DAFTAR LAMPIRAN

Lampiran 1. Script Coding

1. LoadingBarManager.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class LoadingBarManager : MonoBehaviour
{
    SceneHandler sceneHandler;

    public float speed;
    public Slider loadingBar;

    // Start is called before the first frame update
    void Start()
    {
        sceneHandler = SceneHandler.instance;
        loadingBar.value = 0;
    }

    // Update is called once per frame
    void Update()
    {
        loadingBar.value += Time.deltaTime * speed;

        if(loadingBar.value >= 0.99)
        {
            sceneHandler.GoToScene("Menu");
        }
    }
}
```

2. SceneHandler.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneHandler : MonoBehaviour
{
    public static SceneHandler instance;

    private void Awake()
    {
        instance = this;
    }

    public void GoToScene(string name)
    {
        SceneManager.LoadScene(name);
    }

    public void ExitApps()
    {
        Application.Quit();
    }
}
```

3. InfoDataAR.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using System;
using UnityEngine.Events;

public enum TRACK_STATE
{
    SCANMARKER,
    ININFO,
    INIMAGE,
    INVIDEO
}

public class InfoDataAR : MonoBehaviour
{
    public static InfoDataAR Instance;

    [SerializeField]
    TRACK_STATE _currentTrackState =
TRACK_STATE.SCANMARKER;

    public ContentInfo kontenInfo;
    public ImageFotoHandler kontenFoto;

public GameObject panelInfo, panelGambar,panelVideo;
public GameObject btnInfo, btnIMG,btnVideo;

public TextMeshProUGUI txt_Ruangan;

bool _isOnTrackObject = false;

public bool _isCameraFoundTarget { get; set; }

[HideInInspector]
public UnityEvent _currentTrackableEvent;

int currentIndexTracker = 0;


public bool GetIsOnTrackObject
{
    get
    {
        return _isOnTrackObject;
    }
}

public TRACK_STATE GetCurrentTrackState
{
    get
    {
        return _currentTrackState;
    }
}

private void Awake()
{
    if (Instance == null)
        Instance = this;
    else
        Destroy(this);
}

void Start()
{
    _currentTrackState = TRACK_STATE.SCANMARKER;

    panelInfo.SetActive(false);
    panelGambar.SetActive(false);

    btnInfo.SetActive(false);
```

```
        btnIMG.SetActive(false);
        btnVideo.SetActive(false);
        panelVideo.SetActive(false);
    }


    public void PanelAwal(string name)
    {
        _currentTrackState = TRACK_STATE.SCANMARKER;
        panelInfo.SetActive(false);
        panelGambar.SetActive(false);

        btnInfo.SetActive(false);

        btnIMG.SetActive(false);
        btnVideo.SetActive(false);


        txt_Ruangan.text = name;

    }

    public void OnLostTarget()
    {

        btnInfo.SetActive(false);

        btnIMG.SetActive(false);

        btnVideo.SetActive(false);

        txt_Ruangan.text = "SCAN MARKER";
    }

    public void IndexMarker(int index)
    {
        kontenInfo.index = index;
        kontenFoto.indexRuangan = index;
        currentIndexTracker = index;
    }

    public void IsHaveInfo(bool isInfo)
    {
        btnInfo.SetActive(isInfo);

        btnIMG.SetActive(true);
```

```
            btnVideo.SetActive(true);
        }


        public void ShowPanelinfo()
        {
            _currentTrackState = TRACK_STATE.ININFO;
            _isOnTrackObject = true;
            panelInfo.SetActive(true);
            kontenInfo.Info();
            panelGambar.SetActive(false);
            panelVideo.SetActive(false);
        }

        public void ShowPanelGambar()
        {
            _currentTrackState = TRACK_STATE.INIMAGE;
            _isOnTrackObject = true;
            panelGambar.SetActive(true);
            kontenFoto.ChangeFoto(0);
            panelInfo.SetActive(false);
            panelVideo.SetActive(false);
        }

        public void ShowPanelVideo()
        {

    if(VideoPlayerController.Instance.databaseClipVideo[currentIndexT
    racker] != null)
            {
                _isOnTrackObject = true;
                _currentTrackState = TRACK_STATE.INVIDEO;
                panelVideo.SetActive(true);
                panelGambar.SetActive(false);
                panelInfo.SetActive(false);

    VideoPlayerController.Instance.ShowVideo(currentIndexTracker);
            }
        }

        public void ButtonBackPressed()
        {
            if (_currentTrackState != TRACK_STATE.SCANMARKER)
            {
                _currentTrackState = TRACK_STATE.SCANMARKER;
                if (!_isCameraFoundTarget)
                    OnLostTarget();
                else
                    _currentTrackableEvent.Invoke();

                _isOnTrackObject = false;
                panelInfo.SetActive(false);
                panelGambar.SetActive(false);
                panelVideo.SetActive(false);
            }
            else
            {
                SceneHandler.instance.GoToScene("Menu");
            }
        }

}
```

4. ContentInfo.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ContentInfo : MonoBehaviour
{
    public static ContentInfo instance;

    public ScrollRect scrollRect;
    public int index;
    public GameObject[] imgInfo;



    private void Awake()
    {
        instance = this;
    }

    // Start is called before the first frame update
    void Start()
    {
        foreach (GameObject info in imgInfo)
        {
            info.SetActive(false);
        }
        switch (index)
        {
            case 0:
                scrollRect.content =
imgInfo[0].GetComponent<RectTransform>();
                imgInfo[0].SetActive(true);
                break;

            case 5:
                scrollRect.content =
imgInfo[1].GetComponent<RectTransform>();
                imgInfo[1].SetActive(true);
                break;

            case 6:
                scrollRect.content =
imgInfo[2].GetComponent<RectTransform>();
                imgInfo[2].SetActive(true);
                break;

            case 7:
                scrollRect.content =
imgInfo[3].GetComponent<RectTransform>();
                imgInfo[3].SetActive(true);
                break;

            case 9:
                scrollRect.content =
imgInfo[4].GetComponent<RectTransform>();
                imgInfo[4].SetActive(true);
                break;

            case 20:
                scrollRect.content =
imgInfo[5].GetComponent<RectTransform>();
                imgInfo[5].SetActive(true);
                break;

            case 21:
                scrollRect.content =
imgInfo[6].GetComponent<RectTransform>();
                imgInfo[6].SetActive(true);
                break;

            case 22:
                scrollRect.content =
imgInfo[7].GetComponent<RectTransform>();
                imgInfo[7].SetActive(true);
                break;

            case 23:
                scrollRect.content =
imgInfo[8].GetComponent<RectTransform>();
                imgInfo[8].SetActive(true);
                break;

            case 24:
                scrollRect.content =
imgInfo[9].GetComponent<RectTransform>();
                imgInfo[9].SetActive(true);
                break;

            case 25:
                scrollRect.content =
imgInfo[10].GetComponent<RectTransform>();
                imgInfo[10].SetActive(true);
                break;

            case 26:
                scrollRect.content =
imgInfo[11].GetComponent<RectTransform>();
                imgInfo[11].SetActive(true);
                break;

            case 27:
                scrollRect.content =
imgInfo[12].GetComponent<RectTransform>();
                imgInfo[12].SetActive(true);
                break;

            case 28:
                scrollRect.content =
imgInfo[13].GetComponent<RectTransform>();
                imgInfo[13].SetActive(true);
                break;

            case 29:
                scrollRect.content =
imgInfo[14].GetComponent<RectTransform>();
                imgInfo[14].SetActive(true);
                break;
            case 30:
                scrollRect.content =
imgInfo[15].GetComponent<RectTransform>();
                imgInfo[15].SetActive(true);
                break;

            case 31:
                scrollRect.content =
imgInfo[16].GetComponent<RectTransform>();
                imgInfo[16].SetActive(true);
                break;

            case 32:
                scrollRect.content =
imgInfo[17].GetComponent<RectTransform>();
                imgInfo[17].SetActive(true);
                break;
        }
    }

    // Update is called once per frame
    void Update()
    {

    }

    public void Info()
    {
        foreach (GameObject info in imgInfo)
        {
            info.SetActive(false);
        }

        switch (index)
        {
            case 0:
                scrollRect.content =
imgInfo[0].GetComponent<RectTransform>();
                imgInfo[0].SetActive(true);
                break;

            case 5:
                scrollRect.content =
imgInfo[1].GetComponent<RectTransform>();
                imgInfo[1].SetActive(true);
                break;
            case 6:
                scrollRect.content =
imgInfo[2].GetComponent<RectTransform>();
                imgInfo[2].SetActive(true);
                break;

            case 7:
                scrollRect.content =
imgInfo[3].GetComponent<RectTransform>();
                imgInfo[3].SetActive(true);
                break;

            case 9:
                scrollRect.content =
imgInfo[4].GetComponent<RectTransform>();
                imgInfo[4].SetActive(true);
                break;

            case 20:
                scrollRect.content =
imgInfo[5].GetComponent<RectTransform>();
                imgInfo[5].SetActive(true);
                break;

            case 21:
                scrollRect.content =
imgInfo[6].GetComponent<RectTransform>();
                imgInfo[6].SetActive(true);
                break;

            case 22:
                scrollRect.content =
imgInfo[7].GetComponent<RectTransform>();
                imgInfo[7].SetActive(true);
                break;

            case 23:
                scrollRect.content =
imgInfo[8].GetComponent<RectTransform>();
                imgInfo[8].SetActive(true);
                break;

            case 24:
                scrollRect.content =
imgInfo[9].GetComponent<RectTransform>();
                imgInfo[9].SetActive(true);
                break;
            case 25:
                scrollRect.content =
imgInfo[10].GetComponent<RectTransform>();
                imgInfo[10].SetActive(true);
                break;

            case 26:
                scrollRect.content =
imgInfo[11].GetComponent<RectTransform>();
                imgInfo[11].SetActive(true);
                break;

            case 27:
                scrollRect.content =
imgInfo[12].GetComponent<RectTransform>();
                imgInfo[12].SetActive(true);
                break;

            case 28:
                scrollRect.content =
imgInfo[13].GetComponent<RectTransform>();
                imgInfo[13].SetActive(true);
                break;

            case 29:
                scrollRect.content =
imgInfo[14].GetComponent<RectTransform>();
                imgInfo[14].SetActive(true);
                break;

            case 30:
                scrollRect.content =
imgInfo[15].GetComponent<RectTransform>();
                imgInfo[15].SetActive(true);
                break;

            case 31:
                scrollRect.content =
imgInfo[16].GetComponent<RectTransform>();
                imgInfo[16].SetActive(true);
                break;

            case 32:
                scrollRect.content =
imgInfo[17].GetComponent<RectTransform>();
                imgInfo[17].SetActive(true);
```

```
                    break;
                }
            }
}
```

5. ImageFotoHandler.cs

```
        using System.Collections;
        using System.Collections.Generic;
        using UnityEngine;
        using UnityEngine.UI;

        public class ImageFotoHandler : MonoBehaviour
        {
            public int indexRuangan;
            public Image imgFoto;
            public Sprite[] foto;

            int fotoIndex;

            // Start is called before the first frame update
            void Start()
            {
                imgFoto.sprite = foto[0];
                ChangeFoto(0);
            }

            public void ChangeFoto(int index)
            {


                fotoIndex += index;

                if(fotoIndex >= 2)
                {
                    fotoIndex = 2;
                }
                else if (fotoIndex <= 0)
                {
                    fotoIndex = 0;
                }

                imgFoto.sprite = foto[fotoIndex + (indexRuangan*3)];
            }


}
```

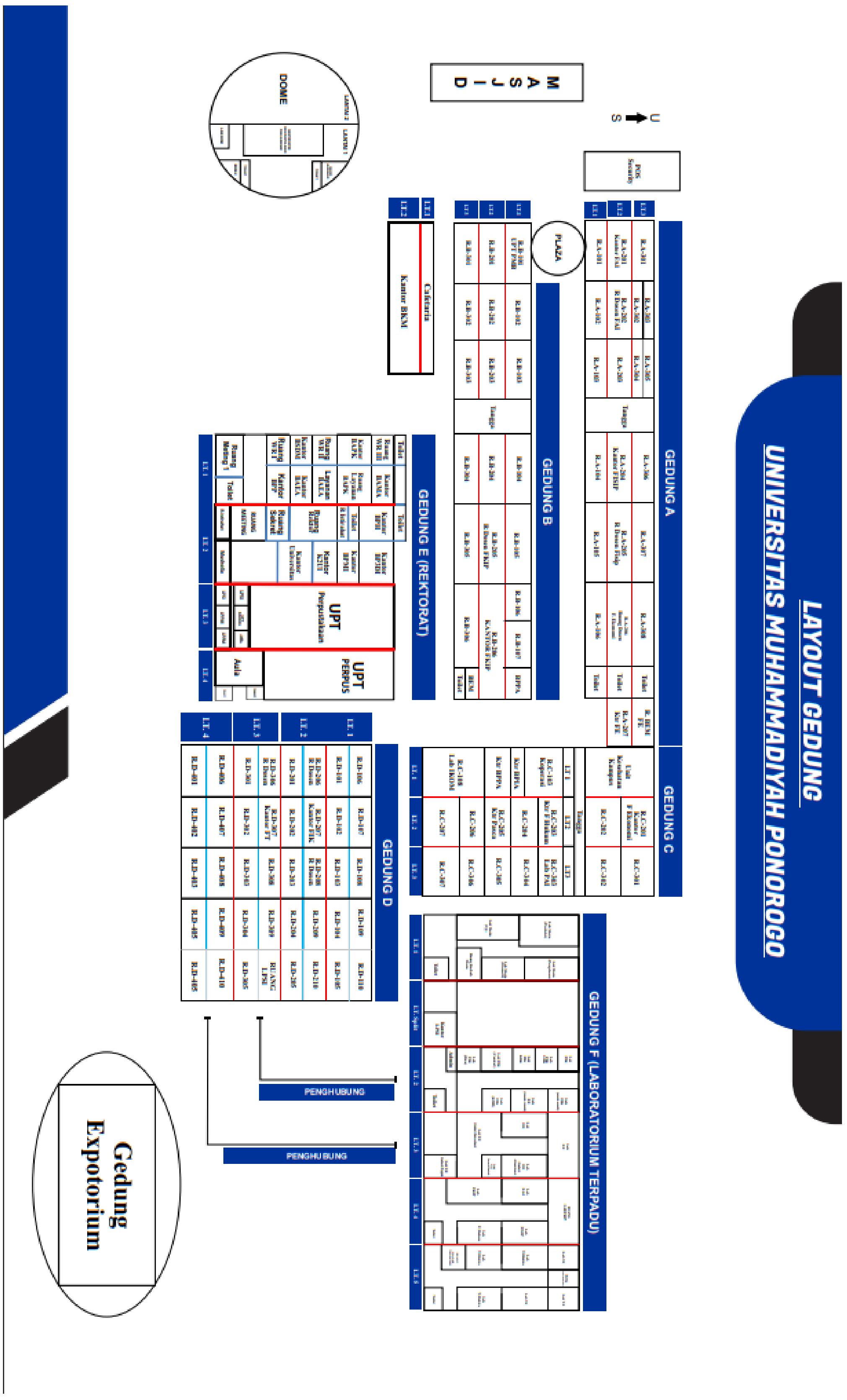
6. VideoPlayerController.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.Video;
using UniRx;

public class VideoPlayerController : MonoBehaviour
{
    public static VideoPlayerController Instance;

    [SerializeField]
    VideoPlayer _videoPlayer;

    [SerializeField]
    Button _buttonPlayVideo;

    [SerializeField]
    Button _buttonPauseVideo;

    [SerializeField]
    Image _fillVideoDuration;

    public VideoClip[] databaseClipVideo;

    private void Awake()
    {
        if (Instance == null)
            Instance = this;
        else
            Destroy(this);
    }


    private void Update()
    {
        if (InfoDataAR.Instance.GetCurrentTrackState ==
TRACK_STATE.INVIDEO)
        {
            _fillVideoDuration.fillAmount = (float)_videoPlayer.time /
(float)_videoPlayer.length;

            if((int)_videoPlayer.time >= (int)_videoPlayer.length)
            {
                _videoPlayer.time = 0;
                _videoPlayer.Pause();
                _buttonPlayVideo.gameObject.SetActive(true);
            }
        }
    }

    private void OnEnable()
    {
        _buttonPlayVideo.onClick.AddListener(() =>
        {
            _videoPlayer.Play();
            _buttonPlayVideo.gameObject.SetActive(false);
        });
        _buttonPauseVideo.onClick.AddListener(() => {
            _videoPlayer.Pause();
            _buttonPlayVideo.gameObject.SetActive(true);
            });
    }

    public void ShowVideo(int index)
    {
        _videoPlayer.clip = databaseClipVideo[index];
        _videoPlayer.time = 0;
        _videoPlayer.Play();
        Observable.Timer(System.TimeSpan.FromSeconds(0.2f))
            .Subscribe(delay =>
                    {
                        _videoPlayer.Pause();
                    }).AddTo(this);
                _buttonPlayVideo.gameObject.SetActive(true);
            }
}
```

7. DefaultTrackableEventHandler.cs

```
using UnityEngine;
using UnityEngine.Events;
using Vuforia;

public class DefaultTrackableEventHandler : MonoBehaviour
{
    public enum TrackingStatusFilter
    {
        Tracked,
        Tracked_ExtendedTracked,
        Tracked_ExtendedTracked_Limited
    }

public TrackingStatusFilter StatusFilter =
TrackingStatusFilter.Tracked_ExtendedTracked_Limited;
    public UnityEvent OnTargetFound;
    public UnityEvent OnTargetLost;



    protected TrackableBehaviour mTrackableBehaviour;
    protected TrackableBehaviour.Status m_PreviousStatus;
    protected TrackableBehaviour.Status m_NewStatus;
    protected bool m_CallbackReceivedOnce = false;

    protected virtual void Start()
    {
        mTrackableBehaviour = GetComponent<TrackableBehaviour>();

        if (mTrackableBehaviour)
        {

mTrackableBehaviour.RegisterOnTrackableStatusChanged(OnTrackableSta
tusChanged);
        }
    }

    protected virtual void OnDestroy()
    {
        if (mTrackableBehaviour)
        {

mTrackableBehaviour.UnregisterOnTrackableStatusChanged(OnTrackable
StatusChanged);
        }
    }

    void
OnTrackableStatusChanged(TrackableBehaviour.StatusChangeResult
statusChangeResult)
    {
        m_PreviousStatus = statusChangeResult.PreviousStatus;
        m_NewStatus = statusChangeResult.NewStatus;

            HandleTrackableStatusChanged();
                }
    protected virtual void HandleTrackableStatusChanged()
    {
        if (!ShouldBeRendered(m_PreviousStatus) &&
            ShouldBeRendered(m_NewStatus))
        {
            OnTrackingFound();
        }
        else if (ShouldBeRendered(m_PreviousStatus) &&
                 !ShouldBeRendered(m_NewStatus))
        {
            OnTrackingLost();
        }
        else
        {
            if (!m_CallbackReceivedOnce &&
!ShouldBeRendered(m_NewStatus))
            {
                // This is the first time we are receiving this callback, and
the target is not visible yet.
                // --> Hide the augmentation.
                OnTrackingLost();
            }
        }

        m_CallbackReceivedOnce = true;
    }

    protected bool ShouldBeRendered(TrackableBehaviour.Status
status)
    {
        if (status == TrackableBehaviour.Status.DETECTED ||
            status == TrackableBehaviour.Status.TRACKED)
        {
            // always render the augmentation when status is
DETECTED or TRACKED, regardless of filter
            InfoDataAR.Instance._isCameraFoundTarget = true;
            return true;
        }

        if (StatusFilter ==
TrackingStatusFilter.Tracked_ExtendedTracked)
        {
            if (status ==
TrackableBehaviour.Status.EXTENDED_TRACKED)
            {
                // also return true if the target is extended tracked
                InfoDataAR.Instance._isCameraFoundTarget = false;
                return false;
            }
        }

        if (StatusFilter ==
TrackingStatusFilter.Tracked_ExtendedTracked_Limited)
        {
            if (status ==
TrackableBehaviour.Status.EXTENDED_TRACKED ||
                status == TrackableBehaviour.Status.LIMITED)
            {
                // in this mode, render the augmentation even if the target's
tracking status is LIMITED.
                // this is mainly recommended for Anchors.
                InfoDataAR.Instance._isCameraFoundTarget = false;
                return false;
            }
        }

        return false;
    }

    protected virtual void OnTrackingFound()
    {


        if (InfoDataAR.Instance.GetIsOnTrackObject)
        {
            InfoDataAR.Instance._currentTrackableEvent =
OnTargetFound;
            return;
        }

        if (mTrackableBehaviour)
        {
            var rendererComponents =
mTrackableBehaviour.GetComponentsInChildren<Renderer>(true);
            var colliderComponents =
mTrackableBehaviour.GetComponentsInChildren<Collider>(true);
            var canvasComponents =
mTrackableBehaviour.GetComponentsInChildren<Canvas>(true);

            // Enable rendering:
            foreach (var component in rendererComponents)
                component.enabled = true;
            // Enable colliders:
            foreach (var component in colliderComponents)
                component.enabled = true;

            // Enable canvas':
            foreach (var component in canvasComponents)
                component.enabled = true;
        }

        if (OnTargetFound != null)
            OnTargetFound.Invoke();
    }

    protected virtual void OnTrackingLost()
    {
        if (InfoDataAR.Instance.GetIsOnTrackObject)
            return;

        if (mTrackableBehaviour)
        {
            var rendererComponents =
mTrackableBehaviour.GetComponentsInChildren<Renderer>(true);
            var colliderComponents =
mTrackableBehaviour.GetComponentsInChildren<Collider>(true);
            var canvasComponents =
mTrackableBehaviour.GetComponentsInChildren<Canvas>(true);

            // Disable rendering:
            foreach (var component in rendererComponents)
                component.enabled = false;
                // Disable colliders:
                foreach (var component in colliderComponents)
                    component.enabled = false;

                // Disable canvas':
                foreach (var component in canvasComponents)
                    component.enabled = false;
            }

            if (OnTargetLost != null)
                OnTargetLost.Invoke();
        }
}
```

Lampiran 2. Denah Universitas Muhammadiyah Ponorogo



Lampiran 3: Identitas Responden

NO	NAMA	NIM	PRODI
1	ANDHIKA SATRIA BINTORO	20520605	TE A
2	CALVIN DWI SUSANTO	20520670	TE A
3	Didik setiawan	20511475	Teknik mesin
4	Chaesar Deserendy Dwiprasetya	20511488	Teknik Mesin
5	Natasya Cindy Rafika Cahyaningrum	20533356	Teknik Informatika
6	Bagas Seto	20533331	Teknik Informatika
7	Fathur Rosyid	20533263	TI
8	Avif Nurrohman	20533276	TI
9	Thomas Wahyu Nugroho	20520638	TE
10	Beny Prayoga	20520660	TE
11	Katon Prasetyo	20520676	TE
12	Ryan Farhan Syarif	20520658	TE
13	Edi Ananda	20520637	TE
14	Muhammad Sholihul Abdillah	20520664	TE
15	Mulki Furqon	20520667	TE
16	Muhamad Wahyu saputra	20520651	TE
17	Bagus seto wiguna	20520655	TE
18	Muhammad Al Ghifari Arifuddin	20520669	TE
19	Andre Kharisma	20520642	TE
20	Taupik kurnianto	20520653	TE
21	Bimbi Nur Fiqron	20520659	TE
22	DIMAS YUSUL PRIJANARGO	20533275	TI
23	Zainul Asrofi	20520654	TE
24	ALFI NURIYATUL HEKMAAH	20533274	TI
25	SEPTIANA MEILANI PUTRI PAWITI	20533273	TI
26	LINA DWI JAYANTI	20533272	TI
27	DINDA APRILIA MUTIARA SARI	20533271	TI
28	TARISA AULIYA RAMADHANI	20533269	TI
29	Muh. Ardiawan	20520665	TE
30	Rizki Wahyu Nur K	20520652	TE
31	Andi Mardani	20520646	TE
32	Dicky Purnomo	20533268	TE
33	Yudha Pratama Irawan	20533267	TI
34	ARDITTA NUGRAHINNI	20533266	TI
35	Lely Mustikasari Mahardhika Fitriani	20533265	TI
36	ADIMAS AMBANG SYAHPUTRA	20533264	TI
37	Muh. Abdil Hagi	20520645	TI
38	Pamuji Rahayu	20520671	TE
39	Muhammad Nur Huda	20520678	TE
40	RIZQI FAJAR RIYANTO	20533242	TI
41	ALIEF VERSA HERDIANSYAH	20533244	TI
42	MOH RIDWAN OKTAVIAN	20533246	TI
43	Mudofar Alfaruqi	20520663	TE
44	Dimas Rahadian Riesananda	20520636	TE
45	Heri Utomo	20520644	TE
46	Bagus Syafur Rizal	20520675	TE

47	Reza Dickiprabowo	20533247	TI
48	Wahyu Krisdiantoro	20520661	TE
49	ANNISA AULIA WARDHANI	20533261	TI
50	LUTHFI AHMAD ANINDITO PUTRA	20533259	TI

Lampiran 4. Jawaban Responden

No.Responden	**No. Pernyataan**									
	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
1	SB	SB	SB	B	B	B	SB	SB	SB	SB
2	B	B	B	B	B	SB	B	B	TB	B
3	B	B	SB	SB	B	SB	B	B	B	B
4	SB	B	SB	SB	B	SB	SB	SB	B	B
5	B	B	B	B	B	B	B	B	B	B
6	B	B	SB	B	B	TB	B	B	B	TB
7	B	B	B	B	B	B	B	B	B	B
8	SB	SB	SB	B	B	SB	SB	B	B	B
9	B	B	B	B	B	B	B	B	B	B
10	B	B	B	B	B	B	B	B	B	B
11	B	B	B	B	B	B	B	B	B	B
12	SB	B	B	B	B	SB	B	B	B	B
13	SB	B	B	B	B	B	B	SB	SB	SB
14	B	B	B	B	B	B	B	B	B	B
15	B	SB	B	B	B	B	B	B	B	B
16	B	B	B	B	B	B	B	B	B	B
17	B	B	B	B	B	B	B	B	B	B
18	B	SB	B	SB	SB	B	B	B	B	B
19	B	B	B	B	B	B	B	B	B	B
20	B	B	B	B	B	B	B	B	B	B
21	B	B	B	B	B	B	B	B	B	B
22	B	B	B	B	B	B	B	B	B	B
23	B	B	B	SB	SB	B	B	B	B	B
24	B	B	B	SB	SB	SB	SB	B	B	SB
25	B	B	B	B	B	B	B	SB	SB	B
26	B	B	B	B	B	B	B	B	B	B
27	B	B	B	SB	B	B	B	B	B	B
28	B	SB	B	B	B	B	B	SB	SB	SB
29	B	B	B	B	B	B	SB	B	B	B
30	SB	SB	SB	SB	SB	SB	SB	SB	SB	SB
31	B	B	B	B	B	B	B	B	B	B
32	B	B	B	B	B	B	B	B	B	B
33	SB	B	B	SB	SB	B	B	B	B	B
34	B	B	SB	SB	B	B	B	B	B	B
35	B	B	B	B	B	B	B	B	B	B
36	B	B	B	B	B	B	B	B	B	B
37	B	B	B	B	B	B	B	B	B	B

38	B	SB	SB	SB	B	B	SB	SB	SB	B
39	SB	SB	SB	B	SB	SB	B	B	SB	B
40	SB	SB	B	SB	SB	SB	B	B	B	B
41	SB	SB	SB	B	B	SB	SB	B	B	B
42	SB	SB	SB	B	B	B	B	SB	SB	B
43	B	SB	SB	B	SB	B	B	B	SB	B
44	B	B	SB	B	SB	B	SB	B	B	B
45	B	B	B	SB	B	B	B	B	B	SB
46	B	B	B	B	B	B	B	B	B	SB
47	B	SB	SB	B	B	B	B	SB	B	B
48	B	B	B	B	B	B	B	B	B	B
49	B	B	B	SB	SB	B	B	B	B	B
50	B	B	SB	B	B	B	SB	B	B	B