**1.Introduction**

**1.1 Purpose of the requirements document:** This Software Requirements Specification (SRS) identifies the requirements for the ROS project(**Robotic Arm Simulation using ROS**). The purpose of the present document is to explain how we organize our project, the different steps of the conception and the project achievement.

**1.2 Scope of the project:**

I)The platform and software used for this robot are extensible and open-source.

Ii) It is a low cost robot.

The benefits of this framework are:

We can simulate any kind of robot using the Gazebo simulation tool

No need to invest on any hardware if we are making any project

ROS provides algorithm for many sensors for robotics

ROS provides many libraries like:

* perception
* object identification
* [segmentation](https://en.wikipedia.org/wiki/Segmentation_(image_processing)) and recognition
* [Face recognition](https://en.wikipedia.org/wiki/Facial_recognition_system)
* [gesture recognition](https://en.wikipedia.org/wiki/Gesture_recognition)
* [motion tracking](https://en.wikipedia.org/wiki/Video_tracking)
* [egomotion](https://en.wikipedia.org/wiki/Egomotion)
* [motion understanding](https://en.wikipedia.org/w/index.php?title=Motion_understanding&action=edit&redlink=1)
* [structure from motion](https://en.wikipedia.org/wiki/Structure_from_motion) (SFM)
* [stereo vision](https://en.wikipedia.org/wiki/Stereopsis): depth perception via two cameras
* motion
* [mobile robotics](https://en.wikipedia.org/wiki/Mobile_robotics)
* control
* planning
* Grasping

**1.3 Overview:**

**2. Description**

**3. Functional requirement**

Here, in our project we have tried to incorporate our functionalities required to complete the task in three modes.

We term these functional modes as mode:0, mode:1, mode:2, which will respectively symbolize our system initialization, design/build and compile/run/output processes accordingly.

Mode 0: System initialization

Mode 1: Design/build dummy\_model

Mode 2: compile/run/output not\_dummy\_model

Further, we introduce nodal sub -structure into our modes, viz. node:0, node:1, node:2, node:3 or, simply,

n:0, n:1, n:2, n:3, which will respectively associate our subtasks of the functionalities to outputs: output\_1, output\_2.

Node 0,1in Mode 0

Node 2 in Mode 1

Node 3 in Mode 2

output\_1 : direct patient to nearest hospital for quarantine facility (patient is Covid -positive here);

output\_2 : provide vaccine and then certification /reschedule for dose\_2 (patient is Covid -negative here).

**FEATURE characteristic functionalities**:

Our feature set **F(x\_i) ⍷** {x1 , x2 , x3……...} as:

**F(x\_i)** = {x\_1, x\_2, x\_3 | x\_1 = body temperature, x\_2 = If dose 1 taken? <Y/N?>, x\_3= class of vaccine}

**y\_i class**: class 0→ covaxin, class 1→ covidshield, class 2→ remdevisir.

set **t\_thresh** =102deg , if **(x\_1) >= (t\_thresh) → output\_1**

And we will collect samples from the input block as : x\_i = = **x\_1**\_<**Y/N?>\_1/0\_0/1/2** (3rd segment is 1 for Y)

**96.7\_Y\_2\_1**: body temp = 96.7deg, dose \_1 taken, dose\_2 to be taken, class 1 : covid -shield to be taken by patient;

**105\_null\_null\_null**: body temp = 105deg → direct patient to nearest hospital for quarantine facility→ **output\_1**

**95\_N\_1\_2** :body temp = 95deg, dose \_1 not taken, dose\_1 to be taken, class 2 : remdevisir to be taken → **output\_2**

In all array segments, we see that Y\_2 and N\_1 blocks in the middle will always be together so we encode them to a single literal as, say, Y\_2 → 2 and N\_1 → 1.

This will pack our former array of **95\_N\_1\_2** (x\_i) now, as (x\_i’) **95\_1\_2** [=token\_id].

[We can also further stylize our token\_id as:

Since we have set the t\_thresh parameter for body temperature limit (s\_thresh classifies output\_1 from output\_2),

We can now discard all values like 96.7, 105, **95** to a more normalised label like for all x\_1 < t\_thresh we set default value at index[0] of x\_i matrix as 95 and else,we just directly pipeline all operations to output\_1(no need to encode).

So, we can say that all labels like 96.7, 105, **95** will now be 95, 105, 95 respectively.]

mode 0

node 0

x\_i’: **95\_1\_2** → b\_2 (binary encoding) → (b\_2 + k) (redundancy addn) → **x\_i|t=t\_n** (decoding)

node 1

x\_i|t=t\_n → Tx → **x\_i’|t=t\_(n + del) → movement\_mat\_i**

mode 1

node 2

(\*movement\_mat\_i → **random joint target** to be interpolated )

movement\_mat\_i(urdf xml)→ arm dynamics (← t \_tests in node 3 ⍷ {t\_test\_0, t\_test\_1})

mode 2

node 3

render movement\_mat\_i **motion**.

Img 1.

Depicted above is the first part of our mode 0, [data]

mode 0

(first part) → dedicated to data (x\_i) flow

node 0 : A\_activation\*\*

1. Input **F(x\_i)** : x\_1, x\_2, x\_3→p\_0 → [p\_0\_0\*, p\_0\_1\*] → p\_1\_0 (= = p\_0) → Encoding x\_i
2. Encoding of **x\_i** → p\_1 → p\_1\_y → Decoding to **x\_i|t=t\_n**

**→** p\_n\_1

Img 2.

Depicted above is the second part of our mode 0, [data]

mode 0

(second part) → dedicated to interfacing

node 1: B\_activation\*\*

Processing dedicated\_x\_i (= = x\_i|t=t\_n → Tx → **x\_i’|t=t\_(n + del) → movement\_mat\_i**).

[dedicated\_x\_i = = **x\_i’|t=t\_(n + del)**].

**→** p\_n\_2

Img 3.

Depicted above are our modes 1 and 2, [data]

[\*]Addl.

[data] show bucket layout, analytical intuition from probabilistic stochastics

[data] show Tx, Movement \_matrix, workflow instance, A\_activation, B\_activation, pipeline skeleton for IR

**4. Interface requirement**

**5. Performance requirement**

[data] show check\_gap() #between dose\_1 && dose\_2, check\_qtty()

[data] show t\_test\_mat\_0, t\_test\_mat\_1, scale\_factor

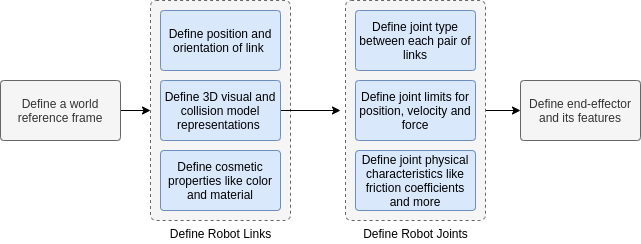
[data] prepare certification /reschedule for dose\_2

**6. Design**

Robotic arms are built out of multiple actuators and passive links forming an actuated chain that can be used to manipulate objects using different end-effectors. A robot arm could have revolute or prismatic joints for rotation or linear motion respectively.

In this project, we design a simple robot arm with revolute joints and prismatic joints.

A Universal Robot Description Format (URDF) file with the robot’s kinematic, visual, and collision model representation is needed for the robot’s interaction with ROS.



*Steps to define a robotic arm URDF file (Image Source - Original)*

The URDF file **robot\_arm\_dummy.urdf** (from movement\_mat\_i) is formed of primitive boxes (cuboids) and cylinders. It may or may not have any end-effector attached.

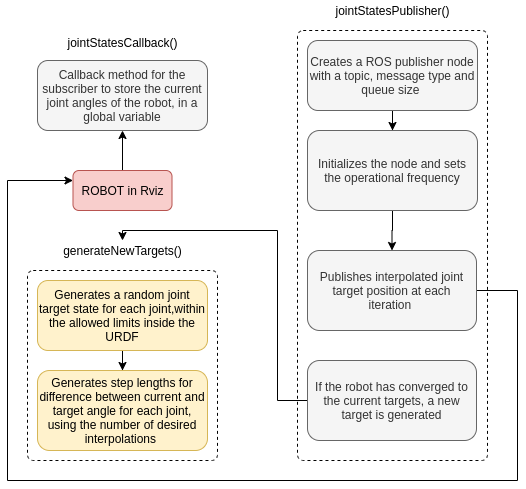
After the robot description is ready, we will follow the steps to create a ROS package to simulate the design.



*Steps involved in the simulation pipeline task (Image Source - Original)*

We design a **custom node that has a ROS subscriber** listening to the robot’s current joint positions.

It creates a **random joint target** (\*movement\_mat\_i) for the robot and then publishes the interpolated joint target position for each iteration of the ROS execution life cycle.



Node Design (Image Source - Original)

After these design requirements, we proceed to build (here we will try to simulate

and run test cases **t –test\_0** and **t –test\_1**) & then we run the **dummy\_model** to get correlational data (metrics like accuracy score).

Then we scale our model based on performance requirements in module 5 to build our final **not**\_**dummy\_model.**

**Here our designed not\_dummy\_model will classify into three object classes of vaccine varieties.**

**7. Non -Functional attribute /requirement**

[data] ..continue from 3. Functional requirement

**8. Schedule & Budget**

**Schedule**

1. **Configurability**

a. Intuitive configuration mechanisms and modular systems.

**Minimal user knowledge requirement.**

b. Automatic system configuration **based on learning**

c. Tools to identify suitable configuration of the robot based on required functionality.

2. **Adaptability**

a. Adaptation to users (**patient, surgeon, caregivers**) and environment

b. Auto-adaptation to **user learnt profile**

3. **Interaction Ability**

a. Multimodal feedback (including force tactile, vision, sound, olfaction, etc.)

b. Transparency of the interaction between the user, the robot and the environment

4. **Dependability**

a. Prediction and identification of future failures to inform the user and activate maintenance

5. **Motion Ability**

a. Capable to produce smooth human-like motion integrated with residual user controlled volitional movements

6. **Perception Ability**

a. Real time situation monitoring (person in conjunction with environment objects):-

i. Perception ability

ii. Tracking Ability

iii. Object Recognition

iv. Scene perception

7. **Cognitive Abilities**

a. Online patient state analysis

b. Flexibility of assistance solution on learned experience by integration of robotic support with residual user capability and support

**Budget**

1. **Equipment Costs** — Equipment cost is an obvious category.

2. **Outsourcing** — Robotic integrators are one option for integrating robots to your business. The costs and advantages are different from training in-house robotics experts.

3. **Depreciation** — As with any technological investment, you also have depreciation to think about. One rough way to calculate the depreciation of a robot is to divide the initial investment by the projected life of the robot.

4. **Maintenance, Upgrades and Replacement** — Hand-in-hand with depreciation comes maintenance and upgrade. There are various maintenance considerations with robotics, including: robotic hardware maintenance, add-on technology maintenance, software upgrades, etc. Each of these will have a different time-frame. Eventually, the robot itself will be in need of replacement.

.**9. Appendices**