A REPORT

ON

HOME FURNISHING USING AUGMENTED REALITY

BY

Kush Mehta 2018B5A70956P

AT

CSIR-CEERI Pilani, Pilani

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

May, 2020

A REPORT

ON

HOME FURNISHING USING AUGMENTED REALITY

BY

Kush Mehta	**2018B5A70956P**	**MSc. Physics and B.E. Computer Science Engineering**

Prepared in partial fulfillment of the
Practice School-I Course Nos.
BITS C221/BITS C231/BITS C241

AT

CSIR-CEERI Pilani, Pilani

A Practice School-I Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

May, 2020

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE PILANI (RAJASTHAN)
Practice School Division

Station: CSIR-CEERI Pilani, Pilani **Centre:** Pilani

Duration: 6 weeks **Date of start:** 18th May 2020

Date of submission: 24th June 2020

Title of the project: Home Furnishing using Augmented Reality

Name: Kush Mehta

Discipline: MSc. Physics and B.E. Computer Science Engineering

Name of expert: Mr. Pramod Tanwar

Designation of expert: Principal Scientist at CSIR-CEERI Pilani

Name of PS faculty: Professor Rakesh Warier

Key Words: Augmented reality, Vuforia, Unity, Ground plane detection, Image Target

Project Areas: Augmented Reality and App Development

Abstract: The project showcases the use of Augmented Reality to create a self-home-furnishing app by using ground plane detection feature of Vuforia through Unity framework. The app allows the user to select a furniture and place it at the desired location to see which arrangement looks best for the house. This project can further be extended to furnish offices, party plots, interior of schools and colleges. The technology allows architects to use it to visualize structures and cities. The app can be customized and deployed for various furniture stores to allow their users to preview the furniture.

Signature of student
Date:

Signature of PS faculty
Date:

ACKNOWLEDGEMENT

I would like to thank BITS Pilani for providing me with this opportunity to work as an intern at CSIR-CEERI Pilani through its Practice School – 1 program. I express my gratitude towards CSIR-CEERI Pilani for accepting me as an intern and providing me with an opportunity to work on a real-life project.

I am highly indebted to Mr. Pramod Tanwar who has guided me throughout the work through his invaluable inputs and reviews. My heartfelt thanks to Professor Rakesh Warier, my faculty in-charge for the PS-1 program for guiding me and providing me with the valuable information regarding the PS-1 procedures.

I am highly thankful to my friends for helping me with the testing of the application and providing technical guidance required for the project respectively. The work would not have been complete without the help and constant motivation of my family and friends.

KUSH MEHTA

TABLE OF CONTENT

1. INTRODUCTION

1.1 THE ORGANISATION

Central Electronics and Engineering Research Institute (CEERI) is a research institute in India and is a constituent laboratory of Council of Scientific and Industrial Research(CSIR), New Delhi. It currently has four centers at Pilani, Delhi, Jaipur, and Chennai with the center at Pilani being the oldest. CEERI was established in 1953 for advanced research and development in the field of Electronics. The main areas of research include:

1. Cyber-physical systems with a focus on the fundamental intellectual problem of conjoining the engineering traditions of the cyber and the physical worlds
2. Smart sensors with an objective of designing and developing robust and accurate sensors and actuators for a variety of applications.
3. Microwave devices and is the only agency in the country which has been associated with R&D of different types of Microwave Tubes.

This year CSIR-CEERI has offered project in four main domains which include Digital Signal Processing, Machine Learning, Deep Learning, App development, Augmented Reality development, IoT systems and VLSI design. CEERI Pilani has a signed MoU with BITS Pilani since 1991 and have since then been indulged in collaborations in taking up Masters and PhD. Programmes.

1.2 PROJECT DOMAIN (Augmented Reality)

The term Augmented Reality means to 'augment' something into the real world. Augmenting something refers to placement of virtual 3D objects in real world through a device(sensor) which provides a feeling as if the object exists. In simple terms we place virtual objects into the real(our) world.

The best devices which capable of demonstrating Augmented reality are smartphones. They take input through the cameras and superpose a virtual 3D object in that location on the screen making it appear as if it were present. Augmenting objects enables a person to visualize and experience a scene which otherwise he wouldn't be able to through mere imagination. Therefore, we can say that AR fulfils three basic features: real-time interaction, accurate 3D registration of virtual and real objects and a combination of real and virtual worlds. You can put your imagination onto the screen to experience how it feels.

Augmented Reality is often used in games where the game objects are augmented to give the user live experience. It can also be used to guide people through unknown things like a device, or even a tourist place. Architects use AR to create interiors and to model cities. The use of AR in education has seen a drastic increase in recent years acting as a virtual teacher. I believe that in these uncertain times of a pandemic where every meeting is at a halt, AR can be the savior and act like a friend, teacher, toy, or a guide. It is a relatively new technology and constant research and technological advancements are being made to extract the full potential of this technology.

2. ABOUT THE PROJECT

2.1 OBJECTIVE

The aim of the project is to self-furnish your house by selecting a furniture for bedroom, hall and dining hall and placing it in a desired location to give it a finished look before buying it. The arrangements are to be stored in a folder by taking screenshots and later retrieving them.

2.2 SOFTWARES USED / SKILLS REQUIRED

The project required a software and an in-built SDK named Unity and Vuforia respectively. Along with this Visual Studio Code was used to write the code for the project. C# programming language was used to develop the project. Below, I have given a summary about the various softwares stated.

2.2.1 Unity

Unity is a cross-platform game engine with a built-in IDE developed by Unity. It the world's most popular development platform for creating 2D and 3D multiplatform games as well as apps and interactive experiences like VR and AR
Unity allows the users to develop their own models or use predefined models through the asset store. It integrates the components of the game through scripts written in C#. It also allows to create interactive UI elements through it drag and drop feature. Unity also has in-built scripts which enable the game objects to interact with the platform and its environment.
Unity also supports various plugins for the creation of AR as well as VR apps. The project utilizes Vuforia plugin of Unity to develop Augmented Reality.

2.2.2 Vuforia

Vuforia is an Augmented reality software development kit (SDK) for mobile devices which enables the creation of augmented reality applications. It uses the camera of a device to detect and recognize planar surfaces called Image Targets and 3D objects. It can recognize planar objects and cylindrical objects for single image tracking and cuboidal objects for multi-image tracking. Its ground plane and mid-air detection features allow the user to place objects on the ground as well as in mid-air.
Object recognition through image targets are called marked tracking whereas for the case of ground plane detection as well as mid-air detection is called marker less tracking. For achieving object recognition through image targets the user must upload an image of the 3D object which he would like to track. Once uploaded the package must be downloaded from the developer portal and be imported into Unity. Now the user can augment the desired 3D virtual object on top of it. The virtual object then tracks the position and orientation of the image in real-time so that the viewer's perspective on the object corresponds with the perspective on the Image Target. It thus appears that the virtual object is a part of the real-world scene.
For marker less tracking a plane finder is used upon which the ground plane is attached to for its detection. After the ground plane is detected it augments the selected model on top of it. According to the Vuforia documentation ground plane detection is limited to only a few devices and hence its app cannot be installed in all mobile applications.

2.2.3 Visual Studio Code

Visual Studio Code is a free source-code editor made by Microsoft for Windows, Linux and macOS. It comes with built-in support for JavaScript, TypeScript and Node.js and has a rich ecosystem of extensions for other languages such as C++, C#, Java, Python, PHP, Go and runtimes such as .NET and Unity.
For using Visual Studio Code with Unity, one must set external script editor preference to support Visual Studio Code. Thereby the scripts created in Unity would automatically open in VSCode. For debugging the code, the user must set the debugger to Unity.

2.3 DETAILS OF THE PROJECT

The project utilizes the ground plane detection feature of Vuforia to enable the app the detect the ground and place selected furniture onto it. The user has an option to select the part of the house which he would like to furnish(virtually). The app includes bedroom, drawing room and dining hall. In each of these regions the user can select models which he would like to place and visualize. Once selected the model appears on the screen placed on the ground. Through finger gestures the user can adjust the position as well as size of the furniture. After finalizing the position, he can set the furniture so that it doesn't move and hence the user can see the entire arrangement created. Fig 2.3.1 shows the main layout of the Unity file.

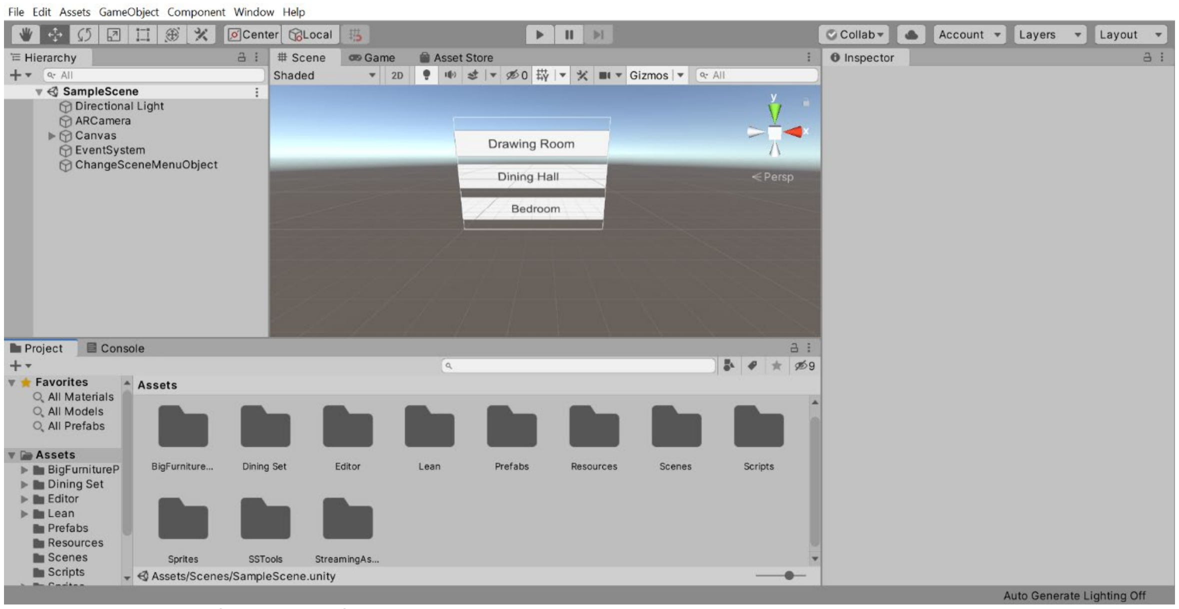
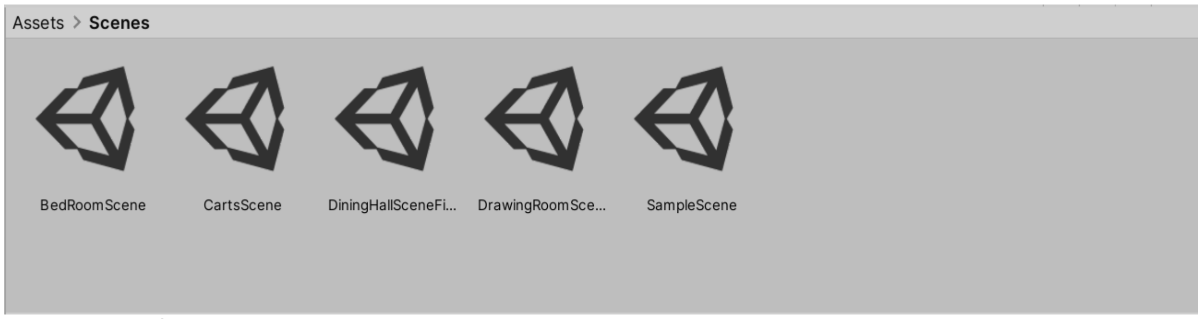


Fig 2.3.1: Layout of the Unity file

2.3.1 Scenes

The app contains five scenes, one each for bedroom, drawing room and dining hall, one for the home screen through which we can toggle between the scenes and the final for the cart which contains all stored screenshots of the arrangements Fig 2.3.2 shows the four scenes created.

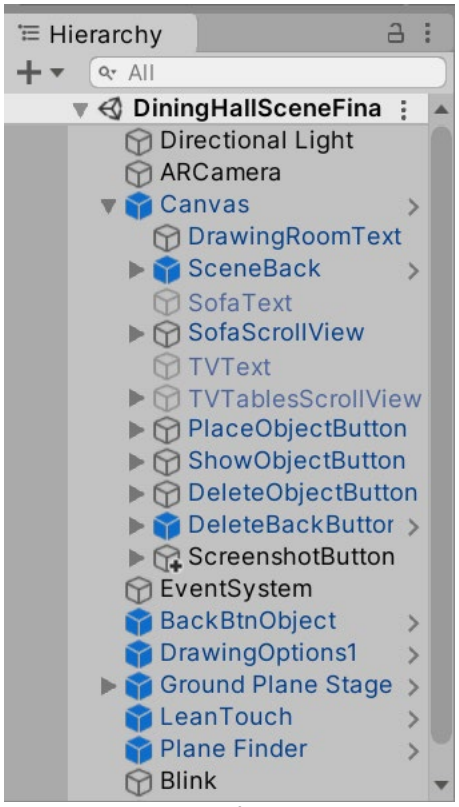

Assets > Scenes

BedRoomScene CartsScene DiningHallSceneFi... DrawingRoomSce... SampleScene

Fig 2.3.2 The four scenes

2.3.2 Hierarchy

The three scenes contain the same components in their hierarchy which include Ground Plane stage which hosts all the models to be displayed on the ground. The ground plane is of 100cm*100cm and hence all the objects must be sized in relation to it. The hierarchy also contains the canvas which enables the user to place UI elements as to be seen on the device. A plane finder is also placed in the hierarchy which takes the ground plane stage as a parameter and detects the ground. Fig 2.3.3 shows the hierarchy of the scenes and Fig 2.3.4 shows the hierarchy of the main scene. The SampleScene hierarchy contains four buttons to toggle between the three scenes and the gallery cart. The Gallery Cart Scene's hierarchy is shown in Fig 2.3.5.


```
Hierarchy
All
DiningHallSceneFina
  Directional Light
  ARCamera
  Canvas
    DrawingRoomText
    SceneBack
    SofaText
    SofaScrollView
    TVText
    TVTablesScrollView
    PlaceObjectButton
    ShowObjectButton
    DeleteObjectButton
    DeleteBackButtor
    ScreenshotButton
  EventSystem
  BackBtnObject
  DrawingOptions1
  Ground Plane Stage
  LeanTouch
  Plane Finder
  Blink
```

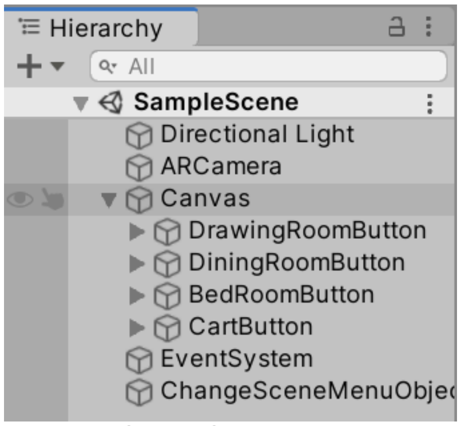

```
Hierarchy
All
SampleScene
  Directional Light
  ARCamera
  Canvas
    DrawingRoomButton
    DiningRoomButton
    BedRoomButton
    CartButton
  EventSystem
  ChangeSceneMenuObjec
```

Fig 2.3.3 DiningHall Scene Hierarchy **Fig 2.3.4 SampleScene Hierarchy**

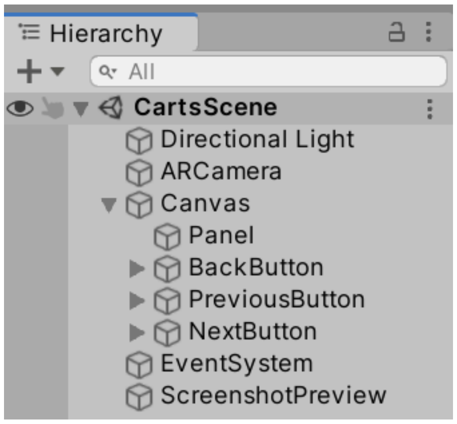


Fig 2.3.5 CartsScene Hierarchy

2.3.3 Canvas

The canvas hold the UI elements which are to be seen on the mobile by the user. The project uses buttons, texts and scrollviews to incorporate all its elements. The scrollviews show 3D buttons(which are the modes themselves) depicting the options of the models which the user has. The 3D buttons can rotate in the canvas to provide a preview as to how they would appear. The canvas of the three scenes is almost the same which specifics being different according to the requirements. The canvas of the main scene contains three buttons to toggle between the scenes. Fig 2.3.6 shows the hierarchy of the canvas of the main(SampleScene) scene.

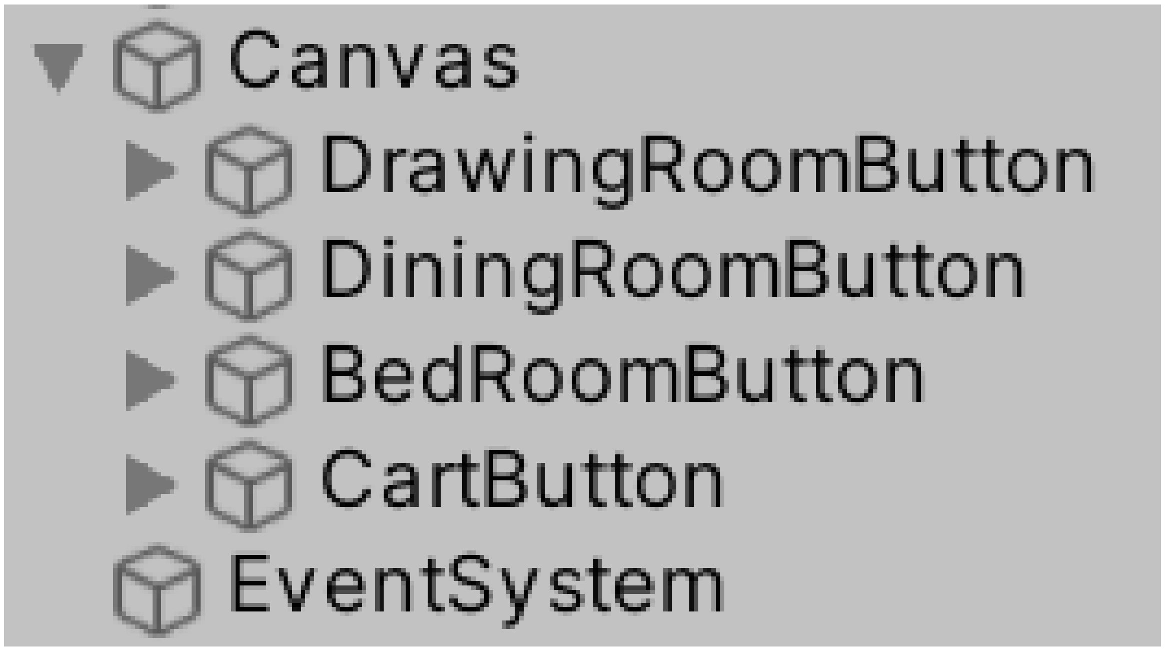


Fig 2.3.6: Canvas Hierarchy of the SampleScene

2.3.4 Scripts

The app uses various scripts written in C# to perform its functionalities. The ChangeSceneScript is used to toggle between the three scenes and the gallery cart through buttons. The three different scenes contain two scripts each, one for selecting and placing the objects and the other for deleting them. Each 3D model button in the scrollview has an additional script attached to it to activate its clickable functionality. These scripts are stored in the folders named by the respective location.

The script which enables selecting and placing objects initially disables all the models. The models are activated only when the corresponding buttons are tapped. The tapping of the buttons is handled by the individual scripts of the models stored in the folders. After the user is satisfied with the position and upon pressing the “place” button all the movements of the object is ceased and the object is fixed at that particular point. Then another model can be selected with the same procedure. The script takes care of all the objects which have been “placed” and doesn’t allow it to move or be selected again. Once the user is satisfied with all the placements and upon clicking the “arrangements” button all the objects are shown at the respective positions.

Finally if the user is not satisfied with the arrangement he can delete an object by clicking the delete button and then tapping the particular object, The delete script takes care that the object is deleted and all its movements are restored back to normal.

Moreover, upon clicking the “Arrangement” button the camera button appears which enables the user to take a screenshot of the arrangement ang store it for future uses. These screenshots can be later viewed by clicking the gallery button through the main scene. The location of the strores screenshot is: File manager -> Android -> Data -> App folder -> Files.

Fig 2.3.7 shows all the 7 scripts in the assets folder and Fig 2.3.8 shows the visual studio code editor. The code for the respective operations is stated in the Appendix.

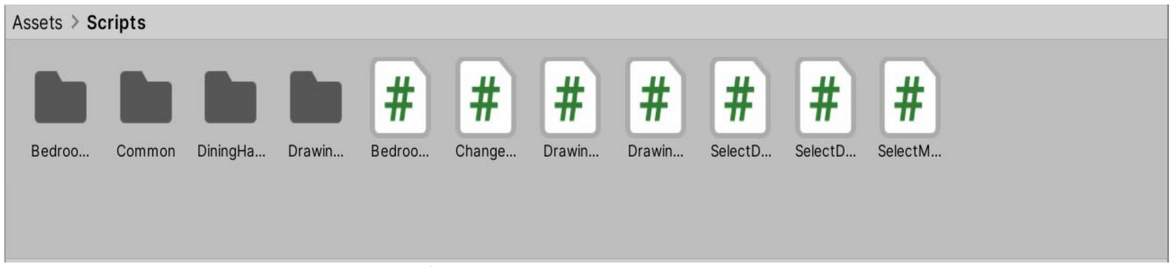
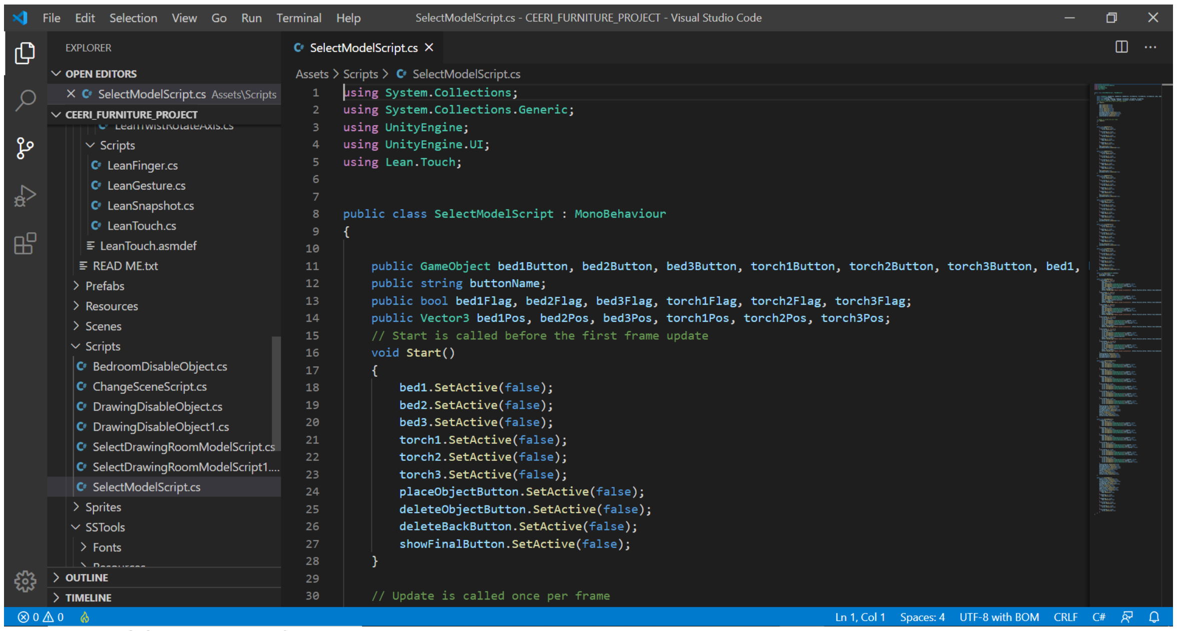


Fig 2.3.6: The 7 scripts in the Assets folder



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


public class SelectModelScript : MonoBehaviour
{

    public GameObject bed1Button, bed2Button, bed3Button, torch1Button, torch2Button, torch3Button, bed1,
    public string buttonName;
    public bool bed1Flag, bed2Flag, bed3Flag, torch1Flag, torch2Flag, torch3Flag;
    public Vector3 bed1Pos, bed2Pos, bed3Pos, torch1Pos, torch2Pos, torch3Pos;
    // Start is called before the first frame update
    void Start()
    {
        bed1.SetActive(false);
        bed2.SetActive(false);
        bed3.SetActive(false);
        torch1.SetActive(false);
        torch2.SetActive(false);
        torch3.SetActive(false);
        placeObjectButton.SetActive(false);
        deleteObjectButton.SetActive(false);
        deleteBackButton.SetActive(false);
        showFinalButton.SetActive(false);
    }

    // Update is called once per frame
```
Fig 2.3.7 VSCode editor for the scripts

2.3.5 Asset Store

Unity contains an in-built asset store which contains both paid and free assets like 3D models, codes for specific tasks, etc. to be used by the user in the app. The project uses the asset store for the models of bed, torch, dining table, sofa, TV table and shelf. An asset called “Lean Touch” was also used to enable the movement of objects across the screen through touch gestures. All the assets are stores in the main project folder through which we can access them. Moreover, the “Materials” folder contains the textures and the materials which can be used on the objects to design them and give them a realistic look.
Fig 2.3.9 shows all the assets used in the project.

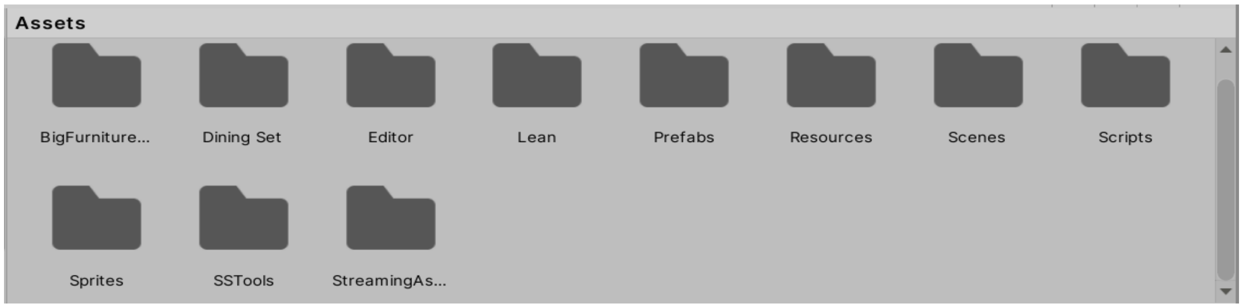


Fig 2.3.8: All the assets used

The main Vuforia folders are also stored in this directory. The components of Vuforia can either be used through right clicking in the hierarchy or through the assets directory.

2.4 WORKING OF THE APP

The app opens with a screen which displays three buttons to toggle between the three scenes of bedroom, drawing room and dining hall. Each button will lead to its scene. On the bottom right corner, a gallery icon is present to lead to the scene where the saved screenshots can be viewed. Fig 2.4.1 displays the look of the intent.

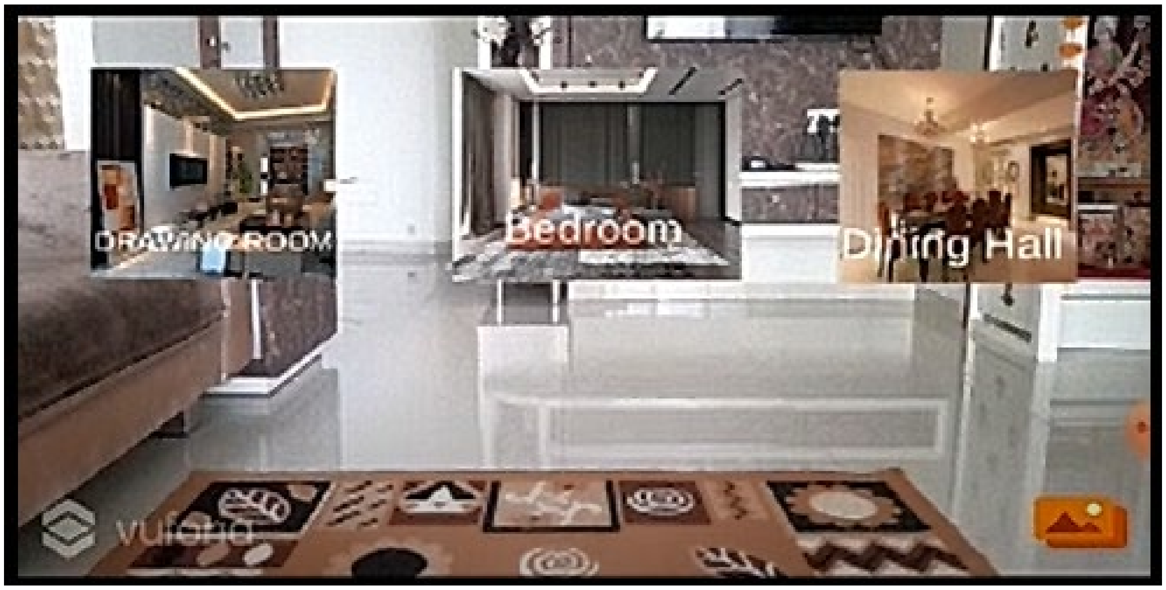


Fig 2.4.1: Main Scene Layout

2.4.1 The Three Scenes

Fig 2.4.2 shows the UI of the scene. The scrollview contain the 3D buttons which upon clicking would display the model on the ground. The buttons show the samples of the 3D models and can be rotated in a similar fashion as the objects. On the bottom right corner, a back button is placed to toggle back to the main scene.

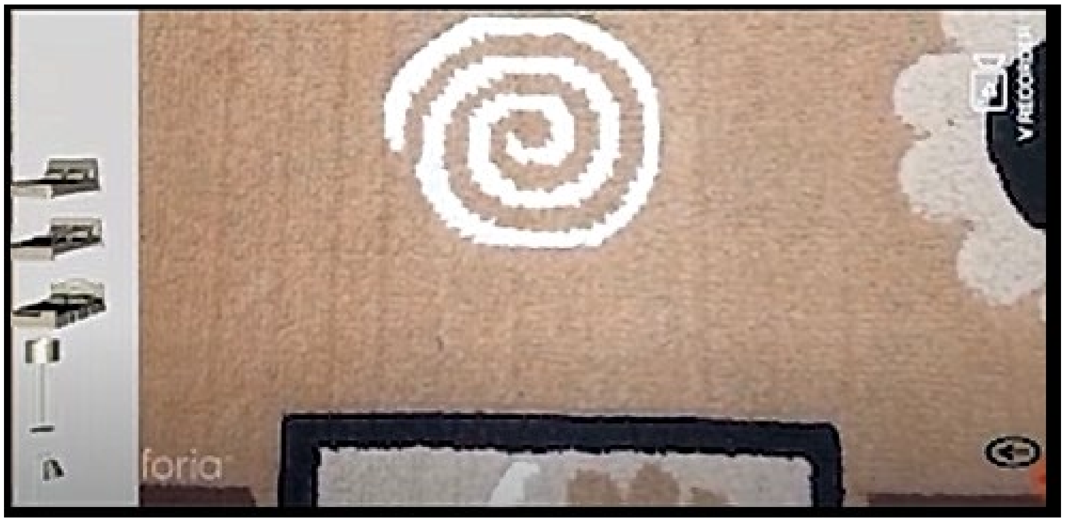


Fig 2.4.2: Drawing room scene

Fig 2.4.3 shows the model displayed on the screen upon clicking on the model's button. Finger gestures can drag the object to the desired position. Similarly pinching can resize the object and twisting can rotate the object to make it fit at the desired location. At the bottom there is a button named "Place". The button is used to fix the model at the desired position and thereby cease all its motion. This enables the inclusion of multiple objects.

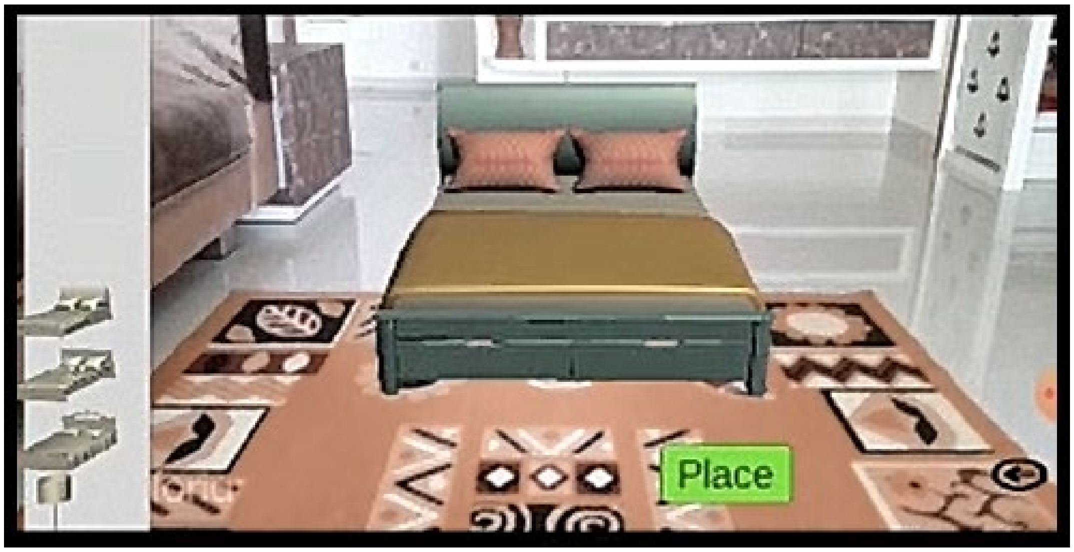


Fig 2.4.3: 3D model augmented through ground plane detection

Upon pressing the button two more buttons appear named "Arrangement" and Delete icon. Fig 2.4.4 shows the screen for the above operation. A message showing "Object placed successfully" appears to notify the user about the operation.

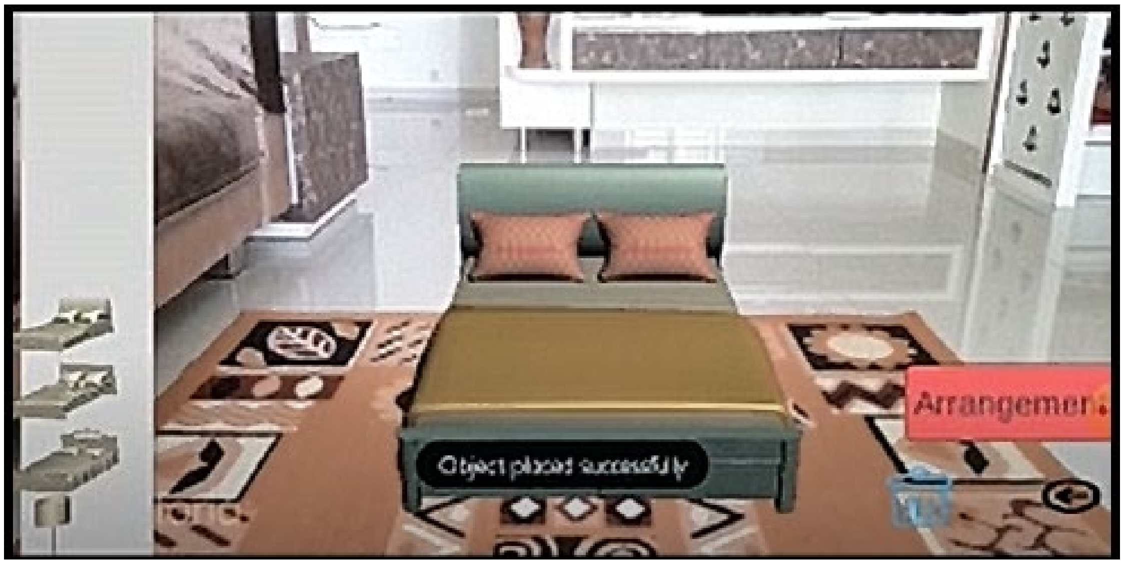


Fig 2.4.4: Place button pressed after placing the object

Upon pressing the “Arrangement” button all the objects whose motion have been ceased would appear and the scrollviews would disappear for the users to be able to see the view completely. If the user is not satisfied with the arrangement, then he can delete the object by clicking on the delete icon and then tapping on the object which he wishes to delete. A camera icon also appear to take a screenshot of the arrangement to be viewed later.
Fig 2.4.5 shows the intent for the following operation.

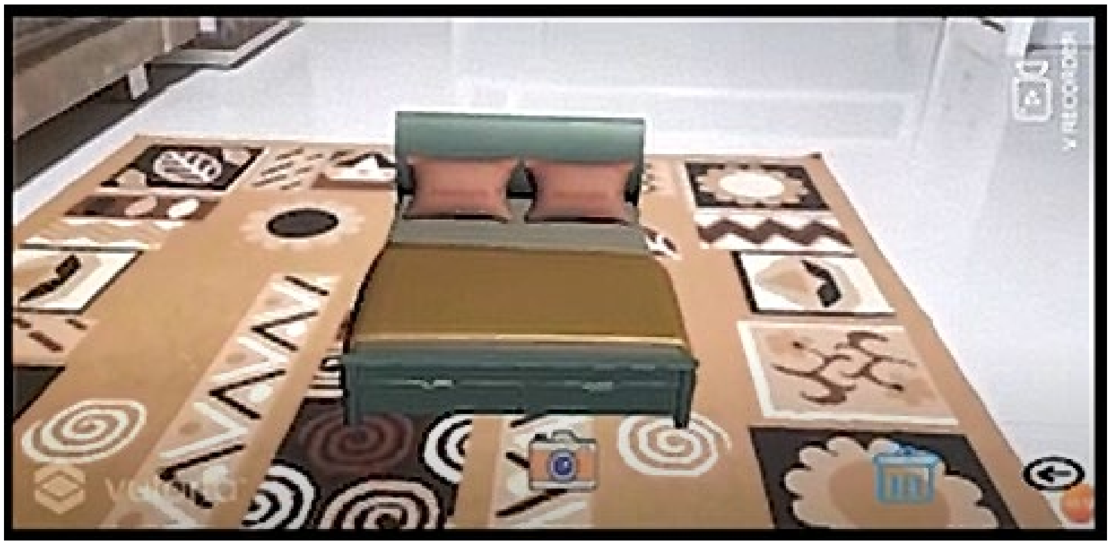


Fig 2.4.5: Show Arrangements button pressed after placing both objects

In the below right corner, there is a back icon which would lead us back to the original display where the scrollviews are present. If we click the “back” button from the intent generated, then we return to the main scene.

After placing an object, the user can place multiple objects and also ensure that no two objects are colliding. Fig 2.4.6 shows the scenario when two objects collide.

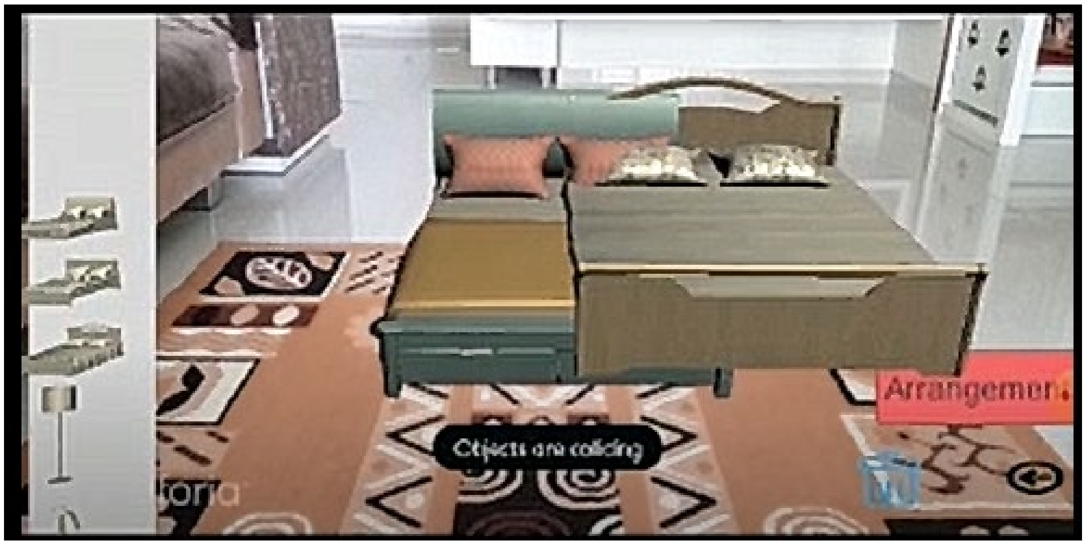


Fig 2.4.6: Generating multiple objects and detecting collision

When two objects collide/overlap the “Place” button is disabled so that the user can’t place the object on top of another (which is what is the real scenario). A message saying “Objects are colliding” displayed to let the user know about the collision. When the arrangement button is pressed in this case only the object which was earlier placed would be visible which in this case would be same as Fig 2.4.5.
To solve this issue and be able to place the object the user must shift the other object to a position which would not overlap the original(placed) object. Fig 2.4.7 demonstrates the scenario. Once the collision has been resolved the place button reappears and a message showing “Functions Restored to Normal” would be displayed.

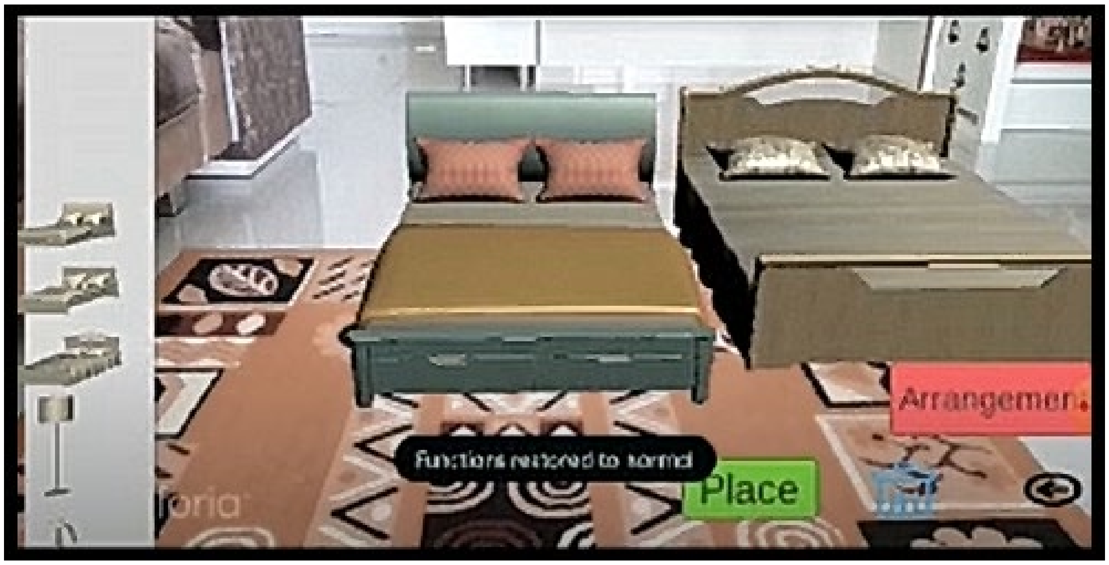


Fig 2.4.7: Collision Management

Once the desired arrangements are made and the user can click on the camera icon which is shown in Fig 2.4.5. Upon clicking the button, a screenshot is taken and stored at the location: File manager -> Android -> Data -> Application name -> Files. A message stating “Screenshot Saved” is displayed to notify the user that the action has been performed. Moreover, to reduce the interference in the screenshots the Delete icon and the back button are disabled momentarily while taking the screenshot creating a blink effect and are restored as soon as the screenshot has been taken. Fig 2.4.8 shows the operation mentioned.

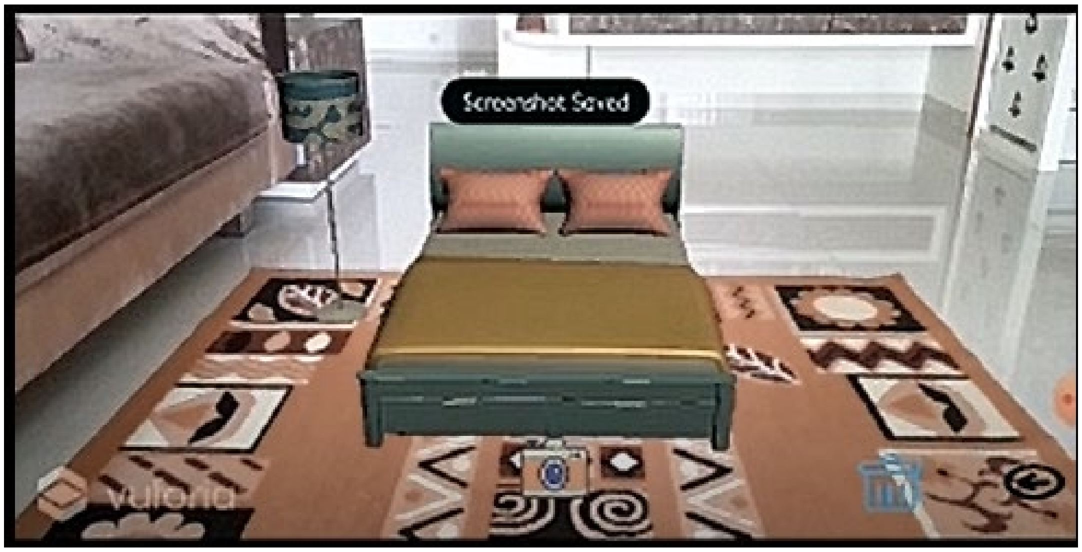


Fig 2.4.8: Screenshot of the arrangement captured

2.4.2 The Gallery Scene

To view the screenshots the user would have to go back to the main scene and click on the gallery icon to bring you to the gallery scene. The screenshots would be displayed sequentially, and the user can toggle between the screenshots through the arrows on the top indicating “next” and “previous”. Fig 2.4.9 shows the gallery scene.

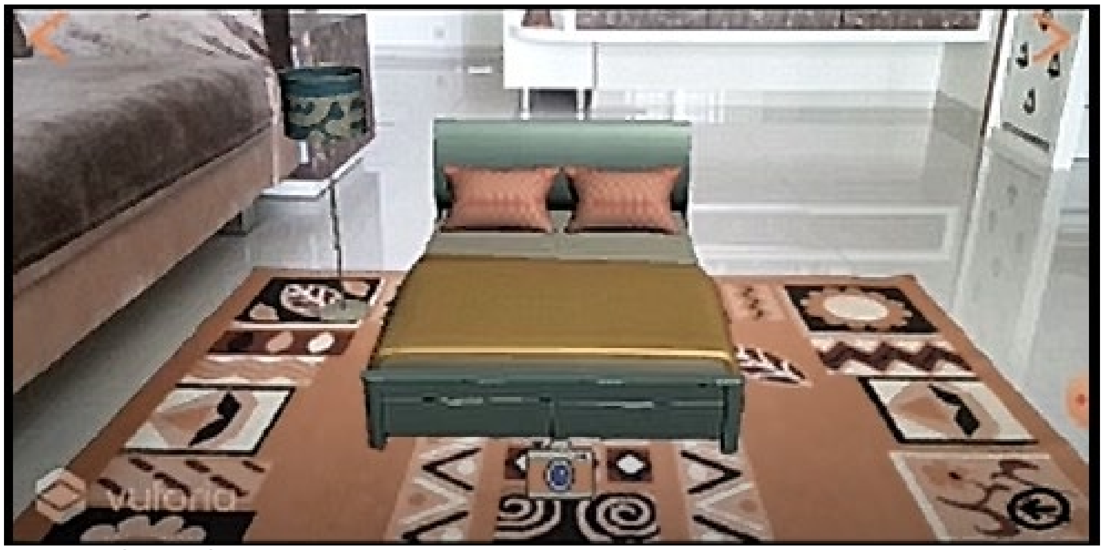


Fig 2.4.9: Gallery Scene

The app works similarly for the other two scenes. There are options for sofa and TV table for drawing room, bed and lamp for bedroom and dining table and shelf for dining hall.

3. CONCLUSIONS AND RECOMMENDATIONS

3.1 CONCLUSIONS

The project was aimed at designing, developing, and implementing an Augmented Reality application to allow users to choose their own furniture while sitting at their house by visualizing the various furniture through Augmentation at various locations of the house. In short it allows the users to furnish their house on their own without wanting to go to the store.

The application allows the users to select furniture for three locations of the house which are bedroom, drawing room and the dining hall. For the bedroom the user has the option to choose among three beds and three lamps. Similarly, for the drawing hall he/she can choose among three sofa sets and three TV tables. Finally, for the dining hall the user has the option between three dining tables and three shelves. Once selected the user can see the complete arrange of the location and decide whether to keep it or delete an object. After finalizing the arrangement, the user can take a screenshot of the arrangement and view it when asked for it.

In conclusion, the application works successfully and gives the users a new perspective as to decorate their house.

3.2 RECOMMENDATIONS

The implications of the project can be extended to furnishing an office area, decorating a party plot, or installing interior of colleges and schools.

Further it can host the furniture of various furniture brands to make the user able to choose furniture from the top brands. The app can also be modelled to design a model house instead of a real one.

With further technological advancements the app can be deployed on vast range of devices as it currently only a few devices can host the application.

4. APPENDICES

4.1 C# CODE TO TOGGLE BETWEEN SCENES

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

public class ChangeSceneScript : MonoBehaviour
{

    public void drawingRoom(){
        SceneManager.LoadScene("DrawingRoomScene");
    }

    public void diningRoom(){
        SceneManager.LoadScene("DiningHallSceneFinal");
    }

    public void bedRoom(){
        SceneManager.LoadScene("BedRoomScene");
    }

    public void backBtn(){
        SceneManager.LoadScene("SampleScene");
    }

    public void addToCart(){
        SceneManager.LoadScene("CartsScene");
    }
}
```

4.2 CODE TO SELECT AND PLACE OBJECT

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

```
public class SelectModelScript : MonoBehaviour
{

    public GameObject bed1Button, bed2Button, bed3Button, torch1Button, torch2Button, tor
ch3Button, bed1, bed2, bed3, torch1, torch2, torch3, showFinalButton, deleteObjectButton,
 deleteBackButton, placeObjectButton, bedText, torchText, bedScrollView, torchScrollView,
 sceneBackBtn, screenshotButton;
    public string buttonName, gameObjectName;
    public bool bed1Flag, bed2Flag, bed3Flag, torch1Flag, torch2Flag, torch3Flag;
    public Vector3 bed1Pos, bed2Pos, bed3Pos, torch1Pos, torch2Pos, torch3Pos, bed1SpawnP
os, bed2SpawnPos, bed3SpawnPos, torch1SpawnPos, torch2SpawnPos, torch3SpawnPos;

    // Start is called before the first frame update
    void Start()
    {
        bed1.SetActive(false);
        bed2.SetActive(false);
        bed3.SetActive(false);
        torch1.SetActive(false);
        torch2.SetActive(false);
        torch3.SetActive(false);
        placeObjectButton.SetActive(false);
        deleteObjectButton.SetActive(false);
        deleteBackButton.SetActive(false);
        showFinalButton.SetActive(false);
        screenshotButton.SetActive(false);
    }

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

    }

    public void OnClick(Button button){
        print(button.name);
        buttonName = button.name;
    }

    public void placeObject(){
        if(gameObjectName == "Bed1"){
            //bed1.SetActive(true);
            bed1Flag = true;
            bed1Pos = bed1.transform.position;
            bed1Pos.z = bed1SpawnPos.z;
            bed1.transform.position = bed1Pos;
            Debug.Log(bed1Pos);
            bed1.GetComponent<LeanDragTranslate>().enabled = false;
            bed1.GetComponent<LeanPinchScale>().enabled = false;
            bed1.GetComponent<LeanTwistRotateAxis>().enabled = false;
        //GameObject.Find("PluginWrapper").GetComponent<PluginWrapper>();
            SSTools.ShowMessage("Object placed successfully", SSTools.Position.bottom, SS
Tools.Time.twoSecond);
        }
        if(gameObjectName == "Bed2"){
            //bed2.SetActive(true);
            bed2Flag = true;
            bed2Pos = bed2.transform.position;
            bed2Pos.z = bed2SpawnPos.z;
            bed2.transform.position = bed2Pos;
            bed2.GetComponent<LeanDragTranslate>().enabled = false;
            bed2.GetComponent<LeanPinchScale>().enabled = false;
            bed2.GetComponent<LeanTwistRotateAxis>().enabled = false;

            Debug.Log(bed2Pos);
            SSTools.ShowMessage("Object placed successfully", SSTools.Position.bottom, SS
Tools.Time.twoSecond);
        }
        if(gameObjectName == "Bed3"){
            //bed3.SetActive(true);
            bed3Flag = true;
            bed3Pos = bed3.transform.position;
            bed3Pos.z = bed3SpawnPos.z;
            bed3.transform.position = bed3Pos;
            bed3.GetComponent<LeanDragTranslate>().enabled = false;
            bed3.GetComponent<LeanPinchScale>().enabled = false;
            bed3.GetComponent<LeanTwistRotateAxis>().enabled = false;

            Debug.Log(bed3Pos);
            SSTools.ShowMessage("Object placed successfully", SSTools.Position.bottom, SS
Tools.Time.twoSecond);
        }
        if(gameObjectName == "torch1"){
            //torch1.SetActive(true);
            torch1Flag = true;
            torch1Pos = torch1.transform.position;
            torch1Pos.z = torch1SpawnPos.z;
            torch1.transform.position = torch1Pos;
            torch1.GetComponent<LeanDragTranslate>().enabled = false;
            torch1.GetComponent<LeanPinchScale>().enabled = false;
            torch1.GetComponent<LeanTwistRotateAxis>().enabled = false;

            Debug.Log(torch1Pos);
            SSTools.ShowMessage("Object placed successfully", SSTools.Position.bottom, SS
Tools.Time.twoSecond);
        }
        if(gameObjectName == "torch2"){
            //torch2.SetActive(true);
            torch2Flag = true;
            torch2Pos = torch1.transform.position;
            torch2Pos.z = torch2SpawnPos.z;
            torch2.transform.position = torch2Pos;
            torch2.GetComponent<LeanDragTranslate>().enabled = false;
            torch2.GetComponent<LeanPinchScale>().enabled = false;
            torch2.GetComponent<LeanTwistRotateAxis>().enabled = false;

            Debug.Log(torch2Pos);
            SSTools.ShowMessage("Object placed successfully", SSTools.Position.bottom, SS
Tools.Time.twoSecond);
        }
        if(gameObjectName == "torch3"){
            //torch3.SetActive(true);
            torch3Flag = true;
            torch3Pos = torch3.transform.position;
            torch3Pos.z = torch3SpawnPos.z;
            torch3.transform.position = torch3Pos;
            torch3.GetComponent<LeanDragTranslate>().enabled = false;
            torch3.GetComponent<LeanPinchScale>().enabled = false;
            torch3.GetComponent<LeanTwistRotateAxis>().enabled = false;

            Debug.Log(torch3Pos);
            SSTools.ShowMessage("Object placed successfully", SSTools.Position.bottom, SS
Tools.Time.twoSecond);
        }
        //PostToDatabase(gameObjectName);
        showFinalButton.SetActive(true);
        deleteBackButton.SetActive(false);
        deleteObjectButton.SetActive(true);
        placeObjectButton.SetActive(false);
        screenshotButton.SetActive(false);
    }

    public void showFinalObjects(){
        if(bed1Flag == true){
            bed1.SetActive(true);
            bed1.GetComponent<LeanDragTranslate>().enabled = false;
            bed1.GetComponent<LeanPinchScale>().enabled = false;
            bed1.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(bed2Flag == true){
            bed2.SetActive(true);
            bed2.GetComponent<LeanDragTranslate>().enabled = false;
            bed2.GetComponent<LeanPinchScale>().enabled = false;
            bed2.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(bed3Flag == true){
            bed3.SetActive(true);
            bed3.GetComponent<LeanDragTranslate>().enabled = false;
            bed3.GetComponent<LeanPinchScale>().enabled = false;
            bed3.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(torch1Flag == true){
            torch1.SetActive(true);
            torch1.GetComponent<LeanDragTranslate>().enabled = false;
            torch1.GetComponent<LeanPinchScale>().enabled = false;
            torch1.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(torch2Flag == true){
            torch2.SetActive(true);
            torch2.GetComponent<LeanDragTranslate>().enabled = false;
            torch2.GetComponent<LeanPinchScale>().enabled = false;
            torch2.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(torch3Flag == true){
            torch3.SetActive(true);
            torch3.GetComponent<LeanDragTranslate>().enabled = false;
            torch3.GetComponent<LeanPinchScale>().enabled = false;
            torch3.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        showFinalButton.SetActive(false);
        deleteBackButton.SetActive(true);
        sceneBackBtn.SetActive(false);
        deleteObjectButton.SetActive(true);
        placeObjectButton.SetActive(false);
        bedText.SetActive(false);
        torchText.SetActive(false);
        bedScrollView.SetActive(false);
        torchScrollView.SetActive(false);
        screenshotButton.SetActive(true);
    }
```

```
public void deleteObject(){
    if(bed1Flag == true){
        bed1.SetActive(true);
        bed1.GetComponent<LeanDragTranslate>().enabled = false;
        bed1.GetComponent<LeanPinchScale>().enabled = false;
        bed1.GetComponent<LeanTwistRotateAxis>().enabled = false;
    }
    if(bed2Flag == true){
        bed2.SetActive(true);
        bed2.GetComponent<LeanDragTranslate>().enabled = false;
        bed2.GetComponent<LeanPinchScale>().enabled = false;
        bed2.GetComponent<LeanTwistRotateAxis>().enabled = false;
    }
    if(bed3Flag == true){
        bed3.SetActive(true);
        bed3.GetComponent<LeanDragTranslate>().enabled = false;
        bed3.GetComponent<LeanPinchScale>().enabled = false;
        bed3.GetComponent<LeanTwistRotateAxis>().enabled = false;
    }
    if(torch1Flag == true){
        torch1.SetActive(true);
        torch1.GetComponent<LeanDragTranslate>().enabled = false;
        torch1.GetComponent<LeanPinchScale>().enabled = false;
        torch1.GetComponent<LeanTwistRotateAxis>().enabled = false;
    }
    if(torch2Flag == true){
        torch2.SetActive(true);
        torch2.GetComponent<LeanDragTranslate>().enabled = false;
        torch2.GetComponent<LeanPinchScale>().enabled = false;
        torch2.GetComponent<LeanTwistRotateAxis>().enabled = false;
    }
    if(torch3Flag == true){
        torch3.SetActive(true);
        torch3.GetComponent<LeanDragTranslate>().enabled = false;
        torch3.GetComponent<LeanPinchScale>().enabled = false;
        torch3.GetComponent<LeanTwistRotateAxis>().enabled = false;
    }
    showFinalButton.SetActive(false);
    deleteBackButton.SetActive(true);
    deleteObjectButton.SetActive(true);
    placeObjectButton.SetActive(false);
    sceneBackBtn.SetActive(false);
    bedText.SetActive(false);
    torchText.SetActive(false);
```

```
        bedScrollView.SetActive(false);
        torchScrollView.SetActive(false);
        screenshotButton.SetActive(false);
    }

    public void deleteBack(){
        bedScrollView.SetActive(true);
        torchScrollView.SetActive(true);
        placeObjectButton.SetActive(false);
        sceneBackBtn.SetActive(true);
        deleteBackButton.SetActive(false);
        bedText.SetActive(false);
        torchText.SetActive(false);
        showFinalButton.SetActive(true);
        bedScrollView.SetActive(true);
        torchScrollView.SetActive(false);
        screenshotButton.SetActive(false);
        buttonName = null;
        if(bed1Flag == true){
            bed1.SetActive(true);
        }
        if(bed2Flag == true){
            bed2.SetActive(true);
        }
        if(bed3Flag == true){
            bed3.SetActive(true);
        }
        if(torch1Flag == true){
            torch1.SetActive(true);
        }
        if(torch2Flag == true){
            torch2.SetActive(true);
        }
        if(torch3Flag == true){
            torch3.SetActive(true);
        }
    }
}
```

4.3 CODE TO DELETE OBJECT AND HANDLE COLLISION

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Lean.Touch;

public class DrawingDisableObject : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

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

    }

    public void OnMouseDown(){
        if(GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScript>(
).buttonName == "DeleteObjectButton"){
            gameObject.SetActive(false);
            if(gameObject.name == "corner_sofa"){
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().cornerSofaFlag = false;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().cornerSofa.GetComponent<LeanDragTranslate>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().cornerSofa.GetComponent<LeanTwistRotateAxis>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().cornerSofa.GetComponent<LeanPinchScale>().enabled = true;
            }
            if(gameObject.name == "sofa_1"){
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().sofa1Flag = false;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().sofa1.GetComponent<LeanDragTranslate>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().sofa1.GetComponent<LeanTwistRotateAxis>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().sofa1.GetComponent<LeanPinchScale>().enabled = true;
            }
            if(gameObject.name == "sofa_2"){
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().sofa2Flag = false;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().sofa2.GetComponent<LeanDragTranslate>().enabled = true;
```

```
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().sofa2.GetComponent<LeanTwistRotateAxis>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().sofa2.GetComponent<LeanPinchScale>().enabled = true;
            }
            if(gameObject.name == "tv_table_1"){
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable1Flag = false;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable1.GetComponent<LeanDragTranslate>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable1.GetComponent<LeanTwistRotateAxis>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable1.GetComponent<LeanPinchScale>().enabled = true;
            }
            if(gameObject.name == "tv_table_2"){
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable2Flag = false;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable2.GetComponent<LeanDragTranslate>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable2.GetComponent<LeanTwistRotateAxis>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable2.GetComponent<LeanPinchScale>().enabled = true;
            }
            if(gameObject.name == "tv_table_3"){
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable3Flag = false;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable3.GetComponent<LeanDragTranslate>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable3.GetComponent<LeanTwistRotateAxis>().enabled = true;
                GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScr
ipt>().tvTable3.GetComponent<LeanPinchScale>().enabled = true;
            }
        }else{

        }
    }

    void OnTriggerEnter(Collider collider){

        if(collider.gameObject.name == "corner_sofa" || collider.gameObject.name == "sofa
_1" || collider.gameObject.name == "sofa_2" || collider.gameObject.name == "tv_table_1" |
| collider.gameObject.name == "tv_table_2" || collider.gameObject.name == "tv_table_3"){
```

```
            GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScript>
().placeObjectButton.SetActive(false);
            SSTools.ShowMessage("Objects are colliding", SSTools.Position.bottom, SSTools
.Time.oneSecond);
        }
    }
    void OnTriggerExit(Collider collider){
        if(collider.gameObject.name == "corner_sofa" || collider.gameObject.name == "sofa
_1" || collider.gameObject.name == "sofa_2" || collider.gameObject.name == "tv_table_1" |
| collider.gameObject.name == "tv_table_2" || collider.gameObject.name == "tv_table_3"){
            GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScript>
().placeObjectButton.SetActive(true);
            SSTools.ShowMessage("Functions restored to normal", SSTools.Position.bottom,
SSTools.Time.oneSecond);
        }
    }
}
```

4.4 ENABLING TOUCH ON 3D MODEL BUTTONS AND SETTING THEIR ROTATION

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using Lean.Touch;

public class Bed1Script : MonoBehaviour
{
    public GameObject bed1;
    // Start is called before the first frame update
    void Start()
    {
        bed1.SetActive(false);
    }

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

    }

    public void OnMouseDown(){
        Debug.Log(gameObject.name);
        GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().gameObjectNam
e = gameObject.name;
        GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed1Button.Ge
tComponent<LeanTwistRotateAxis>().enabled = true;
        GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed1SpawnPos
= GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed1.transform.posi
tion;
        //bed1.SetActive(true);
        GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed1.SetActiv
e(true);
        if(GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed2Flag =
= true){
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed2.SetA
ctive(true);
        } else{
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed2.SetA
ctive(false);
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed2Butto
n.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed3Flag =
= true){
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed3.SetA
ctive(true);
        } else{
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed3.SetA
ctive(false);
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().bed3Butto
n.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch1Flag
 == true){
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch1.Se
tActive(true);
        } else{
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch1.Se
tActive(false);
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch1But
ton.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch2Flag
 == true){
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch2.Se
tActive(true);
        } else{
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch2.Se
tActive(false);
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch2But
ton.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        if(GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch3Flag
 == true){
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch3.Se
tActive(true);
        } else{
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch3.Se
tActive(false);
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().torch3But
ton.GetComponent<LeanTwistRotateAxis>().enabled = false;
        }
        GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().placeObjectBu
tton.SetActive(true);
    }
}
```

4.5 CODE TO TAKE SCREENSHOT AND STORE TO A FILE LOCATION

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

public class TakeScreenshot : MonoBehaviour {

    [SerializeField]
    GameObject blink;
    public string sceneName;
    public GameObject headingText;

    public void TakeAShot()
    {
        if(sceneName == "BedRoomScene"){
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().deleteBac
kButton.SetActive(false);
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().deleteObj
ectButton.SetActive(false);
            //GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().screens
hotButton.SetActive(false);
        }
        if(sceneName == "DrawingRoomScene"){
            GameObject.Find("DrawingOptions").GetComponent<SelectDrawingRoomModelScript1>
().deleteBackButton.SetActive(false);
            GameObject.Find("DrawingOptions").GetComponent<SelectDrawingRoomModelScript1>
().deleteObjectButton.SetActive(false);
            //GameObject.Find("DrawingOptions").GetComponent<SelectDrawingRoomModelScript
1>().screenshotButton.SetActive(false);
        }
        if(sceneName == "DiningHallSceneFinal"){
            GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScript>
().deleteBackButton.SetActive(false);
            GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScript>
().deleteObjectButton.SetActive(false);
            //GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScrip
t>().screenshotButton.SetActive(false);
        }
        headingText.SetActive(false);
        StartCoroutine ("CaptureIt");
    }

    IEnumerator CaptureIt()
    {
        string timeStamp = System.DateTime.Now.ToString("dd-MM-yyyy-HH-mm-ss");
        string fileName = "Screenshot" + timeStamp + ".png";
        string pathToSave = fileName;
        ScreenCapture.CaptureScreenshot(pathToSave);
        yield return new WaitForEndOfFrame();
        Instantiate (blink, new Vector2(0f, 0f), Quaternion.identity);
        SSTools.ShowMessage("Screenshot Saved", SSTools.Position.top, SSTools.Time.twoSec
ond);
        if(sceneName == "BedRoomScene"){
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().deleteBac
kButton.SetActive(true);
            GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().deleteObj
ectButton.SetActive(true);
            //GameObject.Find("BedroomOptions").GetComponent<SelectModelScript>().screens
hotButton.SetActive(false);
        }
        if(sceneName == "DrawingRoomScene"){
            GameObject.Find("DrawingOptions").GetComponent<SelectDrawingRoomModelScript1>
().deleteBackButton.SetActive(true);
            GameObject.Find("DrawingOptions").GetComponent<SelectDrawingRoomModelScript1>
().deleteObjectButton.SetActive(true);
            //GameObject.Find("DrawingOptions").GetComponent<SelectDrawingRoomModelScript
1>().screenshotButton.SetActive(false);
        }
        if(sceneName == "DiningHallSceneFinal"){
            GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScript>
().deleteBackButton.SetActive(true);
            GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScript>
().deleteObjectButton.SetActive(true);
            //GameObject.Find("DrawingOptions1").GetComponent<SelectDrawingRoomModelScrip
t>().screenshotButton.SetActive(false);
        }
        headingText.SetActive(true);
    }

}
```

4.6 CODE TO RETRIEVE SCREENSHOTS FROM LOCATION AND DISPLAY

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

public class ScreenshotPreview : MonoBehaviour {

    [SerializeField]
    GameObject canvas;
    string[] files = null;
    int whichScreenShotIsShown= 0;

    // Use this for initialization
    void Start () {
        files = Directory.GetFiles(Application.persistentDataPath + "/", "*.png");
        if (files.Length > 0) {
            GetPictureAndShowIt ();
        }
    }

    void GetPictureAndShowIt()
    {
        string pathToFile = files [whichScreenShotIsShown];
        Texture2D texture = GetScreenshotImage (pathToFile);
        Sprite sp = Sprite.Create (texture, new Rect (0, 0, texture.width, texture.height
),
            new Vector2 (0.5f, 0.5f));
        canvas.GetComponent<Image> ().sprite = sp;
    }

    Texture2D GetScreenshotImage(string filePath)
    {
        Texture2D texture = null;
        byte[] fileBytes;
        if (File.Exists (filePath)) {
            fileBytes = File.ReadAllBytes (filePath);
            texture = new Texture2D (2, 2, TextureFormat.RGB24, false);
            texture.LoadImage (fileBytes);
        }
        return texture;
    }

    public void NextPicture()
    {
        if (files.Length > 0) {
            whichScreenShotIsShown += 1;
            if (whichScreenShotIsShown > files.Length - 1)
                whichScreenShotIsShown = 0;
            GetPictureAndShowIt ();
        }
    }

    public void PreviousPicture()
    {
        if (files.Length > 0) {
            whichScreenShotIsShown -= 1;
            if (whichScreenShotIsShown < 0)
                whichScreenShotIsShown = files.Length - 1;
            GetPictureAndShowIt ();
        }
    }
}
```

5. REFERENCES

1. https://developer.vuforia.com/
2. www.ceeri.res.in
3. https://answers.unity.com/questions/967128/unable-to-get-horizontal-scroll-bar-to-appear.html
4. https://forum.unity.com/threads/arcore-lean-touch-move-object-in-x-and-z-axis-i-want-to-lock-y-axis.528579/
5. https://stackoverflow.com/questions/57184836/lean-rotate-custom-axis-with-only-one-finger
6. http://carloswilkes.com/Documentation/LeanTouch
7. https://www.reddit.com/r/Unity3D/comments/85gs1n/how_to_create_a_sprite_from_an_png/
8. https://library.vuforia.com/articles/Training/ground-plane-guide.html
9. https://library.vuforia.com/getting-started/overview.html
10. https://www.youtube.com/watch?v=lf8R-xRMYtI
11. https://gamedev.stackexchange.com/questions/106061/how-can-i-detect-a-long-press-then-activate-an-animation/167633
12. https://github.com/sseasycode/SSTools
13. https://docs.unity3d.com/2018.3/Documentation/ScriptReference/EventSystems.EventTrigger.html
14. https://docs.unity3d.com/ScriptReference/Physics2D.Raycast.html?_ga=2.203930256.1871248320.1591074680-1825290488.1589972060
15. https://www.youtube.com/watch?v=DQeylS0l4S4

6. GLOSSARY

1. **AR:** Augmented Reality
2. **VR:** Virtual Reality
3. **SDK:** Software Development Kit
4. **Vuforia:** It is a standard development kit for Unity, which enables the AR libraries functionality
5. **Unity:** A light rendering game engine used to simulate the virtual environment.
6. **VSCode:** Visual Studio Code, used to write scripts in C#.
7. **Image Target:** An image or 3D object uploaded to Vuforia developer platform to enable it to detect the object before augmenting on it.
8. **Ground Plane Detection:** This Vuforia feature enables the app to detect the ground surface.