**NAALAIYA THIRAN – IBM PROJECT**

**PROJECT REPORT**

**IOT BASED GADGET FOR CHILD SAFETY MONITORING AND NOTIFICATION**

Presented by,

**TEAM ID : PNT2022TMID31414**

|  |  |
| --- | --- |
| 710719106017 | CHARULATHA N |
| 710719106006 | APARNA BHUVANESAVRI L |
| 710719106004 | ABINEETHA M |
| 710719106013 | ASWINI G |

**TITLE**

1. **INTRODUCTION**
   1. Project Overview
   2. Purpose
2. **LITERATURE SURVEY**
   1. Existing problem
   2. References
   3. Problem Statement Definition
3. **IDEATION & PROPOSED SOLUTION**
   1. Empathy Map Canvas
   2. Ideation & Brainstorming
   3. Proposed Solution
   4. Problem Solution fit
4. **REQUIREMENT ANALYSIS**
   1. Functional requirement
   2. Non-Functional requirements
5. **PROJECT DESIGN**
   1. Data Flow Diagrams
   2. Solution & Technical Architecture
   3. User Stories
6. **CODING & SOLUTIONING**
   1. Feature 1
   2. Feature 2
7. **TESTING**
   1. User Acceptance Testing
8. **ADVANTAGES & DISADVANTAGES**
9. **CONCLUSION**
10. **FUTURE SCOPE**
11. **APPENDIX**

Source Code

GitHub

# INTRODUCTION

## PROJECT OVERVIEW

The main concept is to create a app that used to check the location of the child as safety system. This concept focus on finding the child and continuously update the current location to the parent or caretaker.

The IBM cloud, Node-red and IBM Watson are used to create the web-application which is to be used in IoT child safety monitoring gadget.

## PURPOSE

Parents can simply leave their children in park, school or somewhere else, child tracker application helps the parent to continuously monitor the child’s location.

Notifications will be sent to the caretaker’s mobile, according to the particular geofence aound the child.

# LITERATURE SURVEY

## EXISTING PROBLEM

* + 1. If the child may in the area where there is no network or move away from the network coverage area, there is a difficulties to identify and track the location of the child.
    2. The wearable device may be removed by someone who try to kidnap the child.
    3. If it was removed or missed by the child, the identification and tracking are worthless then we proposed the method of sense the body temperature of child to identify whether it is with the child or not.

## REFERENCE

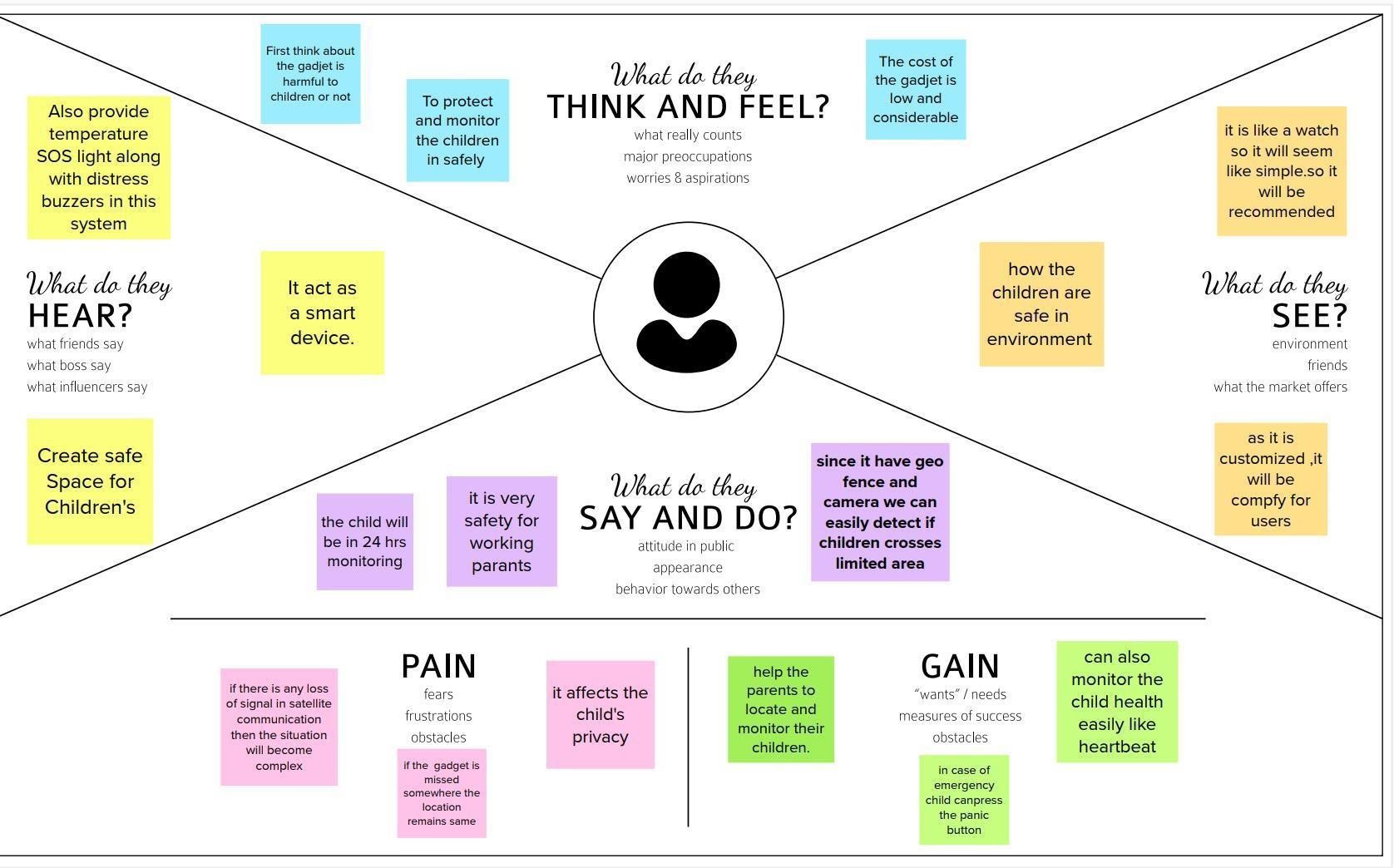
|  |  |  |
| --- | --- | --- |
| **S.No** | **Title** | **Reference** |
| 01 | Safety of a child in large public | https://ieeexpl ore.ieee.org/ab stract/document/9031524 |
| 02 | Intelligent child safety system using Machine Learning in IoT  devices | https://ieeexpl ore.ieee.org/do cument/927713 6 |
| 03 | Smart wearable devices for little children | https://ieeexpl ore.ieee.org/do cument/789953 1 |

* 1. **PROBLEM STATEMENT DEFINITION**

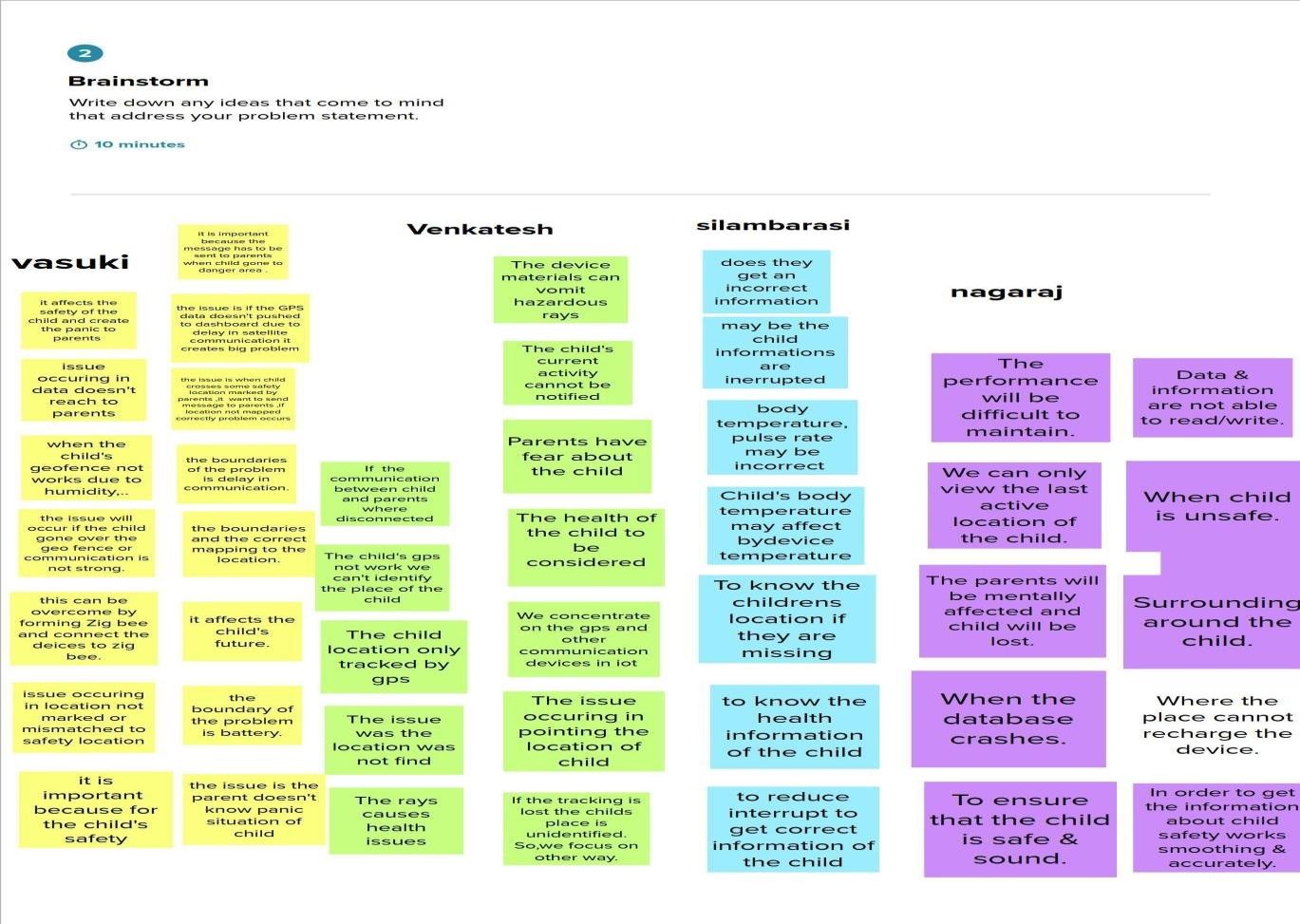
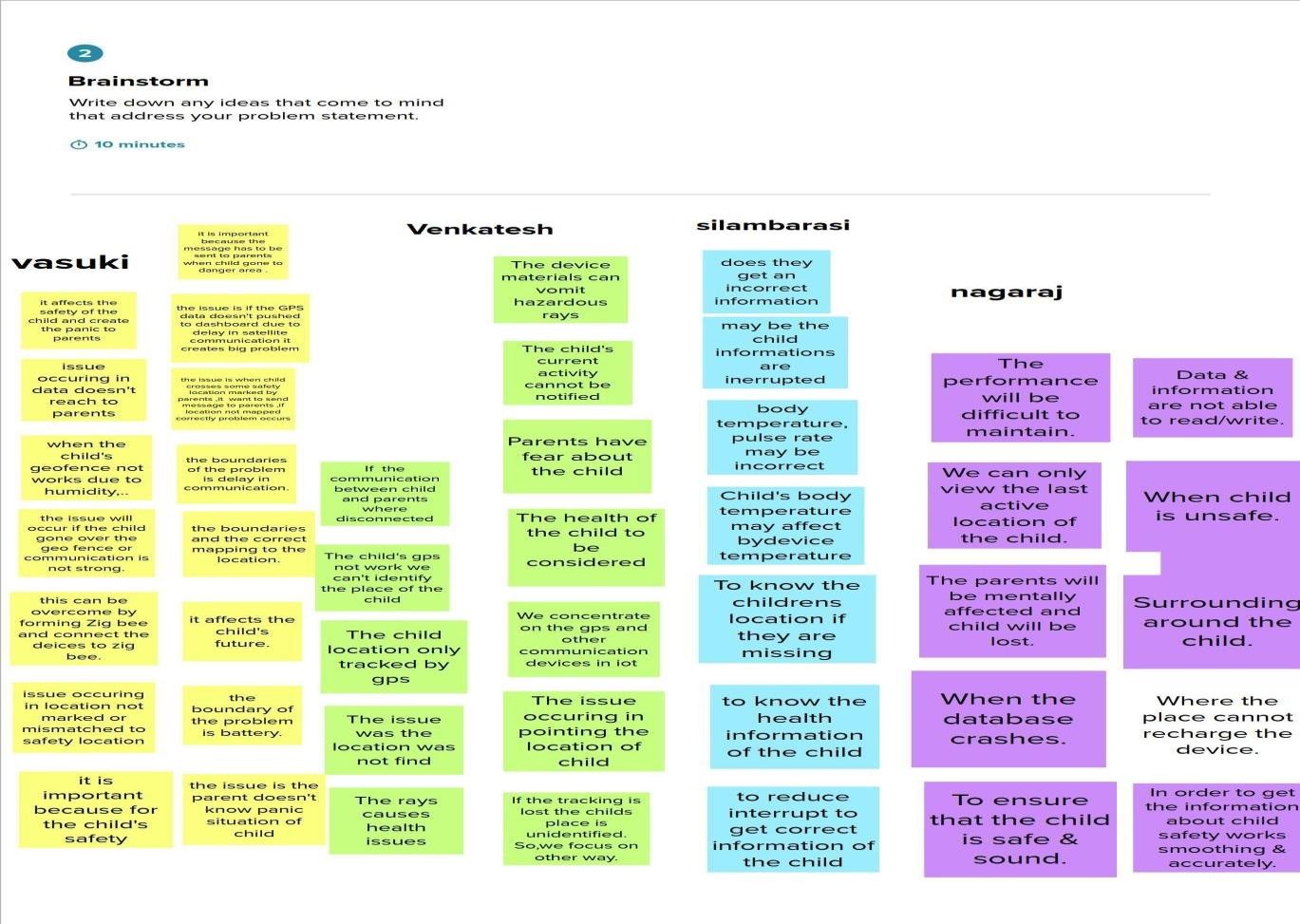
|  |  |  |
| --- | --- | --- |
| I am | Product developer | to make a device |
| I’m trying to | made a device | for child safety |
| But | will not receiving the  propersignal | signal clashes |
| Because | out of coverage and  sometechnical issue | internet issue and  serverdown |
| Which makes me feel | Anxiety | we can't face the  problemdirectly |

# IDEATION & PROPOSED SOLUTION

## EMPATHY MAP CANVAS



* 1. **IDEATION & BRAINSTORMING**



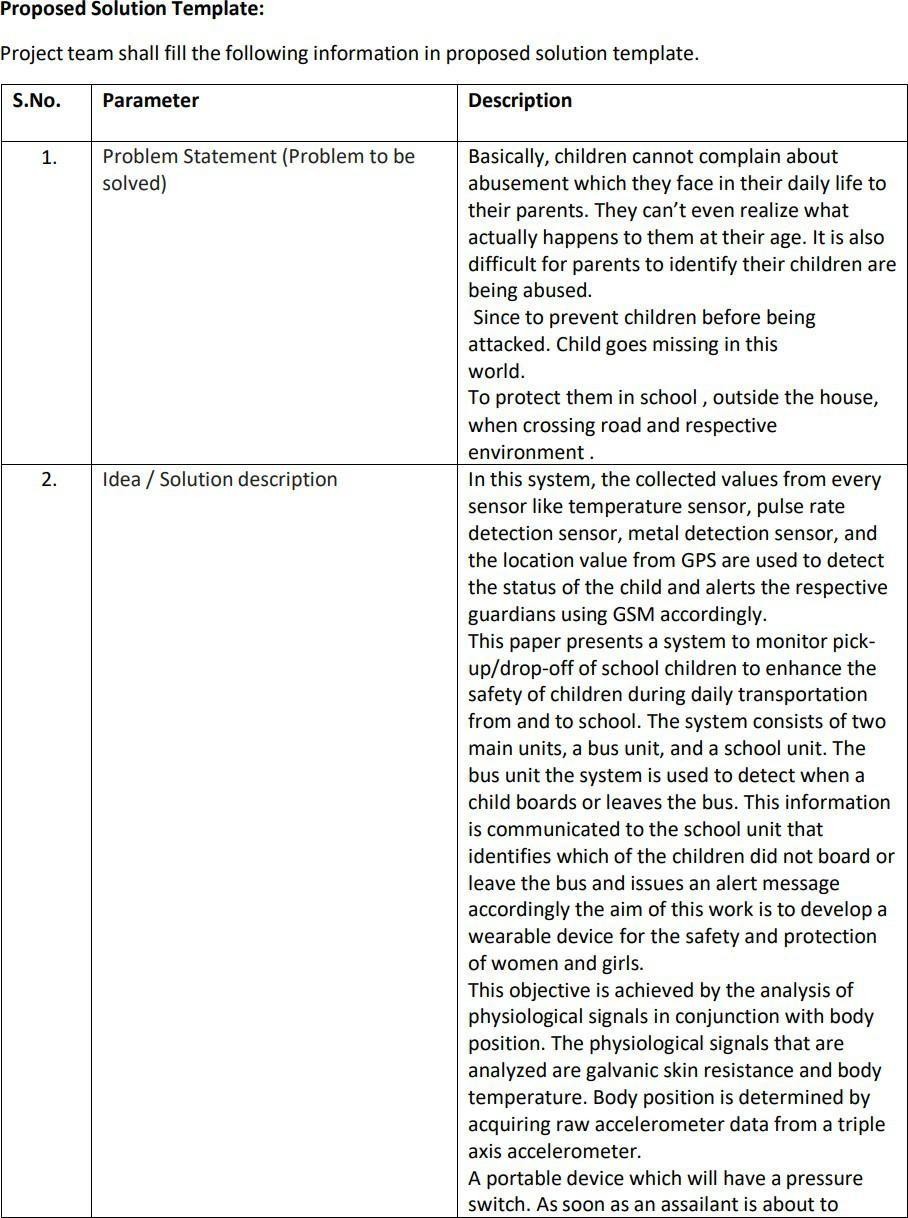
**CHARULATHA**

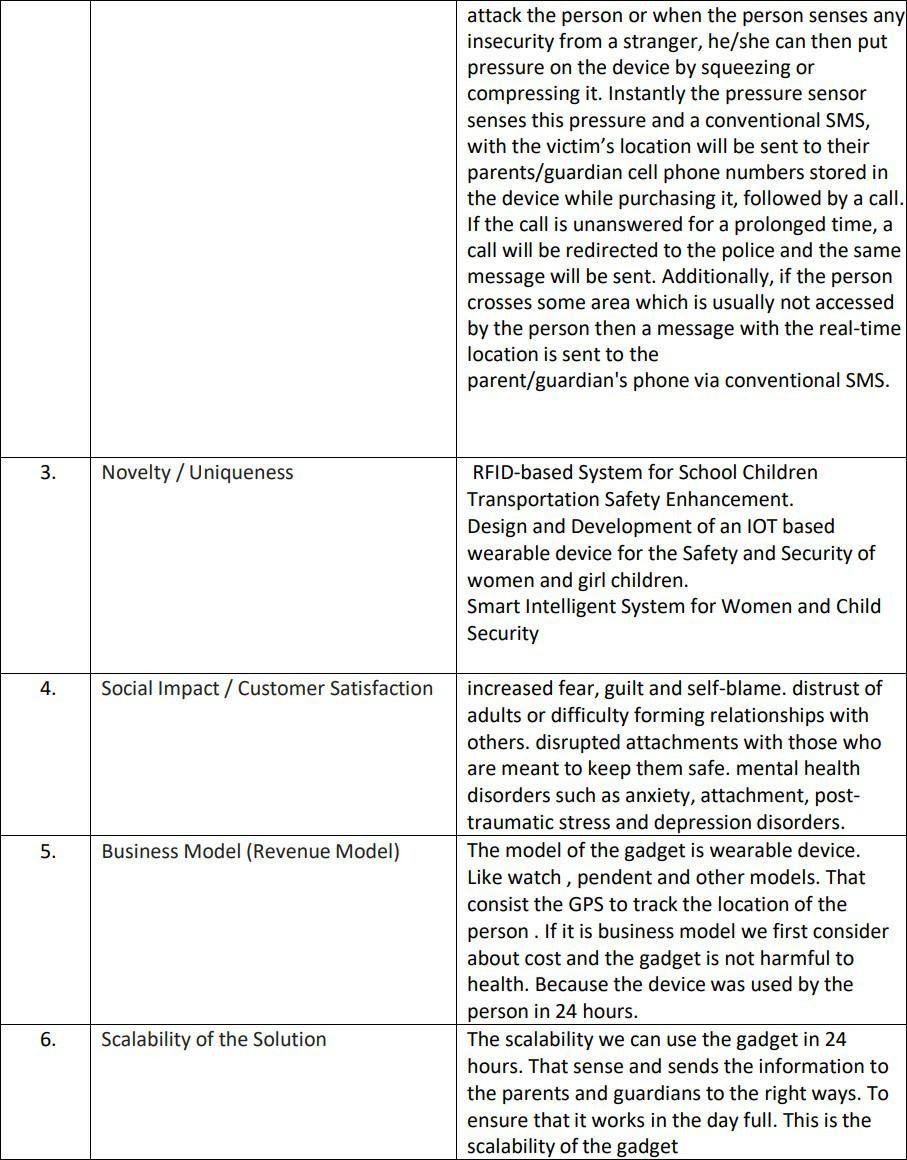
**APARNA BHUVANESVARI**

**ASWINI**

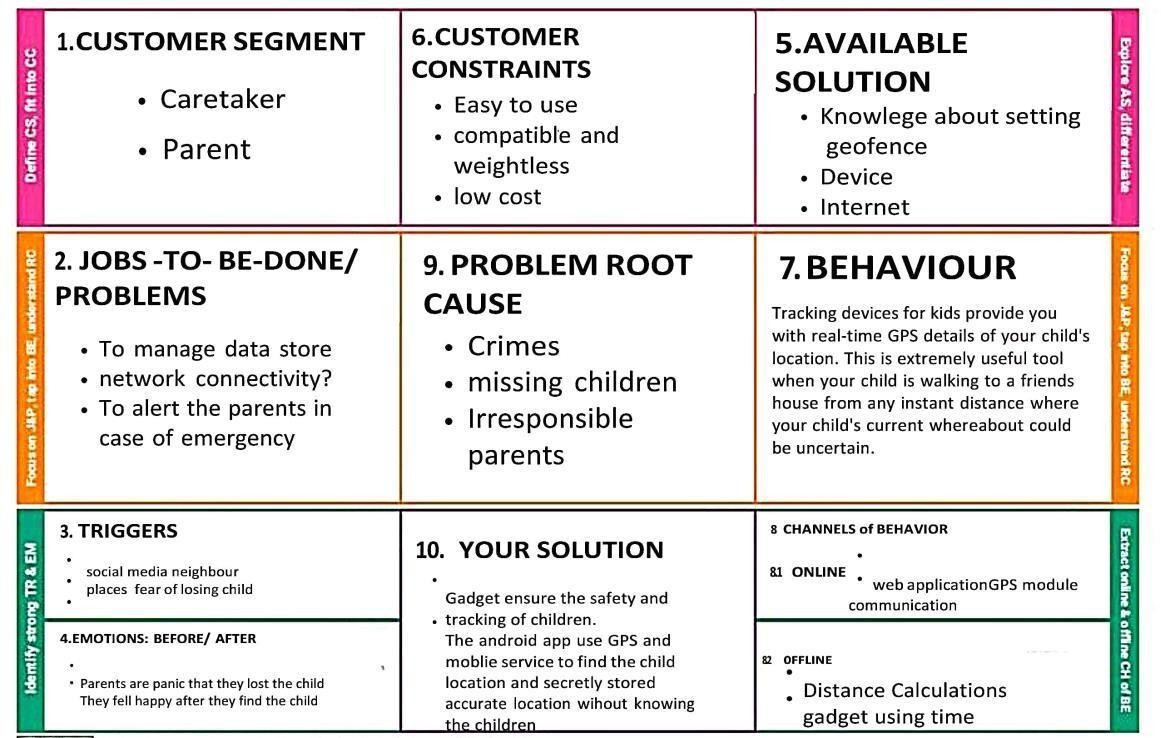
**ABINEETHA**

## PROPOSED SOLUTION



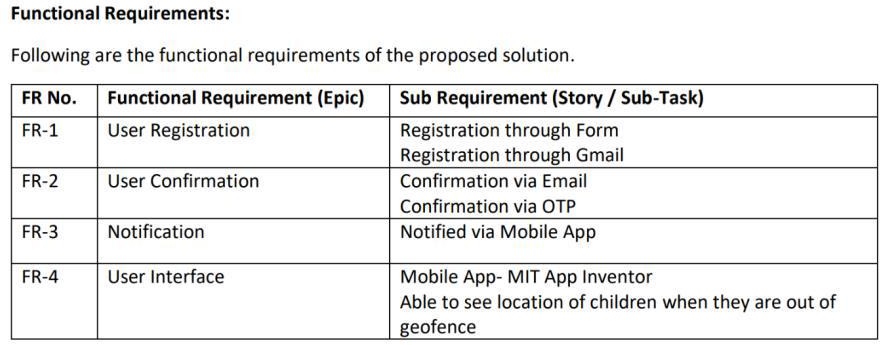


* 1. **PROBLEM SOLUTION FIT**

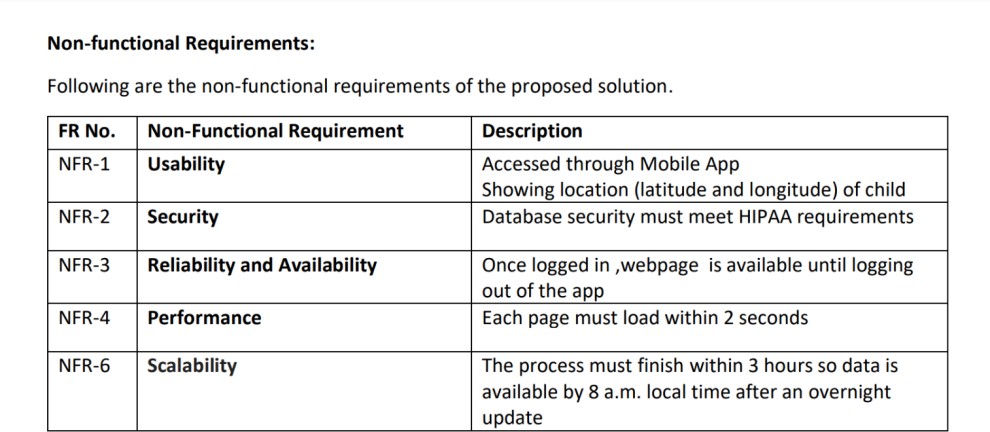


# REQUIREMENT ANALYSIS

## FUNCTIONAL REQUIREMENT

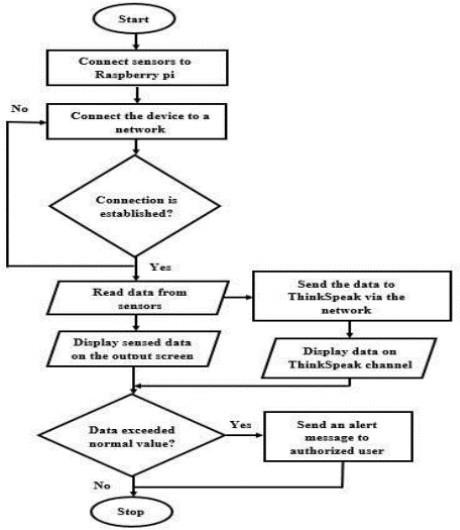


* 1. **NON-FUNCTIONAL REQUIREMENTS**

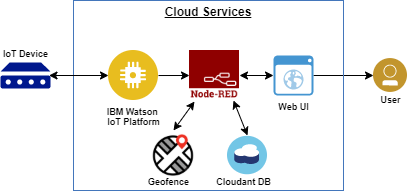


# PROJECT DESIGN

## DATA FLOW DIAGRAMS



* 1. **SOLUTION & TECHNICAL ARCHITECTURE**



## USER STORIES

List all the user stories for the product.

|  |  |  |  |
| --- | --- | --- | --- |
| **USER STORY**  **NUMBER** | **USER STORY/**  **TASK** | **ACCEPTANCE**  **CRITERIA** | **PRIORITY** |
| USN-1 | To use the product when the child needs safety | Parents can access the device with  lock | HIGH |
| USN-2 | Notification will be provided when child is in trouble. | Child cannot access the device as there is a lock | HIGH |
| USN-3 | To safeguard the child when in danger using GPS they can track their  location. | Lock Access Only by parents. | LOW |
| USN-4 | During Emergency there will be  alarm | Lock Access Only by concerned  persons. | MEDIUM |
| USN-5 | When child is missing parents will be  notified | Lock Access Only by Users. | HIGH |

# CODING & SOLUTIONING

## FEATURE 1

* + 1. HTML
    2. JAVA SCRIPT
    3. CSS
    4. PYTHON
  1. **FEATURE 2**

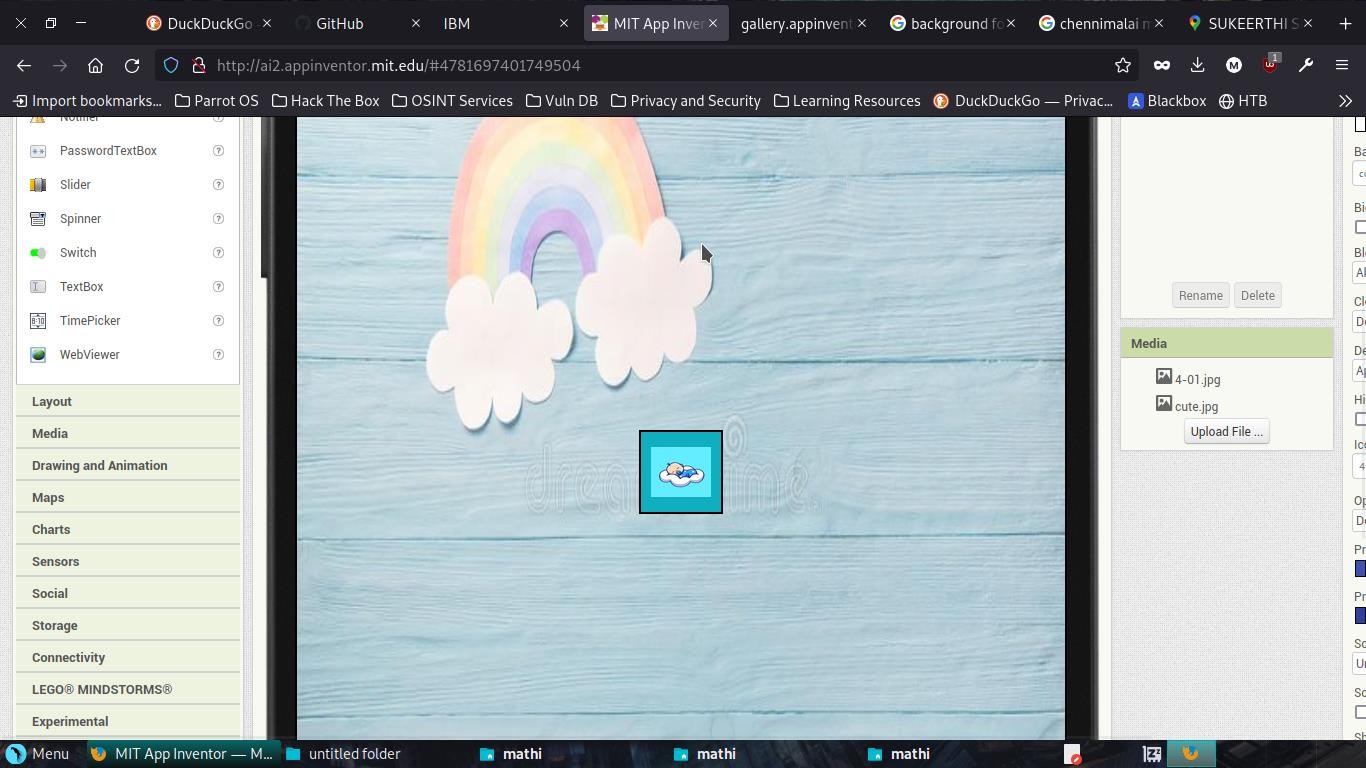
|  |  |  |  |
| --- | --- | --- | --- |
| **S.NO** | **COMPONENT** | **DESCRIPTION** | **TECHNOLOGY** |
| 01 | User Interface | The communication protocol being used in the proposed solution might act as an interface the  way like wifi, bluetooth | MIT APP |
| 02 | Application logic | The data to be collected and send to the authenti- cator via GSM providing the GPS coordinates to easily located access and monitor the child | IBM Watson STT service,python etc |
| 03 | Database | Date to be segregated and secured in the form of re- | MySQL |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | lation DBMS |  |
| 04 | Cloud Database | IBM | IBM Cloudant |
| 05 | File storage | File storage requirements | IBM block stor- age or other storage service or local filesys- tem |
| 06 | External API-1 | To access the children lo- cation | GPS location monitoring etc |
| 07 | Infrastructure | Application deployment on local system/ cloud lo- cal server configuration | Cloud foundry |

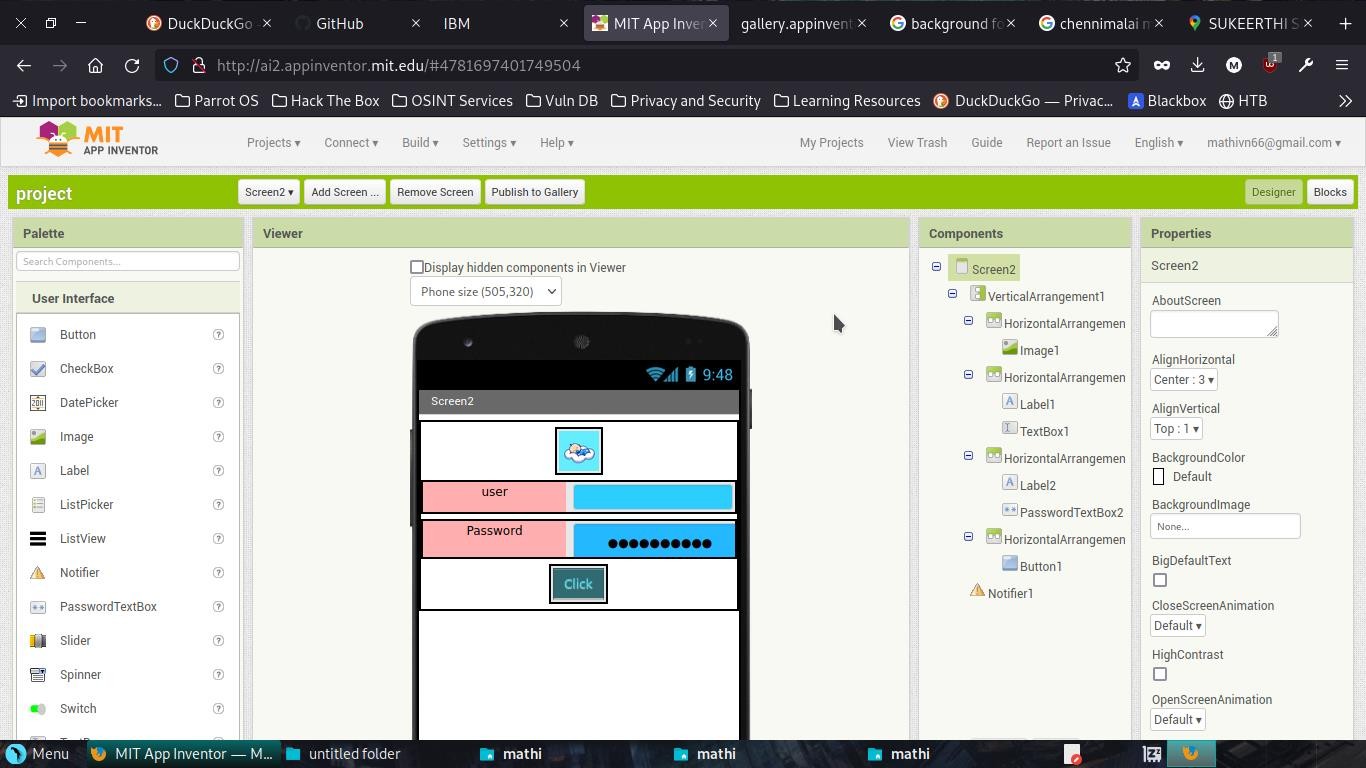
# TESTING

## USER ACCEPTANCE TESTING

MIT APPLICATION INTERFACE

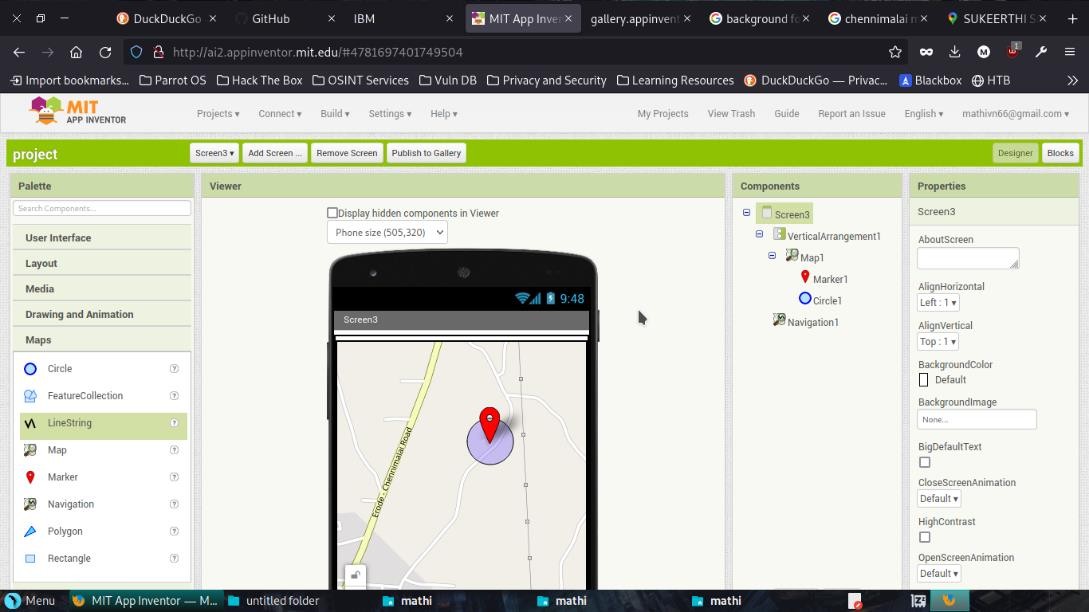


PARENTAL CONTROL INTERFACE

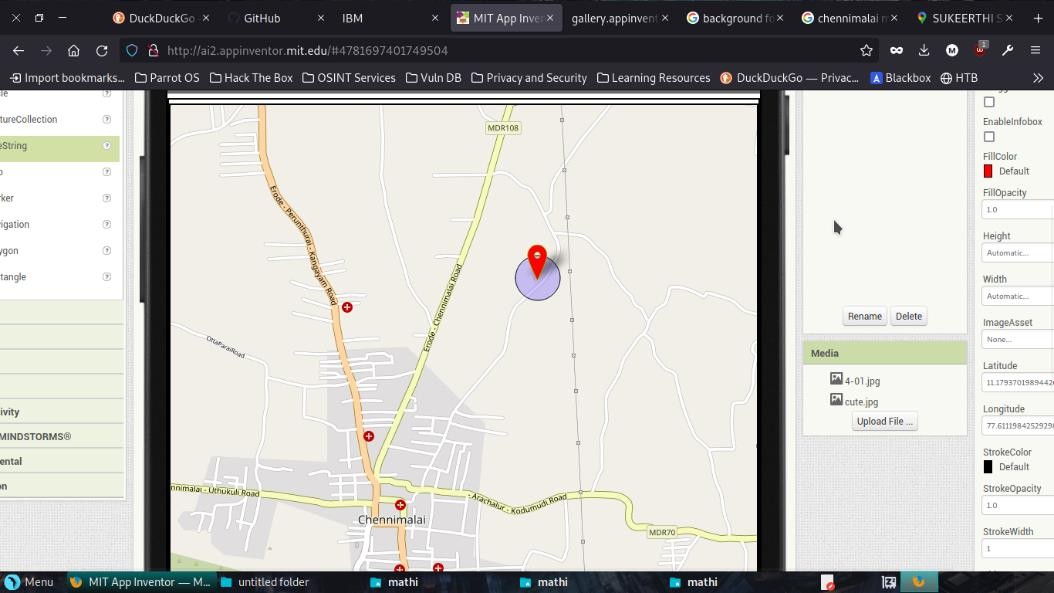


# OUTPUT

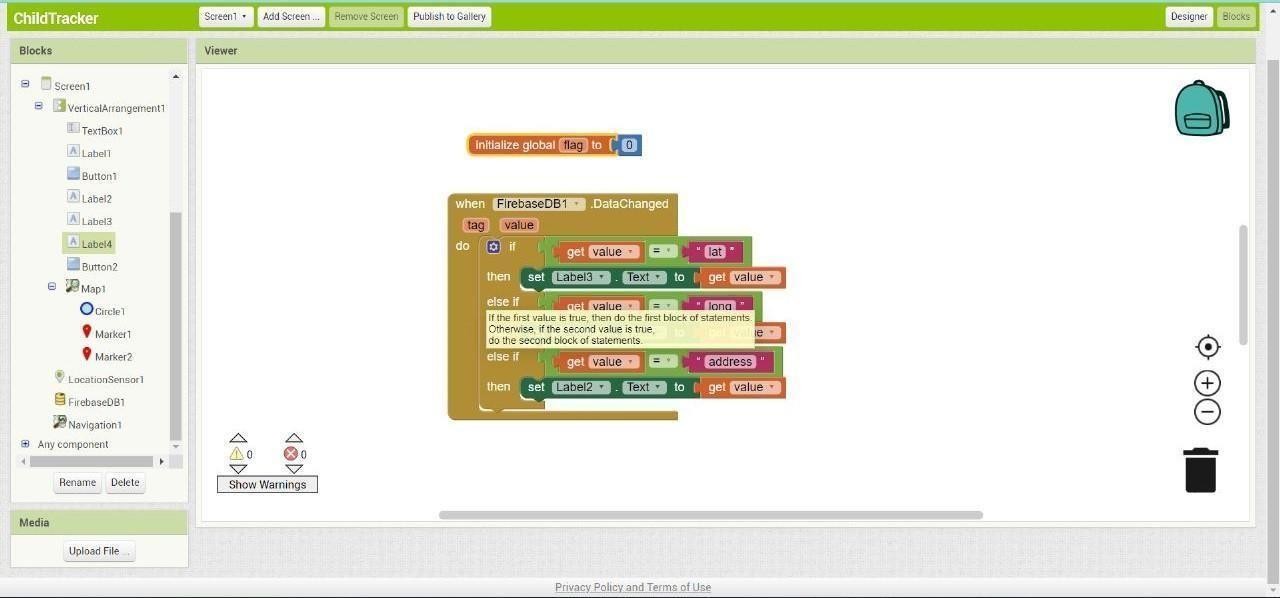
LOCATION STATUS OF CHILD (EMULATOR)



LOCATION STATUS OF CHILD (MONITOR)



BLOCK DIAGRAM



# ADVANTAGES & DISADVANTAGES

## ADVANTAGES:

* Provide safety to the child by allowing their parent to locate.
* This application can be used to monitor the temperature and motion of the child.
* Child safety can be ensured and crime rate against the child can be reduced.
* It can be used to analyze the dynamic environment of the child and alerting system.

## DISADVANTAGES:

* Technical difficulties.
* High cost and difficult to implement.
* Poor data quality.
* Design quality.

# CONCLUSION

This concept demonstrates IoT based gadget for child safety monitoring and notifications is helping the parent to locate and monitor their children. If any abnormal values are read by the sensor then an SMS is sent to the parents mobile. It assists parents to monitor their children remotely. The project aims to create a system that allows the parents to keep track their children when they are out of their sight. This is done by using the IBM cloud, Node-red and IBM Watson which is used to create and store the informations of the project. The child will wear GPS enabled device which is connected to the parent’s smart phone where the child safety web-application is installed.

# FUTURE SCOPE

In this web-application if any abnormalities are read by the sensor an MMS indicating an image or video captured by the serial camera which is to be included with emergency and safety system for accurate surveillance of the child’s surroundings. The future scope of the work is to implement the IoT device which ensures the complete solution for child safety problems.

For this project using the GSM technologies is beneficial as the cellular range is vast and since all the communication between wearable device and the parent is taking place via SMS, therefore no internet connectivity is required. But, still the GSM module possess the added advantage of using GPRS which enables the board to use the internet if required. Whereas for camera module which supports video streaming but due to the constraint of trying to use only SMS, therefore more number of connections will be taking place.

# APPENDIX

## SOURCE CODE

from try:

future

import absolute\_import, division, unicode\_literals

from collections.abc import MutableMapping except ImportError: # Python 2.7

from collections import MutableMapping from xml.dom import minidom, Node

import weakref

from . import base

from .. import constants

from ..constants import namespaces

from ..\_utils import moduleFactoryFactory

def getDomBuilder(DomImplementation):

Dom = DomImplementation

class AttrList(MutableMapping):

def

init

(self, element):

self.element = element

def

iter

(self):

return iter(self.element.attributes.keys())

def

setitem

(self, name, value):

if isinstance(name, tuple):

raise NotImplementedError else:

attr = self.element.ownerDocument.createAttribute(name) attr.value = value

self.element.attributes[name] = attr

def

len

(self):

return len(self.element.attributes)

def items(self):

return list(self.element.attributes.items())

def values(self):

return list(self.element.attributes.values())

def

getitem

(self, name):

if isinstance(name, tuple):

raise NotImplementedError else:

return self.element.attributes[name].value

def delitem\_\_(self, name):

if isinstance(name, tuple): raise NotImplementedError

else:

del self.element.attributes[name]

class NodeBuilder(base.Node):

def

init

(self, element):

base.Node.

init

(self, element.nodeName)

self.element = element

namespace = property(lambda self: hasattr(self.element, "namespaceURI") and self.element.namespaceURI or None)

def appendChild(self, node):

node.parent = self self.element.appendChild(node.element)

def insertText(self, data, insertBefore=None):

text = self.element.ownerDocument.createTextNode(data) if insertBefore:

self.element.insertBefore(text, insertBefore.element) else:

self.element.appendChild(text)

def insertBefore(self, node, refNode): self.element.insertBefore(node.element, refNode.element) node.parent = self

def removeChild(self, node):

if node.element.parentNode == self.element: self.element.removeChild(node.element)

node.parent = None

def reparentChildren(self, newParent):

while self.element.hasChildNodes():

child = self.element.firstChild self.element.removeChild(child) newParent.element.appendChild(child)

self.childNodes = []

def getAttributes(self):

return AttrList(self.element)

def setAttributes(self, attributes):

if attributes:

for name, value in list(attributes.items()): if isinstance(name, tuple):

if name[0] is not None:

qualifiedName = (name[0] + ":" + name[1]) else:

qualifiedName = name[1] self.element.setAttributeNS(name[2], qualifiedName,

value)

else:

self.element.setAttribute( name, value)

attributes = property(getAttributes, setAttributes)

def cloneNode(self):

return NodeBuilder(self.element.cloneNode(False))

def hasContent(self):

return self.element.hasChildNodes()

def getNameTuple(self):

if self.namespace is None:

return namespaces["html"], self.name else:

return self.namespace, self.name nameTuple = property(getNameTuple)

class TreeBuilder(base.TreeBuilder): # pylint:disable=unused-variable def documentClass(self):

self.dom = Dom.getDOMImplementation().createDocument(None, None, None) return weakref.proxy(self)

def insertDoctype(self, token):

name = token["name"] publicId = token["publicId"] systemId = token["systemId"]

domimpl = Dom.getDOMImplementation()

doctype = domimpl.createDocumentType(name, publicId, systemId) self.document.appendChild(NodeBuilder(doctype))

if Dom == minidom: doctype.ownerDocument = self.dom

def elementClass(self, name, namespace=None):

if namespace is None and self.defaultNamespace is None: node = self.dom.createElement(name)

else:

node = self.dom.createElementNS(namespace, name) return NodeBuilder(node)

def commentClass(self, data):

return NodeBuilder(self.dom.createComment(data))

def fragmentClass(self):

return NodeBuilder(self.dom.createDocumentFragment())

def appendChild(self, node): self.dom.appendChild(node.element)

def testSerializer(self, element):

return testSerializer(element)

def getDocument(self):

return self.dom

def getFragment(self):

return base.TreeBuilder.getFragment(self).element

def insertText(self, data, parent=None):

data = data

if parent != self: base.TreeBuilder.insertText(self, data, parent)

else:

if hasattr(self.dom, '\_child\_node\_types'):

if Node.TEXT\_NODE not in self.dom.\_child\_node\_types:

self.dom.\_child\_node\_types = list(self.dom.\_child\_node\_types) self.dom.\_child\_node\_types.append(Node.TEXT\_NODE)

self.dom.appendChild(self.dom.createTextNode(data))

implementation = DomImplementation name = None

def testSerializer(element): element.normalize()

rv = []

def serializeElement(element, indent=0):

if element.nodeType == Node.DOCUMENT\_TYPE\_NODE: if element.name:

if element.publicId or element.systemId: publicId = element.publicId or "" systemId = element.systemId or ""

rv.append("""|%s<!DOCTYPE %s "%s" "%s">""" %

(' ' \* indent, element.name, publicId, systemId))

else:

rv.append("|%s<!DOCTYPE %s>" % (' ' \* indent, element.name)) else:

rv.append("|%s<!DOCTYPE >" % (' ' \* indent,)) elif element.nodeType == Node.DOCUMENT\_NODE:

rv.append("#document")

elif element.nodeType == Node.DOCUMENT\_FRAGMENT\_NODE: rv.append("#document-fragment")

elif element.nodeType == Node.COMMENT\_NODE: rv.append("|%s<!-- %s -->" % (' ' \* indent, element.nodeValue))

elif element.nodeType == Node.TEXT\_NODE: rv.append("|%s\"%s\"" % (' ' \* indent, element.nodeValue))

else:

if (hasattr(element, "namespaceURI") and element.namespaceURI is not None):

name = "%s %s" % (constants.prefixes[element.namespaceURI], element.nodeName)

else:

name = element.nodeName rv.append("|%s<%s>" % (' ' \* indent, name)) if element.hasAttributes():

attributes = []

for i in range(len(element.attributes)):

attr = element.attributes.item(i) name = attr.nodeName

value = attr.value

ns = attr.namespaceURI if ns:

name = "%s %s" % (constants.prefixes[ns], attr.localName) else:

name = attr.nodeName attributes.append((name, value))

for name, value in sorted(attributes): rv.append('|%s%s="%s"' % (' ' \* (indent + 2), name, value))

indent += 2

for child in element.childNodes:

serializeElement(child, indent) serializeElement(element, 0)

return "\n".join(rv) return locals()

getDomModule = moduleFactoryFactory(getDomBuilder)

## GITHUB REPOSITORY LINK :

https://github.com/IBM-EPBL/IBM-Project-39504-1660452006