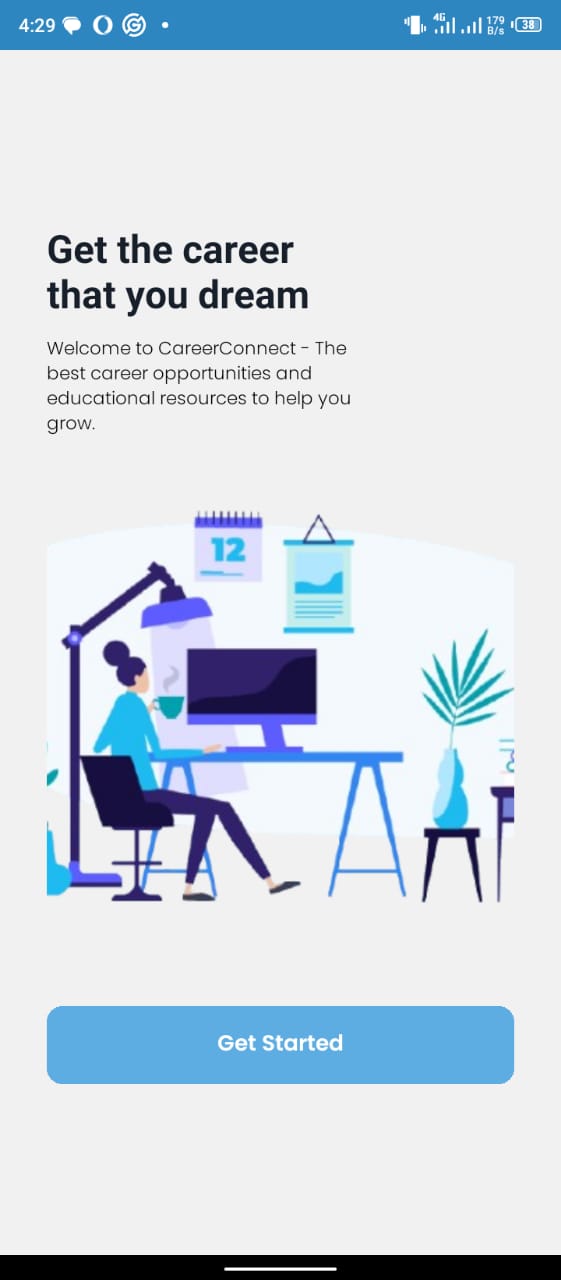
CHAPTER FOUR

SYSTEM IMPLEMENTATION AND DESIGN

* + 1. **LANDING PAGE**

This page is the welcome page about the applications.



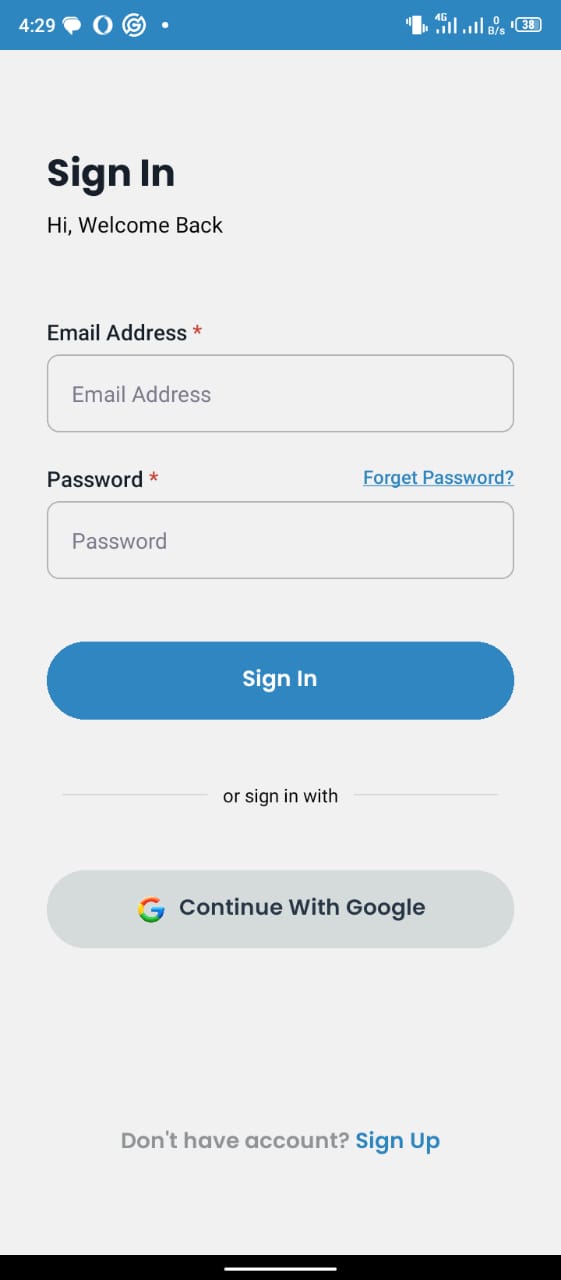
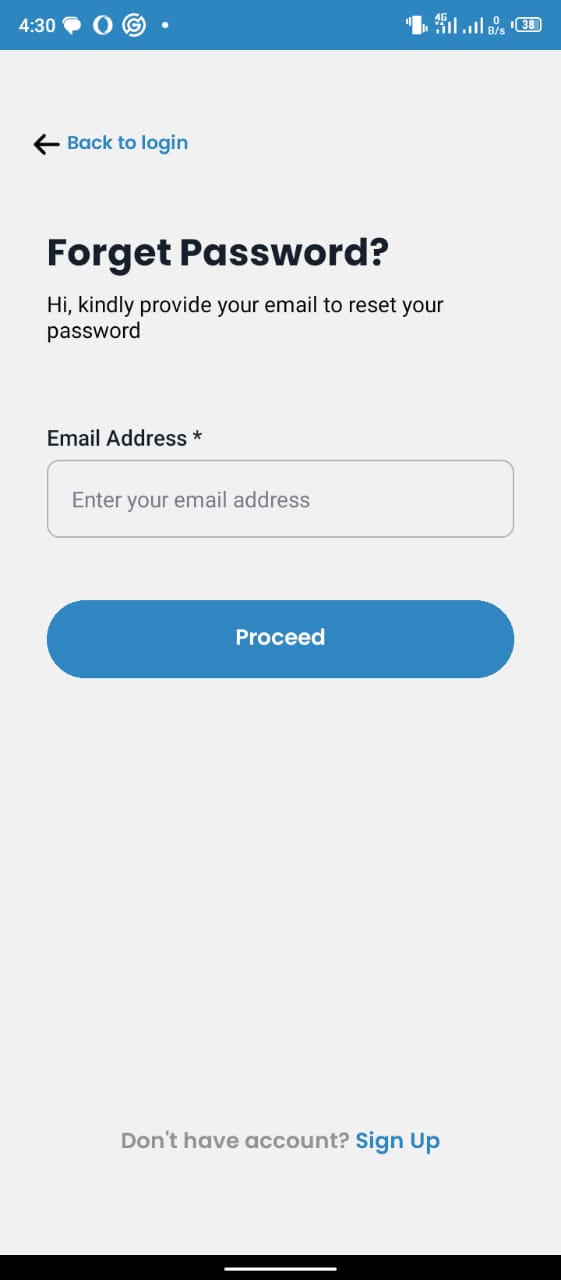
**FIGURE: 1.6 LANDING PAGE**

* + 1. **AUTHENTICATION MANAGEMENT**

Every user who wants to use this software (Career Connect App) by means of email address and password which is registered by the user. This authentication form page is required email as username and password and the reset password page

**FORGET PASSWORD PAGE**

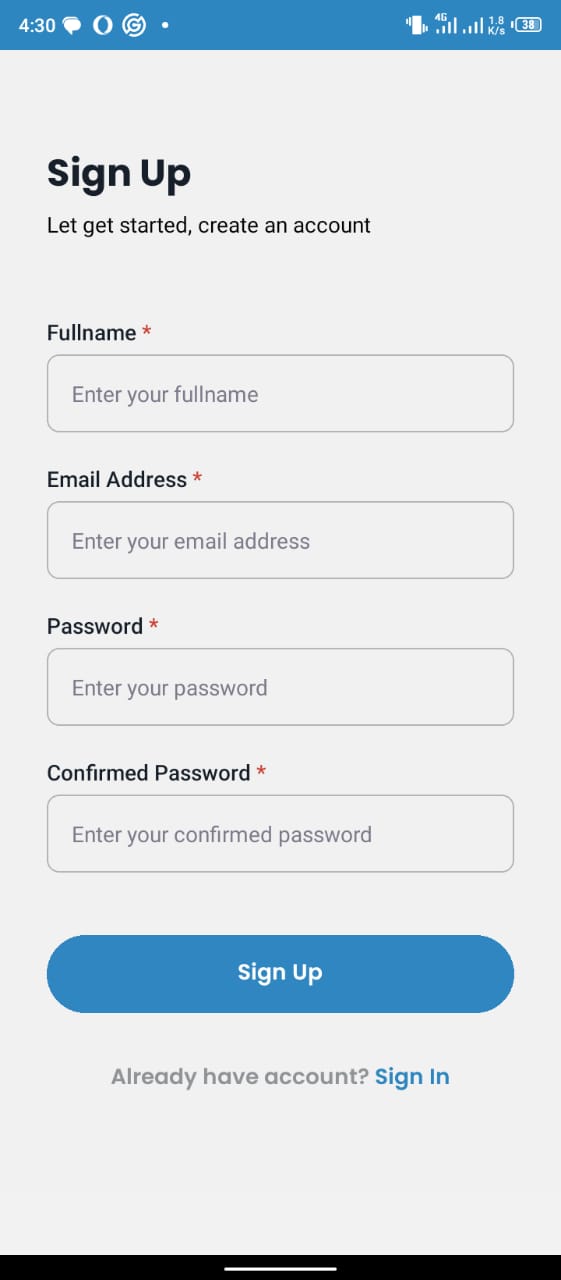
**SIGN IN PAGE**



**FIGURE: 1.7 AUTHENTICATION MANAGEMENT**

* + 1. **SIGN UP FORM PAGE**

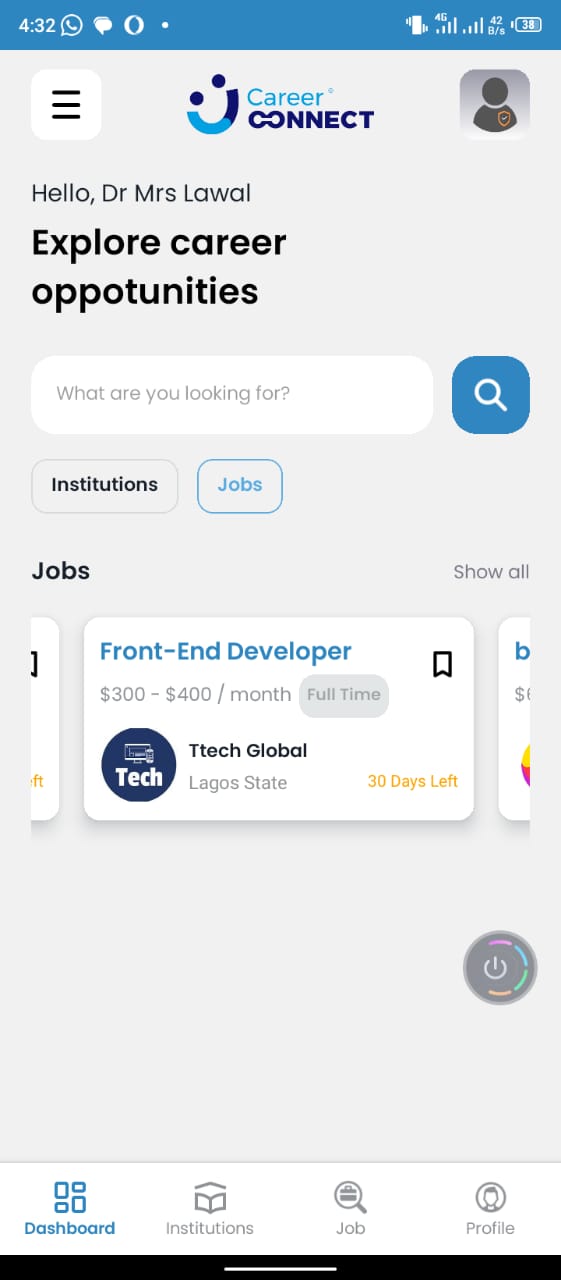
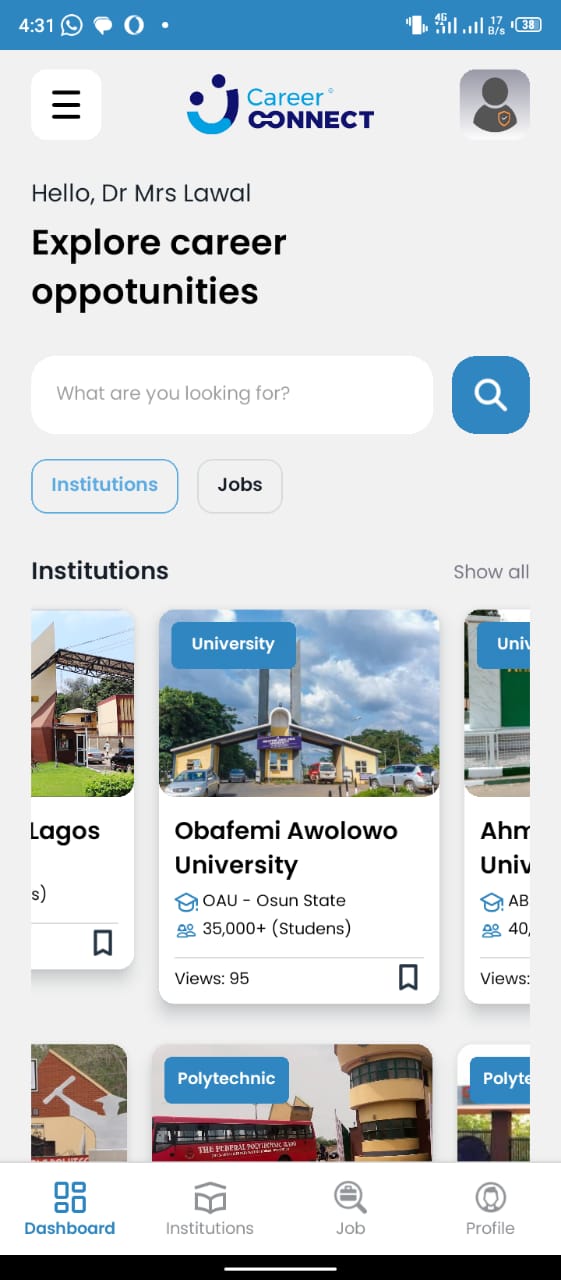
This page is where user to create an account with their fullname, email, and password.



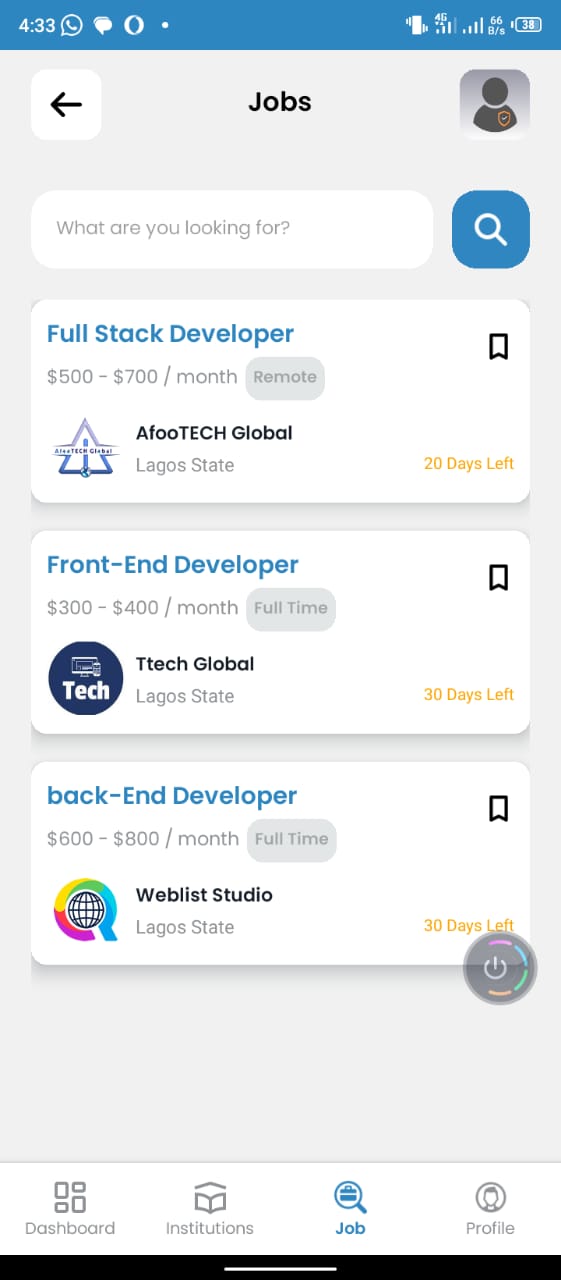
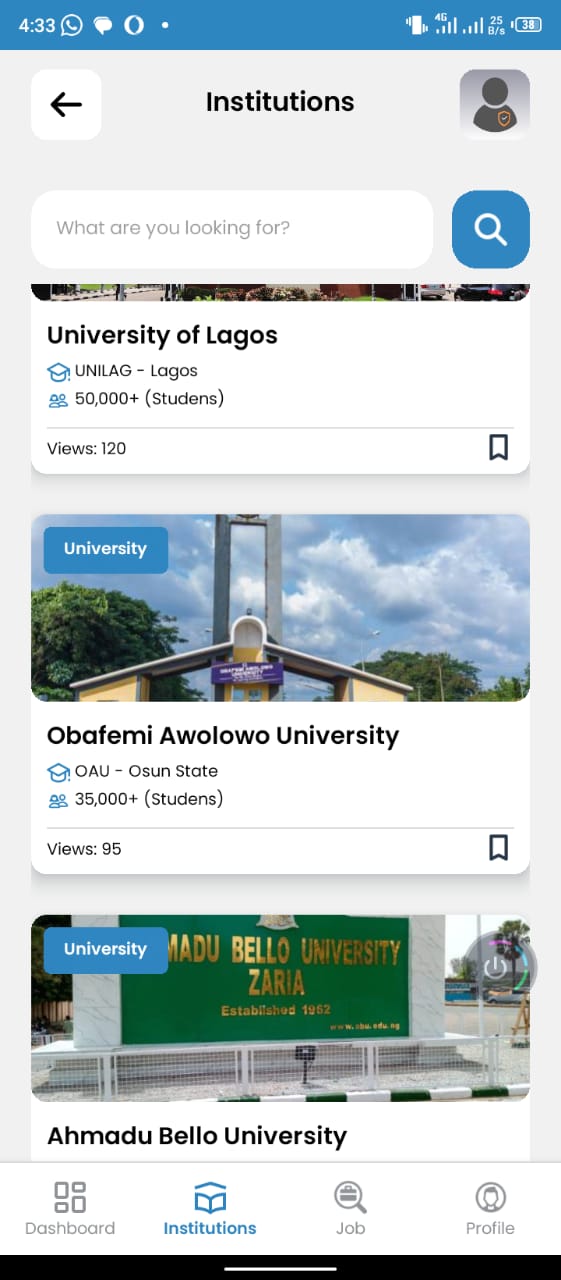
**FIGURE: 1.8 SIGN UP FORM PAGE**

* + 1. **THE USER DASHBOARD PAGE**

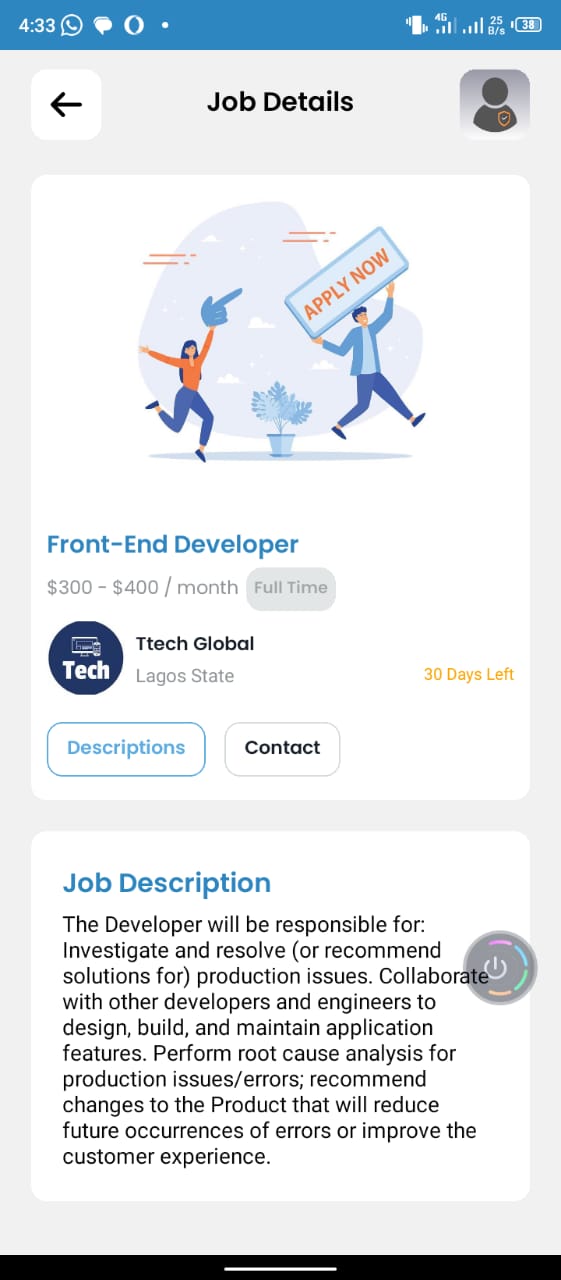
This page is the user dashboard page to access different modules in the application. Modules like Institutions, Jobs, Profile, and Latest news.

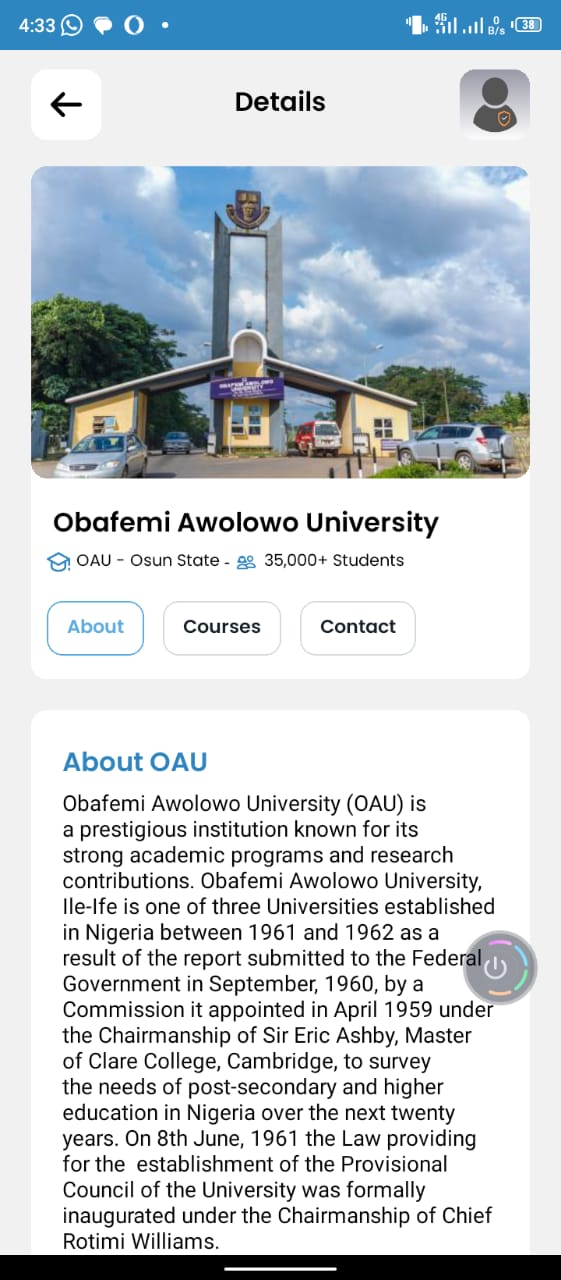
****

**FIGURE: 1.9 USER DASHBOARD**

****

**FIGURE: 1.10 INSTITUTITIONS AND JOBS PAGE**

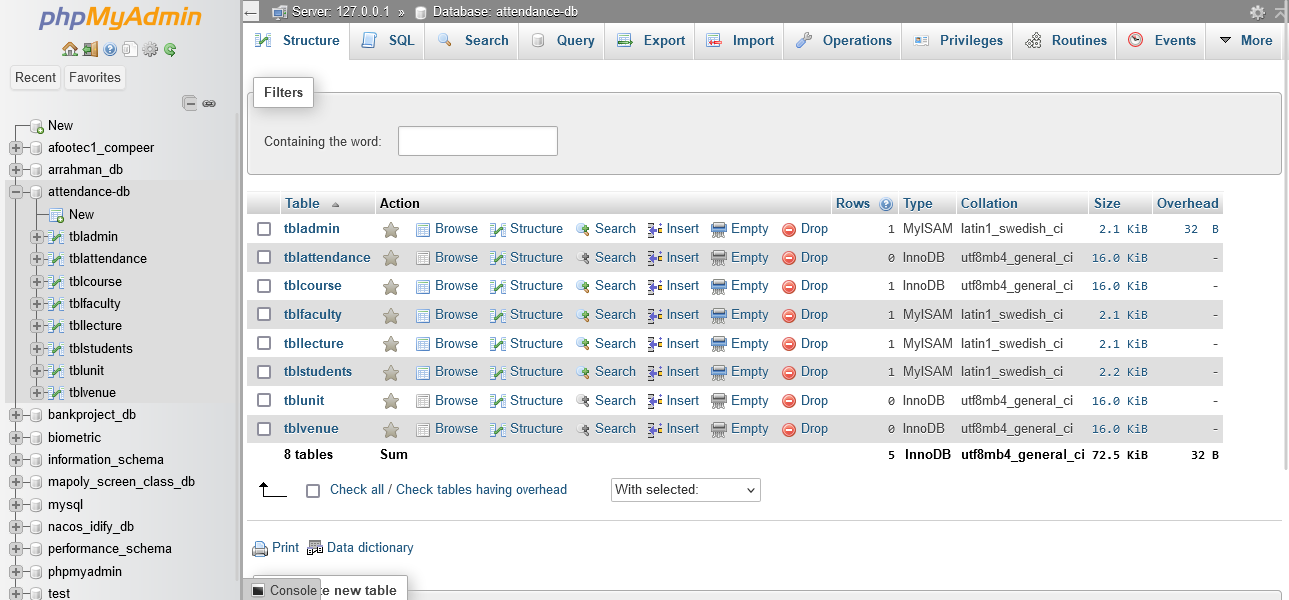
****

****

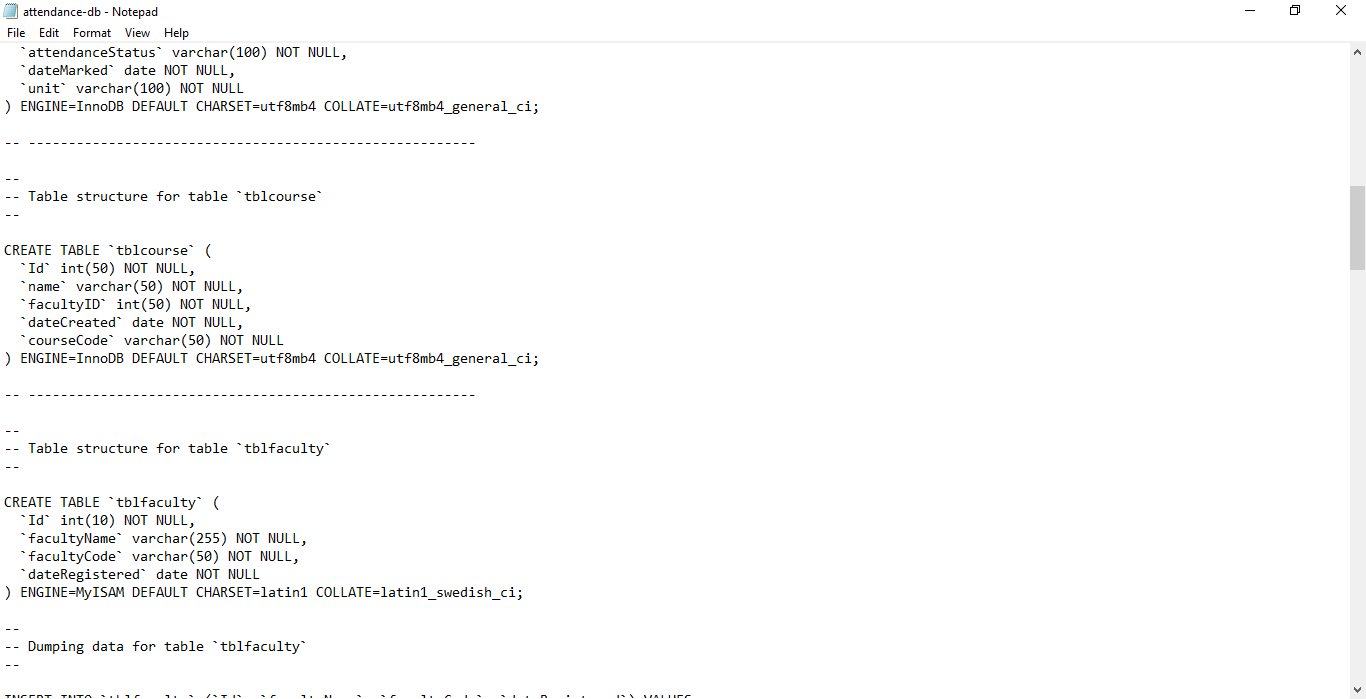
**FIGURE: 1.11 INSTITUTITIONS AND JOBS DETAIL PAGE**

**4.1.5 DATABASE DESIGN**

MySQL database was used in the design. There are ten tables used in the design. The student table, institution table, job table, user table.



**FIGURE: 1.12 DATABASE DESIGN**



**FIGURE: 1.13 MYSQL DATABASE DESIGN**

4.2 SYSTEM REQIREMENT

## Hardware Requirements

## In the design, the software developed needs the following hardware for an effective and efficient operation of the new system.

## Mobile device (Andriod or IOS).

1. At least 2GB RAM.
2. Storage(120 gb)

**Software Requirements**

The software requirements include: -

1. Phone Operating System (Android or IOS)

1. Database server (apache)
2. Graphical application (Adobe Firework)
3. Google Play Store and Apple Store

**Operational Requirement**

For the new system to be operational, Internet access is needed in the mobile devices.

**Personnel Requirement**

A mobile devices with internet access connection

**APPENDIX**

**LANFING PAGE SOUCE CODE**

import { StatusBar } from "expo-status-bar";

import { useRouter } from "expo-router";

import { View, Text, ImageBackground, ScrollView } from "react-native";

import { SafeAreaView } from "react-native-safe-area-context";

import styles from "../mainStyles/welcome.style";

import { images, COLORS } from "../../constants";

import { CustomButton} from "../../components";

const Welcome = () => {

const router = useRouter();

return (

<SafeAreaView style={styles.mainContainer}>

<ScrollView contentContainerStyle={{ height: "100%" }}>

<View style={styles.innerContainer}>

<View style={styles.topTextContainer}>

<Text style={styles.topText}>Get the career that you dream</Text>

<Text style={styles.subText}>Welcome to CareerConnect - The best career opportunities and educational resources to help you grow.</Text>

</View>

<View style={styles.bgContainer}>

<ImageBackground

source={images.bgImage}

style={styles.imgaeBg}

resizeMode="cover"

/>

</View>

<CustomButton

title="Get Started"

handlePress={() => router.push("/(auth)/sign-in")}

/>

</View>

</ScrollView>

<StatusBar backgroundColor={COLORS.primaryColor} style="light" />

</SafeAreaView>

);

};

export default Welcome;

**LOGIN PAGE SOURCE CODE**

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

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

**import { SafeAreaView } from "react-native-safe-area-context";**

**import { useRouter } from "expo-router";**

**import { useGlobalContext } from "../../context/GlobalProvider";**

**import { isValidEmail } from "../../lib/functions/functions";**

**import { CustomButton } from "../../components";**

**import styles from './auth.style';**

**import { COLORS } from "../../constants";**

**//import CheckBox from "react-native-checkbox";**

**const SignIn = () => {**

**const { setUserData, isLogged, setIsLogged } = useGlobalContext();**

**if ( isLogged) return <Redirect href="/dashboard" />;**

**const router = useRouter();**

**const [isLoading, setIsLoading] = useState(false);**

**const [form, setForm] = useState({**

**email: '',**

**password: ''**

**});**

**const [errors, setErrors] = useState({ ...form });**

**const handleLogIn = async () => {**

**let isValid = true;**

**let newErrors = { email: '', password: '' };**

**// Check if email is empty**

**if (!form.email) {**

**newErrors.email = "Please fill in the email field.";**

**isValid = false;**

**} else if (!isValidEmail(form.email)) {**

**newErrors.email = "Invalid email address, try again";**

**isValid = false;**

**}**

**// Check if password is empty**

**if (!form.password) {**

**newErrors.password = "Please fill in the password field.";**

**isValid = false;**

**}**

**setErrors(newErrors);**

**setIsLoading(true);**

**try {**

**if (isValid) {**

**// const userLoginDataSession = await AsyncStorage.getItem('userLoginDataSession');**

**// console.log(userLoginDataSession);**

**///await AsyncStorage.setItem('userLoginDataSession', JSON.stringify(userLoginDataSession[0])); // Store user data or token**

**setIsLogged(true);**

**// setUserData(userLoginDataSession);**

**Alert.alert("SUCCESS", "Login Successful");**

**router.push(`/(tabs)/dashboard`);**

**setForm("");**

**}**

**} catch (error) {**

**Alert.alert('Error', 'Something went wrong!');**

**} finally {**

**setIsLoading(false);**

**}**

**};**

**return (**

**<SafeAreaView style={styles.formContainer}>**

**<View style={styles.innerForm}>**

**<View style={styles.textContainer}>**

**<Text style={styles.topText}>Sign In</Text>**

**<Text style={styles.subTopText}>Hi, Welcome Back</Text>**

**</View>**

**<View style={styles.inputContainer}>**

**<Text style={styles.inputLabel}>Email Address <Text style={styles.errorText}>\*</Text></Text>**

**<TextInput**

**style={[styles.inputField, errors.email ? styles.required : null]} // Apply error style if email has an error**

**placeholder="Email Address"**

**placeholderTextColor={COLORS.textFieldColor}**

**value={form.email}**

**onChangeText={(value) => setForm({ ...form, email: value })}**

**/>**

**{errors.email ? <Text style={styles.errorText}>{errors.email}</Text> : null}**

**</View>**

**<View style={styles.inputContainer}>**

**<Text style={styles.inputLabel}>Password <Text style={styles.errorText}>\*</Text></Text>**

**<TouchableOpacity activeOpacity={.8} style={styles.forgetPass}>**

**<Text style={styles.forgetPass} onPress={() => router.push("/(auth)/forget-password")}>Forget Password?</Text>**

**</TouchableOpacity>**

**<TextInput**

**style={[styles.inputField, errors.password ? styles.required : null]}**

**placeholder="Password"**

**placeholderTextColor={COLORS.textFieldColor}**

**secureTextEntry={true}**

**value={form.password}**

**onChangeText={(value) => setForm({ ...form, password: value })}**

**/>**

**{errors.password ? <Text style={styles.errorText}>{errors.password}</Text> : null}**

**</View>**

**{/\* <View style={styles.container}>**

**<CheckBox**

**value={isChecked}**

**onChange={setIsChecked}**

**checkboxStyle={[styles.checkBox, !isChecked ? styles.cbFalse : styles.cbTrue]}**

**/>**

**<Text>Keep me signing in</Text>**

**</View> \*/}**

**<CustomButton**

**isLoading={isLoading}**

**title={isLoading ? "Signing In..." : "Sign In"}**

**handlePress={handleLogIn}**

**containerStyles={styles.button}**

**/>**

**<View style={styles.footer}>**

**<View style={styles.ftIn}>**

**<Text style={styles.ftLines}></Text>**

**<Text>or sign in with</Text>**

**<Text style={styles.ftLines}></Text>**

**</View>**

**<CustomButton**

**title="Continue With Google"**

**icon={true}**

**textStyles={styles.ftTextBtn}**

**// handlePress={() => router.push("/(auth)/sign-up")}**

**containerStyles={styles.ftBtn}**

**/>**

**</View>**

**<TouchableOpacity TouchableOpacity={.8} onPress={() => router.push("/sign-up")}>**

**<Text style={styles.ftText}>Don't have account? <Text style={{ color: COLORS.primaryColor }} >Sign Up</Text></Text>**

**</TouchableOpacity>**

**</View>**

**</SafeAreaView>**

**)**

**}**

**export default SignIn**

**DASHBOARD PAGE SOURCE CODE**

**import React, { useState } from "react";**

**import { SafeAreaView } from 'react-native-safe-area-context';**

**import { ScrollView, FlatList, View, Text, TouchableOpacity } from "react-native";**

**import { StatusBar } from "expo-status-bar";**

**import { Redirect, useRouter } from "expo-router";**

**import { COLORS, SIZES } from "../../constants";**

**import { Header, Welcome, SchoolList, JobList } from "../../components";**

**import { useGlobalContext } from "../../context/GlobalProvider";**

**import { Universities, Polytechnic, College, jobs } from "../../lib/data";**

**import styles from "../mainStyles/tab.style";**

**//import AsyncStorage from '@react-native-async-storage/async-storage';**

**const tabDetails = ["Institutions", "Jobs"];**

**const Dashboard = () => {**

**const { userData, isLogged } = useGlobalContext();**

**if (!isLogged) return <Redirect href="/sign-in" />;**

**const router = useRouter();**

**const [activeTab, setActiveTab] = useState("Institutions");**

**const [selectedSchool, setSelectedSchool] = useState();**

**const handleSchoolPress = (item, index) => {**

**// router.push(`/school-details/${index}`);**

**router.push(`/school-details/detail?categories\_school=${item.categories\_school}&index=${index}`);**

**setSelectedSchool(item.school\_id);**

**};**

**const [selectedJob, setSelectedJob] = useState();**

**const handleJobsPress = (item, index) => {**

**router.push(`/job-details/${index}`);**

**setSelectedJob(item.job\_id);**

**};**

**const displayTabContent = () => {**

**switch (activeTab) {**

**case "Institutions":**

**return (**

**<>**

**<View style={styles.subHeaderText}>**

**<Text style={styles.leftText}>{activeTab}</Text>**

**<TouchableOpacity activeOpacity={.7} onPress={() => router.push("/school")}>**

**<Text style={styles.rightText}>Show all</Text>**

**</TouchableOpacity>**

**</View>**

**<FlatList**

**data={Universities}**

**renderItem={({ item, index }) => (**

**<SchoolList**

**schoolCategory={item.categories\_school}**

**item={item}**

**index={index}**

**selectedSchool={selectedSchool}**

**handlePress={handleSchoolPress}**

**/>**

**)}**

**keyExtractor={(item, index) => item.school\_id || index.toString()} // Ensure unique keys**

**contentContainerStyle={{ columnGap: SIZES.medium }}**

**horizontal**

**showsHorizontalScrollIndicator={false}**

**/>**

**<FlatList**

**data={Polytechnic}**

**renderItem={({ item, index }) => (**

**<SchoolList**

**schoolCategory={item.categories\_school}**

**item={item}**

**index={index} // Pass index to the component if needed**

**selectedSchool={selectedSchool}**

**handlePress={handleSchoolPress}**

**/>**

**)}**

**keyExtractor={(item, index) => item.school\_id || index.toString()} // Ensure unique keys**

**contentContainerStyle={{ columnGap: SIZES.medium }}**

**horizontal**

**showsHorizontalScrollIndicator={false}**

**/>**

**<FlatList**

**data={College}**

**renderItem={({ item, index }) => (**

**<SchoolList**

**schoolCategory={item.categories\_school}**

**item={item}**

**index={index} // Pass index to the component if needed**

**selectedSchool={selectedSchool}**

**handlePress={handleSchoolPress}**

**/>**

**)}**

**keyExtractor={(item, index) => item.school\_id || index.toString()} // Ensure unique keys**

**contentContainerStyle={{ columnGap: SIZES.medium }}**

**horizontal**

**showsHorizontalScrollIndicator={false}**

**/>**

**</>**

**);**

**case "Jobs":**

**return (**

**<>**

**<View style={styles.subHeaderText}>**

**<Text style={styles.leftText}>{activeTab}</Text>**

**<TouchableOpacity activeOpacity={.7} onPress={() => router.push("/job")}>**

**<Text style={styles.rightText}>Show all</Text>**

**</TouchableOpacity>**

**</View>**

**<FlatList**

**data={jobs}**

**keyExtractor={(item) => item.job\_id || index.toString()}**

**renderItem={({ item, index }) => (**

**<JobList**

**item={item}**

**index={index}**

**selectedJob={selectedJob}**

**handlePress={handleJobsPress}**

**/>**

**)}**

**keyboardShouldPersistTaps="handled"**

**contentContainerStyle={{ columnGap: SIZES.medium, marginTop: 15 }}**

**horizontal**

**showsHorizontalScrollIndicator={false}**

**/>**

**</>**

**);**

**default:**

**return null;**

**}**

**};**

**return (**

**<SafeAreaView style={styles.areaView}>**

**<Header isLogged={isLogged} />**

**<ScrollView showsVerticalScrollIndicator={false}>**

**<Welcome fullname={userData.fullname} tabDetails={tabDetails} activeTab={activeTab} setActiveTab={setActiveTab} />**

**<View style={styles.flatContainer}>**

**{displayTabContent()}**

**</View>**

**</ScrollView>**

**<StatusBar backgroundColor={COLORS.primaryColor} style="light" />**

**</SafeAreaView >**

**)**

**}**

**export default Dashboard**