BUILDING AN AUGMENTED REALITY MOBILE APPLICATION USING REACT NATIVE FOR E-COMMERCE

Akshaya Asok[1], Lekshmy M G[2], Snigdha Santhosh D[3], Sreelekshmi Remesh[4], Dileep V K[5]

[1,2,3,4]Student, Information Technology, LBS Institute of Technology for Women, Poojappura ,Kerala, India
[5]Professor, Dept. of Computer Science and Engineering, LBS Institute of Technology for Women, Poojappura, Kerala, India

---***---

Abstract - *Augmented Reality have emerged as a new technology basically progressing its importance in the field of entertainment and gaming. Although AR has built a huge importance in those two fields, its use in other fields are still to be explored. This paper focuses on bringing AR technology to the field of e-commerce and also building an AR application with react native. React Native is an application framework that provides lots of features, instant updates and allows cross-platform development. So developing an AR application with react native does not compromise in bringing out any of the newer possibilities.*

Key Words: Augmented Reality, React Native, ViroReact, E-commerce, Mobile Application.

1. INTRODUCTION

Augmented Reality (AR) is a growing technology that has enhanced capabilities of displaying a computer generated output with the real world. Augmented reality combines digital image processing, computer graphics, artificial intelligence, multimedia technology and other areas.

AR is going to make the next biggest disruptive change in the 21st century around the world. This paper presents an overview of the fast emerging technology Augmented Reality, its use in e-commerce, along with AR application development using React Native, which is new unlike other platforms like Unity3D or SceneKit which has been largely used in this technology long before.

E-commerce is the process of buying or selling of products by electronic means using websites or online shopping applications on the internet. This method has gained worldwide acceptance as it has made the process of buying and selling goods easier and comfortable. As more and more brands enter the e-commerce spectrum, the competition gets tougher. Innovations have to be made to get past the competition and also help customers to choose wiser.

Many customers have complained of not being satisfied with design, or colour of the product or with it not suiting them when they wear it. For example, a pair of shoes bought from a website may not satisfy the customer if it doesn't suit the person. To overcome this problem, an application is proposed to be developed using the AR technology and ReactNative capabilities for the e-commerce spectrum, the structure of which is explained further in this paper below.

2. LITERATURE REVIEW

The most commonly accepted definitions, researcher Ron Azuma [1] defines that Augmented Reality is technology that has three key requirements [Azuma, 1997]:

1) It combines real and virtual content [1]

2) It is interactive in real time [1]

3) It is registered in 3D [1]

These three characteristics also define the prerequisites for the AR system. An AR system must have a display that can combine real and virtual images. It must be a computer system that can generate interactive graphics that responds to user input in real time. It should also have a tracking system that can find the position of the user's viewpoint and enable the virtual image to appear fixed in the real world [1].

2.1 Types of Augmented Reality

Marker-based AR: Marker-based AR uses a marker or a static image in the physical world that act as a trigger to the augmented experience. The developer can place any digital content over or near the marker when it is identified [2] [3].

Marker-less AR: Marker-less AR works by scanning the surrounding environment and it allows the user to decide where to put the virtual object. It relies on the device's hardware, such as camera, GPS, digital compass etc. [2] [3]

Location-based AR: Unlike the former two types of AR, to determine the user's device location as well as its position, location based apps uses GPS and digital compass of the device. When a user's location matches the predetermined spot, the virtual object is displayed on the screen. Pokemon Go, is an example of location-based AR [2] [4].

Superimposition AR*:* Superimposition AR recognizes an object in the physical world and it changes either partially or completely the view of an object [2].

Projection-based AR: Projection-based AR is different from other types of marker-less augmented reality. It is used to create 3D objects that can be interacted with by the user. To show a prototype of a new product or its inner working, projection based AR can be used [2].

2.2. Augmented Reality SDKs

Augmented Reality SDK provides several components within the AR app i.e AR recognition which is the brain of AR app, AR tracking the eyes and AR content rendering which is the visual treat the user gets to experience through the interface. A set of tools is also provided to developers through SDK which is required for the components to be developed.

AR SDKs can be organized into these categories: Geo-located AR Browsers, Marker based, Natural Feature Tracking. AR Browser SDK is used to create geo-located augmented reality apps using the GPS and IMU in mobile appliances. Marker based SDKs employ special images, markers, to create AR experiences. Natural Feature Tracking SDKs depend on the patterns that are actually present in the real world to add digital content by tracking planar images or based on a SLAM (Simultaneous Location and Mapping) approach [5].

The most commonly known AR SDKs include Metaio, Vuforia, Wikitude, D'Fusion, AR ToolKit, AR Media.

AR ToolKit: The AR toolkit [6] component is comprises class 'ARSceneView.' It is the subclass of 'SceneView' that helps to display an AR experience in the developed application. Augmented Reality framework of the platform is used to display the live camera feed. It synchronises with runtime SDK's 'SceneView' and helps in real world tracking ARCore/ARKit session is started and managed by 'ARSceneView'. To get an initial GPS location or when continuous GPS tracking is required, ARSceneView uses a user-provided 'LocationDataSource'.

ARCore: ARCore helps to build AR apps on **Android**. The three main technologies that the ARCore uses to integrate virtual content with the real world when seen through a device's camera are Motion Tracking, Environmental Understanding, Light Estimation [7].

ARKit: ARKit is Apple's augmented reality (AR) platform for iOS devices which enables app developers to quickly and easily build AR experiences into their apps and games using your iOS device's camera, processors, and motion sensors. ARKit uses Visual Inertial Odometry that helps to keep track of the world around your iPad or iPhone [8].

2.3. React Native JS

React Native [9] is an open-source mobile application framework which enables developers to create real and exciting mobile apps based on JavaScript and React along with native platform capabilities. It is based on ReactJS, a Facebook library for building user interfaces.

With React Native Framework, user interfaces can be rendered in both iOS and Android platforms. In the near future, it will be compatible with other platforms like Windows and tvOS [9] [11].

React Native [9] application development is comparatively simple, faster, efficient and has lower cost of development. Since React Native [11] components have counterpart rights, these components can be reused (just code once) in building both Android and iOS apps which helps to save development time i.e; it helps in Seamless cross platform development and Code reuse. With the help of JavaScript, once a change in code is done, as soon as the code is saved, it iterates the native builds at a lighting speed [9][11].

2.4. ViroReact

ViroReact [10] [12] is an open source developer platform for developers to rapidly build native cross platform AR/VR applications using React Native Framework. It supports both ARKit and ARCore and can be integrated with the existing app. This platform is mainly composed of two main components. One of them is a high performance native 3D rendering engine which runs on native thread and uses graphical hardware chip of device. This rendering engine is Lag free but need to use optimized 3D Files and preferably low-poly files for mobile devices. The supported 3D file formats are .obj, .fbx and .gltf files. The other component is a custom extension of React for AR and VR development.

In ViroReact [10] [12], coding is to be done once and this code can be used across all mobile AR (ARKit, ARCore) and VR (Cardboard iOS/Android, GearVR, Daydream). With the help of a free Viro Testbed app it is easy to view the developed app and to see the changes made quickly i.e; it helps in Rapid Iteration. The usage of declarative markup language makes it an easy-to-learn platform, as well as the usage of Xcode or Android Studio is also not necessary. Developer does not need to create a new API if in future VR features are also to be included in the application, since same API used for AR can be reused in Viro React.

This paper tries to give an overview to create an AR app with the help of Viro components which allows you to use the same codebase in different platforms. ViroReact provides all functionalities that are expected in an AR/VR platform like PBR, HDR, Realtime Lighting, Physics, Particles, 3D Animation etc. Unlike UNITY, which is game-centric, ViroReact is application-centric which needs to be extensible to other platforms also. It is also very easy to upgrade the data from 2D experience to 3D experience or AR.

3. PROPOSED SYSTEM

This application is built using ViroReact, developed from react-native. Before using ViroReact, the user has to make sure that the kev machine react-native-cli and react-viro-cli are installed as global packages using the command ***npm install -g react-native-cli react-viro-cli***. After the installation, use the react-viro cli to create a new project using the special command ***react-viro init***, followed by the project name.

3.1. Project Structure

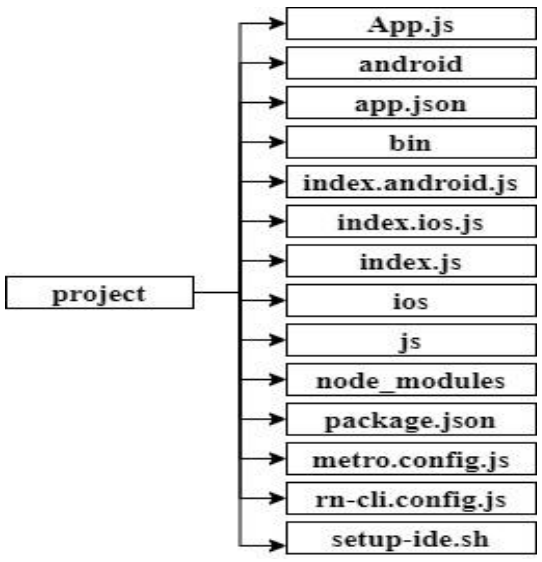

Fig -1: React Native project structure

A basic React Native project structure is shown in figure 1. It includes an **App.js** which is the main Javascript file containing the logic for the application. Project directory contains an **android**, **ios** and **js** directories containing the Android, iOS and Javascript source codes respectively. A miscellaneous React Native property file is also included i.e; **app.json**. **bin** directory contains various scripts which supports the project. **index.android.js** and **index.ios.js** are the legacy files that points to App.js to launch the application for Android and iOS respectively whereas **index.js** is the entry file that points to the main application in App.js. **node_modules** directory contain all the node modules as specified by the **package.json** file. **metro.config.js** file configures the React Native CLI / React Native's metro bundler and **rn-cli.config.js** file is used by older React Native bundler which would be removed in a subsequent version of react-viro, also configures the React Native CLI .

To run it natively on the device, run *./setup-ide.sh*. This script will set up the app development environment for iOS, Android, or both.

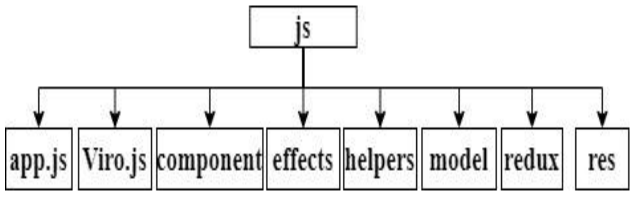

Fig -2: js framework

The JS directory contains the javascript source files for the AR app. The **app.js** file acts as the entry point of the app. This class also connects and orchestrates the interaction between 2D UI components and 3D Viro components using redux. When the app state changes and propagate to **Viro.js** via **redux**, AR Scene will be rendered on the MAIN Screen of the native app. **Viro.js** file renders the AR Scene shown in the App. All 3D Viro Components are handled and rendered in this file. Viro Components such as Objects, Portals, Effects are added, removed and manipulated in the AR scene using 2D RN UI components via redux. In AR scene, **Objects** represent the 3D animating models (OBJ, VRX). **Portals** mainly represent an entry way into a virtual world where the virtual world can be a 360 image / video or a 2D image / video. It can be rendered using **ViroPortal** [12] that represent an entry way into a **ViroPortalScene** [12].

```
<ViroPortalScene>
<ViroPortal position={[0, 0, -1]} scale={[.5, .5, .5]}>
<Viro3DObject
source={require('./res/portalFrame.obj')}
type="OBJ"/>
</ViroPortal>
</ViroPortalScene>
```

Effects include interesting effects such as Particle Emitters or Post Processing for videos. Viro's particle system create and configure quad emitters for building complex and intricate particle effects like smoke, rain, confetti and fireworks.

The **component** directory includes Javascript files for including properties to the native app. It includes an ARIntializationUI component, ButtonComponent, PhotoSelector component etc. The **effects** directory contains a javascript file **effects.js** which is a collection of functions that configure each "Effect" in the app.

The **helpers** directory contains a renderif javascript file in which a Helper function is included for conditional rendering. It takes two arguments: condition (evaluates to TRUE or FALSE) and content (JS content to render if the give condition evaluates to true).

The **model** directory includes Item files with data model for Effects, Models, Portals and Emitters that include particle emitter for candle flame, smoke and Helper functions to add these particle emitters. The **redux** directory contains constants for identifying different Effect names from ListView. Redux actions used to change app state based on events in the app. The **res** folder contains images in **jpg** and **vrx** format.

4. IMPLEMENTATION

4.1. Building Block of AR scene

The main building block of any AR scene is <ViroARSceneNavigator> component in the App.js which acts as the entry point for AR applications. Each SceneNavigator requires a property i.e initialScene which is an object having a scene attribute with the scene component as value [12].

<ViroARSceneNavigator

initialScene={{scene: InitialScene} />

ViroReact uses React Native constructs to create native AR applications. Coding begins by importing React, StyleSheet from React Native and react-viro components like ViroARScene, ViroConstants, ViroMaterials, ViroAmbientLight, ViroDirectionalLight, ViroSpotLight etc. that the app will use.

Create an ES6 class that extends a React Component which cohere to the react component lifecycle. First start with the constructor(), call the super()/parent constructor (in Component) and initialize the state. Below that, "bind" this to the functions declared in this class so that it will reference this object. The render() function determines how the scene is displayed.

4.2. Adding a Scene

Viro Scenes behaves as box for all our UI Objects, Lights and 3D objects. There are 2 types of Scene components i.e **ViroScene** and **ViroARScene** [12]. Each AR scene is a hierarchical tree structure of nodes. These scenes are managed by a 3D scene graph engine. In the return statement, **ViroARScene** is declared. It is the top level component since all the other components are children of ViroARScene.

Sometimes the tracking will be lost in AR apps hence the callback prop, **onTrackingUpdated**, is used to call the **_onInitialized()** function which gets fired, when the app receives AR Tracking state changes from ViroARScene. If the tracking state is not NORMAL then show the user AR Initialization animation which guides them to move the device around to get better AR tracking till the tracking status is **TRACKING_NORMAL**.

<ViroARScene ref="arscene"

physicsWorld={{gravity:[0, -9.81, 0]}}
postProcessEffects={[this.props.postProcessEffects]}
onTrackingUpdated={this._onInitialized}>

An optional property of ViroARScene is **postProcessEffects** that specifies which post-process effects to enable like grayscale, sepia, sincity, barallel distortion, pin cushion distortion, thermal vision, crosshatch, pixelated. The optional property physicsWorld help to contain and process the object which is physically enabled in AR scene. Using physicsWorld, properties such as gravity can also be applied. The **onCameraARHitTest** is performed to find the correct position where to place a new object being added to the scene. It also returns the camera's current orientation, and performs an AR Hit Test with Ray along the camera's orientation. The object is then placed at the intersection of the Ray and identified AR point is returned by the system along that ray.

4.3. Placing a 3D Animating Object/Model to the AR Scene

To place a Model/3D object to the scene the js folder in the root directory must contain a res folder that should have the images of the model in .obj/.vrx/.fbx format **(**.vrx is a FBX format extension passed through Viro compression script**).**

To represent positions and transformations for an object in 3D space, **ViroNode** [12] component can be used. It accepts an array of vectors (x, y, z) for representing position, scale and rotation.

The **Viro3DObject** component helps to create 3D objects that can be placed on the AR scene. These 3D objects are created from 3D structure/texture files. ViroMaterials components consists a set of attributes which defines the appearance of an object's surface when rendered. Each model in a scene can be assigned one or more materials. Usually all UI elements, and most basic 3D models, utilize only one material whereas complex 3D objects, represented by <Viro3DObject>, typically have multiple materials, one for each defined mesh surface in the 3D object.

<Viro3DObject
source={require("./res/object_bday_cake.obj")}
resources={[require('../res/object_bday_cake/blinn1_Roughness.png'),
require('../res/object_bday_cake/blinn1_Metallic.png'),
require('../res/object_bday_cake/blinn1_Normal_OpenGL.png'),

require('../res/object_bday_cake/blinn1_Base_Color.png')] } position={[1, 3, -5]} scale={[.2, .2, .2]} rotation={[0, .165234, 0]} type="OBJ"}/>

The source property of Viro3DObject accepts a remote URL or a local file resource. To specify the 3D model file type being loaded, type property is used.

In the return statement, three Light objects are declared. **ViroAmbientLight** [12] is a light object that emits ambient light that affects all objects equally i.e; without this light, no object is visible. It has properties colour and intensity. **ViroDirectionalLight** [12] pours some light to our scene i.e; it helps in emitting a directional light, directing away from the user, pointed upwards, to light up the "face" of the model. ViroDirectionalLight specifies two props colour and direction. **ViroSpotLight** [12] emits Spotlight on top of the model to highlight it. This SpotLight is placed directly above the 3D Object, directed straight down, and is responsible for creating "shadows" underneath the object in addition to providing lighting. The properties such as innerAngle, outerAngle, attenautionDistances, nearZ, farZ are configured so that it create "as tight as possible" spotlight frustrum around the object for optimizing performance.

<ViroAmbientLight color="#ffffff" intensity={20}/>
<ViroDirectionalLight color="#ffffff" direction={[0,-1,-.2]}/>
<ViroSpotLight innerAngle={5} outerAngle={20} direction={[0,-1,-.2]} position={[0, 5, 1]} color="#ffffff" castsShadow={true} shadowNearZ={.1} attenuationStartDistance={0.1} attenuationEndDistance={22} shadowFarZ={5} shadowOpacity={.9} />

4.4. Placing Objects in Real-World

In an AR app, the device's camera permissions are setup in such a way that it presents the user a live, onscreen view of the real world. 3D virtual objects are superimposed over this view to create an apparition that they actually exist.

The concept of attaching virtual objects to a real-world point is called **Anchoring** [13]. **Anchors** are used for this which are actually vertical or horizontal planes, or images (often markers) found in the real world by the AR system on which we can rely to build a virtual world. Using different detection methods like ViroARPlane, ViroARPlaneSelector etc. an anchor object can be found through markers.

When an AR system discovers a plan, the **ViroARPlane** [12] component allows developers to attach components corresponding to that plane, whereas the **ViroARPlaneSelector** [12] allows the user to choose a plane where the user wants to display the AR component. The **ViroARImageMarker** [12] tries to find the given image target where the user can place the object. When the AR system finds a plane/image target that matches the given dimensions, the plane/image target will be anchored to the real world plane and the child components will be made visible.

To add some animations, import **ViroAnimations**. ViroReact provides a global animation registry called **ViroAnimations** [14] which is a class used to create reusable animations that are applied to a ViroReact component. It accepts an animation prop.

5. RESULTS AND DISCUSSIONS

The following figure3 is a screenshot of the application user interface.

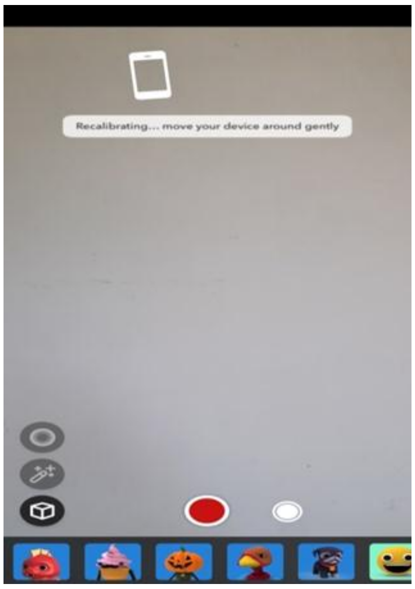

Fig -3: AR Application User Interface js framework

The application's user interface includes an AR initialization UI and user have to move device around until AR tracking is initialized and once initialized, later the user sees a "Re-Calibrating" tag. There are Buttons in the bottom-left menu to switch List View contents between Portals, Objects, Effect and contextual buttons on the right, that gets activated when an object is tapped. User Interface provides a red and a white button at the bottom for recording AR scene and to capture images. User can choose from a variety of AR objects and effects from the individual items in the Listview.

User can take a picture or video of the objects/effects, share the picture user took as shown in figures 4 and 5 and if the user wishes he/she can delete the object.

This application when coming in terms of online marketing strategy provides a new way of interfacing for customers through an AR application. The objects or merchandise for sale is rendered in the application and their AR model is available in it. So customers can select the object from the application and view them through the interface provided. It is like experiencing the product in real world without actually touching it. So, the customer can virtually wear this product to check if it suits them or not thus overcoming the problem of not bring satisfied enough with the product. This method helps in better understanding of the product like when a customer actually wears the product in shopping malls or stores and identifies it as suiting them.

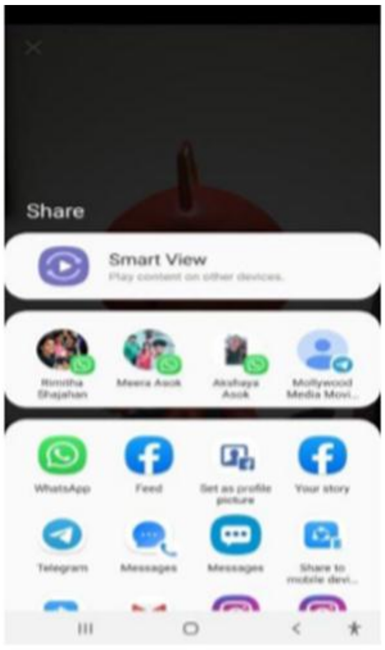

Fig -4: User interface to share the image into other social media applications

Fig -5: Image of an AR object rendered in real world

5.1. React Native Image Crop Picker

A community-made module, react-native-image-crop-picker is used which allows to use native UI to select a photo from the device photo library/gallery or directly from the camera. It can also be used to compress image with quality (from 0 to 1, where 1 is best quality), multiple image selection and cropping of photo.

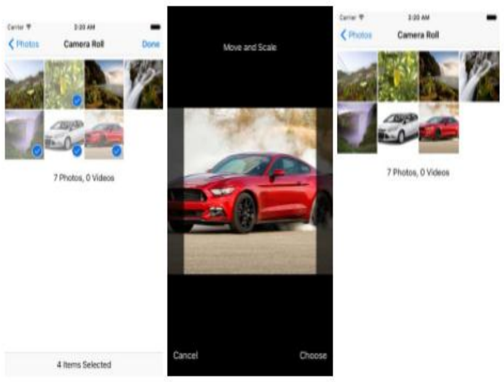

Fig -6: The images of gallery interface that can be provided in the application to select and image from the gallery and projecting it in the real space with the help of AR application.

Source: Adapted from Rathore,D.(2019).7 best ReactNative image picker libraries. https://www.dunebook.com/react-native-image-picker/

This application provides the ability to select an image from the gallery, crop it if needed like shown in figure 6 and project it in onscreen. Now, it has not reached a point of real time rendering of the image and projecting it as an AR object, but can be used to project the available gallery images as such with the help of react-native ***module react-native-crop-image-picker*** [14].

6. FUTURE WORKS THAT CAN BE INCORPORATED

The application has a large platform of innovations to be included to make it better. The improved experience of real time rendering of gallery images can be included to make it convenient for users to select and project an object from gallery.

It can include location based service provided by augmented reality technology. The community-made module ***react-native-geolocation-service*** [13] is used to import the real time longitude and latitude of an area. Using Google map APIs, real time location in react can be imported. This location service can be combined with the location based service in augmented reality to scan an identify real time places using the camera interface in the application, and users can easily navigate to locations. It helps in finding out nearby places in the locality to visit like shops, museum etc. if the user is hostile to the environment. An example is shown in the figure 7.

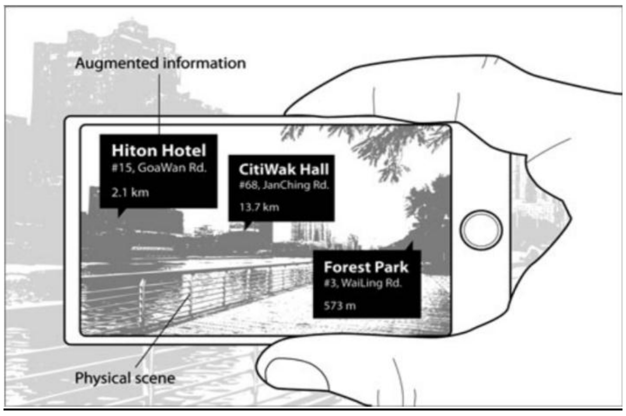

Fig -7: Location based service in Augmented reality

Source: Adapted from Cheng, K. H., & Tsai, C. C. (2013). Affordances of augmented reality in science learning: Suggestions for future research. *Journal of science education and technology*, *22*(4), 449-462.

7. CONCLUSION

Using react for AR based applications is different as React's on the go save, repeat procedure makes the project easy to be dealt with. Viro react library makes project contain less storage as it completely relies on lines of code than bulky software. Viro react also makes it easy if VR need to be incorporated in the project in the future as in this library AR and VR can be rendered using the same API. This application is a boon to many buyers who buys products online without trying them on and later finds it to be a mismatch on them.

ACKNOWLEDGEMENT

This project was supported by LBS Institute of Technology for Women. We thank our guide from LBS Institute of Technology for Women who provided insight and expertise that greatly assisted this project.

We thank Deepu G.S and Gokul.P for assistance with providing us with all the infrastructural facilities and for the proper guidance to React-Native Framework. We would also like to show our gratitude to Deekshit.M, Senior Software Development Manager, Oracle for providing expertise during the course of this project.

REFERENCES

[1] Azuma, R. T. (1997). A survey of augmented reality. Presence: Teleoperators & Virtual Environments, 6(4), 355-385.

[2] Poetker, B., (2019, July 24). What Is Augmented Reality? Most Common Types of AR Used Today. Learning Hub, https://learn.g2.com/augmented-reality.

[3] Gatis, Z., (2019, October 18). Marker-based vs markerless augmented reality: pros, cons & examples. ARinsights. https://overlyapp.com/blog/marker-based-vs-markerless-augmented-reality-pros-cons

[4] Lypchenko, S,.(2019 October 29).How to Build a Location-Based AR Application.AR Post. https://arpost.co/2019/10/29/how-to-build-a-location-based-ar-application

[5] Amin, D., & Govilkar, S. (2015). Comparative study of augmented reality SDKs. International Journal on Computational Science & Applications, 5(1), 11-26.

[6] ARToolKit,https://esri.github.io/arcgis-toolkit-dotnet/ar.html

[7] Introduction to ARCore in Unity, https://codelabs.developers.google.com/codelabs/arcore-intro/#0

[8] Maggie T., Dan G., 2019. Apple ARKit explained: Everything you need to know about Apple's augmented reality platform. https://www.pocket-lint.com/ar-vr/news/apple/141615-apple-ar-kit-explained

[9] Kuitunen, Mika, "Cross-Platform Mobile Application Development with React Native", Tampere University of Technology, December 2018.

[10] ViroReact, https://viromedia.com/viroreact

[11] "React Native: Getting Started," 2016, https://facebook.github.io/react-native/docs/.

[12] "ViroReact-Documentation," October 16, 2019, https://docs.viromedia.com/docs/tutorial

[13] Maciocci, G., Everitt, A. J., Mabbutt, P., & Berry, D. T. (2016). U.S. Patent No. 9,384,594. Washington, DC: U.S. Patent and Trademark Office.

[14] Majchrzak, T., & Grønli, T. M. (2017, January). Comprehensive analysis of innovative cross-platform app development frameworks. In Proceedings of the 50th Hawaii International Conference on System Sciences.

[15] "React Native: Geolocation," 2016, https://facebook.github.io/react-native/docs/geolocation.html.