ISSN No:-2456-2165

Fusion Reality

Akhildev MJ
Computer Science and Engineering
Sahrdaya College of Engineering and Technology,
Kodakara, Kerala 680684, India

Albin Saj Chalissery
Computer Science and Engineering
Sahrdaya College of Engineering and Technology,
Kodakara, Kerala 680684, India

Maritta Stephen
Computer Science and Engineering
Sahrdaya College of Engineering and Technology,
Kodakara, Kerala 680684, India

Deepa Devassy
Computer Science and Engineering
Sahrdaya College of Engineering and Technology,
Kodakara, Kerala 680684, India

Abstract:- Fusion Reality is the next generation reality that will change the aspect of everything. We talk about a lot of reality technologies including augmented reality and virtual reality. But experiencing mixed reality is really expensive. This paper deals with fusion reality which aims to develop a software development kit that includes support for developing mixed reality applications within minutes. Our development kit is included in the official website of our fusion reality. Developers can download and install the Unity 3D software. After downloading the software developers can import the fusion reality kit package which is available at our website and can start to create theirs on mixed reality applications.

The website contains the necessary files for developing fusion reality applications as well as integrated with an artificially intelligent chatbot which can be helpful in solving any doubts and queries the developers might have while developing the application. The chatbot is also available on telegram social media applications and developers can utilize the telegram application also to solve doubts and queries.

Atoms, CoronaAR are android applications developed with a fusion reality kit of ours. Atoms deal with chemistry and aim to decrease the complexity of learning periodic table and other chemistry related things. CoronaAR also built with the latest technologies and tools associated with the fusion reality SDK and assets. Atoms are capable of augmenting the periodic table and elements with details of each particular element having its basic properties with isotopes, compounds and video displays to learn or experience periodic table like never before. CoronaAR enables peoples to interact with Indian rupees notes to get a better understanding about the Corona virus, the precautions to be taken and the better ways to overcome disease with tips. Atoms, CoronaAR are just examples of what people can build with Fusion reality kit. All the existing augmented reality and virtual reality applications can also be modified with fusion reality kit to make it more understandable and productive.

Keywords:- *Fusion Reality, Mixed Reality, Virtual Reality, Augmented Reality, Unity, Vuforia.*

I. INTRODUCTION

Fusion Reality is the next generation reality that will change the aspect of everything. We talk about a lot of reality technologies including augmented reality and virtual reality. But experiencing mixed reality is really expensive. This paper deals with fusion reality which aims to develop a software development kit that includes support for developing mixed reality applications within minutes.

Our development kit is included in the official website of our fusion reality. Developers can download and install the Unity 3D software. After referring website as well as integrated with an artificially intelligent chatbot which can be helpful in solving any doubts and queries the developers might have while developing the application. The chatbot is also available on telegram social media applications and developers can utilize the telegram application also to solve doubts and queries.

Atoms, CoronaAR are android applications developed with a fusion reality kit of ours. Atoms deal with chemistry and aim to decrease the complexity of learning periodic table and other chemistry related things. CoronaAR also built with the latest technologies and tools associated with the fusion reality SDK and assets. Atoms are capable of augmenting the periodic table and elements with details of each particular element having its basic properties with isotopes, compounds and video displays to learn or experience periodic table like never before.

CoronaAR enables peoples to interact with Indian rupees notes to get a better understanding about the Corona virus, the precautions to be taken and the better ways to overcome disease with tips. Atoms, CoronaAR are just examples of what people can build with Fusion reality kit. All the existing augmented reality and virtual reality applications can also be modified with fusion reality kit to make it more understandable and productive.

ISSN No:-2456-2165

II. ARCHITECTURE

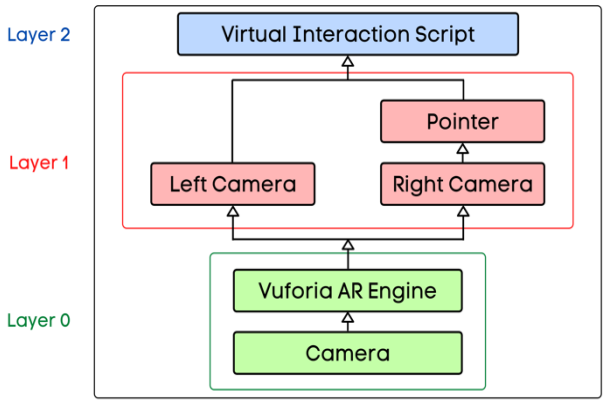


Fig 1:- Fusion Reality Architecture

Following are the architecture components.

➢ *Camera*

A camera is a visual device used to record images. At their most basic, cameras are sealed cases (the camera body) with a little hole (the opening) that lets light in to grab a picture on a light-sensitive surface (usually photographic film or a digital sensor). Cameras have various devices to check how the light falls onto the light-sensitive surface. The still image camera is the central device in the art of photography and taken pictures may be photographed later as a part of the means of photography, digital imaging, photographic printing. Downloading the software developers can import the fusion reality kit package which is available at our website and can start to create theirs on mixed reality applications. The website contains the necessary files for developing fusion reality

➢ *Vuforia AR Engine*

Vuforia AR engine provides the basic functionalities of Augmented reality. It provides various 3D Model projections and augmenting the same into ground/terrain or mid-air. This layer helps to 1279tilize all the features of the Augmented reality.

➢ *Left Camera*

The viewport of the left camera is set from w=1 to w=0.5. This setup will make the screen render in the virtual reality mode that is the AR camera is divided into two horizontal spaces. Left horizontal space is left camera.

➢ *Right Camera*

The viewport of the right camera is set from w=1 to w=0.5. This setup will make the screen render in the virtual reality mode that is the AR camera is divided into two horizontal spaces. Right horizontal space is right camera.

➢ *Pointer*

Pointer is used as a visual aid for aiming. The position of the pointer is either at a default position in space or on the surface of a VRInteractiveItem as determined by the VREyeRaycaster.

➢ *Virtual Interaction Script*

Virtual interaction Script consists of several scripts that helps to interact with the virtual objects in the Augmented environment. This module enables the user to complete operations in an easy way compared to other existing methods used in virtual reality.

III. COMPARISON TABLE

Mixed Reality	Fusion Reality
LCMR Uses Google VR SDK.	Fusion Reality Uses Fusion Reality SDK
Only Supported by Unity 2017 Version	Supported by Almost all Unity Versions
Development is Very Hard	Development is Easy Compared to LCMR
Virtual Reality Screen is small compared to Fusion Reality	Virtual Reality Screen is Large compared to LCMR
Low Clarity Compared to the Fusion Reality	High Clarity Compared to the LCMR
Large Area of Screen is Blank.	Only Small Area of Screen is Blank.
Application Only Works with some Android Phones.	Application Works with any Android Phones.
Requires Google Cardboard Application to Run this Application which is Supported by Only few Android Phones.	Requires no Additional or Specific Application to Run the Application.
Only Few Virtual Interactions are Supported.	Comparatively More Virtual Interactions are Supported
Development Time is High.	Comparatively Less Development Time.

Fig 2

IV. IMPLEMENTATION

The implementation includes the following phases.

➢ Fusion Reality Kit development
➢ Chat Bot integrated Website Development.
➢ Atoms - Application built with Fusion Reality Kit.

A. Fusion Reality Kit development

The fusion reality kit supports the Unity 3D environment. Vuforia is a public platform available to create and develop an Augmented reality application. The kit is developed with the support of the vuforia augmented reality plugin. In the package manager, vuforia plugin is installed and imported. AR camera is the default camera provided by the vuforia to scan the environment as well as markers. In Fusion reality, we created a camera inside the AR camera. The newly added camera will act like the Left camera and the script CameraFieldView.cs is added. The script will set the field of view of the newly added camera to that of the AR camera which is default camera of the Vuforia.

ISSN No:-2456-2165

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CameraFieldView : MonoBehaviour
{
  void Start()
  {
    StartCoroutine(Wait());
  }

  IEnumerator Wait()
  {
    yield return new WaitForSeconds(1);
this.GetComponent<Camera>().fieldOfView = GameObject.Find("ARCamera").GetComponent<Camera>().fieldOfView;
  }
}
```

After creating the Left camera which is duplicated to create the right camera in which together with the CameraFieldView script CameraFocusControl.cs script is added

```
using UnityEngine;
using Vuforia;
public class CameraFocusController : MonoBehaviour {
 private bool myVuforiaStarted = false;
 void Start ()
 {
  VuforiaARController vuforia = VuforiaARController.Instance;

  if (vuforia != null)

vuforia.RegisterVuforiaStartedCallback(StartAfterVuforia);
 }
 private void StartAfterVuforia()
 {
  myVuforiaStarted = true;
  SetAutofocus();
 }
 void OnApplicationPause(bool pause)
 {
  if (!pause)
  {
   if (myVuforiaStarted)
   }
 }
 private void SetAutofocus()
 {
  if (CameraDevice.Instance.SetFocusMode(CameraDevice.FocusMode.FOCUS_MODE_CONTINUOUSAUTO))
   Debug.Log("Autofocus is set");
  else
Debug.Log("Sorry this device doesn't support auto focus");
 }
}
```

The script will set the auto focus mode for the right camera after vuforia instance is created. The focus mode used in the right camera is CONTINUOUS_AUTO_FOCUS. The viewport of the left camera and right camera is set from w=1 to w=0.5. This setup will make the screen render in the virtual reality mode that is the AR camera is divided into two horizontal spaces. Fusion reality not only deals with mixing augmented reality and virtual reality. To add virtual reality input to the software kit under the Right camera new canvas added and named it as Reticle. The reticle will help to navigate through in the mixed reality environment. Under reticle, the necessary UI contents to visualize the reticle(cursor) is designed. To enable the functionalities of the reticle in the camera context Reticle.cs, VRInput.cs, VREyeRayCastor.cs is added. Reticle.cs is used as a visual aid for aiming. The position of reticle will be default in space or the surface of a VRInteractiveItem as determined by the VREyeRaycaster.

```
using UnityEngine;
using UnityEngine.UI;
public class myReticle : MonoBehaviour
{
    [SerializeField] public float m_DefaultDistance = 5f;
    [SerializeField] private bool m_UseNormal;
    [SerializeField] private Image m_Image;
    [SerializeField] private Transform m_ReticleTransform;
    [SerializeField] private Transform m_Camera;

    private Vector3 m_OriginalScale;
    private Quaternion m_OriginalRotation;

    private float timeToClick = 1f;
    private bool filling = false;

    public bool UseNormal
    {
      get { return m_UseNormal; }
      set { m_UseNormal = value; }
    }


    public Transform ReticleTransform { get { return m_ReticleTransform; } }


    private void Awake()
    {
      m_OriginalScale = m_ReticleTransform.localScale;
      m_OriginalRotation = m_ReticleTransform.localRotation;
    }

    public void Hide()
    {
      Debug.Log ("Hiding image");
      m_Image.enabled = false;
    }
```

```
    public void Show()
    {
        Debug.Log ("Hiding image");
        m_Image.enabled = true;
    }

    public void SetPosition ()
    {
        m_ReticleTransform.position = m_Camera.position
+ m_Camera.forward * m_DefaultDistance;
        m_ReticleTransform.localScale = m_OriginalScale *
m_DefaultDistance;
        m_ReticleTransform.localRotation =
m_OriginalRotation;
    }


    public void SetPosition (RaycastHit hit)
    {
        m_ReticleTransform.position = hit.point;
        m_ReticleTransform.localScale = m_OriginalScale *
hit.distance;
        if (m_UseNormal)
            m_ReticleTransform.rotation =
Quaternion.FromToRotation (Vector3.forward, hit.normal);
        else
            m_ReticleTransform.localRotation =
m_OriginalRotation;



    }

    public void fillInTime(float mainTimeToClick) {
        m_Image.fillAmount = 0f;
        filling = true;
        timeToClick = mainTimeToClick;
    }

    public void stopFilling() {
        filling = false;
        m_Image.fillAmount = 0f;
    }

    private void Update() {
        if (filling) {
            m_Image.fillAmount +=
Time.deltaTime/timeToClick;
            if (m_Image.fillAmount > 1) {
                m_Image.fillAmount = 0f;
                filling = false;
            }
        }
    }
}
```

VRInput class encapsulates all the input required for most VR games. It has events that can be subscribed to by classes that need specific input. This class must exist in every scene and so can be attached to the main camera for ease.

```
using System;
using UnityEngine;
namespace VRStandardAssets.Utils
{
  public class VRInput : MonoBehaviour
  {
    public enum SwipeDirection
    {
      NONE,
      UP,
      DOWN,
      LEFT,
      RIGHT
    };
    public event Action<SwipeDirection> OnSwipe;
    public event Action OnClick;
    public event Action OnDown;
    public event Action OnUp;
    public event Action OnDoubleClick;
    public event Action OnCancel;
    [SerializeField] private float my_DoubleClickTime =
0.3f;
    [SerializeField] private float my_SwipeWidth = 0.3f;
    private Vector2 my_MouseDownPosition;
    private Vector2 my_MouseUpPosition;
    private float my_LastMouseUpTime;
    private float my_LastHorizontalValue;
    private float my_LastVerticalValue;
    public float DoubleClickTime{ get { return
my_DoubleClickTime; } }
    private void Update()
    {
      CheckInput();
    }
    private void CheckInput()
    {
      SwipeDirection swipe = SwipeDirection.NONE;
      if (Input.GetButtonDown("Fire1"))
      {
        my_MouseDownPosition = new
Vector2(Input.mousePosition.x, Input.mousePosition.y);
        if (OnDown != null)
          OnDown();
      }
      if (Input.GetButtonUp ("Fire1"))
      {
        my_MouseUpPosition = new Vector2
(Input.mousePosition.x, Input.mousePosition.y);
        swipe = DetectSwipe ();
      }
      if (swipe == SwipeDirection.NONE)
        swipe = DetectKeyboardEmulatedSwipe();
      if (OnSwipe != null)
        OnSwipe(swipe);
      if(Input.GetButtonUp ("Fire1"))
      {
        if (OnUp != null)
          OnUp();
        if (Time.time - my_LastMouseUpTime <
my_DoubleClickTime)
        {
```

```
        if (OnDoubleClick != null)
          OnDoubleClick();
       }
       else
       {
        if (OnClick != null)
          OnClick();
       }
       my_LastMouseUpTime = Time.time;
      }
      if (Input.GetButtonDown("Cancel"))
      {
       if (OnCancel != null)
         OnCancel();
      }
    }
    private SwipeDirection DetectSwipe ()
    {
     Vector2 swipeData = (my_MouseUpPosition -
my_MouseDownPosition).normalized;
     bool swipeIsVertical = Mathf.Abs (swipeData.x) <
my_SwipeWidth;
     bool swipeIsHorizontal = Mathf.Abs(swipeData.y) <
my_SwipeWidth;
     if (swipeData.y > 0f && swipeIsVertical)
       return SwipeDirection.UP;
     if (swipeData.y < 0f && swipeIsVertical)
       return SwipeDirection.DOWN;
     if (swipeData.x > 0f && swipeIsHorizontal)
       return SwipeDirection.RIGHT;
     if (swipeData.x < 0f && swipeIsHorizontal)
       return SwipeDirection.LEFT;
     return SwipeDirection.NONE;
    }
    private SwipeDirection DetectKeyboardEmulatedSwipe
()
    {
     float horizontal = Input.GetAxis ("Horizontal");
     float vertical = Input.GetAxis ("Vertical");
     bool noHorizontalInputPreviously = Mathf.Abs
(my_LastHorizontalValue) < float.Epsilon;
     bool noVerticalInputPreviously =
Mathf.Abs(my_LastVerticalValue) < float.Epsilon;
     my_LastHorizontalValue = horizontal;
     my_LastVerticalValue = vertical;
     if (vertical > 0f && noVerticalInputPreviously)
       return SwipeDirection.UP;
     if (vertical < 0f && noVerticalInputPreviously)
       return SwipeDirection.DOWN;
     if (horizontal > 0f && noHorizontalInputPreviously)
       return SwipeDirection.RIGHT;
     if (horizontal < 0f && noHorizontalInputPreviously)
       return SwipeDirection.LEFT;
     return SwipeDirection.NONE;
    }
    private void OnDestroy()
    {
     OnSwipe = null;
     OnClick = null;
     OnDoubleClick = null;
     OnDown = null;
     OnUp = null;
    }
   }
}
```

VREyeRayCastor.cs used in order to interact with objects in the scene. This class casts a ray into the scene and if it finds a VRInteractiveItem it exposes it for other classes to use. This script should generally be placed on the camera.

```
using System;
using UnityEngine;
namespace VRStandardAssets.Utils
{
 public class VREyeRaycaster : MonoBehaviour
 {
  public event Action<RaycastHit> OnRaycasthit;
  [SerializeField] private Transform my_Camera;
  [SerializeField] private LayerMask
my_ExclusionLayers;
  [SerializeField] private Reticle my_Reticle;
  [SerializeField] private VRInput my_VrInput;
  [SerializeField] private bool my_ShowDebugRay;
  [SerializeField] private float my_DebugRayLength =
50f;
  [SerializeField] private float my_DebugRayDuration =
1f;
  [SerializeField] private float my_RayLength = 5000f;
        [SerializeField] private float autoClickTime = 1f;
  private VRInteractiveItem my_CurrentInteractible;
  private VRInteractiveItem my_LastInteractible;
  public VRInteractiveItem CurrentInteractible
  {
   get { return my_CurrentInteractible; }
  }
  private void OnEnable()
  {
   my_VrInput.OnClick += HandleClick;
   my_VrInput.OnDoubleClick += HandleDoubleClick;
   my_VrInput.OnUp += HandleUp;
   my_VrInput.OnDown += HandleDown;
  }
  private void OnDisable ()
  {
   my_VrInput.OnClick -= HandleClick;
   my_VrInput.OnDoubleClick -= HandleDoubleClick;
   my_VrInput.OnUp -= HandleUp;
   my_VrInput.OnDown -= HandleDown;
  }
  private void Update()
  {
   EyeRaycast();
  }
  private void EyeRaycast()
  {
   if (my_ShowDebugRay)
   {
    Debug.DrawRay( my_Camera.position,
my_Camera.forward * my_DebugRayLength, Color.blue,
my_DebugRayDuration);
   }
```

```
    Ray ray = new Ray( my_Camera.position, my_Camera.forward);
    RaycastHit hit;
    if (Physics.Raycast(ray, out hit, my_RayLength, ~ my_ExclusionLayers))
    {
     VRInteractiveItem interactible = hit.collider.GetComponent<VRInteractiveItem>(); //attempt to get the VRInteractiveItem on the hit object
      my_CurrentInteractible = interactible;
          if (interactible && interactible != my_LastInteractible) {

            interactible.setAutoClickTime (autoClickTime);

            interactible.Over ();

my_Reticle.fillInTime (autoClickTime);
                                    }
      if (interactible != my_LastInteractible)
       DeactiveLastInteractible();
      my_LastInteractible = interactible;
      if (my_Reticle)
       my_Reticle.SetPosition(hit);

      if (OnRaycasthit != null)
       OnRaycasthit(hit);
     }
     else
     {
      DeactiveLastInteractible();
      my_CurrentInteractible = null;
      if (my_Reticle)
       my_Reticle.SetPosition();
     }
   }
   private void DeactiveLastInteractible()
   {
    if (my_LastInteractible == null)
     return;
                              my_Reticle.stopFilling ();
    my_LastInteractible.Out();
    my_LastInteractible = null;
   }
   private void HandleUp()
   {
    if (my_CurrentInteractible != null)
     my_CurrentInteractible.Up();
   }
   private void HandleDown()
   {
    if (my_CurrentInteractible != null)
     my_CurrentInteractible.Down();
   }
   private void HandleClick()
   {
                          Debug.Log ("HandleClick of raycaster");
     if (my_CurrentInteractible != null)
      my_CurrentInteractible.Click();
   }
   private void HandleDoubleClick()
   {
    if (my_CurrentInteractible != null)
     my_CurrentInteractible.DoubleClick();

   }
  }
}
```

This is the setup for the Virtual reality input in the context of an AR camera. Now a sample Image Target is created with default Vuforia_mars database. And under image target, 3D models are created. The interaction with the item will be like our reticle is going over the 3D model, on the 3D model, out the 3D model, click the 3D model, Double-clicking the 3D model. To add these interactions to the 3D model VRInteractiveItem.cs is added. Which will enable the 3D models to be interactive.

```
using System;
using UnityEngine;
namespace VRStandardAssets.Utils
{
 public class VRInteractiveItem : MonoBehaviour
 {
  public event Action OnOver;
  public event Action OnOut;
  public event Action OnClick;
  public event Action OnDoubleClick;
  public event Action OnUp;
  public event Action OnDown;
                public bool autoClick = false;
                private float autoClickTime = 1f;
                private float clickTimerState = 0f;
                private bool clicked = false;
  protected bool m_IsOver;
                private void Update () {
                        if (autoClick && m_IsOver && !clicked) {
                                clickTimerState += Time.deltaTime;
                                if (clickTimerState >= autoClickTime) {
                                        clicked = true;
                                        Click ();
                                }
                        }
                }
  public bool IsOver
  {
   get { return m_IsOver; }
  }

                public void setAutoClickTime(float time) {
                        autoClickTime = time;
                }
  public void Over()
  {
   m_IsOver = true;
```

```
    if (OnOver != null)
      OnOver();
  }
  public void Out()
  {
    m_IsOver = false;
                              clicked = false;
                              clickTimerState = 0f;

    if (OnOut != null)
      OnOut();
  }
  public void Click()
  {
    if (OnClick != null)
      OnClick();
  }
  public void DoubleClick()
  {
    if (OnDoubleClick != null)
      OnDoubleClick();
  }
  public void Up()
  {
    if (OnUp != null)
      OnUp();
  }
  public void Down()
  {
    if (OnDown != null)
      OnDown();
  }
 }
}
```

Now the type of interactions are defined and we need to perform some actions and get the output we desire. To achieve interactions in real-time a new script ExampleInteractiveItem.cs is added. The new script will define ways to deal with the interactive item which is defined by the above script.

```
using System;
using UnityEngine;
using VRStandardAssets.Utils;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
namespace VRStandardAssets.Examples
{
 public class ExampleInteractiveItem : MonoBehaviour
 {
  [SerializeField] private VRInteractiveItem
my_InteractiveItem;
  [SerializeField] private Renderer my_Renderer;
  [SerializeField] private Material my_OverMaterial;
  [SerializeField] private Material my_NormalMaterial;
  [SerializeField] private Material my_ClickedMaterial;
  [SerializeField] private GameObject secondObject;
  private void Awake()
  {
   my_NormalMaterial = my_Renderer.material;
  }
  private void OnEnable()
  {
   Debug.Log("Enabled");
   my_InteractiveItem.OnOver += HandleOver;
   my_InteractiveItem.OnOut += HandleOut;
   my_InteractiveItem.OnClick += HandleClick;
  }
  private void OnDisable()
  {
   Debug.Log("Disabled");
   my_InteractiveItem.OnOver -= HandleOver;
   my_InteractiveItem.OnOut -= HandleOut;
   my_InteractiveItem.OnClick -= HandleClick;
  }
  private void HandleOver()
  {
   my_Renderer.material = my_OverMaterial;
   Debug.Log("Showed over state");
  }
  private void HandleOut()
  {
   my_Renderer.material = my_NormalMaterial;
   Debug.Log("Showed out state");
  }
  private void HandleClick()
  {
   my_Renderer.material = my_ClickedMaterial;
   Debug.Log("Showed click state");
   secondObject.SetActive(true);

  }
 }
}
```

These files are the basic files for the development of any fusion reality application. The files together with assets and demo scenes are integrated to create the software development kit.

B. Chat Bot integrated Website Development.

The website is created with HTML and CSS. Using HTML the front end of the website script is created. Using CSS the styling of the HTML elements is done.

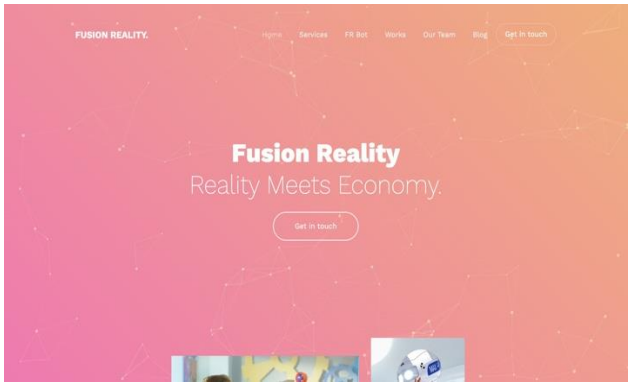


Fig 3:- Fusion Reality Website Home Page

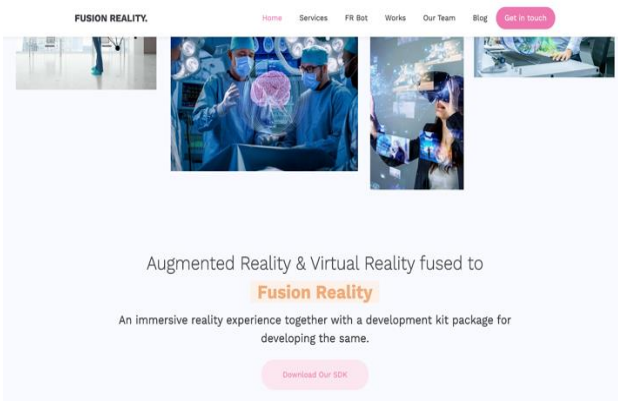


Fig 4:- Fusion Reality Website Download Page

A chatbot is created with the Dialog Flow platform of google. In the chatbot intents and entities are created to solve any doubts and queries. These possible queries are given as input to the bot and using which the bot is trained. After the bot is made using the integration module of google, the chatbot is integrated into the website and telegram.

The chatbot is available at the website and can solve any queries that the developer has in the context of fusion reality application development using our software development kit.

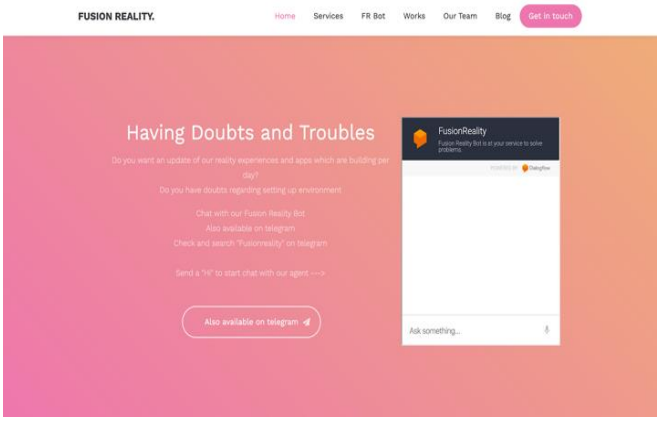


Fig 5:- Fusion Reality Website Chatbot Page

C. Fusion Reality - Application built with Fusion Reality Kit.

Fusion Reality is built with the latest technologies and tools associated with the fusion reality SDK and assets which are capable of augmenting the periodic table and elements with details of each particular element having its basic properties with isotopes, compounds, and video displays to learn or experience periodic table like never before. This is built by downloading the software development kit from the website which is developed in the first phase. After downloading the files imported to the new project in unity.

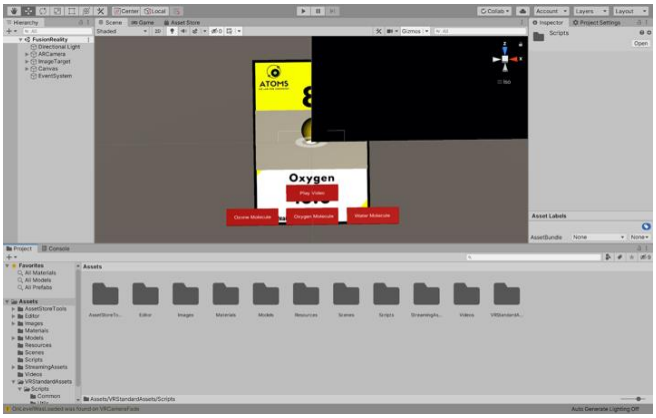


Fig 6:- Unity After uploading files

In vuforia a developer account is created and a new license key created. The license key is pasted into the key column in the Inspector menu of AR Camera. A new database of periodic elements markers created using canva which is a web application to create posters and images.

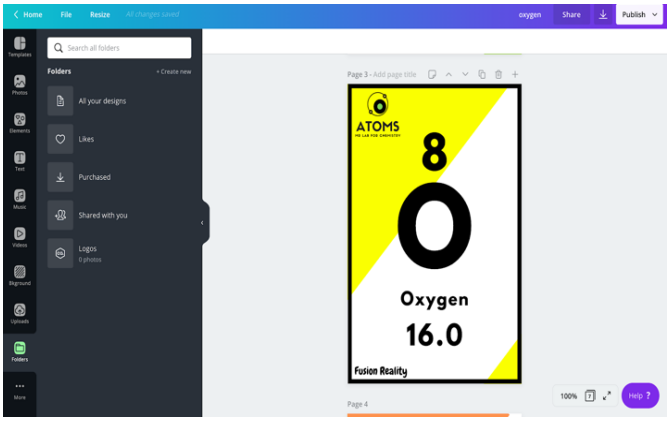


Fig 7:- Oxygen marker created in canva

The database is downloaded from the vuforia website and imported to unity.

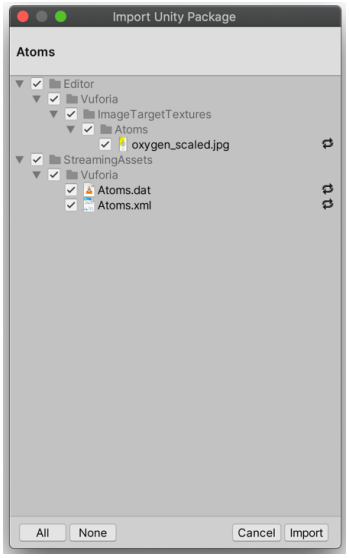


Fig 8:- Importing Image Targets

A new image target is created and the target is selected from a new database. One of the image target oxygen is used. Under the image target, the cubes are added for interacting with particular elements. Each cube acts as buttons and the ExampleInteractiveItem.cs is modified to the current use and scripts are added to the cubes. A cube with enabling oxygen molecule 3D model added with oxygen.cs and others with similar code.

```
using System;
using UnityEngine;
using VRStandardAssets.Utils;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

namespace VRStandardAssets.Examples
{
  public class oxygen : MonoBehaviour
  {
    [SerializeField] private VRInteractiveItem m_InteractiveItem;
    [SerializeField] private Renderer m_Renderer;
    [SerializeField] private Material m_OverMaterial;
    [SerializeField] private Material m_NormalMaterial;
    [SerializeField] private Material m_ClickedMaterial;
    [SerializeField] private GameObject renderObject,removeObject1,removeObject2,removeObject3;

    private void Awake()
    {
      m_NormalMaterial = m_Renderer.material;
    }

    private void OnEnable()
    {
      Debug.Log("Enable");
      m_InteractiveItem.OnOver += HandleOver;
      m_InteractiveItem.OnOut += HandleOut;
      m_InteractiveItem.OnClick += HandleClick;
    }

    private void OnDisable()
    {
      Debug.Log("Disable");
      m_InteractiveItem.OnOver -= HandleOver;
      m_InteractiveItem.OnOut -= HandleOut;
      m_InteractiveItem.OnClick -= HandleClick;
    }

    private void HandleOver()
    {
      m_Renderer.material = m_OverMaterial;
      Debug.Log("Show over state");
    }
    private void HandleOut()
    {
      m_Renderer.material = m_NormalMaterial;
      Debug.Log("Show out state");
    }
    private void HandleClick()
    {
      m_Renderer.material = m_ClickedMaterial;
      Debug.Log("Show click state");
      renderObject.SetActive(true);
      removeObject1.SetActive(false);
      removeObject2.SetActive(false);
      removeObject3.SetActive(false);

    }
  }
}
```

Fig 9:- Fusion Reality Application icon on Android

Fig 10:- Earth View Using Fusion Reality

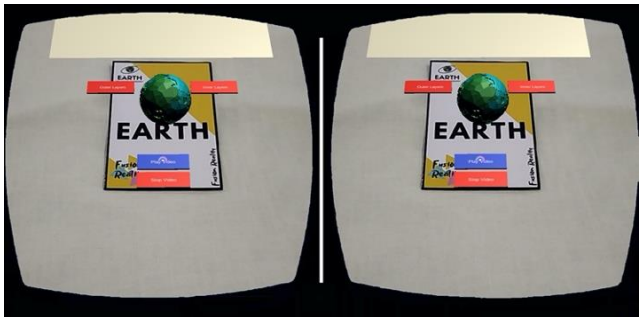


Fig 11:- Clicked Button Colour in Fusion Reality

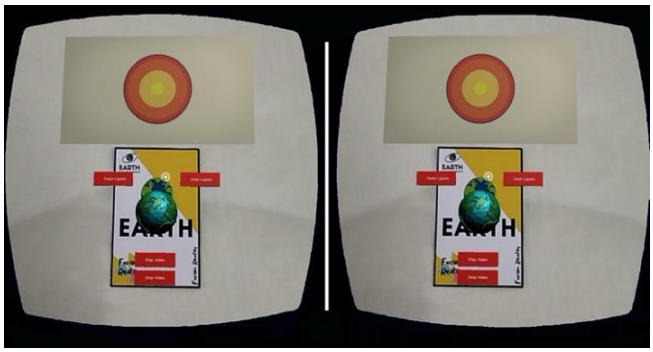


Fig 12:- Watching Video of Earth Using Fusion Reality

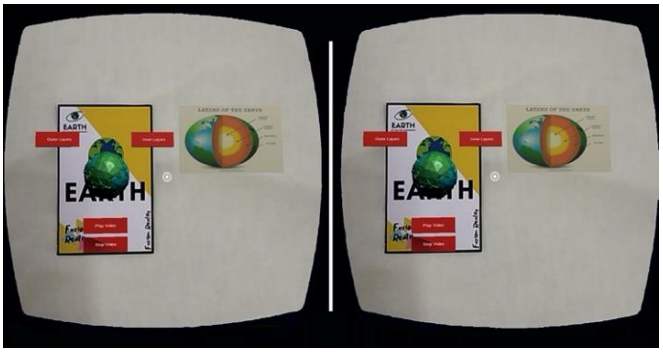


Fig 13:- Viewing Inner Layers of Earth Using Fusion Reality

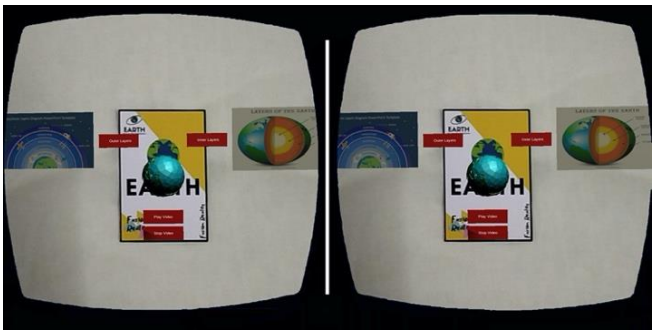


Fig 14:- Viewing Inner and Outer Layers of Earth Using Fusion Reality

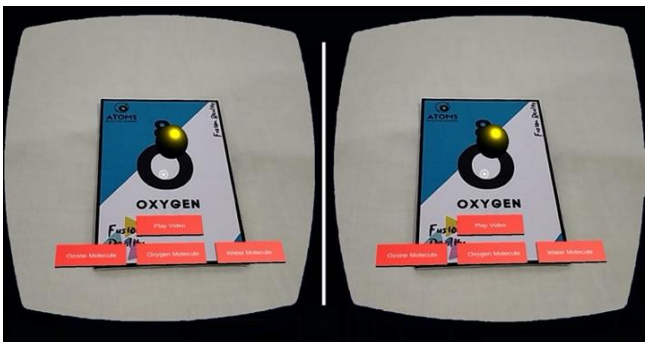


Fig 15:- Viewing Oxygen Molecule Using Fusion Reality

Fig 16:- Viewing Ozone Molecule Using Fusion Reality

Fig 17:- Viewing Water Molecule Using Fusion Reality

Fig 18:- Viewing Oxygen Video Using Fusion Reality

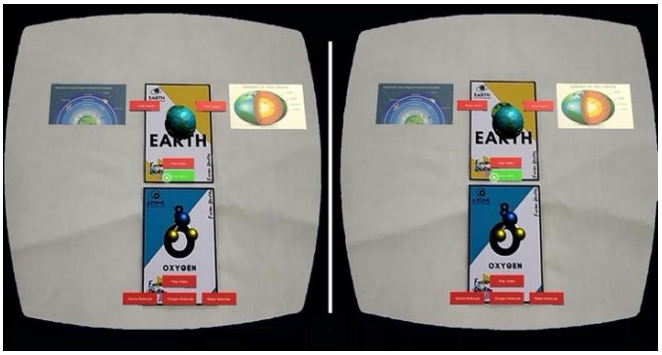


Fig 19:- Combined View of Earth and Oxygen Using Fusion Reality

V. HARDWARE SPECIFICATION

The hardware components are Low-cost virtual reality headset and Android mobile device.

Fig 20:- Virtual Reality Headset

ISSN No:-2456-2165

A virtual reality headset is a head-mounted device which provides virtual reality for the user. Virtual reality headsets commonly used with games but they and in other applications, including simulators and trainers. Some VR headsets have eye tracking sensors and gaming controllers. They includes a stereoscopic stereo sound, head mounted display, and also head motion tracking sensors. The lenses of the headset help for mapping the up-close display to a wide field of view, while also providing a more comfortable distant point of focus.

Fig 21:- Android device

Android is a Linux based OS (operating system) which is designed for touch screens mobile devices such as smartphones and tablet computers. The operating system has developed a lot in the past fifteen years starting from black and white phones to recent smartphones or mini computers. One of the most commonly used mobile operating system these days is android. Android is a powerful operating system and it supports a large number of softwares in Smartphones. These softwares are more user friendly and advanced for users. The hardware supports for android software is ARM architecture platform. The android is an open-source operating system which means it's free and anyone can use it.

VI. SOFTWARE SPECIFICATION:

Softwares used are Unity 3D, Visual Studio Code, Vuforia Plugin.

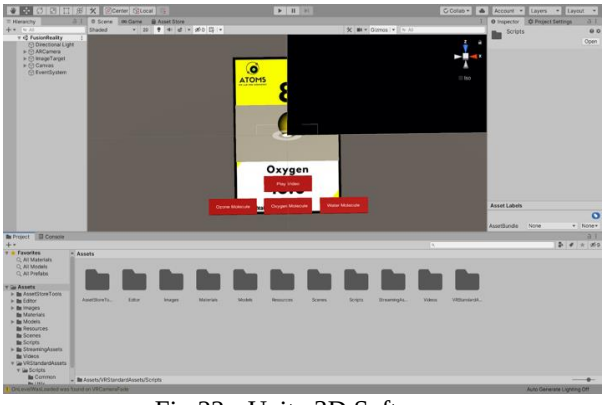
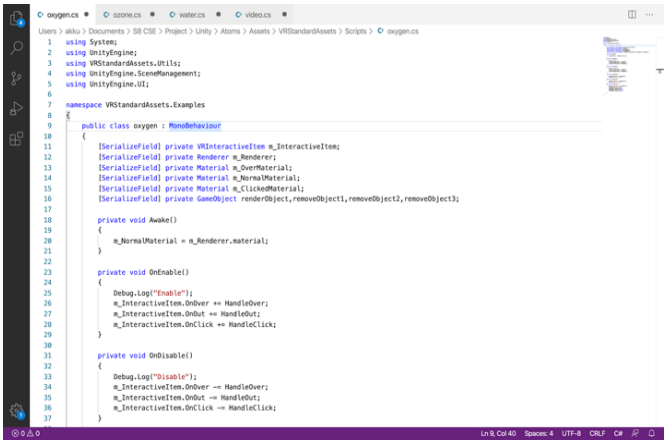


Fig 22:- Unity 3D Software

Unity is a cross platform game engine which supports large number of platforms developed by Unity Technologies, is used to design and integrate the toolkit developed. The engine can be used to create 3D, 2D, VR(virtual reality), and AR(augmented reality) games, as well as simulations and also other experiences.



```
using System;
using UnityEngine;
using VRStandardAssets.Utils;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

namespace VRStandardAssets.Examples
{
    public class oxygen : MonoBehaviour
    {
        [SerializeField] private VRInteractiveItem m_InteractiveItem;
        [SerializeField] private Renderer m_Renderer;
        [SerializeField] private Material m_OverMaterial;
        [SerializeField] private Material m_NormalMaterial;
        [SerializeField] private Material m_ClickedMaterial;
        [SerializeField] private GameObject renderObject,removeObject1,removeObject2,removeObject3;

        private void Awake()
        {
            m_NormalMaterial = m_Renderer.material;
        }

        private void OnEnable()
        {
            Debug.Log("Enable");
            m_InteractiveItem.OnOver += HandleOver;
            m_InteractiveItem.OnOut += HandleOut;
            m_InteractiveItem.OnClick += HandleClick;
        }

        private void OnDisable()
        {
            Debug.Log("Disable");
            m_InteractiveItem.OnOver -= HandleOver;
            m_InteractiveItem.OnOut -= HandleOut;
            m_InteractiveItem.OnClick -= HandleClick;
        }
```

Fig 23:- Visual Studio Code Software

A highly modifiable source code editor. Visual Studio code is very useful for debugging, embedded Git control and GitHub, syntax highlighting. The users allowed to change the keyboard shortcuts, preferences, theme, and install extensions that add additional functionality.The compiled binaries are freeware for any purposes. The source code is free and also open-source, released under the permissive MIT License.

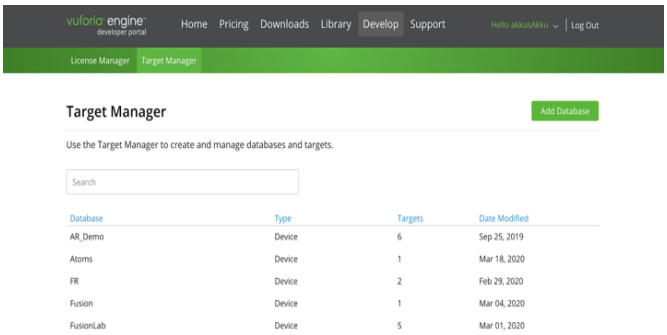


Fig 24:- Vuforia Web App

Vuforia is an augmented reality software development kit) used for computer vision technology and allows the production of augmented reality applications. It can identify and track planar pictures and 3D objects in real-time. When viewed within the camera of mobile the image registration ability enables developers to position and orient virtual objects, such as 3D figures and other media, about real-world objects. The position and orientation of the image in real-time are traced by the virtual object so that the viewer's perspective on the object corresponds with the appearance on the target and appears to be a part of the real-world display.

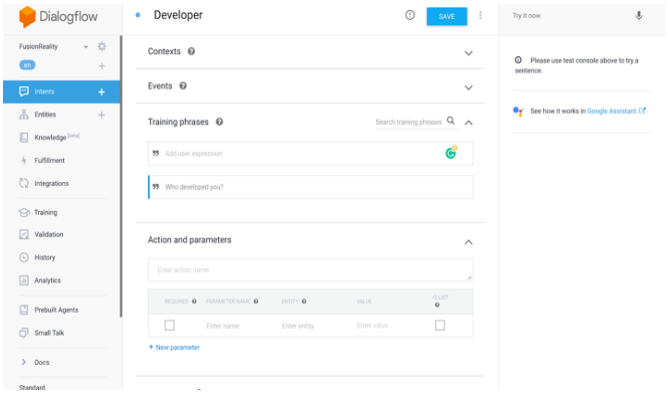


Fig 25:- Dialog Flow Web App

The FR Bot made accessible to the user within the website and telegram which is a chatbot created on a platform called DialogFlow. DialogFlow is a platform related to design and combine conversational user interfaces in different mobiles, devices, web applications, and chatbots which is a natural language recognition platform. Various intents for user interface are created and are made as many users as effectively as possible. The user can interact with the chatbot and post or ask his questions and doubts. The SDK's contain voice recognition, natural language recognition, and text-to-speech. api.ai grants a web interface to create and test conversation situations.

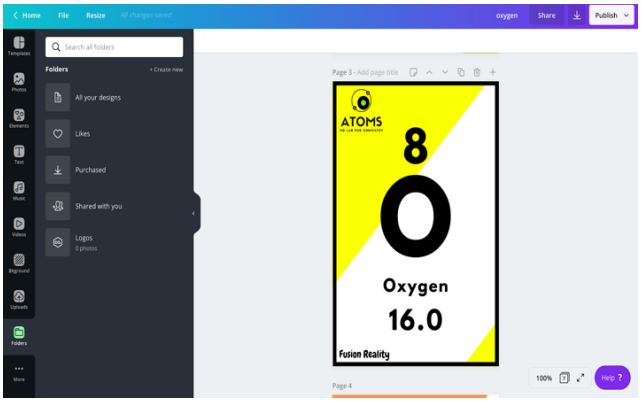


Fig 26:- Canva Web App

VII. CONCLUSION

Technology is changing rapidly. Most people are using modern technology to do various activities. Life has become more convenient and enjoyable. You will realize that the recent development of technology has made it possible for us to lead more comfortable lives.

In this era where technology is advancing each second, the devices and humans must be technically efficient. The project aims to develop an architecture which incorporates various technologies. This project deals with creating an architecture which combines different technologies.

The Fusion Reality is a layered architecture Incorporates Reality and Artificial Intelligence technologies into daily lives so that our routines are made technically advanced. The project aims to develop an architecture incorporating various technologies like AR, VR, MR into a single platform. It also build an SDK which can be used to develop any Fusion Reality applications that run on any system to perform various tasks including daily routines. This makes our daily lives technically advanced and promising.

The scope of the fusion reality kit is much more. All the existing augmented reality and virtual reality applications can also be modified with a fusion reality kit to make it more understandable and productive. Developers can integrate fusion reality in various fields including Geography, Biology, Architecture, Medical, etc. to make these fields more understandable and easier to learn.

FUTURE SCOPE

We have successfully developed a Fusion Reality software development kit, and tested it's working . This project can be expanded to replace live target tracking mechanism to object and space tracking mechanism so that we may not require any target images for the augmentation. It can also be extended by incorporating machine learning to do facial recognition, object recognition, live target selection etc. We successfully developed an application using this development kit "ATOM", like Atom we can develop more complicated applications like AUTOCAD which enables users to design modules using their hand in live 3D word instead of monitors and keyboards .In such application multiple users can develop on the same 3D project at same time thus reducing the time and increasing efficiency of their project.

REFERENCES

[1]. Krichenbauer, Max & Yamamoto, Goshiro & Taketomi, Takafumi & Sandor Christian & Kato, Hirokazu. (2017). Augmented Reality vs Virtual Reality for 3D Object Manipulation. IEEE Transactions on Visualization and Computer Graphics. PP. 1-1. 10.1109/TVCG.2017.2658570.

[2]. T. Blum, V. Kleeberger, C. Bichlmeier, N. Navab, "Mirracle: An Augmented Reality Magic Mirror System for Anatomy Education", Proc. IEEE Virtual Reality Short Papers and Posters (VRW), pp. 115-116, Mar. 2012.

[3]. Krummenacher, G.; Ong, C.S.; Koller, S.; Kobayashi, S.; Buhmann, J.M. Wheel Defect Detection with Machine Learning. IEEE Trans. Intell. Transp. Syst. 2017

[4]. B. Luo, H. Wang, H. Liu, B. Li, F. Peng, "Early fault detection of machine tools based on deep learning and dynamic identification", IEEE Trans. Ind. Electron., vol. 66, no. 1, pp. 509-518, Jan. 2019

[5]. Li, Gang & Liu, Yuanan & Wang, Yongtian. (2017). Evaluation of labelling layout methods in augmented reality. 351-352. 10.1109/VR.2017.7892321.

[6]. Agrawal, Diptanshu & Mane, S. & Pacharne, Apoorva & Tiwari, Snehal. (2018). IoT Based Augmented Reality System of Human Heart: An Android Application. 899-902. 10.1109/ICOEI.2018.8553807.

[7]. Ju, Xiaohui & Yang, Feng & Liang, Daojun. (2019). LightNets: The Concept of Weakening Layers. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2923983.
[8]. Leibe, Bastian & Cornelis, Nico & Van Gool, Luc. (2008). Coupled Object Detection and Tracking from Static Cameras and Moving Vehicles. IEEE transactions on pattern analysis and machine intelligence. 30. 1683-98.10.1109/TPAMI.2008.170.
[9]. Karunasekera, Hasith & Wang, Han & Zhang, Handuo. (2019). Multiple Object Tracking with attention to Appearance, Structure, Motion and Size. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2932301.
[10]. Hussein, Sarfaraz & Kandel, Pujan & Bolan, Candice & Wallace, Michael & Bagci, Ulas. (2019). Lung and Pancreatic Tumor Characterization in the Deep Learning Era: Novel Supervised and Unsupervised Learning Approaches. IEEE Transactions on Medical Imaging. PP. 1-1. 10.1109/TMI.2019.2894349.
[11]. Srinivasan, Srinikethan & Truong-Huu, Tram & Gurusamy, Mohan. (2018). Machine Learning-based Link Fault Identification and Localization in Complex Networks.