### CHAPTER 1

### INTRODUCTION

#### 1.1 Android:

Android is a mobile operating system based on a modified version of the Linux kernel and other open-source software, designed primarily for touch screen mobile devices such as smart phones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance, with the main contributor and commercial marketer being Google. Initially developed by Android Inc., which Google bought in 2005, Android was unveiled in 2007, with the first commercial Android device launched in September 2008. The current stable version is Android 10, released on September 3, 2019.

#### Android Architecture:

Android OS is a stack of software components which is roughly divided into five sections and four main layers as shown below in the architecture diagram. GLUT gives your ability to create a window, handle input and render to screen without being OS dependent.

#### Libraries:

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

#### Android Runtime:

This is the third section of the architecture and available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android. The Dalvik VM makes use of Linux core features like memory management and multithreading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine. The Android runtime also provides a set of core libraries which enable Android application developers to write Android applications using standard Java programming language.

**1.2 React Native:**

React Native is an open-source UI software framework created by Meta Platforms, Inc. It is used to develop applications for Android and iOS.

**Architecture of React Native**

The React Native renderer goes through a sequence of work to render React logic to a host platform. This sequence of work is called the render pipeline and occurs for initial renders and updates to the UI state.

The render pipeline can be broken into three general phases:

1. **Render**: React executes product logic which creates a React Element Trees in JavaScript. From this tree, the renderer creates a React Shadow Tree in C++.

2. **Commit**: After a React Shadow Tree is fully created, the renderer triggers a commit. This promotes both the React Element Tree and the newly created React Shadow Tree as the “next tree” to be mounted. This also schedules calculation of its layout information.

3. **Mount**: The React Shadow Tree, now with the results of layout calculation, is transformed into a Host View Tree.

**Cross Platform Implementation**

The React Native renderer utilizes a core render implementation to be shared across platforms.

The current renderer was designed to be a cross-platform solution by sharing a core C++ implementation.

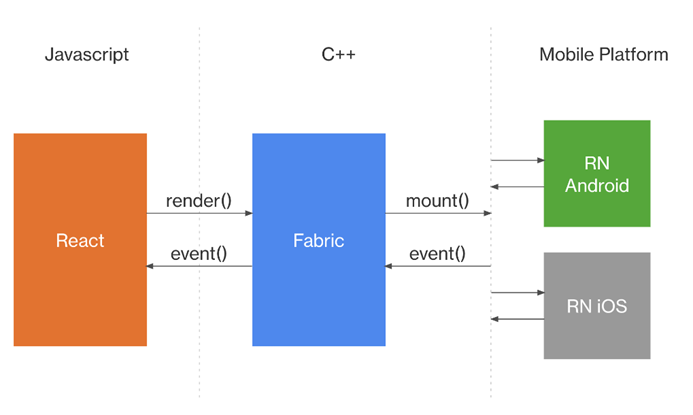
The React Native team intends to incorporate an animation system into the render system and also extend the React Native render system to new platforms such as Windows, and operating systems in game consoles, televisions, and more.

Leveraging C++ for the core render system introduces several advantages. A single implementation reduces the cost of development and maintenance. It improves the performance of creating React Shadow Trees and layout calculation because the overhead of integrating with Yoga the renderer is minimized on Android. Finally, the memory footprint of each React Shadow Node is smaller in C++ than it would be if allocated from Kotlin or Swift.

The team is also leveraging C++ features that enforce immutability to ensure there are no issues related to concurrent access to shared but not protected resources.

The renderer provides two sides of its C++ APIs:

* to communicate with React
* to communicate with the host platform



**Figure 1.2.1 :** React Native Architecture

#### 1.3 Problem statement / Aim:

The main aim of developing this Android application is to provide the user with the ability to attend the exams from anywhere with the suitable internet connection and android device.

#### CHAPTER 2

#### REQUIREMENT SPECIFICATION

The requirement specification is a comprehensive description of both software and hardware requirements to run the project successfully.

* 1. **Hardware Requirements**:
* Processor: AMD Ryzen 5 / Intel Core 9th Gen i5 processor or higher processor
* RAM: 8GB or more
* ROM: 256GB SSD or more
* Input: Keyboard and Mouse
* Display: 1920 X 1080 resolution display or more
* Memory: 4GB
  1. **Software Requirements:**
* Operating System: Window 11
* Languages used: JAVA
* Software: Android Studio

**2.3 Requirements Analysis of System:**

**Feasibility analysis:**

This section verified that it is feasible to add exams and attend the exams on the Android system from the aspects of economic, technical, and social feasibility.

**Economic feasibility:**

To design an Android mobile Examination System App given a computer has the Android development and application development of Android is free. In addition, mobile Examination is basic needs for students and Faculty. The information that which functions are necessary form all the consumers, which functions are needed for some people, and which features are seldom to use is easy to understand. And a lot of research is eliminated, thus saved the spending. Therefore, the whole process of development doesn’t need to spend any money that is economic feasibility.

**Technical feasibility**:

To design a Examination System appl which meets the basic requirements, a deep understand of JAVA language, the Android system architecture, application of framework and other technical knowledge are needed. (Framework is the core of the application, and rules that all the programmers participating in the development must abide by).

### CHAPTER 3

### IMPLEMENTATION

#### Project description:

Implementation is the stage where all planned activities are put into action. Before the implementation of a project, the implementors (spearheaded by the project committee or executive) should identify their strength and weaknesses (internal forces), opportunities and threats (external forces).

# Create an Examination System App in React Native:

Examination app is built using React Native Where its an Cross Platform App development framework where it convert the native code to java built code for android For Backend we are using the Nodejs for development of API where it directly interacts with the database and fetch the data and send the data to the database. Front end of the app call and request and gets the response from the API.

**About** **Examination System App:**

In Examination App Both students and staff have different gateway to access the workspace. Here staff can allocate the exams in the form of quiz with option and students those who wants to attend the exam. Students can open the app and attend the allocated exams.

**Features Of Examination System App:**

1. Different login space for both students and staff.
2. Staff can add the exam for the selected students.
3. Students can attend the allotted exams.
4. History of all attended exam will be stored in the Database.
5. Ensures faster results.

#### Flowchart:

Diagram

Description automatically generated

**Figure 3.2.1 Flowchart**

#### Source Code:

QUIZ.js

// Reads The Questions From Database and Display Them

import React, { useState, useEffect } from 'react';

import axios from 'axios';

import { View, StyleSheet, Text, Button } from 'react-native';

import { CurrentRenderContext } from '@react-navigation/native';

import { useNavigation } from '@react-navigation/native';

import storage from '../../FACULTY/auth/StorageHelper';

const API\_URL = Platform.OS === 'web' ? 'http://localhost:5000' : 'http://10.0.2.2:5000';

const Quiz = () => {

const [data, setData] = useState([]);

const [number, setNumber] = useState(0);

const [pts, setPts] = useState(0);

const [email, setEmail] = useState('');

const [quizid, setQuizid] = useState('');

const navigation = useNavigation();

const shuffle = (a) => {

a.sort(() => Math.random() - 0.5);

}

const pickAnswer = (ans) => {

if(data[number].answer === ans ) {

setPts(pts + 1);

}

setNumber(number + 1);

}

console.log(pts);

useEffect(() => {

storage.load({ key: 'Suser', })

.then(ret1 => {

storage.load({ key: 'SQuizId' })

.then(ret2 => {

console.log(ret1+"abc"+ret2);

setEmail(ret1);

setQuizid(ret2);

axios.post(`${API\_URL}/student/viewQuiz`, {

quizid: ret2

})

.then(res => {

console.log(res.data.result);

console.log(res.data.result);

setData(res.data.result.map(item => (

{

question: item.question,

answer: item.answer,

options: [item.op1, item.op2, item.op3, item.op4]

})));

})

})

.catch(err => {

console.warn(err.message);

});

})

.catch(err => {

console.warn(err.message);

}

)}, []);

console.log(data);

return (

<View style={styles.background}>

{

data[number] &&

<View style={styles.QuizWindow} >

<Text style={styles.question}>{data[number].question}</Text>

<View style={styles.button}>

{

data[number].options.map(item => {

return (

<View>

<Button key={item} title={item} onPress={() => pickAnswer(item)} />

<View style={styles.space} />

</View>

)

})

}

</View>

</View>

}

{

number === data.length &&

<View style={styles.QuizWindow} >

<Text style={styles.question}>Your Score: {pts}</Text>

<Button style = {styles.button} title="Restart Quiz" onPress={() => {

setNumber(0);

setPts(0);

}

}/>

</View>

}

{

number === data.length &&

<View style={styles.QuizWindow}>

<Button title="Submit Quiz" onPress={() => {

alert("Submit Quiz");

axios.post(`${API\_URL}/student/insertMarks`, {

quizid: quizid,

email: email,

marks: pts,

total : data.length

}).then(res => {

console.log(res.data.result);

alert(res.data.result);

}

)

}

}/>

</View>

}

</View>

)

}

const styles = StyleSheet.create({

QuizWindow: {

paddingTop : 150,

alignItems: 'center',

justifyContent: 'center',

padding: 20

},

question: {

marginBottom: 25,

textAlign: 'center',

FontFace :'sans-serif',

backgroundColor: 'red',

width: '100%',

color : 'black',

borderRadius: 15,

},

button: {

display: 'flex',

flexDirection: 'column',

justifyContent: 'space-around',

marginTop: 20,

},

text: {

fontSize: 20,

marginBottom: 20,

textAlign: 'center'

}});

export default Quiz;

//Add Question.js

import React, {useEffect, useState} from 'react'

import axios from "axios";

import {View, Text, StyleSheet, TouchableOpacity, TextInput, Alert} from 'react-native'

const API\_URL = Platform.OS === 'web' ? 'http://localhost:5000' : 'http://10.0.2.2:5000';

import storage from '../auth/StorageHelper';

import { useNavigation } from '@react-navigation/native';

import DropDownPicker from 'react-native-dropdown-picker';

export default function AddQuestion() {

const navigation = useNavigation();

const [open, setOpen] = useState(false);

const [value, setValue] = useState(null);

const [quizid, setQuizid] = useState('');

const [state , setState] = useState({

question: '',

option1: '',

option2: '',

option3: '',

option4: '',

answer: '',

quizid:''

});

const handleChange = (e) => {

setState({

...state,

[e.target.name] : e.target.value

})

}

//clear the form

const clearForm = () => {

setState({

question: '',

option1: '',

option2: '',

option3: '',

option4: '',

answer: '',

quizid:''

})

}

useEffect(() => {

storage.load({key: 'FQuizId'})

.then(ret => {

// console.log('AddStudent')

// console.log(ret+'1');

setQuizid(ret);

}

)

.catch(err => {

console.warn(err.message);

}

);

}

,[])

const handleSubmit = (e) => {

e.preventDefault();

// props.addQuestion(state);

const payload = {

question: state.question,

option1: state.option1,

option2: state.option2,

option3: state.option3,

option4: state.option4,

answer:value,

quizid: quizid

}

alert(payload.answer)

axios.post(`${API\_URL}/faculty/enterquestion`, payload)

.then(res => {

if (res.status !== 200) {

Alert('Data Not Sent');

} else {

alert('Data Sent');

}

}

).catch(err => {

Alert(err.message);

}

);

}

//add new question

const addnQ = () => {

clearForm()

}

const endQues = () =>{

alert('once clicked u cant insert question')

storage.remove({key: 'FQuizId'});

navigation.navigate('FHome');

}

return (

//React Native Elements To Add Questions

<View style={styles.container}>

<View style={styles.header}>

<Text style={styles.headerText}>Add Question</Text>

</View>

<View style={styles.form}>

<TextInput

style={styles.input}

placeholder="Enter Question"

name="question"

onChangeText={(question) => handleChange({target: {name: 'question', value: question}})}

value={state.question}

/>

<TextInput

style={styles.input}

placeholder="Enter Option 1"

name="option1"

onChangeText={(option1) => handleChange({target: {name: 'option1', value: option1}})}

value={state.option1}

/>

<TextInput

style={styles.input}

placeholder="Enter Option 2"

name="option2"

onChangeText={(option2) => handleChange({target: {name: 'option2', value: option2}})}

value={state.option2}

/>

<TextInput

style={styles.input}

placeholder="Enter Option 3"

name="option3"

onChangeText={(option3) => handleChange({target: {name: 'option3', value: option3}})}

value={state.option3}

/>

<TextInput

style={styles.input}

placeholder="Enter Option 4"

name="option4"

onChangeText={(option4) => handleChange({target: {name: 'option4', value: option4}})}

value={state.option4}

/>

<DropDownPicker

items={[

{label: 'Option 1', value:state.option1},

{label:'Option 2', value : state.option2},

{label: 'Option 3', value : state.option3},

{label: 'Option 4', value: state.option4}

]}

open={open}

value = {value}

setOpen={setOpen}

setValue={setValue}

/>

<TouchableOpacity

style={styles.button}

onPress={handleSubmit}

>

<Text style={styles.buttonText}>Add Question</Text>

</TouchableOpacity>

<TouchableOpacity

style={styles.button}

onPress={addnQ}

>

<Text style={styles.buttonText}>Add More Question</Text>

</TouchableOpacity>

<TouchableOpacity

style={styles.button}

onPress={endQues}

>

<Text style={styles.buttonText}>End Quiz</Text>

</TouchableOpacity>

</View>

</View>

)

}

const styles = StyleSheet.create({

container: {

flex: 1,

backgroundColor: '#fff',

alignItems: 'center',

justifyContent: 'center'

},

header: {

backgroundColor: '#0066ff',

alignItems: 'center',

justifyContent: 'center',

height: 100,

width: '100%',

marginTop: 20

},

headerText: {

color: '#fff',

fontSize: 30,

fontWeight: 'bold'

}

,

form: {

width: '100%',

padding: 20,

marginTop: 20

}

,

input: {

borderWidth: 1,

borderColor: '#0066ff',

padding: 10,

marginTop: 10,

height : 50,

width: '100%'

}

,

button: {

backgroundColor: '#0066ff',

padding: 10,

marginTop: 10,

borderRadius: 10,

height: 50,

width: '100%'

}

,

buttonText: {

color: '#fff',

fontSize: 20,

fontWeight: 'bold'

}

})

### CHAPTER 4

### SNAPSHOTS

|  |  |
| --- | --- |
|  |  |
| |  |  |  | | --- | --- | --- | | **Figure 4.1: Home Page**  This contains options to login as  A Student or a Faculty. |  | **Figure 4.2: Signup Page**  Sign Up page Were Student and Faculty Signup. | |  |  |  | |  |

|  |  |
| --- | --- |
|  |  |
| |  |  |  | | --- | --- | --- | | **Figure 4.3: Add Exam Page**  Faculty can add the exams to the students. |  |  | |  |  |  | | **Figure 4.4: Profile Page**  Users after login can view this  page |

|  |  |
| --- | --- |
|  |  |
| **Figure 4.5 : Home Page**  This is Home Page after the user is  Login.   |  |  |  | | --- | --- | --- | |  |  | . | |  |  |  | | **Figure 4.6: Result Page**  This page contains the result of students  who attended the exams. |
|  |  |

|  |  |
| --- | --- |
|  |  |
| |  |  |  | | --- | --- | --- | | **Figure 4. 7: Result Page**  This contains the Result of previously attended exams.  . |  | **Figure 4.8: View Exam Page**  Students can view the specified exam | |  |  |  | |  |

**CONCLUSION**

The Examination system provides better functionality for an examination to be more efficient and reduce manual paperwork by automating all tasks.

This development implicated the popular mobile terminal development technology. This is the combination management of Java and C++ in the open-source mobile platform based on Linux system configuration file. The system realized the Examination System.

This design of Examination System is based on Android system requires elaborate design of the React native framework, by adopting ANDROID STUDIO 2021.2.1 + Java language as technical support of this system, with the Android plug-in tools, and combination of Latest Android SDK version and Nodejs version led to the comprehensive and smoothly design and development of the mobile terminal.

### REFERENCES

1. Google Developer Training, "Android Developer Fundamentals Course – Concept Reference”, Google Developer Training Team, 2017.

https://www.gitbook.com/book/google-developer-training/android-developer-fundamentalscourse-concepts/details

1. https://nodejs.org/en/docs/guides/
2. https://www.github.com/
3. https://www.stackoverflow.com/
4. <https://reactnative.dev/docs/getting-started>/
5. https://www.freecodecamp.org/