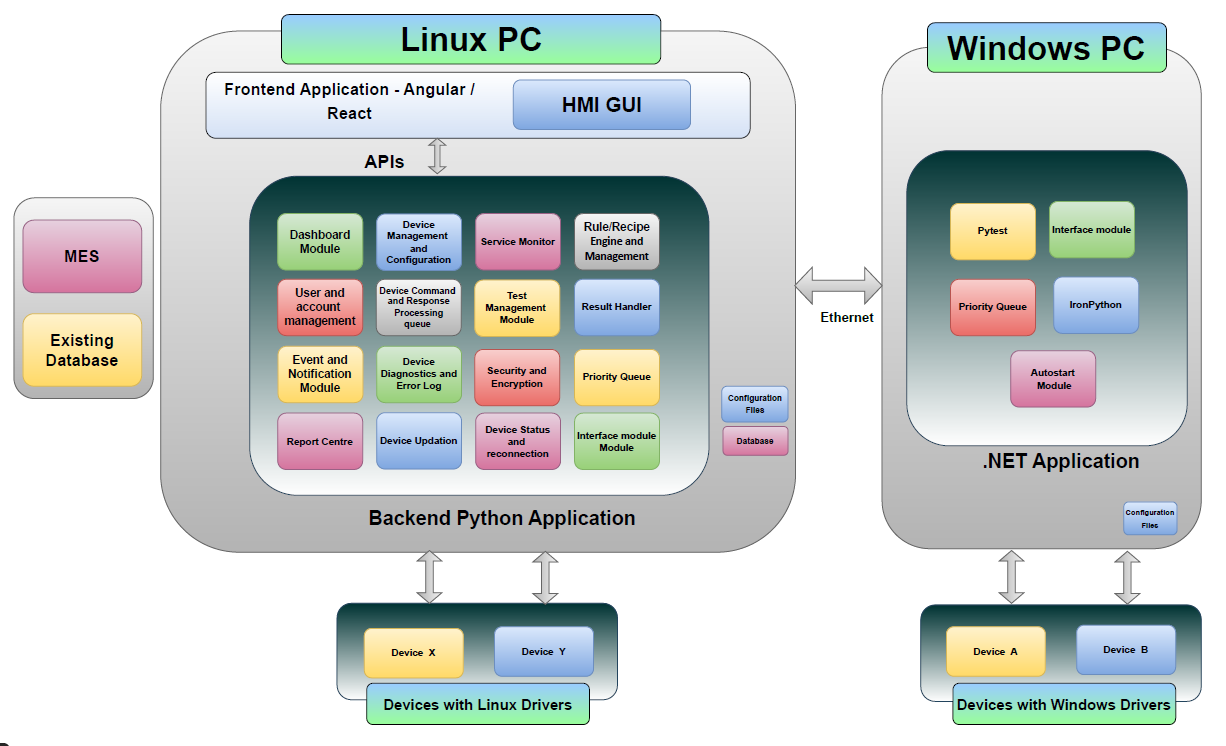
Furthermore, it is crucial to incorporate different user privileges within the HMI application software. This feature allows for multiple user roles with varying levels of access and permissions. Users with appropriate privileges can create, modify, and manage the test cases, ensuring flexibility and security within the system. Additionally, users should have the capability to execute the test cases, generating comprehensive reports on the outcomes and performance of the interconnected hardware devices.

# Approach

Linux PC

NUC

# Architecture Block Diagram



The proposed approach utilizes a Windows NUC for Windows devices and a Linux system for Linux devices, with an HMI software running on the Linux system to control and communicate with both sets of devices. By implementing appropriate drivers, IPC mechanisms, drag and drop controls, user privileges, and reporting capabilities, a unified control system can be achieved, facilitating efficient management, testing, and control of the interconnected hardware devices.

# Linux PC

## Frontend: HMI GUI

The Human-Machine Interface (HMI) is the application that enables wiring together of hardware devices. The HMI allows:

* Production-line monitoring – displays real-time information about the information and status of each hardware device connected to the system.
* Fault detection and diagnostics – detects faults and abnormalities in the production line/ test setup.
* Production Planning and scheduling – allows users to manage and optimize the production line/ testing setup, schedule tests, monitor progress and adjust schedules based on real-time data.
* Error tracking and reporting – track and log errors or faults encountered during execution of test recipes. It can generate reports or notifications that summarize the types, frequency, and causes of errors, helping to identify areas for improvement and implement preventive measures.
* Operator guidance and training – provides visual instructions, standard operating procedures, or work instructions to guide operators through the assembly process. It can also support training modules and interactive tutorials for new operators or when introducing new devices.
* Performance analytics - offers performance analytics and reporting capabilities, providing key performance indicators (KPIs) and metrics related to production efficiency, downtime, cycle times and other relevant parameters.
* Energy monitoring and optimization - tracks and analyses energy consumption of the devices and helps in overall energy optimization.

## Interface Module

The interface module enables communication and interaction between the Linux PC and the NUC.  
It serves as an intermediary, allowing different modules to exchange information, commands,   
or data while abstracting the underlying complexity of the systems involved.  
It uses a Unified Messaging Protocol for achieving the same.  
The devices connected to the Windows PC communicate with the HMI running on the Linux PC using the Interface module.   
Various modules of the Linux PC application also uses this module to communicate between each other.

## Device Status and Reconnection:

This module is responsible for maintaining the status of all the system modules by periodically pinging each hardware device connected to the system. It also performs re-initialization of devices in case of abnormal state. It maintains the continuity of device-live status. It will initiate reconnection if connectivity is lost.

## Device Updation:

This module handles the new device configuration when the device model is updated. It initiates the command sequence required to perform the specific configuration updation. It allows the user to add a new device, add a new instance of the device or edit the existing device configuration.

It provides an interface to update the device firmware as provided by the vendor.

## Report Centre:

This module is responsible for generating reports that provide valuable insights and facilitates decision-making processes. It collects real-time data from the hardware devices stores them in the database. It provides interfaces for aggregating and filtering the data based on various parameters like time, number of events, anomalies or correlations (e.g. Last 5 test results etc.). This module employs visualization techniques like graphs, charts and tables. It also provides the facility to export the reports in various file formats like pdf, csv etc.

## Priority Queue:

This module creates a queue with each packet assigned a priority value. Packets are inserted with associated priority and packets with highest priority is dequeued first. If multiple packets have the same priority, they are dequeued based on the order on which they were inserted. This module ensures that the most critical packets are processed at the earliest.

## Security and Encryption:

This module ensures a robust defence against security threats like unauthorized access, data breaches, tampering or interception. It employs encryption algorithms to encode the data that is being exchanged between the Linux PC and the Windows PC. This encryption prevents unauthorized access to the critical data that is being transferred between devices. This module incorporates protocols to establish secure channels, authenticate endpoints, and safeguard against eavesdropping or data interception. It also handles secure storage mechanisms. Encryption algorithms like Advanced Encryption Standard (AES), Elliptic Curve Cryptography (ECC) maybe used to secure the communication.

## Device Diagnostics and Error log:

This module focuses on monitoring and diagnosing the health and performance of the hardware devices. It collects the data related to device health, performance metrics and operational parameters. It uses built-in-self-tests, fault detection algorithms and sensor data analysis to identify and pin point potential issues or failures. When a fault or error is detected, this module alerts and notifies users.

This module also maintains detailed logs and audit trails of diagnostic activities and events. These logs can be used for troubleshooting, historical analysis, compliance purposes, or generating performance reports.

## Event and Notification Module:

Event and Notification scheduling based on the triggers.

The Events and Notifications module is responsible in managing events like device events, user actions etc. Notifications are generated by the system to inform users about important updates, status changes, job completion, error conditions etc.

This module acts as a central hub for handling all events, enabling all the components or services to react and respond to events in a coordinated manner.

Administrator can define event handling rules, notification preferences, and recipient lists.

This module integrates with other components like device management, dashboard module and hardware devices to trigger specific actions or initiate automated processes based on the events and notifications received.

## Result Handler

This module runs on Linux PC and handles test results from the secondary pc and updates the system accordingly. It processes the outcome of the operation based on the specific test case. It may extract relevant data from the result, transform or format the data, and update the application's state or user interface accordingly. The result handler may also involve flow control logic to determine the subsequent actions or operations based on the result. It can trigger additional operations, switch to different states or screens, or perform any necessary follow-up tasks based on the outcome. Result handlers may handle cases where an operation is cancelled or times out. They can perform appropriate cleanup actions, update the application's state, or notify the user about the cancellation or timeout

## Test Management Module:

This module provides an interface to manage test cases and track testing progress. It allows users to define test cases and test suites. It helps in organizing and categorizing tests. Users can specify test inputs, expected outputs, preconditions and post-conditions for each test case, test priorities, dependencies, triggers and custom alerts. Users can schedule tests, record actual outcomes, log defects etc.

## Device Command and Response Processing queue:

Handles the device command reception to perform a specific action, sets the priority for the request, and sends a response.

This module handles the processing of commands sent to the hardware devices and generates appropriate responses. It also houses the communication method and the protocol stack specific to each of the devices. The module receives the commands, interprets it, identifies the action/information requested and transmits the response to the device using the device-specific communication protocol.

## User and Account Management:

Handles creating, managing and authenticating user accounts for Admin, Engineer, Operator, Technician etc. and mapping permissions. The Admin and Engineer have permission to add ingredients and recipes. Additionally, the Admin will also have permission to add new users and define their roles. The Technician and Operator will only have the permission to add and execute recipes.

## Recipe/Rule Engine and Management:

Ingredients are nothing but test scripts that are used to test a specific functionality of the device or the process of the device’s operation. A series of ingredients form a recipe.

A “Rule” is included between ingredients to define the execution of ingredients in a specific flow. In simple terms, a recipe is a series of tests that are executed one after the other based on the rules. The “Rule” also defines the input to the test cases and the conditions under which the tests are executed. Execution of the recipe can include multiple devices.

## Service Monitor:

The module continuously monitors the connected devices to gather real-time information about their operational status. It collects data such as device health, performance metrics, sensor readings, and connectivity status. Users can view this information through a dashboard interface, providing them with insights into the current state of the devices. This module also reports if any process is down.

## Device Management and Configuration:

This module provides an interface to monitor, control and customize the devices initializations and configurations. It facilitates adding and managing device instances. This module allows customization of device and interface specific features such as network settings, COM port settings etc.

It provides customization of device discovery and initialization procedures of hardware devices. The devices are assigned unique identifiers and gathers device-specific information for future management and configuration tasks.

## Dashboard Module:

The dashboard module gathers and aggregates data from various sources, such as databases, APIs, and hardware devices. It allows users to create various tabs, widgets, and controllers, initiate processes, or modify settings directly from the dashboard interface. The dashboard will contain a snapshot of system events that are of concern. It provides an interface for the user to drag and drop the widgets into the dashboard page allowing the user to customize their dashboards according to their requirement.

## Database

Maintains all the data required by the Backend application module (Ex: Dashboard widgets, User account details etc.)

## MES

Client Databases

## Configuration files

Maintains all the test configurations of devices required for executing recipes

# Windows PC

## Autostart module

Handles autostart and autoboot process of the Windows PC

## IronPython

Enables the interface between the .net application and Pytest

## Priority Queue

Handles the request process in the order of priority

## Interface Module

Handles interface and communication between Linux and Windows PC

## Pytest

Executes the Recipes based on Backend application Module Request

# Failure Handling and Recovery

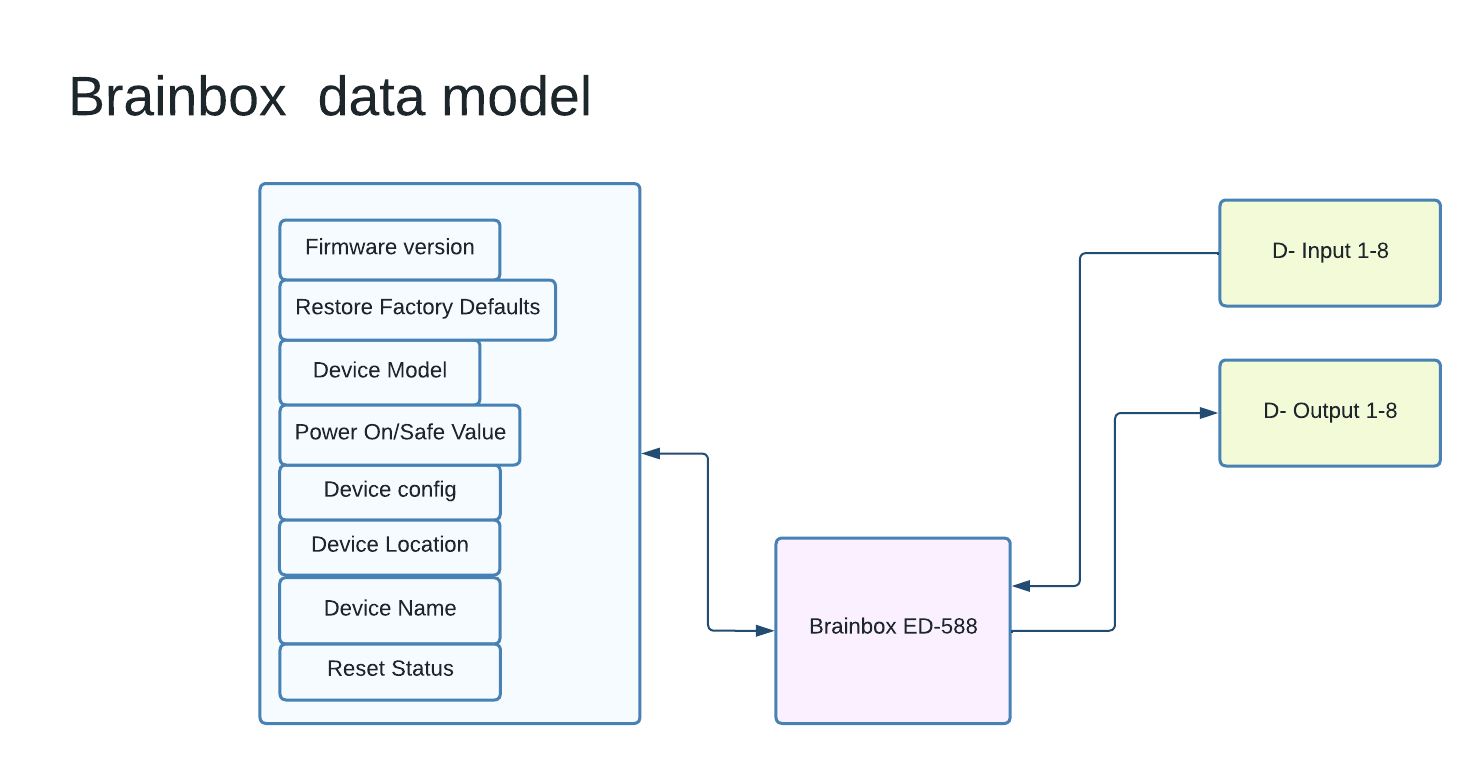
The Windows system failures can be handled by issuing restart commands from the Linux PC. This can be done in the following ways:

* Remote Access
  + Remote access the Windows NUC using SSH (Secure Shell) using the NUC’s IP address.
  + Run the command shutdown /r /t 0 to restart the Windows based NUC.
  + The autostart module in the Windows system will handle the starting up of all application modules and services.
* Wake-on-LAN (WoL) utility
  + A network management protocol Wake-on-LAN can be used to remotely power on or restart a computer over a network by sending a magic packet to its Network Interface Card (NIC).
  + The Windows NUC should have the Wake-on-LAN capability enabled in the BIOS/UEFI settings.
  + The Linux PC should have the wakeonlan utility installed
  + The Wake-on-LAN utility needs the MAC address of the target Windows NUC system as a parameter and it sends a magic packet to the NUC initiating a restart.
  + The utility will broadcast the magic packet, and only the device with the specified MAC address will act upon it, while the rest of the devices will simply disregard it. Hence, the Wake-on-LAN utility can run on the same Ethernet channel as other devices on the LAN.
  + The autostart module in the Windows system will handle the starting up of all application modules and services.

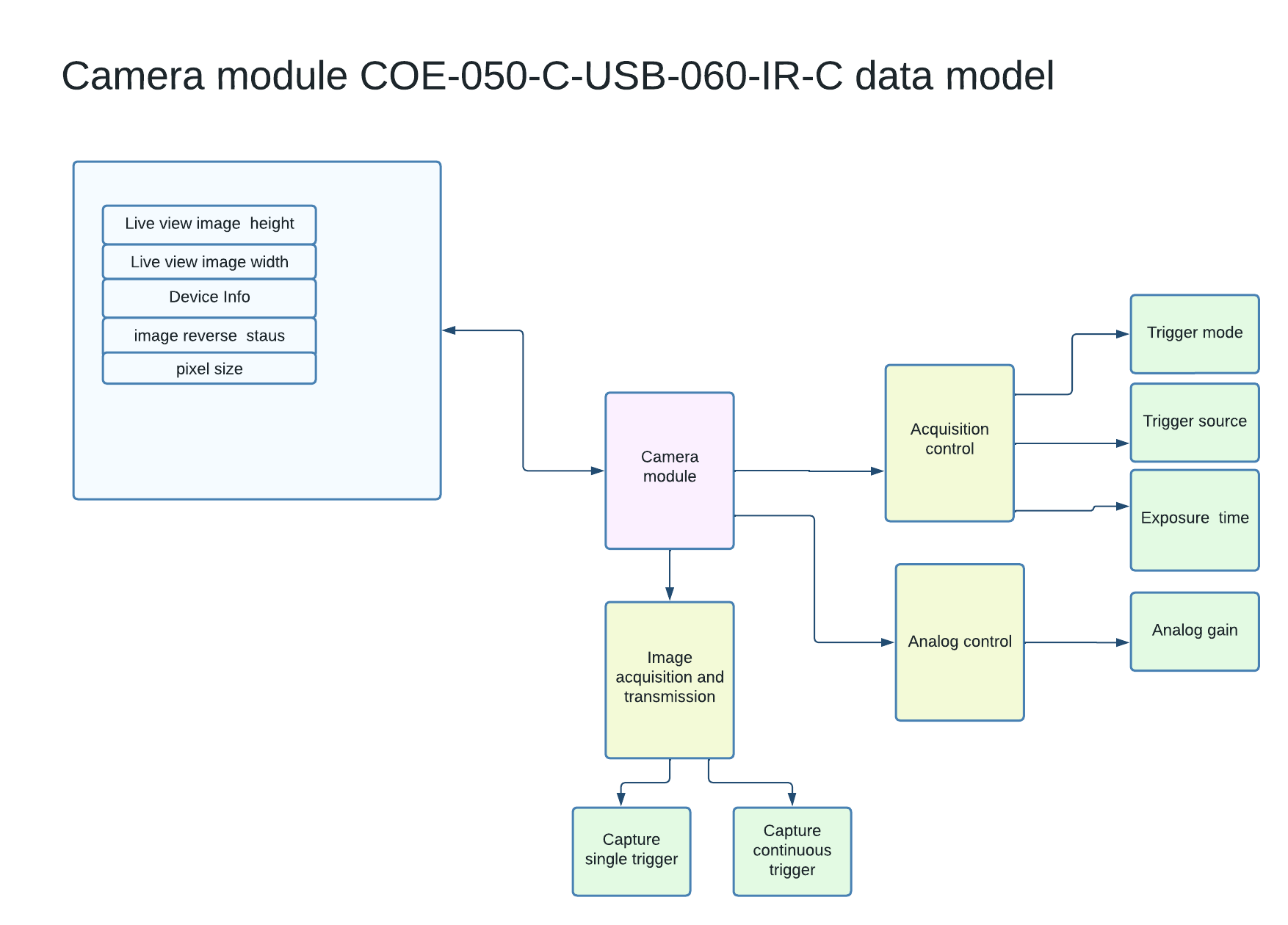
# Device Data attributes

The salient device attributes for the devices are listed below.

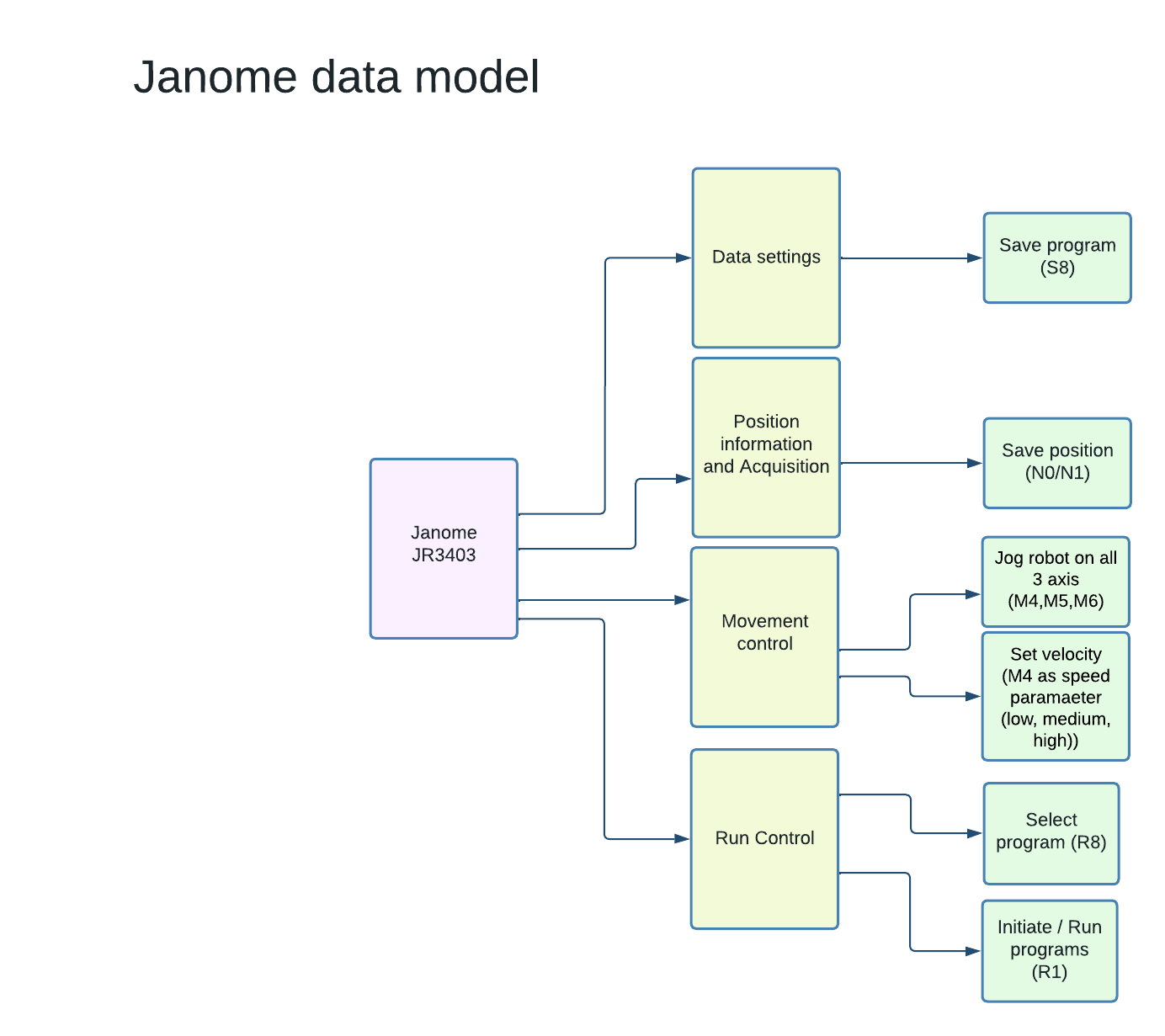
## Brainbox



## Camera

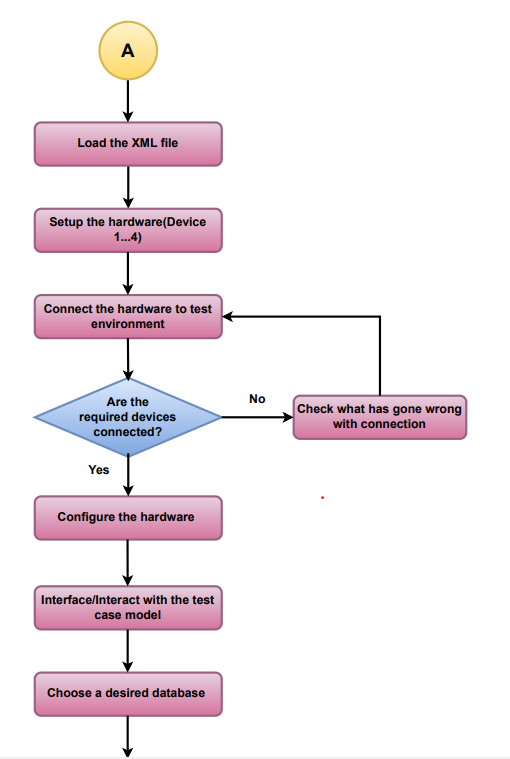


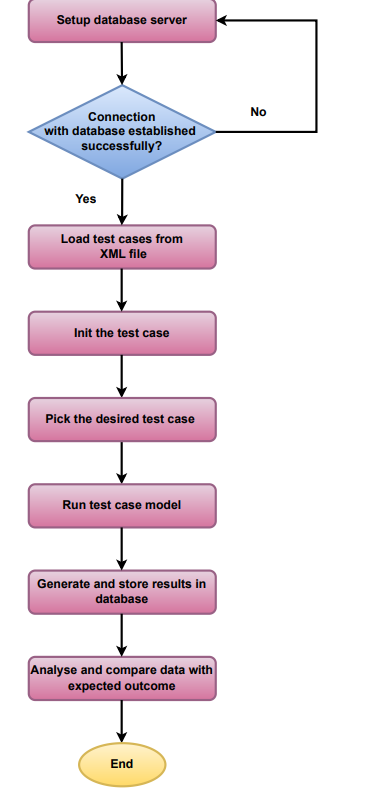
## Janome



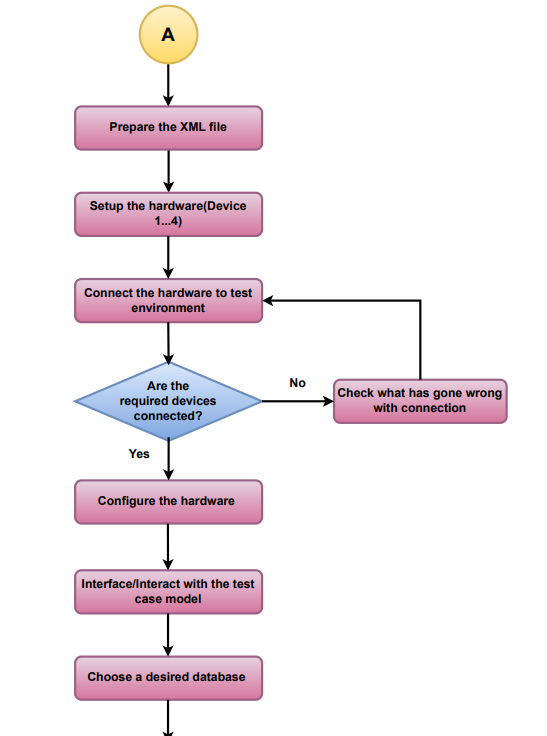
# Test Case Recipe Model

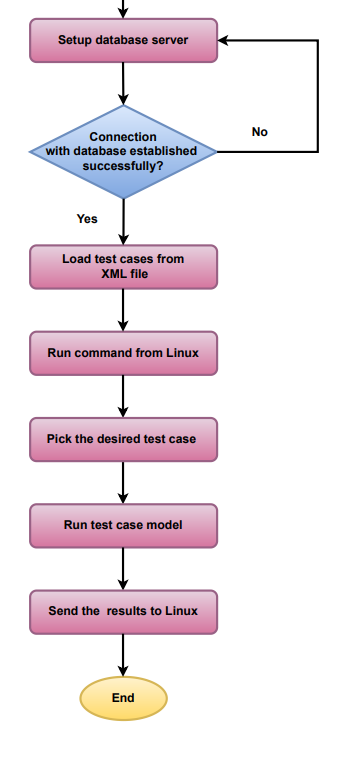
## For Linux





## For Windows





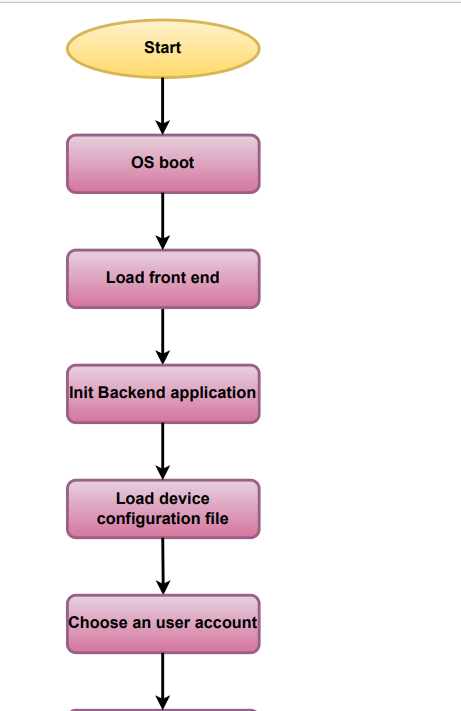
# Programming Languages Finalization

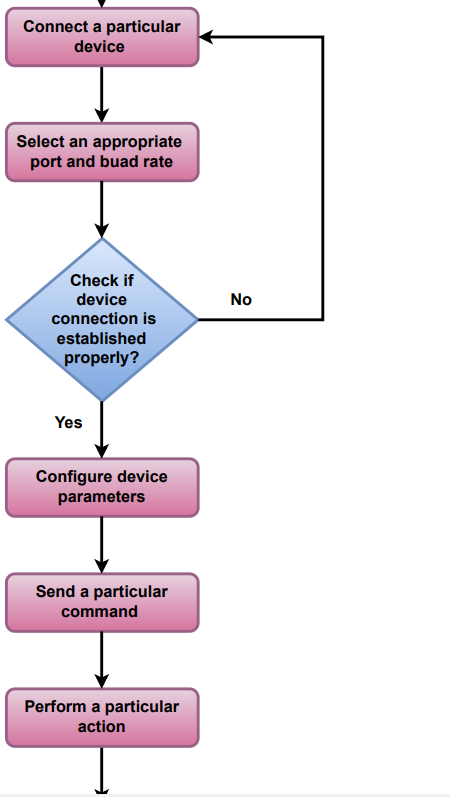
There are various programming languages used in the process of achieving automation. For Linux application, Angular/React JS is used on the front-end side. Python is preferred for backed modules.

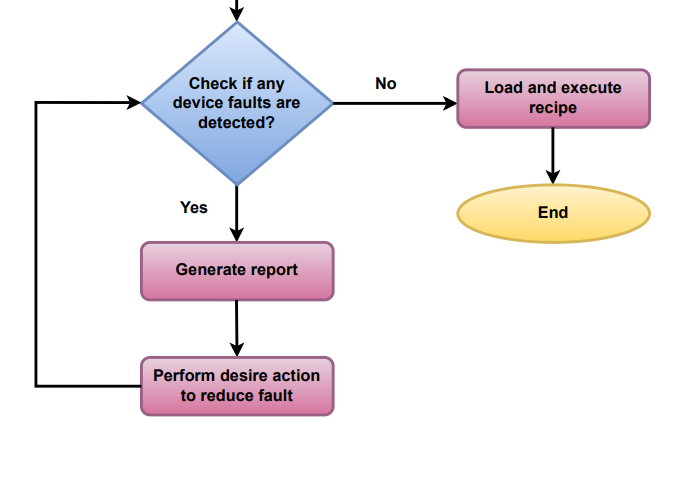
On Windows side, C# is used to invoke required tests execution.

# BootUp Sequence

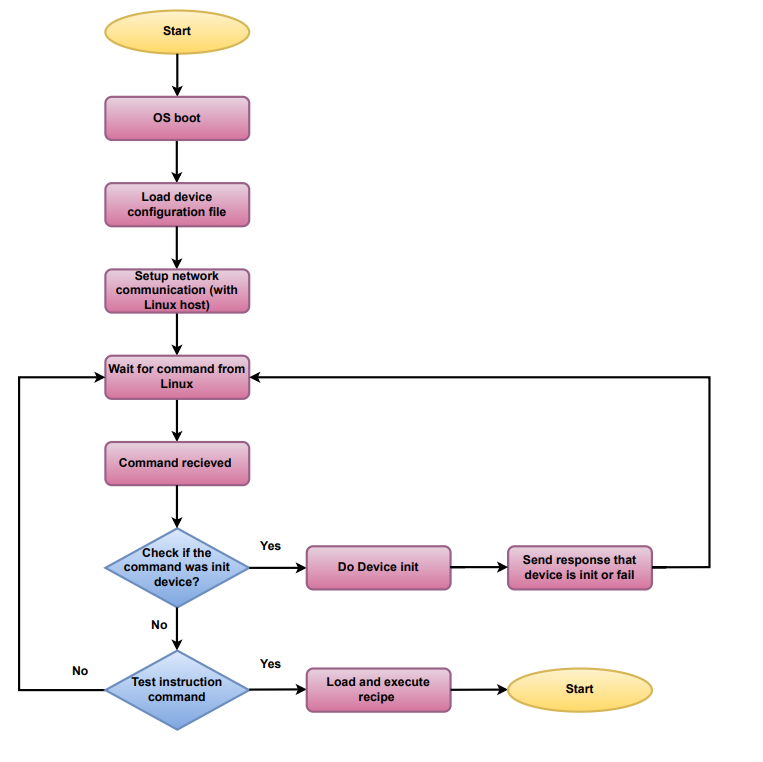
## Linux







## Windows



# API/ASCII References

This section explains different device driver APIs or ASCII command references that are invoked for test automation. In future, there will be different sub sections for each device and their respective references.

# Project Management tools

1. Asana
2. Trello
3. Jira
4. Monday.com
5. ClickUp
   1. **Brief Explanation of Project Management Tools**
6. **Asana**

Asana is one of the more popular project management software platforms available. Built for businesses of many sizes, this multifaceted tool combines file storage, project roadmaps, dashboards, and more in one attractive interface.

1. **Trello**

Trello is one of the earliest Kanban-based tools. It still allows for simple project visualization and effective drag-and-drop task management. It’s familiar to many people, allowing for easy sharing and collaboration, even among external users.

1. **Jira**

Jira is part of the Atlassian Group now, but it originated as a software development solution many years ago. Since then, it has developed into an umbrella platform that includes Jira Software, Jira Core, and Jira Work Management, all of which combine to offer work management assistance for teams of every size.

1. **Monday.com**

Monday.com is a popular project management software option that includes various templates and tools to optimize operations and help your teams work toward higher productivity levels. The software displays a lot of information for teams, but key data can get overlooked in long chains of events.

1. **ClickUp**

ClickUp is a well-known project management software solution that works well for most sized teams, including those operating in remote environments. It has all the standard management features that support the planning, organization, and management of various tasks.

## Feature Comparison of Project Management tools

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Sr. No | Project manage ment tools | Bug tracking feature | Integration with Git and Bitbucket | Collab- oration tools | Support for issue tracking | Kanban Board with custom  grouping | Road  -map plan | Project decomp  -osition |
| 1 | Asana | No | Yes | Yes | Yes | Yes | Yes | Yes |
| 2 | Trello | No | Yes | Yes | No | Yes | No | Yes |
| 3 | Jira | Yes | Yes | Yes | Yes | Yes | Yes | Yes |
| 4 | Monday. com | No | Yes | Yes | Yes | Yes | Yes | Yes |
| 5 | ClickUp | Yes | Yes | Yes | Yes | Yes | Yes | Yes |

## Price Comparison of Project Management tools

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Sr. No | Software tools | Free | Basic | Standard | Premium | Business |
| 1 | Asana | $0 | $0 |  | $13.49  /user/month | $30.49  user/month |
| 2 | Trello | $0 |  | $6  /user/month | $12.50  /user/month | $17.50  user/month |
| 3 | Jira | $0 |  | $7.75 | $15.25  user/month |  |
| 4 | Monday.com | $0 | $10/user/month | $12user  /month (Standard) | $20 user  /month (Pro) |  |
| 5 | ClickUp | $0 |  | $9 user/month |  | $19  /user/month |

## Conclusion on Project Management Tools

Based on the analysis on the features and pricing of the Project Management Tools, for our project we can use Jira.

# Version Control Tools Identification

1. Git
2. CVS
3. SVN
4. Mercurial
5. Monotor

## Brief Explanation of version control Tools

1. **Git**

Git is the most popular option and has become synonymous with "source code management." Git is an open source distributed system that is used for software projects of any size, making it a popular option for startups, enterprise, and everything in between.

1. **CVS**

CVS (Concurrent version system) is a version control system intended to allow multiple developers to work concurrently on a project. Each developer has a copy of the source tree(s) that they need to do their work. Their copies are checked out of a central repository that keeps track of versions. Various commands are available to keep developers synchronized with the repository and each other. CVS can also operate in a client-server mode that allows remote developers (i.e. not sharing the file system) to work with the repository.

1. **SVN**

SVN (Sub-Version) is a widely adopted centralized VCS. This system keeps all of a project's files on a single codeline making it impossible to branch, so it's easy to scale for large projects. It's simple to learn and features folder security measures, so access to subfolders can be restricted.

1. **Mercurial**

Mercurial is a distributed VCS that offers simple branching and merging capabilities. The system enables rapid scaling and collaborative development, with an intuitive interface. The flexible command line interface enables users to begin using the system immediately.

1. **Monotone**

Monotone is a free, distributed version control system. It provides fully disconnected operation, manages complete tree versions, keeps its state in a local transactional database, supports overlapping branches and extensible metadata, uses an efficient peer-to-peer network protocol, performs history-sensitive merging, and delegates trust functions to client-side RSA certificates.

## Feature Comparison of Version Control tools

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr. No | Software tools | Non- linear develop ment | Distributed repo model | Client- server Repo model | Efficient in handling small to large sized projects | Compatible with existing systems and protocols like HTTP,FTP,  ssh | Support for pluggable merge strategies |
| 1 | Git | Yes | Yes | No | Yes | Yes | Yes |
| 2 | CVS | No | No | Yes | Less | No | No |
| 3 | SVN | No | No | Yes | Yes | Yes | No |
| 4 | Mercurial | Yes | Yes | No | Yes | Yes | Yes |
| 5 | Monotone | Yes | Yes | No | Yes | Yes | No |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Sr. No | Software tools | Popularity and Community Support | Ease to Maintain and Robust | Com- mand line utility | Free-form versioned metadata | Parallel develo- pment | Copy, delete, move and rename operation |
| 1 | Git | Best | Yes | Yes | No | Yes | Yes |
| 2 | CVS | Good | No | Yes | No | Yes | Yes |
| 3 | SVN | Good | Yes | Yes | No | Yes | Yes |
| 4 | Mercurial | Good | Yes | Yes | Yes | Yes | Yes |
| 5 | Monotone | Good | No | Yes | Yes | Yes | Yes |

## Price Comparison of Version Control tools

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sr. No | Software tools | Open source | Cost | Team Plan |
| 1 | Git | Yes | Free | GitHub: $4 /user/month (Team)  $21 /user/month (Enterprise) GitLab: $19 /user/month (Premium editions)  $99 /user/month (Ultimate editions) |
| 2 | CVS | Yes | Free |  |
| 3 | SVN | Yes | Free |  |
| 4 | Mercurial | Yes | Free |  |
| 5 | Monotone | Yes | Free |  |

## Conclusion on Version Control Tools

Based on the analysis on the features and pricing of the Version Control Tools, for our project we can use Git.