**PROJECT ON HYBRID MOBILE APP NAMED**



**BIST TESTOBIT**

(An online test portal for students)

**Project Members:**

**ABHIJIT KUMAR**

**REGISTRATION NUMBER: 003918 of 2018-2019**

**SNEHA AGARWAL**

**REGISTRATION NUMBER: 003918 of 2018-2019**

**MANISHA MUKHERJEE**

**REGISTRATION NUMBER: 003918 of 2018-2019**

**SOUMYA DARIPA**

**REGISTRATION NUMBER: 003918 of 2018-2019**

Under the supervision of

**Head of Department Mr. ABHIJIT ROY**

**Department of BCA**

**Bengal Institute of Science and Technology**

**CERTIFICATE**

The report of the Project titled Hybrid Mobile app named BIST **TESTOBIT** submitted by Abhijit Kumar (Roll No : 24), Sneha Agarwal (Roll No : 42),Manisha Mukherjee (Roll No : 35), Soumya Daripa (Roll No : 28), of BCA 6th Semester of 2018-19 has been prepared under my supervision for the partial fulfilment of the requirements for BCA degree in Sidho-kanho-birsha University.

*Countersigned By*

(Pradip Kumar Mahato)

**Principle**

(Abhijit Roy)

Dept. of BCA

**Project Guide**

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to **Mr. Abhijit Roy**, Head of Department (BCA) and Bengal Institute Of Science and Technology, whose role as project guide was invaluable for the project. We are extremely thankful for the keen interest he took in advising us, for the books and reference materials provided for the moral support extended to us.

I am also thankful to **Mr. Satyajit kumar** for his unconditional help and support.

Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Abhijit Kumar)

Reg. No.: 003918 of 2018-2019

(Sneha Agarwal)

Reg. No.: 003918 of 2018-2019

(Manisha Mukherjee)

Reg. No.: 003918 of 2018-2019

(Soumya Daripa)

Reg. No.: 003952 of 2018-2019

BCA 6th Semester, BIST

|  |  |  |
| --- | --- | --- |
| **Sl No.** | **Topics** | **Page No.** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**INDEX**

**1.Introduction :**

**2.Software Requirement Specification**

**3.SYSTEM PLANNING & SYSTEMS DEVELOPMENT LIFECYCLE (SDLC)**

We have chosen “Iterative Life Cycle Model” for developing this application, because an iterative life cycle model does not attempt to start with a full specification pf requirements. Instead, development begins by specifying and implementing just part of the software, which can then be reviewed in order to identify further requirements. This process is then repeated, producing a new version of the software for each cycle of the model. Here is the diagram of the iterative life cycle model which depicts its working flow.

Few advantages for choosing the SDLC are –

* In iterative model we are building and improving the product step by step. Hence, we can track the defects at early stages. This avoids the downward flow of the defects.
* Testing and debugging in smaller iteration is easy.
* In iteration model we can get the reliable user feedback. When presenting sketches and blueprints of the product to user for their feedback, we are effectively asking them to imagine how the product will work.
* Progress can be measured.
* In iterative model less time is spent on documentation and more time is given for designing.
* Risk are identified and resolved during iteration; and each iteration is an easily managed milestone. It supports changing requirements.

**4.CONSTRAINTS OF USE**

* Admin and user must remember login id and password.
* User need to have cell phone with internet connection.

**5.TECHNOLOGY USED**

**5.1.Android Frame : Android WebView**

The WebView class is an extension of Android's View class that allows you to display web pages as a part of your activity layout. It does not include any features of a fully developed web browser, such as navigation controls or an address bar. All that WebView does, by default, is show a web page.

**5.2 Front-end Design and Validations: HTML, CSS, Bootstrap , React JS (JavaScript Library)**

Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript, it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document.

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content, including layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Bootstrap is a free and open-source front-end library for designing websites and web applications. It contains HTML- and CSS-based design templates for typography, forms, buttons, navigation and other interface components, as well as optional JavaScript extensions. Unlike many web frameworks, it concerns itself with front-end development only.

React (also known as React.js or React JS) is an open-source front-end JavaScript library for building user interfaces or UI components. It is maintained by Facebook and a community of individual developers and companies React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing, as well as certain client-side functionality.

**5.3Business logic: PHP, LARAVEL (PHP Framework)**

PHP: Hypertext Pre-processor (or simply PHP) is a server-side scripting language designed for web development but also used as a xgeneral-purpose programming language. It was originally created by Rasmus Lerdorf in 1994; the PHP reference implementation is now produced by The PHP Group. PHP originally stood for Personal Home Page, but it now stands for the recursive acronym PHP: Hypertext Pre-processor. PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

Laravel : Laravel is a web application framework with expressive, elegant syntax. We believe development must be an enjoyable, creative experience to be truly fulfilling. Laravel attempts to take the pain out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, and caching002E,

Laravel is accessible, yet powerful, providing powerful tools needed for large, robust applications. A superb inversion of control container, expressive migration system, and tightly integrated unit testing support give you the tools we need to build any application with which we are tasked.

**5.4 Database: MySQL**

MySQL is an open source relational database management system. For proprietary use, several paid editions are available, and offer additional functionality. In this project MySQL has been used to store, update, retrieve and delete related to user’s data and other additional data about projects.

**5.5 Web Server: Apache**

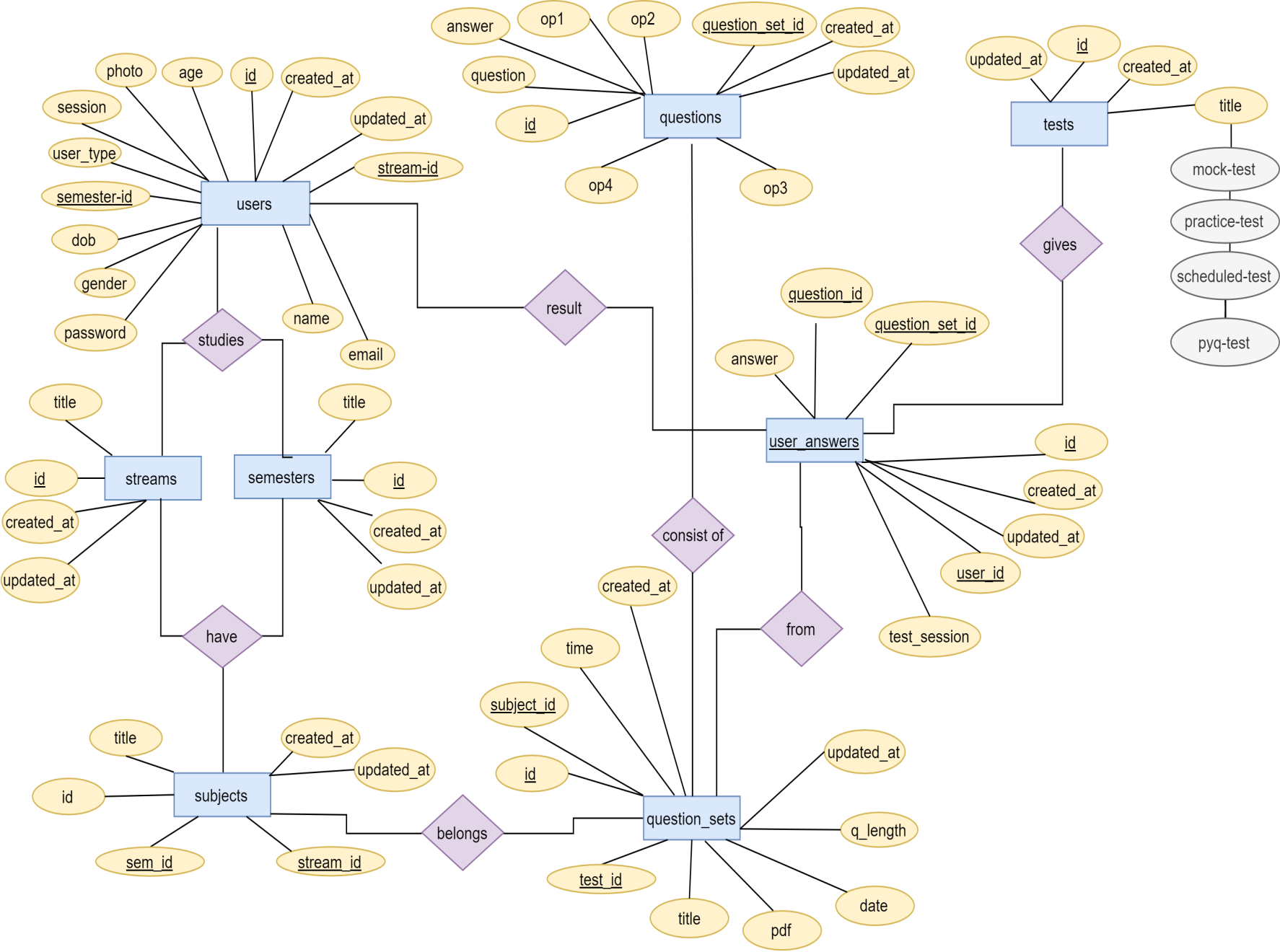
The Apache HTTP Server, colloquially called Apache, is a free and open-source cross-platform web server, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The Apache HTTP Server is cross platform; as of 1 June 2017 92% of Apache HTTPS Server copies run on Linux distributions. Version 2.0 improved support for non-Unix operating systems such as Windows and OS/2. Old versions of Apache were ported to run on OpenVMS and NetWare.

**6.FEASIBILITY STUDY**

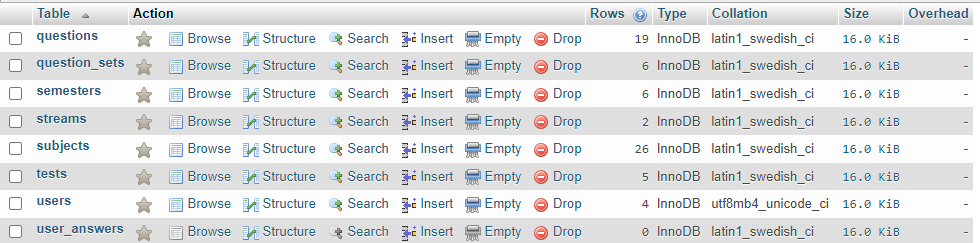
A feasibility study is an analysis of how successfully a project can be completed, accounting for factors that affect it such as economic, technological and operational. Project managers use feasibility studies to determine potential positive and negative outcomes of a project before investing a considerable amount of time and money into it.

During the stage of our feasibility study, we had to undergo the following steps as described under:

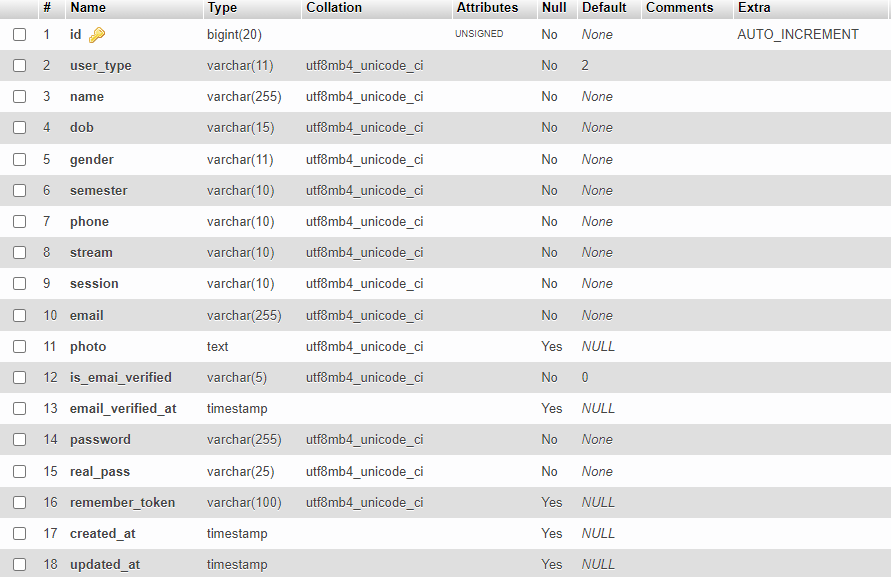
* Identify the origin of data at different levels of the system.
* Identify the expectation of end user from the finished product/system.
* Analyze the drawback(s) of the existing system.
* **Technical feasibility study:** It lays out details on how a good or service will be delivered, which includes transportation, business location, technology needed, materials and labour.
* **Financial feasibility study:** It is a projection of the amount of funding or start-up capital needed, what sources of capital can and will be used and what kind of return can be expected on the investment.
* **Organizational feasibility study:** It is a definition of the corporate and legal structure of the business; this may include information about the founders, their professional background and the skills they possess necessary to get the company off the ground and keep it operational.**7. ENTITY RELATIONSHIP DIAGRAM (ERD)**

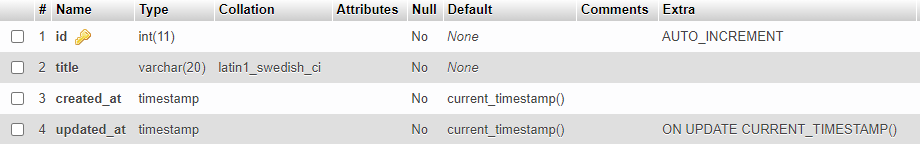
****

**8 DATABASE DESIGNED**

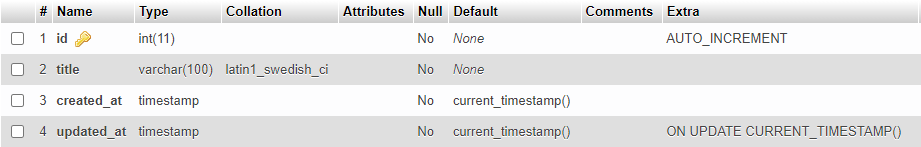
**8.1 Database: testobit**

**8.2.Database tables:**

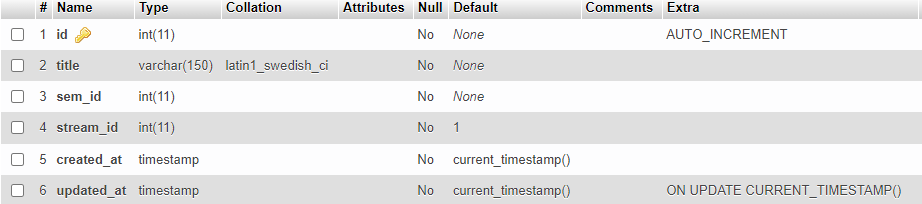
** 1.Users :**

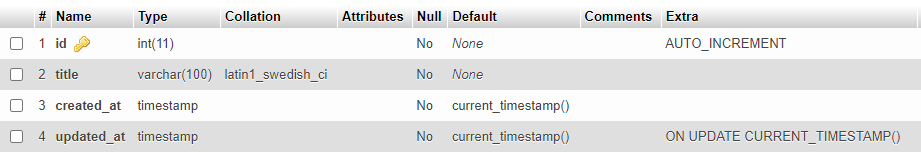
**2.Semesters:**

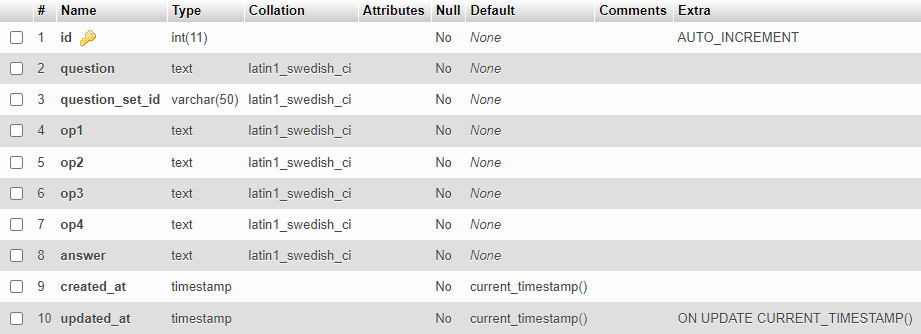
**3.Streams :**

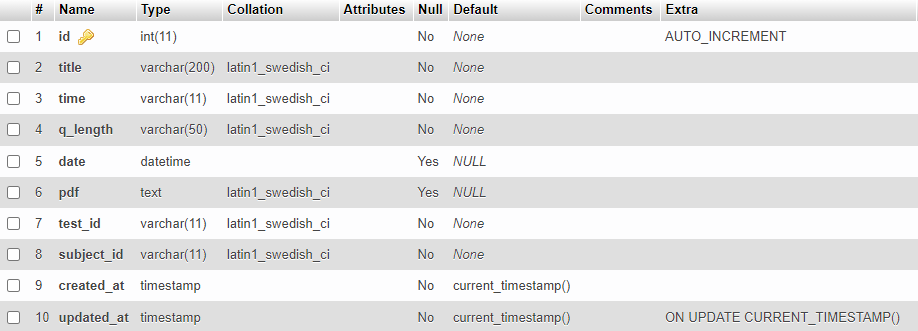
****

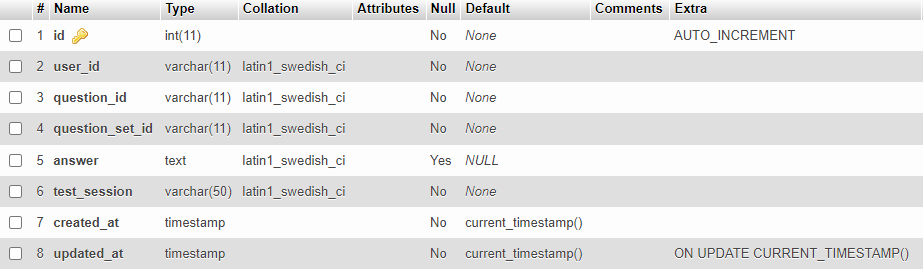
**4.Subjects:**

****

** 5.Tests :**

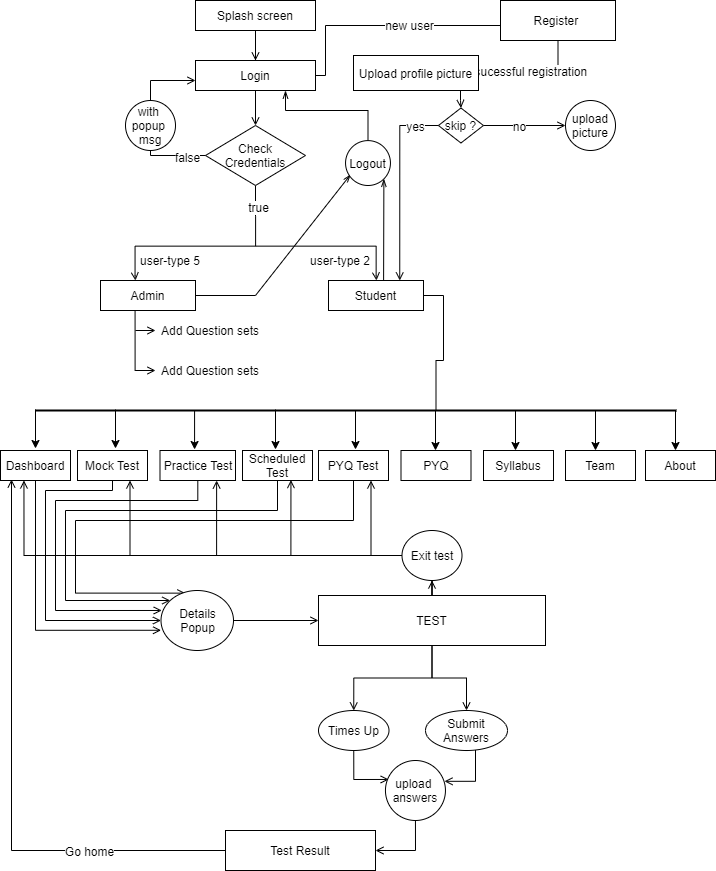
**6.Questions:**

**7.Question-sets:**

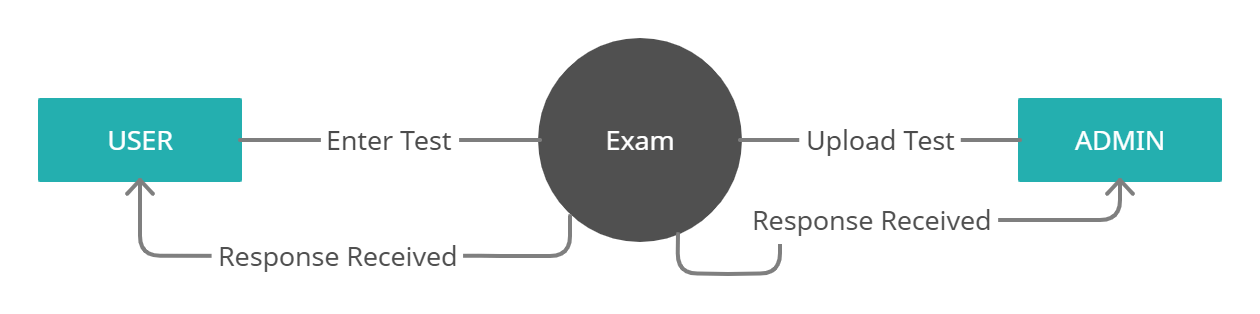
****

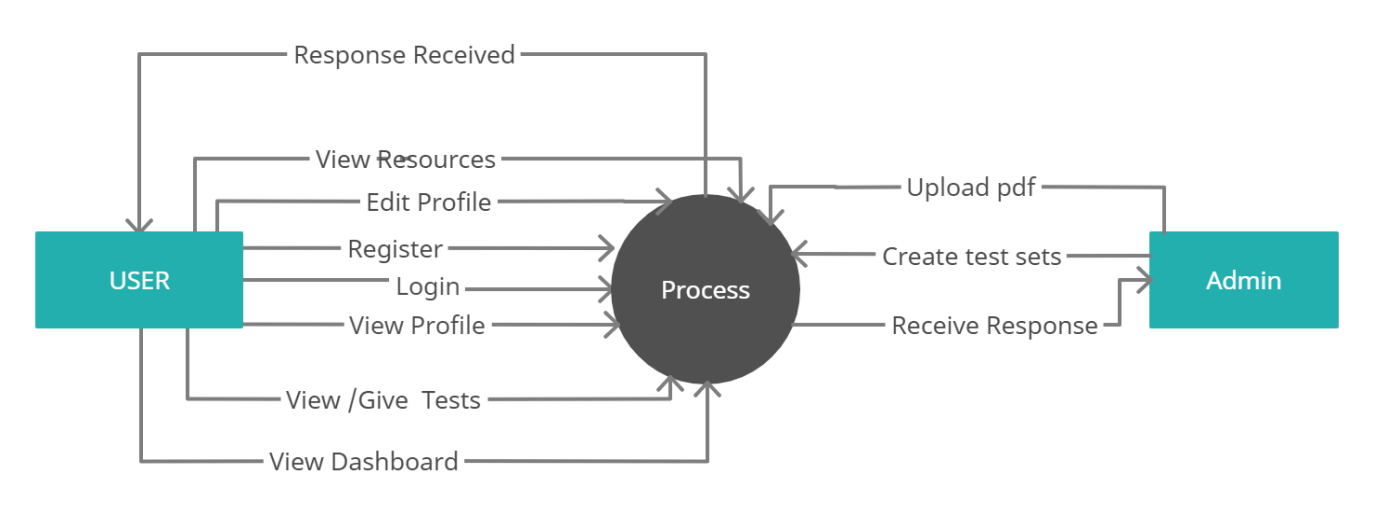
**8.User-answers :**

**9.Activity Diagram**

****An activity diagram is used to specify the functionality of the system from the point of view of a user .Each activity describes a logical task that may be performed by a user. It mainly shows the interaction between the system and the outside world.

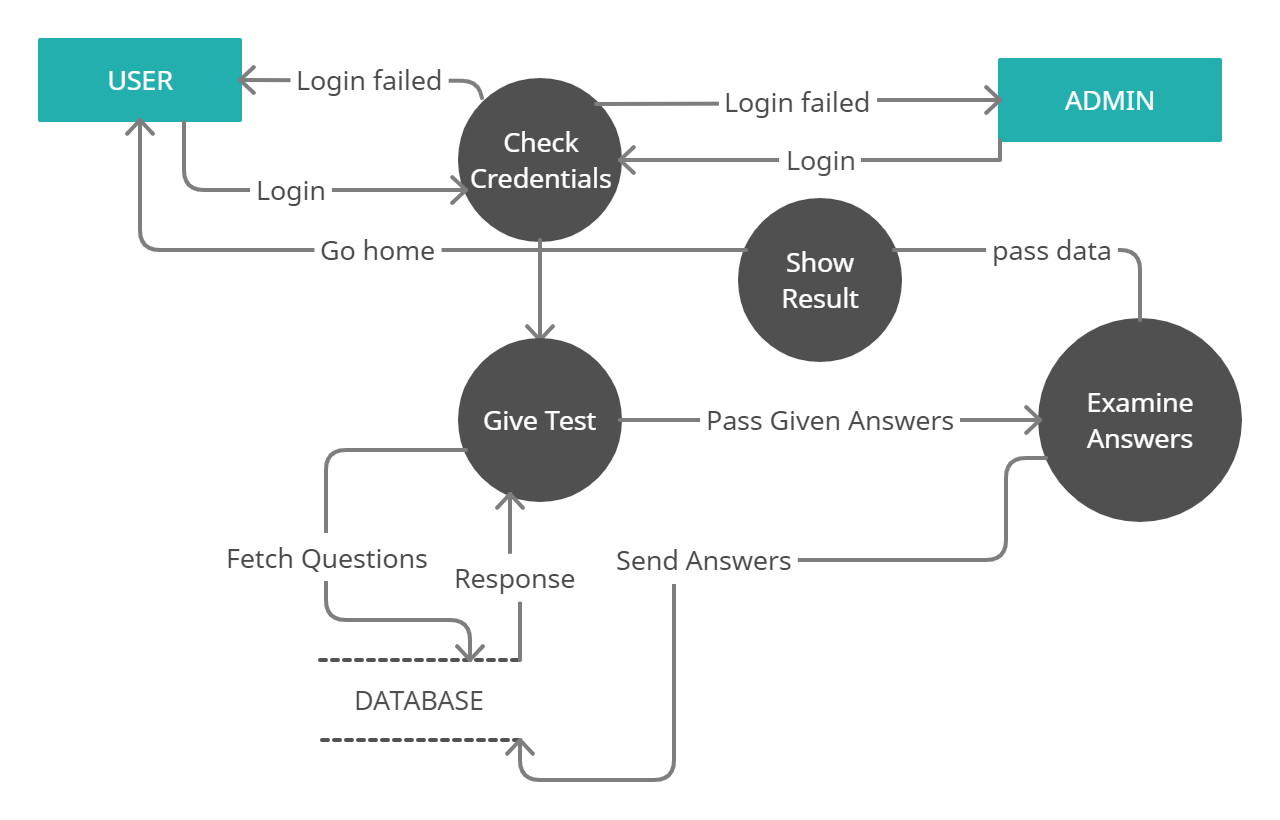
**10.Data Flow Diagram (DFD)**



**Level – 0 DFD**

**Level – 1 DFD**

**Level -2 DFD**



**11. SOURCE CODE**

* **11.1 Front-End:**

**Index.js(Entry - Point)**

import React from 'react';

import ReactDOM from 'react-dom';

import './index.css';

import App from './App';

import reportWebVitals from './reportWebVitals';

import { Provider } from "react-redux"

import store from "./redux/store"

import { HashRouter as Router} from 'react-router-dom';

import { rootURL } from './globals/\_\_gobal\_vars';

import axios from 'axios'

import { LOADEROFF, LOADERON } from './globals/\_\_global\_funcs';

const renderApp = () => {

  ReactDOM.render(

    <Provider store={store}>

      <React.StrictMode>

        <Router>

          <App />

        </Router>

      </React.StrictMode>

    </Provider>,

    document.getElementById('root')

  );

}

renderApp();

if(localStorage.getItem('token')){

  LOADERON();

  axios.defaults.headers.common['Authorization'] = `Bearer ${localStorage.getItem('token')}`;

  axios.post(`${rootURL}/auth/getUser`).then(res => {

    store.dispatch({ type: 'SET\_LOGIN', payload: res.data.data })

    renderApp();

    LOADEROFF();

  }).catch((err)=>{

    console.log(err);

    renderApp();

    LOADEROFF();

  })

} else {

  renderApp();

}

reportWebVitals();

**App.js**

import React from 'react';

import './css/style.css';

import AuthRoute from "./auth/AuthRoute"

import GuestRoute from "./auth/GuestRoute"

import { Route, Switch, useLocation } from 'react-router';

import { createMuiTheme } from '@material-ui/core/styles';

import { ThemeProvider } from '@material-ui/styles';

import { useHistory } from "react-router-dom";

import Login from './views/guest/Login';

import Register from './views/guest/Register';

import VerifyEmail from './views/guest/VerifyEmail';

import AllSet from './views/guest/AllSet';

import StudentLayout from './views/student/layouts/StudentLayout';

import Page404 from './views/guest/Page404';

import { connect } from 'react-redux';

import AdminLayout from './views/admin/layout/AdminLayout';

function App(props) {

  const theme = createMuiTheme({

    palette: {

      primary: {

        // Purple and green play nicely together.

        main: '#432fbf',

      },

    },

  });

  const activeUser = (UserType) => {

    switch (UserType) {

      case 7:

        return <React.Fragment>

          <AuthRoute path="/admin" component={AdminLayout} />

        </React.Fragment>

      case 5:

        return <React.Fragment>

          <AuthRoute path="/admin" component={AdminLayout} />

        </React.Fragment>

      case 2:

        return (<React.Fragment>

          <AuthRoute path="/student" component={StudentLayout} />

          <AuthRoute path="/all-set/:change" component={AllSet} />

        </React.Fragment>)

      case 0:

        return <Route exact path="/" component={Login} />

      default:

        return ""

    }

  }

  let history = useHistory();

  const location = useLocation()

  const [goUrl, SetGoUrl] = React.useState("")

  const [hide, setHide] = React.useState(true)

  return (

    <React.Fragment>

      <ThemeProvider theme={theme}>

        <form className={`fixed-bottom transit ${(hide) ? "goUrl-hide" : ""}`} onSubmit={(e) => { e.preventDefault(); history.push(goUrl) }}>

          <div  onClick={()=>{history.goBack()}} className="back-button "><i className={`fas fa-chevron-left`}></i> </div>

          <div className="input-group flex-nowrap">

            <input type="text" value={goUrl} onChange={(e) => { SetGoUrl(e.target.value) }} placeholder="Go URL (debug mode)" className="form-control " />

            <button className="btn btn-primary rounded-0 input-group-text" id="addon-wrapping">Go</button>

          </div>

        </form>

              <Switch location={location} >

                <GuestRoute exact path="/" component={Login} />

                <GuestRoute path="/register" component={Register} />

                <GuestRoute path="/verify" component={VerifyEmail} />

                {

                  activeUser(parseInt(props.user.user\_type))

                }

                <Route component={Page404} />

              </Switch>

      </ThemeProvider>

    </React.Fragment>

  )

}

const mapStateToProps = state =>({

  user : state.auth.user

})

export default connect(mapStateToProps,null)(App);

**Login.js**

import React, { Component } from 'react'

import { Link } from 'react-router-dom'

import { LOADEROFF, LOADERON, notification, redirect, validate } from '../../globals/\_\_global\_funcs'

import { setLogin } from '../../redux/actions/AuthActions'

import { connect } from 'react-redux'

import { rootURL } from '../../globals/\_\_gobal\_vars'

import axios from "axios"

class Login extends Component {

    state = {

        username: "",

        password: "",

    }

    loginSubmit = (event) => {

        event.preventDefault();

        if (!(this.state.username == "" || this.state.username == null || this.state.password == "" || this.state.password == null)) {

            LOADERON();

            let config = {

                headers: {

                    "Content-Type": "application/json",

                }

            }

            axios.post(`${rootURL}/login`, {

                email : this.state.username.toLowerCase(),

                password : this.state.password,

            },config)

                .then(res => {

                    if(res.data.status){

                        localStorage.setItem('token', res.data.access\_token);

                        localStorage.setItem('email', res.data.user.email);

                        localStorage.setItem('id', res.data.user.id);

                        localStorage.setItem('user\_type', res.data.user.user\_type);

                        this.props.setLogin(res.data.user);

                        this.props.history.push(redirect(res.data.user.user\_type));

                        LOADEROFF();

                    }else{

                        LOADEROFF();

                        notification(res.data.error,"error");

                    }

                }).catch(err => {

                    if(err.message == "Network Error"){

                        notification("Network error","error");

                    }else{

                        console.log(err)

                    }

                    LOADEROFF();

                });

        }

    }

    componentDidMount() {

        validate('login-validation');

    }

    render() {

        return (

            <div className="wrapper d-flex align-items-center justify-content-center">

                <div style={{marginTop: "50px"}} className="card">

                    <div className="card-body">

                        <div className="login-logo">

                            <img className="w-100" src="./assets/img/logo-white.png" alt />

                        </div>

                        <table width="100%">

                            <tbody><tr>

                                <td><hr className="bg-dark" /></td>

                                <td width="100px" style={{ textAlign: 'center', color: 'rgb(105, 105, 105)', fontFamily: 'verdana', fontSize: 20 }}>LOGIN</td>

                                <td><hr className="bg-dark" /></td>

                            </tr>

                            </tbody>

                        </table>

                        <form className="login-validation" noValidate onSubmit={this.loginSubmit}>

                            <div className="row justify-content-center">

                                <div className="col-11 my-1">

                                    <label >Email</label>

                                    <div className="input-group flex-nowrap mt-1">

                                        <span className="input-group-text" id="addon-wrapping"><i className="fas fa-user-shield text-primary" /></span>

                                        <input onChange={(e) => { this.setState({ username: e.target.value }) }} value={this.state.username} type="text" className="form-control" placeholder="Enter email" required aria-label="Username" aria-describedby="addon-wrapping" />

                                        {/\* <div class="invalid-feedback"><i class="fas fa-info-circle"></i> Field is required.</div> \*/}

                                    </div>

                                </div>

                                <div className="col-11 my-1">

                                    <label >Password</label>

                                    <div className="input-group flex-nowrap mt-1">

                                        <span className="input-group-text" id="addon-wrapping"><i className="fas fa-key text-primary" /></span>

                                        <input onChange={(e) => { this.setState({ password: e.target.value }) }} value={this.state.password} type="password" className="form-control" placeholder="Enter Password" required aria-label="Password" aria-describedby="addon-wrapping" />

                                        {/\* <div class="invalid-feedback"><i class="fas fa-info-circle"></i> Field is required.</div> \*/}

                                    </div>

                                </div>

                                <div className="col-11 mt-3">

                                    <button className="btn btn-primary btn-block">Login</button>

                                </div>

                                <div className="col-12  mt-3">

                                    <p className="text-center">Not have an Account ? click to <Link to="/register" > Register </Link> </p>

                                </div>

                            </div>

                        </form>

                    </div>

                </div>

            </div>

        )

    }

}

export default connect(null,{setLogin})(Login)

**Register.js**

import React, { Component } from 'react'

import { Link } from 'react-router-dom'

import TextField from '@material-ui/core/TextField';

import { FormControl, FormControlLabel, FormLabel, InputLabel, MenuItem, Radio, RadioGroup, Select } from '@material-ui/core';

import { LOADEROFF, LOADERON, notification, validate } from '../../globals/\_\_global\_funcs';

import Animate from '../../components/Animate';

import Topbar from '../../components/Topbar';

import { rootURL } from '../../globals/\_\_gobal\_vars';

// import { InputAdornment } from '@material-ui/core';

import axios from 'axios';

import { connect } from 'react-redux';

import { setLogin } from '../../redux/actions/AuthActions';

class Register extends Component {

  state = {

    formdata: {

      name: '',

      dob: "2000-01-25",

      semester: '',

      gender: 'male',

      email: '',

      phone: '',

      stream: '',

      session: '',

      password: '',

      c\_password: '',

    },

    errors: {

      name: false,

      dob: false,

      semester: false,

      email: false,

      phone: false,

      stream: false,

      session: false,

      password: false,

      c\_password: false,

    },

    semesterList: [],

    streamList: [],

  }

  validateEmail = (email) => {

    const re = /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)\*)|(\".+\"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;

    return re.test(email);

  }

  isValid = () => {

    let noErr = true

    let data = this.state.formdata

    let errObj = this.state.errors

    for (let [key, value] of Object.entries(data)) {

      if (value == "" || value == null) {

        errObj[key] = true

      } else {

        errObj[key] = false

      }

    }

    if (data.password !== data.c\_password) {

      errObj['c\_password'] = true

    } else {

      errObj['c\_password'] = false

    }

    if (!errObj.email) {

      if (this.validateEmail(data.email)) {

        errObj['email'] = false

      } else {

        errObj['email'] = true

      }

    }

    for (let [key, value] of Object.entries(errObj)) {

      if (value) {

        noErr = false

      }

    }

    this.setState({ errors: errObj })

    return noErr;

  }

  registerSubmit = (event) => {

    event.preventDefault();

    if (this.isValid()) {

      LOADERON();

      axios.post(`${rootURL}/register`, {

        "name": this.state.formdata.name,

        "email": this.state.formdata.email,

        "dob": this.state.formdata.dob,

        "gender": this.state.formdata.gender,

        "semester": this.state.formdata.semester,

        "phone": this.state.formdata.phone,

        "stream": this.state.formdata.stream,

        "session": this.state.formdata.session,

        "password": this.state.formdata.password,

      }).then(res => {

        //success

        if (res.data.status) {

          console.log(res);

          localStorage.setItem('token', res.data.access\_token);

          localStorage.setItem('email', res.data.user.email);

          localStorage.setItem('id', res.data.user.id);

          localStorage.setItem('user\_type', res.data.user.user\_type);

          // this.setState({ progress: false })

          this.props.setLogin(res.data.user);

          LOADEROFF();

          this.props.history.push('/all-set/0');

        } else {

          notification(res.data.errors[0], 'error');

          LOADEROFF();

        }

      }).catch((err) => {

        console.log(err);

        LOADEROFF();

      })

    }

  }

  setData = (event, key) => {

    this.setState({

      formdata: { ...this.state.formdata, [key]: event.target.value }

    })

  }

  componentDidMount() {

    validate("register-form")

    LOADERON();

    axios.defaults.headers.common['Authorization'] = `Bearer ${localStorage.getItem('token')}`;

    let stream = axios.post(rootURL + "/getStream", {}).then((res) => {

      //success

      this.setState({

        streamList: res.data.data

      })

      LOADEROFF();

    }).catch((err) => {

      LOADEROFF();

    })

    axios.defaults.headers.common['Authorization'] = `Bearer ${localStorage.getItem('token')}`;

    let semester = axios.post(rootURL + "/getSemester", {}).then((res) => {

      //success

      this.setState({

        semesterList: res.data.data

      })

      LOADEROFF();

    }).catch((err) => {

      LOADEROFF();

    })

    // Promise.all([stream, semester]).then(() => {

    //     LOADEROFF();

    // })

  }

  render() {

    return (

      <div className="wrapper mb-3">

        <Topbar text="Register"> </Topbar>

        <div className="container-fluid">

          <div className="card">

            <div className="card-body">

              <form className="register-form" onSubmit={this.registerSubmit} noValidate autoComplete="off">

                <div className="row justify-content-center">

                  <div className="col-12 my-1">

                    <TextField

                      value={this.state.formdata.name}

                      onChange={(e) => { this.setData(e, 'name') }}

                      error={this.state.errors.name}

                      helperText={(this.state.errors.name) ? "Field is required" : ""}

                      // InputProps={{

                      // startAdornment: (

                      //     <InputAdornment position="start">

                      //     <i className="fas fa-eye text-primary"></i>

                      //     </InputAdornment>

                      // ),

                      // }}

                      label="Name" />

                  </div>

                  <div className="col-12 my-1">

                    <TextField

                      id="date"

                      value={this.state.formdata.dob}

                      onChange={(e) => { this.setData(e, 'dob') }}

                      label="Birthday"

                      type="date"

                      helperText={(this.state.errors.dob) ? "Field is required" : ""}

                      // defaultValue="2000-01-01"

                      InputLabelProps={{

                        shrink: true,

                      }}

                    />

                    {/\* <TextField

                                            value={this.state.formdata.dob}

                                            onChange={(e) => { this.setData(e, 'dob') }}

                                            error={this.state.errors.dob}

                                            helperText={(this.state.errors.dob) ? "Field is required" : ""}

                                            id="date" type="date" placeholder="select date" label="Date of birth" /> \*/}

                  </div>

                  <div className="col-12 my-1">

                    <FormControl component="fieldset">

                      <FormLabel component="label">Gender</FormLabel>

                      <RadioGroup aria-label="gender" name="gender1" value={this.state.formdata.gender} onChange={(e) => { this.setData(e, 'gender') }}>

                        <FormControlLabel value="male" control={<Radio />} label="Male" />

                        <FormControlLabel value="female" control={<Radio />} label="Female" />

                        <FormControlLabel value="other" control={<Radio />} label="Other" />

                      </RadioGroup>

                    </FormControl>

                  </div>

                  <div className="col-12 my-1">

                    <FormControl error={this.state.errors.semester}>

                      <InputLabel id="semester">Semester</InputLabel>

                      <Select

                        labelId="semester"

                        value={this.state.formdata.semester}

                        onChange={(e) => { this.setState({ formdata: { ...this.state.formdata, semester: e.target.value } }) }}

                        error={this.state.errors.semester}

                        helperText={(this.state.errors.semester) ? "Field is required" : ""}

                      >

                        {

                          this.state.semesterList.map((item, index) => (

                            <MenuItem key={index} value={item.id}>{item.title}</MenuItem>

                          ))

                        }

                      </Select>

                    </FormControl>

                  </div>

                  <div className="col-12 my-1">

                    <TextField

                      value={this.state.formdata.email}

                      onChange={(e) => { this.setData(e, 'email') }}

                      error={this.state.errors.email}

                      helperText={(this.state.errors.email) ? "Email must contains '@' and '.'" : ""}

                      label="Email" />

                  </div>

                  <div className="col-12 my-1">

                    <TextField

                      value={this.state.formdata.phone}

                      onChange={(e) => { if (e.target.value.length < 11) { this.setData(e, 'phone') } }}

                      error={this.state.errors.phone}

                      helperText={(this.state.errors.phone) ? "Field is required" : ""}

                      maxLength="10"

                      label="Phone" />

                  </div>

                  <div className="col-12 my-1">

                    <FormControl error={this.state.errors.stream} >

                      <InputLabel id="stream">Stream</InputLabel>

                      <Select

                        labelId="stream"

                        value={this.state.formdata.stream}

                        onChange={(e) => { this.setState({ formdata: { ...this.state.formdata, stream: e.target.value } }) }}

                        error={this.state.errors.stream}

                        helperText={(this.state.errors.stream) ? "Field is required" : ""}

                      >

                        {

                          this.state.streamList.map((item, index) => (

                            <MenuItem disabled={item.id == "2"} key={index} value={item.id}>{item.title}</MenuItem>

                          ))

                        }

                      </Select>

                    </FormControl>

                  </div>

                  <div className="col-12 my-1">

                    <TextField

                      value={this.state.formdata.session}

                      onChange={(e) => { this.setData(e, 'session') }}

                      error={this.state.errors.session}

                      helperText={(this.state.errors.session) ? "Field is required" : ""}

                      label="Session" />

                  </div>

                  <div className="col-12 my-1">

                    <TextField

                      value={this.state.formdata.password}

                      onChange={(e) => { this.setData(e, 'password') }}

                      error={this.state.errors.password}

                      helperText={(this.state.errors.password) ? "Field is required" : ""}

                      type="password" label="Password" />

                  </div>

                  <div className="col-12 my-1">

                    <TextField

                      value={this.state.formdata.c\_password}

                      onChange={(e) => { this.setData(e, 'c\_password') }}

                      error={this.state.errors.c\_password}

                      helperText={(this.state.errors.c\_password) ? "Password mismatche !" : ""}

                      type="password" label="Confirm Password" />

                  </div>

                  <div className="col-12">

                    {/\* <Link to="/verify" > \*/}

                    <button type="submit" className="btn btn-primary btn-block">Register</button>

                    {/\* </Link> \*/}

                  </div>

                  <div className="col-12  mt-3">

                    <p className="text-center">Already have an Account ? Go to <Link to="/" > Login </Link> </p>

                  </div>

                </div>

              </form>

            </div>

          </div>

        </div>

      </div>

    )

  }

}

export default connect(null, { setLogin })(Register);

**Page404.js**

import { IconButton } from '@material-ui/core'

import React, { Component } from 'react'

import { Redirect } from 'react-router'

import Animate from '../../components/Animate'

import Topbar from '../../components/Topbar'

export default class Page404 extends Component {

  render() {

    return (

      <div>

        <Redirect to="/" />

      </div>

    )

  }

}

**AllSet.js**

import React, { Component } from 'react'

import { Link } from 'react-router-dom';

import axios from 'axios';

import { LOADERON, LOADEROFF } from "../../globals/\_\_global\_funcs"

import { imageURL, rootURL } from '../../globals/\_\_gobal\_vars';

import { connect } from 'react-redux';

import { editUser } from '../../redux/actions/AuthActions';

import Cropper from 'react-easy-crop'

import Button from '@material-ui/core/Button';

import getCroppedImg from '../../components/cropImage';

import { Grid, IconButton, Slider } from '@material-ui/core';

class AllSet extends Component {

  constructor(props) {

    super(props);

    this.state = {

      imageSource: this.props.user.photo ? imageURL + this.props.user.photo : "./assets/img/user.png",

      image: {},

      tempName: "",

      tempImage: "",

      progress: false,

      visible: false,

      crop: {

        x: 0,

        y: 0,

      },

      zoom: 1,

      cropData: {}

    }

  }

  onCropComplete = async (croppedArea, croppedAreaPixels) => {

    // console.log(croppedArea, croppedAreaPixels)

    var CroppedImage = await getCroppedImg(this.state.tempImage, croppedAreaPixels, 0)

    // console.log(CroppedImage);

    // let blob = await fetch(CroppedImage).then(r => r.blob());

    // blob = {name : this.state.tempName , ...blob}

    this.setState({

      imageSource: CroppedImage,

      cropData: croppedAreaPixels

    })

  }

  componentDidMount() {

  }

  uploadImage = event => {

    event.preventDefault();

    if (this.state.image.name) {

      this.setState({

        progress: true

      })

      let image = new FormData();

      image.append("photo", this.state.image, this.state.image.name);

      image.append("h", this.state.cropData.height)

      image.append("w", this.state.cropData.width)

      image.append("x", this.state.cropData.x)

      image.append("y", this.state.cropData.y)

      axios.post(rootURL + "/auth/uploadImage", image, {

        headers: {

          'Content-Type': 'multipart/form-data',

          'Authorization': `Bearer ${localStorage.getItem('token')}`

        }

      }).then(res => {

        //success

        console.log(res);

        if (res.data.status) {

          this.props.editUser({ photo: res.data.image\_path })

          if (this.props.match.params.change == 1) {

            this.props.history.push("/student/profile");

          } else {

            this.props.history.push("/student/dashboard");

          }

          this.setState({

            progress: false

          })

        } else {

          this.setState({

            progress: false

          })

        }

      }).catch((err) => {

        console.log(err);

        this.setState({

          progress: false

        })

      })

    }

  }

  handleImageChoose = (e) => {

    let defaultImg = this.props.user.photo ? imageURL + this.props.user.photo : "./assets/img/user.png"

    this.setState({

      visible: true,

      image: e.target.files[0],

      tempImage: e.target.files[0] ? URL.createObjectURL(e.target.files[0]) : defaultImg

    })

  }

  render() {

    let change = this.props.match.params.change ?? "0"

    let defaultImg = this.props.user.photo ? imageURL + this.props.user.photo : "./assets/img/user.png"

    return (

      <div className="wrapper ">

        <div className="d-flex align-items-center h-100">

          <div className="container-fluid ">

            <Link to="/student/dashboard" className="text-right text-primary d-block mr-3">{(change == 1) ? "Back to dashboard" : "Skip"} &nbsp;&nbsp;<i class="fas fa-caret-right"></i> </Link>

            <div className>

              <div className>

                <h2 className="font-weight-light text-center">{(change == 1) ? "Change your picture" : "You are all set."} </h2>

                <form onSubmit={this.uploadImage}>

                  <div className="row justify-content-center my-5">

                    <div className="col-12 ">

                      <div className={`upload-avatar ${this.state.progress ? "upload-avatar-loader" : ""}`}>

                        <label htmlFor="upload-pic" className={`upload-icon ${this.state.progress ? "upload-icon-loader" : ""}`}>

                          <i class="fas fa-camera fa-2x"></i>

                        </label>

                        <label style={{ borderRadius: "inherit", overflow: 'hidden', display: "block", width: "100%", height: "100%", }} htmlFor="upload-pic" >

                          <img width="100%" src={this.state.imageSource} alt="" />

                        </label>

                      </div>

                      <input onChange={this.handleImageChoose} className="d-none" type="file" name="" id="upload-pic" />

                    </div>

                    <p className="text-center my-3">Upload Your Profile Picture</p>

                    {

                      (this.state.imageSource !== defaultImg) ?

                        <div className="col-12 mt-1">

                          <button disabled={this.state.progress} type="submit" className="btn btn-primary btn-block">{this.state.progress ? "Uploading please wait ..." : " Upload & Finish"}</button>

                        </div>

                        : ""

                    }

                  </div>

                </form>

              </div>

            </div>

          </div>

        </div>

        {

          this.state.visible ?

            <div className="crop-overlay">

              <div className="crop-area">

                <div className="container-fluid">

                  <div className="row justify-content-center align-items-center">

                    <div className="col-12 pt-3">

                      <Cropper

                        style={{

                          containerStyle: {

                            position: "relative",

                            height: "350px",

                          }

                        }}

                        image={this.state.tempImage}

                        crop={this.state.crop}

                        zoom={this.state.zoom}

                        aspect={1 / 1}

                        onCropChange={(value) => { this.setState({ crop: value }) }}

                        onCropComplete={this.onCropComplete}

                        onZoomChange={(zoom) => { this.setState({ zoom }) }}

                      />

                    </div>

                    {/\* <div className="col-2 text-center">

                                            <IconButton><i style={{ fontSize: "15px", color: "var(--primary)" }} className="fas fa-minus-circle"></i> </IconButton>

                                        </div>

                                        <div className="col-8 py-4">

                                            <Slider min={1} step={0.3} max={7} value={this.state.zoom} onChange={(e, zoom) => { this.setState({ zoom }) }} aria-labelledby="continuous-slider" />

                                        </div>

                                        <div className="col-2 text-center">

                                            <IconButton><i style={{ fontSize: "15px", color: "var(--primary)" }} className="fas fa-plus-circle"></i> </IconButton>

                                        </div> \*/}

                    <div className="col-4">

                      <Button onClick={() => { this.setState({ visible: false, tempImage: "" }) }} className="float-left" color="primary"> Done </Button>

                    </div>

                    <small className="text-danger text-center"><i className="fas fa-exclamation-circle"></i> Please use small size images for quick uploading</small>

                  </div>

                </div>

              </div>

            </div> : null

        }

      </div>

    )

  }

}

const mapStateToProps = state => ({

  user: state.auth.user

})

export default connect(mapStateToProps, { editUser })(AllSet);

**StudentLayout.js**

import { Menu, MenuItem } from '@material-ui/core';

import React, { useEffect} from 'react'

import { connect } from 'react-redux';

import { Route, Switch, useLocation } from 'react-router';

import Sidebar from '../../../components/Sidebar'

import { setLogout } from '../../../redux/actions/AuthActions';

import Page404 from '../../student/Page404';

import About from '../About';

import Dashboard from '../Dashboard';

import MockTest from '../MockTest';

import Notice from '../Notice';

import PracticeTest from '../PracticeTest';

import PYQ from '../PYQ';

import PastYearTest from '../PastYearTest';

import Syllabus from '../Syllabus';

import Team from '../Team';

import SchedulledTest from '../SchedulledTest';

import TestCMP from '../TestCMP';

import Profile from '../Profile';

import TestResult from '../TestResult';

import { setStudentData } from '../../../redux/actions/TestActions';

function StudentLayout(props) {

    const [state, setStateValue] = React.useState({

        open: false,

        popperOpen: null,

    })

    const setState = (valueObj) => {

        setStateValue({ ...state, ...valueObj })

    };

    const toggleDrawer = (anchor, open) => (event) => {

        if (event && event.type === 'keydown' && (event.key === 'Tab' || event.key === 'Shift')) {

            return;

        }

        setState({ open })

    };

    const togglePopper = (event) => {

        setState({ popperOpen: event.currentTarget })

    };

    const logout = () => {

        props.setLogout();

    }

    useEffect(() => {

        props.setStudentData();

    }, [])

    const location = useLocation()

    return (

        <div className="wrapper p-0">

            <Sidebar

                toggleDrawer={toggleDrawer}

                open={state.open}

                path="/student" />

            <Menu

                id="simple-menu"

                anchorEl={state.popperOpen}

                keepMounted

                open={Boolean(state.popperOpen)}

                onClose={() => { setState({ popperOpen: null }) }}

            >

                <MenuItem onClick={logout}>Logout</MenuItem>

            </Menu>

                    <Switch>

                        <Route path="/student/dashboard" render={(p) => (<Dashboard {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/mock-test" render={(p) => (<MockTest {...p} studentData={props.studentData ?? {}} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/pyq-test" render={(p) => (<PastYearTest {...p} studentData={props.studentData ?? {}} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/pyq" render={(p) => (<PYQ {...p} toggleDrawer={toggleDrawer} studentData={props.studentData ?? {}} togglePopper={togglePopper} />)} />

                        <Route path="/student/practice-test" render={(p) => (<PracticeTest {...p} studentData={props.studentData ?? {}} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/scheduled-test" render={(p) => (<SchedulledTest {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/notice" render={(p) => (<Notice {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/syllabus" render={(p) => (<Syllabus {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/team" render={(p) => (<Team {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/test/:id/:time" render={(p) => (<TestCMP {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/test-result" render={(p) => (<TestResult {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/profile" render={(p) => (<Profile {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route path="/student/about" render={(p) => (<About {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                        <Route render={(p) => (<Page404 {...p} toggleDrawer={toggleDrawer} togglePopper={togglePopper} />)} />

                    </Switch>

        </div>

    )

}

const mapStateToProps = state =>({

    studentData : state.studentData

})

export default connect(mapStateToProps, { setLogout,setStudentData })(StudentLayout)

**Dashboard.js**

import React, { Component } from 'react'

import Animate from '../../components/Animate'

import { IconButton } from '@material-ui/core';

import Topbar from '../../components/Topbar'

import { Line } from 'react-chartjs-2';

import SchedulledTest from './SchedulledTest';

import Notice from './Notice';

import { CircularProgressbarWithChildren, buildStyles } from 'react-circular-progressbar';

import 'react-circular-progressbar/dist/styles.css';

import { connect } from 'react-redux';

class Dashboard extends Component {

  state = {

  }

  push = (path) => {

    this.props.history.push(path);

  }

  render() {

    if (Object.keys(this.props.studentData).length !== 0) {

      let total = this.props.studentData.right + this.props.studentData.wrong;

      let labels = [];

      let dataRight = []

      let dataWrong = []

      let totalTests = Object.keys(this.props.studentData.set\_data).length

      Object.entries(this.props.studentData.set\_data).reverse().map((item, index) => {

        if (Object.keys(this.props.studentData.set\_data).length - 15 < index + 1) {

          labels = [...labels, "T-" + (index + 1)];

          let tempr = 0

          let tempw = 0

          item[1].map((nItem) => {

            if (nItem.answer !== null) {

              if (nItem.answer == nItem.question.answer) {

                tempr++;

              } else {

                tempw++;

              }

            }

          });

          dataRight = [...dataRight, tempr]

          dataWrong = [...dataWrong, tempw]

        }

      })

      let chartState = {

        labels: labels,

        datasets: [

          {

            label: 'Right',

            fill: false,

            lineTension: 0.5,

            backgroundColor: 'rgb(21, 212, 36)',

            borderColor: 'rgb(21, 212, 36)',

            borderWidth: 2,

            data: dataRight

          },

          {

            label: 'Wrong',

            fill: false,

            lineTension: 0.5,

            backgroundColor: 'rgb(233, 30, 99)',

            borderColor: 'rgb(233, 30, 99)',

            borderWidth: 2,

            data: dataWrong

          },

        ]

      }

      return (

        <>

          <div className="wrapper">

            <Topbar text={<span><i className="fas fa-tachometer-alt"></i> Dashboard</span>}>

              <IconButton onClick={this.props.toggleDrawer('left', true)}> <i style={{ fontSize: "20px" }} className="fas fa-bars  text-white"></i> </IconButton >

              <IconButton className="float-right" onClick={this.props.togglePopper}> <i style={{ fontSize: "20px" }} className="fas fa-ellipsis-v  text-white"></i> </IconButton >

            </Topbar>

            <div className="card my-3">

              <h5 className="card-header font-weight-light"><i className="fas fa-chart-line"></i> Test statistics</h5>

              <div className="card-body">

                <div className="statics-boxes">

                  <div className="row justify-content-center align-items-center my-2">

                    <div className="col-4">

                      <CircularProgressbarWithChildren

                        strokeWidth={1}

                        value={(total != 0) ? Math.round((this.props.studentData.right / (total)) \* 100) : 0}

                        styles={buildStyles({

                          pathColor: `var(--primary)`,

                          // textColor: '#f88',

                          trailColor: '#d6d6d6',

                          backgroundColor: '#3e98c7',

                        })}

                      >

                        {(total != 0) ? Math.round((this.props.studentData.right / (total)) \* 100) : 0} %

                        <h6 style={{ fontSize: "11px", textAlign: "center" }} className="font-weight-light">Accuracy</h6>

                      </CircularProgressbarWithChildren>

                    </div>

                    <div style={{ fontSize: "13px" }} className="col-8">

                      <div className="row">

                        <div className="col-6">Right</div>

                        <div className="col-6 text-right">{this.props.studentData.right}</div>

                      </div>

                      <div className="row">

                        <div className="col-6">Wrong</div>

                        <div className="col-6 text-right">{this.props.studentData.wrong}</div>

                      </div>

                      <div className="row">

                        <div className="col-6">Unanswered</div>

                        <div className="col-6 text-right">{this.props.studentData.unanswerd}</div>

                      </div>

                    </div>

                  </div>

                  <div className="row justify-content-center align-items-center my-2">

                    <div className="col-4">

                      <CircularProgressbarWithChildren

                        strokeWidth={1}

                        value={100}

                        styles={buildStyles({

                          pathColor: `var(--secondary)`,

                          // textColor: '#f88',

                          trailColor: '#d6d6d6',

                          backgroundColor: '#3e98c7',

                        })}

                      >

                        {total}

                        <h6 style={{ fontSize: "11px", textAlign: "center" }} className="font-weight-light">Questions <br />attempted</h6>

                      </CircularProgressbarWithChildren>

                    </div>

                    <div style={{ fontSize: "13px" }} className="col-8">

                      <div className="row">

                        <div className="col-6">Mock Tests</div>

                        <div className="col-6 text-right">{totalTests}</div>

                      </div>

                    </div>

                  </div>

                </div>

              </div>

              <div className="col-12 px-1">

                <Line

                  data={chartState}

                  options={{

                    title: {

                      display: true,

                      text: 'Average Rainfall per month',

                      fontSize: 20

                    },

                    legend: {

                      display: true,

                      position: 'right'

                    }

                  }}

                />

              </div>

            </div>

            <div className="card my-3">

              <h5 className="card-header font-weight-light"> Scheduled Tests</h5>

              <SchedulledTest history={{ push: this.push }} limit={3} topbar={false} />

            </div>

            <Notice topbar={false} />

            <br />

            <br />

            <br />

            <br />

            <br />

            <br />

            <br />

          </div>

        </>

      )

    } else {

      return <div style={{ height: "100vh" }} className="d-flex justify-content-center align-items-center">

        <div style={{ height: "100px", width: "100px" }} className="spinner-border text-primary" ></div>

      </div>

    }

  }

}

const mapStateToProps = (state) => ({

  studentData: state.studentData

})

export default connect(mapStateToProps, null)(Dashboard);

**MockTest.js**

import { Button, Dialog, DialogActions, DialogContent, DialogContentText, DialogTitle, Divider, IconButton, Link, List, ListItem, ListItemSecondaryAction, ListItemText, ListSubheader } from '@material-ui/core'

import React, { Component } from 'react'

import LongTopbar from '../../components/LongTopbar'

import axios from 'axios';

import { rootURL } from '../../globals/\_\_gobal\_vars';

import { LOADEROFF, LOADERON } from '../../globals/\_\_global\_funcs';

export default class MockTest extends Component {

  constructor(props) {

    super(props);

    this.state = {

      scrollTop: 0,

      testList: [],

      visible: false,

      selectedItem: {}

    }

  }

  handleScroll = (e) => {

    this.setState({

      scrollTop: e.target.scrollTop

    })

  }

  componentDidMount() {

    LOADERON();

    axios.defaults.headers.common['Authorization'] = "Bearer " + localStorage.getItem('token');

    axios.post(rootURL + "/auth/getSubjectWiseQuestionSets", {

      test\_type: 1

    }).then((res) => {

      //success

      if (res.data.status) {

        this.setState({

          testList: res.data.data,

        })

        LOADEROFF();

      } else {

        LOADEROFF();

        console.log(res);

      }

    }).catch((err) => {

      LOADEROFF();

      console.log(err);

    })

  }

  render() {

    let total = this.props.studentData.right + this.props.studentData.wrong;

    return (

      <div>

        <div ref={this.myRef} onScroll={this.handleScroll} className="wrapper">

          <LongTopbar collapse={this.state.scrollTop > 65} text={<span><i className="fas fa-book"></i>&nbsp;&nbsp; Mock Test</span>}>

            <IconButton onClick={this.props.toggleDrawer('left', true)}> <i style={{ fontSize: "20px" }} className="fas fa-bars  text-white"></i> </IconButton >

            <IconButton className="float-right" onClick={this.props.togglePopper}> <i style={{ fontSize: "20px" }} className="fas fa-ellipsis-v  text-white"></i> </IconButton >

            <div className={`static-card-box ${this.state.scrollTop > 65 ? "static-card-box-collapse" : ""}`}>

              <div className="row justify-content-around">

                <div className="col-4">

                  <div className="static-card">

                    <div className="text-right"><i className="fas fa-percent text-primary"></i> &nbsp;&nbsp; </div>

                    <p className="text-left">{(total != 0) ? Math.round((this.props.studentData.right / (total)) \* 100) : 0}</p>

                  </div>

                </div>

                {/\* <div className="col-4">

                                    <div className="static-card">

                                        <div className="text-right"><i className="fas fa-stopwatch text-primary"></i> &nbsp;&nbsp; </div>

                                        <p className="text-left">2:39</p>

                                    </div>

                                </div> \*/}

                <div className="col-4">

                  <div className="static-card">

                    <div className="text-right"><i className="fas fa-pen-fancy text-primary"></i> &nbsp;&nbsp; </div>

                    <p className="text-left">{total}</p>

                  </div>

                </div>

              </div>

            </div>

          </LongTopbar>

          <div className="list-box">

            {

              this.state.testList.map((subject, index) => (

                <React.Fragment key={index}>

                  <List dense

                    subheader={

                      <ListSubheader component="div" id="nested-list-subheader">

                        {subject.title}

                      </ListSubheader>

                    } >

                    {

                      (subject.question\_sets.length > 0) ?

                        subject.question\_sets.map((set, i) => {

                          return (

                            <ListItem

                              key={i}

                              style={{

                                background: "#5c5c5c1f",

                                padding: "10px 15px",

                                margin: "5px 0"

                              }}

                              button>

                              <ListItemText id={`checkbox-list-secondary-label-${i}`} primary={set.title} />

                              <ListItemSecondaryAction>

                                <IconButton onClick={() => { this.setState({ visible: true, selectedItem: set }) }}> <i className="fas text-primary fa-arrow-alt-circle-right"></i> </IconButton >

                                {/\* <IconButton onClick={()=>{this.setState({visible : true , selectedItem : item})}}> <i className="fas text-success fa-check-circle"></i> </IconButton > \*/}

                              </ListItemSecondaryAction>

                            </ListItem>

                          );

                        })

                        :

                        <ListItem

                          style={{

                            background: "#5c5c5c1f",

                            padding: "10px 15px",

                            margin: "5px 0"

                          }}

                          button>

                          <ListItemText id={`checkbox-list-secondary-label-${1}`} primary={<>{"Sorry no Question sets available ..."}<i className="fas fa-sad-tear"></i></>} />

                        </ListItem>

                    }

                  </List>

                  <Divider />

                </React.Fragment>

              ))

            }

          </div>

        </div>

        <Dialog

          open={this.state.visible}

          onClose={() => { this.setState({ visible: false }) }}

          aria-labelledby="alert-dialog-title"

          aria-describedby="alert-dialog-description"

        >

          <DialogTitle id="alert-dialog-title">{"Test Details"}</DialogTitle>

          <DialogContent>

            <DialogContentText id="alert-dialog-description">

              {

                Object.keys(this.state.selectedItem).length !== 0 ?

                  <div className="row">

                    <div className="col-4">Title</div>

                    <div className="col-8">{this.state.selectedItem.title}</div>

                    <div className="col-4">Questions</div>

                    <div className="col-8">{this.state.selectedItem.q\_length}</div>

                    <div className="col-4">Time</div>

                    <div className="col-8">{parseInt(this.state.selectedItem.time)} min(s)</div>

                  </div> : ""

              }

            </DialogContentText>

          </DialogContent>

          <DialogActions>

            <Button onClick={() => { this.setState({ visible: false }) }} color="primary">

              Close

            </Button>

            <Button onClick={() => { this.setState({ visible: false }); this.props.history.push("/student/test/" + this.state.selectedItem.id + "/" + (parseInt(this.state.selectedItem.time))) }} color="primary" autoFocus>

              Start

            </Button>

          </DialogActions>

        </Dialog>

      </div>

    )

  }

}

**About.js**

import { IconButton } from '@material-ui/core'

import React, { Component } from 'react'

import Topbar from '../../components/Topbar'

export default class About extends Component {

  render() {

    return (

      <div>

        <div className="wrapper">

          <Topbar text={<span><i className="fas fa-info-circle"></i> About</span>}>

            <IconButton onClick={this.props.toggleDrawer('left', true)}> <i style={{ fontSize: "20px" }} className="fas fa-bars  text-white"></i> </IconButton >

            <IconButton className="float-right" onClick={this.props.togglePopper}> <i style={{ fontSize: "20px" }} className="fas fa-ellipsis-v  text-white"></i> </IconButton >

          </Topbar>

          <div className="card">

            <h4 className="card-header font-weight-light">Aim and Objective</h4>

            <div className="card-body">

              <p>

                The tremendous popularity of using online education system could

                never have been realized before. In this COVID-19 pendemic this become an

                important way of learning without going or travelling another place we can continue our learnings from home.

                That's why we approach this pattern of learning an we created a test taking hybrid mobile application as our college project named as

                <span className="text-primary"> BIST TestOBit</span>.

              </p>

              <ul>

                <li>

                  The scope of Test O Bit is provide support our student that make himself/herself better.

                </li>

                <li>

                  The bright future of our college students is also grows our college reputation.

                </li>

                <li>

                  We provide our app support into mobile phones that students can easily access it.

                </li>

              </ul>

            </div>

          </div>

        </div>

      </div>

    )

  }

}

**Notice.js**

import { IconButton } from '@material-ui/core'

import React, { Component } from 'react'

import Topbar from '../../components/Topbar'

export default class Notice extends Component {

    static defaultProps = {

        topbar: true,

    }

    render() {

        return (

            <div>

                <div className={`${this.props.topbar ? "wrapper" : ""}`}>

                    {

                        this.props.topbar ?

                        <Topbar text={<span><i className="fas fa-sticky-note"></i> Notice</span>}>

                            <IconButton onClick={this.props.toggleDrawer('left', true)}> <i style={{ fontSize: "20px" }} className="fas fa-bars  text-white"></i> </IconButton >

                            <IconButton className="float-right" onClick={this.props.togglePopper}> <i style={{ fontSize: "20px" }} className="fas fa-ellipsis-v  text-white"></i> </IconButton >

                        </Topbar>: ""

                    }

                    <div className="card">

                        <h5 className="card-header font-weight-ligh">Notice</h5>

                        <div className="card-body">

                            <ul className="notice-board">

                                <li> <a target="\_blank" rel="noreferrer" href="https://www.bistpurulia.org/"><span class="blink-badge">NEW</span> Admissions Open </a></li>

                                <li><a target="\_blank" rel="noreferrer" href="https://www.bistpurulia.org/uploads/creditcardscheme.jpeg"><span class="blink-badge">IMPORTANT</span>Notice for West Bengal Students Credit Card Scheme.</a> </li>

                                <li><a target="\_blank" rel="noreferrer" href="https://www.bistpurulia.org/uploads/regulation.jpeg"><span class="blink-badge">IMPORTANT</span> Regulation relating to Under Graduate Course of Studies and Examination.</a> </li>

                                <li>You’re braver than you believe and stronger than you seem, and smarter than you think - so follow guidelines and stay safe.</li>

                            </ul>

                        </div>

                    </div>

                </div>

            </div>

        )

    }

}

**Profile.js**

import { Button, IconButton } from '@material-ui/core'

import React, { Component } from 'react'

import { connect } from 'react-redux'

import { imageURL } from '../../globals/\_\_gobal\_vars'

class Profile extends Component {

    render() {

        return (

            <div className="wrapper p-0">

                <div className="back">

                <div className="row">

                    <div className="col-3">

                        <IconButton onClick={this.props.toggleDrawer('left', true)}> <i style={{ fontSize: "20px" }} className="fas fa-bars  text-white"></i> </IconButton >

                    </div>

                    <div className="col-6">

                        <div className="topbar-text topbar-long-text h-30"><i className="fas fa-user-circle"></i> Profile</div>

                    </div>

                    <div className="col-3">

                        <IconButton className="float-right" onClick={this.props.togglePopper}> <i style={{ fontSize: "20px" }} className="fas fa-ellipsis-v  text-white"></i> </IconButton >

                    </div>

                </div>

                </div>

                <div className="front-card">

                    <div className="img-avatar">

                        <img src={(this.props.user.photo !== null)? imageURL+this.props.user.photo : "./assets/img/user.png"} alt="image" />

                    </div>

                    <div className="front-card-body">

                        <div className="row justify-content-center">

                            <div className="col-12 text-center">

                                <label  className="text-primary " onClick={()=>{this.props.history.push('/all-set/1')}} color="primary">Change picture</label>

                            </div>

                        </div>

                        <div className="user-name">{this.props.user.name ?? "" }</div>

                        <div className="text-muted text-center">{this.props.user.email ?? ""}</div>

                        <center><hr width="85%" /></center>

                        <div className="row py-2">

                            <div className="col-6">Phone</div>

                            <div className="col-6 text-right">{this.props.user.phone ?? ""}</div>

                        </div>

                        <div className="row py-2">

                            <div className="col-6">Date of birth</div>

                            <div className="col-6 text-right">{this.props.user.dob ?? ""}</div>

                        </div>

                        <div className="row py-2">

                            <div className="col-6">Semester</div>

                            <div className="col-6 text-right">{this.props.user.semester\_title ?? ""}</div>

                        </div>

                        <div className="row py-2">

                            <div className="col-6">Department</div>

                            <div className="col-6 text-right">{this.props.user.stream\_title ?? ""}</div>

                        </div>

                        <div className="row py-2">

                            <div className="col-6">Session</div>

                            <div className="col-6 text-right">2018-2019</div>

                        </div>

                        <p className="text-muted text-center">User created at 19-2-2021</p>

                    </div>

                </div>

            </div>

        )

    }

}

const mapStateToProps = (state) =>(

    {

        user : state.auth.user

    }

)

export default connect(mapStateToProps,null)(Profile)

**SchedulledTest.js**

import { Link } from "react-router-dom"

import React, { Component } from 'react'

import Topbar from '../../components/Topbar'

import axios from 'axios';

import { rootURL } from '../../globals/\_\_gobal\_vars';

import { Button, Dialog, DialogActions, DialogContent, DialogContentText, DialogTitle, Divider, IconButton, List, ListItem, ListItemSecondaryAction, ListItemText, ListSubheader } from '@material-ui/core'

import { futurePastDetecter, LOADEROFF, LOADERON, timeConvert } from '../../globals/\_\_global\_funcs';

export default class SchedulledTest extends Component {

    static defaultProps = {

        topbar: true,

        limit: 50

    }

    state = {

        testList: [],

        visible: false,

        selectedItem: {}

    }

    dataFetcher = (loader) => {

        if (loader) {

            LOADERON();

        }

        axios.defaults.headers.common['Authorization'] = "Bearer " + localStorage.getItem('token');

        axios.post(rootURL + "/auth/getScheduledTests", {

            limit: this.props.limit

        }).then((res) => {

            //success

            if (res.data.status) {

                this.setState({

                    testList: res.data.data,

                })

                if (loader) {

                    LOADEROFF();

                }

            } else {

                if (loader) {

                    LOADEROFF();

                }

                console.log(res);

            }

        }).catch((err) => {

            if (loader) {

                LOADEROFF();

            }

            console.log(err);

        });

    }

    componentDidMount() {

        this.dataFetcher(true)

        // console.log(futurePastDetecter("2021-05-19", "14"));

    }

    colorDetector = (date) => {

        var dateStr = date.split(" ")[0]

        switch (futurePastDetecter(dateStr, date.split(" ")[1])) {

            case 'Upcoming today':

                return "var(--primary)"

            case 'Running':

                return "var(--success)"

            case 'Upcoming':

                return "var(--primary)"

            case 'Missed':

                return "var(--secondary)"

            default:

                break;

        }

    }

    futurePastDetecter2 = (givenDate, hours) => {

        let now = new Date();

        let date = new Date(givenDate);

        // return date.getHours();

        let [h, m, s] = hours.split(":");

        let d = new Date()

        if (date.setHours(0, 0, 0, 0) > now.setHours(0, 0, 0, 0)) {

            return "Upcoming";

        } else if (date.setHours(0, 0, 0, 0) < now.setHours(0, 0, 0, 0)) {

            return "Missed"

        } else {

            // return (parseInt(h)\*60 + parseInt(m)+60 )+  " = "+ (d.getHours()\*60 + d.getMinutes())

            if ((parseInt(h) \* 60 + parseInt(m)) < (d.getHours() \* 60 + d.getMinutes())) {

                if ((parseInt(h) \* 60 + parseInt(m) + 60) > (d.getHours() \* 60 + d.getMinutes())) {

                    return "Running"

                } else {

                    return "Missed"

                }

            } else {

                return "Upcoming today"

            }

        }

    }

    revealQuestions = (str) => {

        switch (str) {

            case "Running":

                return false

            case "Missed":

                return false

            default:

                return true

        }

    }

    render() {

        return (

            <div className={`${this.props.topbar ? "wrapper" : ""}`}>

                {

                    this.props.topbar ?

                        <Topbar text={<span><i className="fas fa-users"></i> SchedulledTest</span>}>

                            <IconButton onClick={this.props.toggleDrawer('left', true)}> <i style={{ fontSize: "20px" }} className="fas fa-bars  text-white"></i> </IconButton >

                            <IconButton className="float-right" onClick={this.props.togglePopper}> <i style={{ fontSize: "20px" }} className="fas fa-ellipsis-v  text-white"></i> </IconButton >

                        </Topbar> : ""

                }

                <div className="card my-3">

                    {

                        (this.state.testList.length > 0) ?

                            <div className="card-body p-2">

                                <div className="row justify-content-end">

                                    <div className="col-4">

                                        <Button onClick={() => { this.dataFetcher(true); }} color="primary"> Refresh </Button>

                                    </div>

                                </div>

                                {

                                    this.state.testList.map((item, index) => (

                                        <div key={index} style={{ borderColor: item.date ? this.colorDetector(item.date) : "var(--primary)" }} className="schedule-box">

                                            <div style={{ fontSize: "12px" }} className="row justify-content-center">

                                                <div className="col-8 "><i style={{ color: item.date ? this.colorDetector(item.date) : "var(--primary)" }} className="fas fa-calendar-day"></i> {item.date ? item.date.split(" ")[0].split("-").reverse().join("-") : ""} at {item.date ? timeConvert(item.date.split(" ")[1]) : ""}</div>

                                                <div className="col-4 text-right"><i style={{ color: item.date ? this.colorDetector(item.date) : "var(--primary)" }} className="fas fa-clock"></i> <i>{item.time ? parseInt(item.time) : ""} min</i></div>

                                                <div className="col-9 "><i style={{ color: item.date ? this.colorDetector(item.date) : "var(--primary)" }} className="fas fa-angle-double-right"></i> {item.subject.title ?? ""}</div>

                                                <div className="col-3 text-right"><i style={{ color: item.date ? this.colorDetector(item.date) : "var(--primary)" }} className="fas fa-angle-double-right"></i> {item.q\_length} Qs</div>

                                                <div className="col-11 my-2">

                                                    <h4 className="font-weight-light">{item.title}</h4>

                                                    <br />

                                                </div>

                                            </div>

                                            <div className="bottom-box ">

                                                <div className="row align-items-end">

                                                    <div className="col-8">

                                                        <div className="text-muted">{item.date ? futurePastDetecter(item.date.split(" ")[0], item.date.split(" ")[1]) : ""} ...</div>

                                                    </div>

                                                    <div className="col-4">

                                                        <button disabled={item.date ? (this.revealQuestions(futurePastDetecter(item.date.split(" ")[0], item.date.split(" ")[1]))) : false} className="btn btn-block  btn-sm float-right" onClick={() => { this.setState({ visible: true, selectedItem: item }) }} style={{ borderRadius: "0", borderTopLeftRadius: "20px", cursor: "pointer", background: item.date ? this.colorDetector(item.date) : "var(--primary)", color: "white" }}  >Go  &nbsp;<i className="fas fa-caret-right"></i></button>

                                                    </div>

                                                </div>

                                            </div>

                                        </div>

                                    ))

                                }

                            </div>

                            :

                            <h3 className="font-weight-light text-center p-5">Sorry no tests are shedulled .</h3>

                    }

                </div>

                <Dialog

                    open={this.state.visible}

                    onClose={() => { this.setState({ visible: false }) }}

                    aria-labelledby="alert-dialog-title"

                    aria-describedby="alert-dialog-description"

                >

                    <DialogTitle id="alert-dialog-title">{"Test Details"}</DialogTitle>

                    <DialogContent>

                        <DialogContentText id="alert-dialog-description">

                            {

                                Object.keys(this.state.selectedItem).length !== 0 ?

                                    <div className="row">

                                        <div className="col-4">Title</div>

                                        <div className="col-8">{this.state.selectedItem.title}</div>

                                        <div className="col-4">Questions</div>

                                        <div className="col-8">{this.state.selectedItem.q\_length}</div>

                                        <div className="col-4">Time</div>

                                        <div className="col-8">{parseInt(this.state.selectedItem.time)} min</div>

                                    </div> : ""

                            }

                        </DialogContentText>

                    </DialogContent>

                    <DialogActions>

                        <Button onClick={() => { this.setState({ visible: false }) }} color="primary">

                            Close

                    </Button>

                        <Button onClick={() => { this.setState({ visible: false }); this.props.history.push("/student/test/" + this.state.selectedItem.id + "/" + (parseInt(this.state.selectedItem.time))) }} color="primary" autoFocus>

                            Start

                    </Button>

                    </DialogActions>

                </Dialog>

            </div>

        )

    }

}

**TestCMP.js**

import { Button, Divider } from '@material-ui/core'

import React, { Component } from 'react'

import Radio from '@material-ui/core/Radio';

import RadioGroup from '@material-ui/core/RadioGroup';

import FormControlLabel from '@material-ui/core/FormControlLabel';

import FormControl from '@material-ui/core/FormControl';

import { LOADEROFF, LOADERON, notification } from '../../globals/\_\_global\_funcs';

import { rootURL } from '../../globals/\_\_gobal\_vars';

import axios from "axios";

import { connect } from 'react-redux';

import { setTestResult } from '../../redux/actions/TestActions';

class TestCMP extends Component {

    constructor(props) {

        super(props);

        this.state = {

            questionList: [],

            choice: null,

            q\_index: 1,

        }

    }

    componentDidMount() {

        LOADERON();

        axios.defaults.headers.common['Authorization'] = "Bearer " + localStorage.getItem('token');

        let questionCall = axios.post(rootURL + "/auth/getQuestions", {

            set\_id: this.props.match.params.id

        }).then((res) => {

            //success

            if (res.data.status) {

                let temp = res.data.data.map((item, index) => {

                    return {

                        ...item,

                        selectedAnswer: ""

                    }

                })

                this.setState({

                    questionList: temp

                })

                LOADEROFF();

                // setTimeout(() => {

                // }, 3000);

            } else {

                LOADEROFF();

                console.log(res);

            }

        }).catch((err) => {

            LOADEROFF();

            console.log(err);

        })

        Promise.all([questionCall]).then(() => {

            this.countdown("timer", parseInt(this.props.match.params.time), 0).then((isTimeup) => {

                if (isTimeup) {

                    notification("Time's Up !!!", "onClickReturn").then((isTimeup) => { if (isTimeup) { this.submitTest(); } });

                }

            })

        })

    }

    countdown = (elementName, minutes, seconds) => {

        return new Promise((resolve, reject) => {

            var element, endTime, hours, mins, msLeft, time;

            function twoDigits(n) {

                return (n <= 9 ? "0" + n : n);

            }

            function updateTimer() {

                msLeft = endTime - (+new Date());

                if (msLeft <= 0) {

                    resolve(true)

                } else if (!document.getElementById(elementName)) {

                    resolve(false)

                } else {

                    time = new Date(msLeft);

                    hours = time.getUTCHours();

                    mins = time.getUTCMinutes();

                    element.innerHTML = (hours ? hours + ':' + twoDigits(mins) : mins) + ':' + twoDigits(time.getUTCSeconds());

                    setTimeout(updateTimer, time.getUTCMilliseconds() + 500);

                }

            }

            element = document.getElementById(elementName);

            endTime = (+new Date()) + 1000 \* (60 \* minutes + seconds) + 500;

            updateTimer()

        })

    }

    submitTest = () => {

        let unanswered = 0;

        this.state.questionList.map((item, index) => {

            if (item.selectedAnswer === "") {

                unanswered++;

            }

        })

        if (unanswered > 0) {

            notification(`You have ${unanswered} Unanswered question(s) are you sure to submit ??`, "confirm").then((isConfirm) => {

                if (isConfirm) {

                    this.props.setTestResult(this.state.questionList)

                    this.props.history.push("/student/test-result");

                }

            })

        } else {

            this.props.setTestResult(this.state.questionList)

            this.props.history.push("/student/test-result");

        }

    }

    selectAnswer = (id, value) => {

        let temp = this.state.questionList.map((item, index) => {

            if (item.id == id) {

                return {

                    ...item,

                    selectedAnswer: value

                }

            }

            return item

        })

        this.setState({ questionList: temp })

    }

    render() {

        let presentQuestion = this.state.questionList[this.state.q\_index - 1]

        return (

            <div className="wrapper p-0">

                <div className="test-main">

                    {

                        (this.state.questionList.length > 0) ?

                            <React.Fragment>

                                <div className="">

                                    <div className="row mb-2">

                                        <div className="col-8">

                                            <Button onClick={() => { notification("Are you sure to leave test ??", "confirm").then((isConfirm) => { if (isConfirm) { this.props.history.goBack() } }) }} color="primary"> Exit </Button>

                                            <b className="text-primary" >Q</b> : {this.state.q\_index}/{this.state.questionList.length}

                                        </div>

                                        <div className="col-4 text-right">

                                            <i className="fas fa-stopwatch text-primary"></i> <span id="timer">wait..</span>

                                        </div>

                                    </div>

                                    <Divider />

                                    <div className="test-area">

                                        <div className="row my-2">

                                            <div className="col-12">

                                                <p className="text-justify">

                                                    <b>Q.{this.state.q\_index} &nbsp; &nbsp;&nbsp;&nbsp;</b>

                                                    <span>{presentQuestion.question} </span>

                                                </p>

                                            </div>

                                        </div>

                                        <div className="row my-2 pl-3">

                                            <div className="col-12">

                                                <FormControl>

                                                    <RadioGroup aria-label="choices" name="gender1" value={presentQuestion.selectedAnswer} >

                                                        <FormControlLabel onChange={(e) => { this.selectAnswer(presentQuestion.id, e.target.value) }} value={presentQuestion.op1} control={<Radio />} label={presentQuestion.op1} />

                                                        <FormControlLabel onChange={(e) => { this.selectAnswer(presentQuestion.id, e.target.value) }} value={presentQuestion.op2} control={<Radio />} label={presentQuestion.op2} />

                                                        <FormControlLabel onChange={(e) => { this.selectAnswer(presentQuestion.id, e.target.value) }} value={presentQuestion.op3} control={<Radio />} label={presentQuestion.op3} />

                                                        <FormControlLabel onChange={(e) => { this.selectAnswer(presentQuestion.id, e.target.value) }} value={presentQuestion.op4} control={<Radio />} label={presentQuestion.op4} />

                                                    </RadioGroup>

                                                </FormControl>

                                            </div>

                                        </div>

                                    </div>

                                </div>

                                <div className="bottom-nav">

                                    <div className="">

                                        <div className="row">

                                            <div className="col-6 px-0">

                                                {

                                                    (this.state.q\_index > 1) ?

                                                        <button onClick={() => { this.setState({ q\_index: this.state.q\_index - 1 }) }} className="btn btn-sm btn-block py-3 rounded-0 text-white btn-primary" >Prev</button>

                                                        : ""

                                                }

                                            </div>

                                            {/\* <div className="col-4 px-0">

                                                <button onClick={()=>{this.setState({q\_index : this.state.q\_index + 1})}} className="btn btn-sm btn-block py-3 rounded-0 text-white btn-primary" >Skip</button>

                                            </div> \*/}

                                            <div className="col-6 px-0">

                                                {

                                                    (this.state.q\_index == this.state.questionList.length) ?

                                                        <button onClick={() => { this.submitTest() }} className="btn btn-sm btn-block py-3 rounded-0 text-white btn-success" >Submit</button>

                                                        :

                                                        <button onClick={() => { this.setState({ q\_index: this.state.q\_index + 1 }) }} className="btn btn-sm btn-block py-3 rounded-0 text-white btn-primary" >Next</button>

                                                }

                                            </div>

                                        </div>

                                    </div>

                                </div>

                            </React.Fragment> :

                            <>

                                <div className="row mb-2">

                                    <div className="col-8">

                                        <Button onClick={() => { this.props.history.goBack() }} color="primary"> Back </Button>

                                    </div>

                                </div>

                                <h3 className="font-weight-light p-3 text-center" >

                                    Sorry No Questions available in this set .

                                </h3 >

                            </>

                    }

                </div>

                <div className="side-q ">

                    {

                        this.state.questionList.map((item, index) => (

                            <div className="q-box">

                                <button key={index} onClick={() => { this.setState({ q\_index: index + 1 }) }} className={`btn btn-floating ${index == this.state.q\_index - 1 ? "bg-secondary text-white" : "bg-white"}  mx-auto`}>{index + 1}</button>

                            </div>

                        ))

                    }

                </div>

            </div>

        );

    }

}

export default connect(null, { setTestResult })(TestCMP)

**TestResult.js**

import { Button, IconButton } from '@material-ui/core';

import React, { Component } from 'react'

import { connect } from 'react-redux'

import Topbar from '../../components/Topbar';

import { CircularProgressbarWithChildren, buildStyles } from 'react-circular-progressbar';

import 'react-circular-progressbar/dist/styles.css';

import { setStudentData } from '../../redux/actions/TestActions';

class TestResult extends Component {

    constructor(props) {

        super(props);

        this.state = {

            result: this.props.result,

            right : 0,

            wrong : 0,

            unanswered : 0,

            totalQuestion : 0,

        }

    }

    componentDidMount(){

        let res =  this.props.result

        let right = 0;

        let wrong = 0;

        let unanswered = 0;

        res.map((item,index)=>{

            if(item.answer === item.selectedAnswer ){

                right++;

            }else if(item.answer !== item.selectedAnswer && item.selectedAnswer !== "") {

                wrong++;

            }else{

                unanswered++;

            }

        })

        this.setState({

            right,

            wrong,

            unanswered,

            totalQuestion : res.length

        })

    }

    iconReturner = (item,op) =>{

        if(item.selectedAnswer === ""){

            return "far fa-dot-circle text-primary";

        }else{

            if(item[op] !== item.answer){

                if(item[op] === item.selectedAnswer ){

                    return "fas fa-times-circle text-secondary"

                }else{

                    return "far fa-dot-circle text-primary";

                }

            }else{

                if( item.answer === item.selectedAnswer  ){

                    return "fas fa-check-double text-success"

                }else{

                    return "fas fa-check-circle text-success"

                }

            }

        }

    }

    render() {

        return (

            <div className="wrapper">

                <Topbar text={<span><i className="fas fa-invoice"></i> Result </span>}>

                    <IconButton onClick={this.props.toggleDrawer('left', true)}> <i style={{ fontSize: "20px" }} className="fas fa-bars  text-white"></i> </IconButton >

                    <Button style={{backgroundColor : "#ffffff4a"}} className="float-right  text-white  rounded-0" onClick={() => { this.props.setStudentData() ; this.props.history.push("/student/dashboard") } }> Go Home </Button >

                </Topbar>

                <div className="performance container-fluid">

                    <div className="row  justify-content-center">

                        <div className="col-8">

                        <CircularProgressbarWithChildren

                                strokeWidth={3}

                                value={Math.round((this.state.right / this.state.totalQuestion)\*100)}

                                styles={buildStyles({

                                    pathColor: `rgb(255, 134, 0)`,

                                    // textColor: '#f88',

                                    trailColor: '#d6d6d6',

                                    backgroundColor: '#3e98c7',

                                })}

                            >

                            <h2 className="font-weight-light">Your Score</h2>

                            <div className="row">

                                <div className="col-6 text-left">{this.state.right}/{this.state.totalQuestion}</div>

                                <div className="col-6 text-right">{Math.round((this.state.right / this.state.totalQuestion)\*100)}%</div>

                            </div>

                            </CircularProgressbarWithChildren>

                        </div>

                    </div>

                    <div className="row my-3">

                        <div className="col-4 p-2">

                            <CircularProgressbarWithChildren

                                strokeWidth={3}

                                value={Math.round((this.state.right / this.state.totalQuestion)\*100)}

                                styles={buildStyles({

                                    pathColor: `var(--success)`,

                                    // textColor: '#f88',

                                    trailColor: '#d6d6d6',

                                    backgroundColor: '#3e98c7',

                                })}

                            >

                                {this.state.right}

                            <h6 className="font-weight-light">Right</h6>

                            </CircularProgressbarWithChildren>

                        </div>

                        <div className="col-4 p-2">

                            <CircularProgressbarWithChildren

                                strokeWidth={3}

                                value={Math.round((this.state.wrong / this.state.totalQuestion)\*100)}

                                styles={buildStyles({

                                    pathColor: `var(--secondary)`,

                                    // textColor: '#f88',

                                    trailColor: '#d6d6d6',

                                    backgroundColor: '#3e98c7',

                                })}

                            >

                                {this.state.wrong }

                            <h6 className="font-weight-light">Wrong</h6>

                            </CircularProgressbarWithChildren>

                        </div>

                        <div className="col-4 p-2">

                            <CircularProgressbarWithChildren

                                strokeWidth={3}

                                value={Math.round((this.state.unanswered / this.state.totalQuestion)\*100)}

                                styles={buildStyles({

                                    pathColor: `var(--primary)`,

                                    // textColor: '#f88',

                                    trailColor: '#d6d6d6',

                                    backgroundColor: '#3e98c7',

                                })}

                            >

                              {this.state.unanswered }

                            <h6 className="font-weight-light">Unanswerd</h6>

                            </CircularProgressbarWithChildren>

                        </div>

                    </div>

                </div>

                    <div  style={{fontSize : "14px"}} className="row">

                        <div className="col-4 text-center">

                            <div className="circle"> <i className="fas fa-times-circle text-secondary"></i> </div>

                            <div >Given wrong answer</div>

                        </div>

                        <div className="col-4 text-center">

                            <div className="circle"> <i className="fas fa-check-circle text-success"></i> </div>

                            <div >Correct answer</div>

                        </div>

                        <div className="col-4 text-center">

                            <div className="circle"> <i className="fas fa-check-double text-success"></i> </div>

                            <div >Given right answer</div>

                        </div>

                    </div>

                    <br />

                {

                    this.props.result.map((item,index)=>(

                        <div key={index} className="answer-box">

                            <h5>Question : {index + 1}</h5><p style={{textIndent : "20px"}} className="text-justify"><i className="fas fa-angle-double-right"></i> {item.question}</p>

                            <ul>

                                <li><span className="circle"> <i className={this.iconReturner(item,"op1")}></i> </span> &nbsp;&nbsp;&nbsp; {item.op1}</li>

                                <li><span className="circle"> <i className={this.iconReturner(item,"op2")}></i> </span> &nbsp;&nbsp;&nbsp; {item.op2}</li>

                                <li><span className="circle"> <i className={this.iconReturner(item,"op3")}></i> </span> &nbsp;&nbsp;&nbsp; {item.op3}</li>

                                <li><span className="circle"> <i className={this.iconReturner(item,"op4")}></i> </span> &nbsp;&nbsp;&nbsp; {item.op4}</li>

                            </ul>

                        </div>

                    ))

                }

            </div>

        )

    }

}

const mapStateToProps = (state) => ({

    result: state.testResult.result

})

export default connect(mapStateToProps, {setStudentData})(TestResult)

* **11.2 Backend :**

**1.AuthController.php**

**1.1 Login User :**

public function login()

    {

        $credentials = request(['email', 'password']);

        if (! $token = auth()->attempt($credentials)) {

            return response()->json([

                'status' => false,

                'error' => 'Incorrect Email and Password'

            ]);

        }

        return $this->respondWithToken($token);

    }

    public function checkCredentials(Request $req)

    {

        $rules = [

            'password' => 'required',

            'user\_id' => 'required',

        ];

        $messages = [

            'password.required' => 'password is required!',

            'user\_id.required' => 'user\_id is required!',

        ];

        $valid = Validator::make($req->all(), $rules, $messages);

        if($valid->passes()) {

            try{

                $credentials = ["email"=> auth()->user()->email , "password" => $req->password];

                if (auth()->validate($credentials) && auth()->user()->user\_type == 7) {

                    // credentials are valid

                    $users = User::where(["user\_type"=> "2","id"=>$req->user\_id])->select(["id","name","dob","gender","photo","email","session","real\_pass","created\_at",'stream','semester'])->first();

                    return response()->json([

                        'status' => true,

                        'data' => $users

                    ]);

                }else{

                    return response()->json([

                        'status' => false,

                        'error' => 'Invalid Credentials'

                    ]);

                }

            }catch(\Exception $e) {

                return response()->json([

                    'status' => false,

                    'data' => "Failed to Load data",

                    'msg' => $e->getMessage()

               ]);

            }

        }else {

            return response()->json([

                'status' => false,

                'data' => (object)[],

                'errors' => $valid->errors()->all() ?? []

            ]);

        }

    }

**1.2 Register User :**

public function register(Request $request)

    {

        $validator = Validator::make($request->all(), [

            'email' => 'required|email|max:255|unique:users',

            'name' => 'required',

            'dob' => 'required',

            'gender' => 'required',

            'semester' => 'required',

            'phone' => 'required|max:10',

            'stream'=> 'required',

            'session'=> 'required',

            'password'=> 'required',

        ]);

        if ($validator->fails()) {

            return response()->json([

                'status' => false,

                'errors' => $validator->errors()->all()

            ]);

        } else {

            $sem = Semester::find($request->semester)->first();

            $stream = Stream::find($request->stream)->first();

            $user = new User();

            $user->name = $request->name;

            $user->email = strtolower($request->email);

            $user->dob = $request->dob;

            $user->gender =  $request->gender;

            $user->semester =  $request->semester;

            $user->phone =  $request->phone;

            $user->stream = $request->stream;

            $user->session =  $request->session;

            $user->real\_pass = $request->password;

            $user->password = bcrypt($user->real\_pass);

            $user->user\_type = 2;

            $user->save();

            $user['semester\_title'] = $sem->title;

            $user['stream\_title'] = $stream->title;

            $token = JWTAuth::fromUser($user);

            return response()->json([

                'status' => true,

                'access\_token' => $token,

                'token\_type' => 'bearer',

                'user' => $user ?? (object)[],

                'expires\_in' => auth()->factory()->getTTL() \* 60

            ]);

        }

    }

**1.3 Upload picture:**

public function uploadImage(Request $req)

    {

        $rules = [

            'photo' => 'required',

            'w' => 'required',

            'h' => 'required',

            'x' => 'required',

            'y' => 'required',

        ];

        $messages = [

            'photo.required' => 'photo is required!',

            'w.required' => 'w is required!',

            'h.required' => 'h is required!',

            'x.required' => 'x is required!',

            'y.required' => 'y is required!',

        ];

        $valid = Validator::make($req->all(), $rules, $messages);

        if($valid->passes()) {

            try{

                if ($req->file('photo')) {

                    $user = User::where(["id"=>auth()->user()->id])->first();

                    $image = $req->file('photo');

                    $image\_name = time().".".$image->getClientOriginalExtension();

                    // print\_r($image\_name);

                    $destination\_path = public\_path('/images');

                    $image->move($destination\_path,$image\_name);

                    $im\_path = $destination\_path."/".$image\_name;

                    $croppedImage = Image::make($im\_path);

                    $croppedImage->crop($req->w, $req->h, $req->x, $req->y);

                    $croppedImage->save($im\_path);

                    $user->photo = "images/".$image\_name;

                    $user->save();

                }

                return response()->json([

                    'status' => true,

                    'msg' => "image Uploded successfully",

                    'image\_path'=> "images/".$image\_name

                ]);

            }catch(\Exception $e) {

                return response()->json([

                    'status' => false,

                    'data' => (object)[],

                    'msg' => $e->getMessage()

               ]);

            }

        }else {

            return response()->json([

                'status' => false,

                'data' => (object)[],

                'errors' => $valid->errors()->all() ?? []

            ]);

        }

    }

**2. TestController.php**

**2.1 Get Schedulled Test:**

public function getScheduledTests(Request $req){

        $rules = [

            'limit' => 'required',

        ];

        $messages = [

            'limit.required' => 'limit is required!',

        ];

        $valid = Validator::make($req->all(), $rules, $messages);

        if($valid->passes()) {

            try{

                $data\_set = Question\_set::where(['test\_id'=> 4])->with('subject')->orderBy('date', 'DESC')->take($req->limit)->get();

                return response()->json([

                    'status' => true,

                    'data' => $data\_set,

               ]);

            }catch(\Exception $e) {

                return response()->json([

                    'status' => false,

                    'data' => (object)[],

                    'msg' => $e->getMessage()

               ]);

            }

        }else {

            return response()->json([

                'status' => false,

                'data' => (object)[],

                'errors' => $valid->errors()->all() ?? []

            ]);

        }

    }

**2.2 GetSubjectWiseQuestionSets :**public function getSubjectWiseQuestionSets(Request $req){

        $rules = [

            'test\_type' => 'required',

        ];

        $messages = [

            'test\_type.required' => 'test\_type is required!',

        ];

        $valid = Validator::make($req->all(), $rules, $messages);

        if($valid->passes()) {

            try{

                if($req->test\_type == "5" ){

                    $data\_set = Subject::where(['stream\_id'=>auth()->user()->stream])->with('question\_sets')->get();

                }else{

                    $data\_set = Subject::where(['sem\_id'=> auth()->user()->semester])->where(['stream\_id'=>auth()->user()->stream])->with('question\_sets')->get();

                }

                // $data\_set = Question\_set::select()->where(["test\_id" => $req->test\_type,"subject\_id"=>1])->get();

                $filterData =array();

                foreach($data\_set->toArray() as $value){

                     $q\_setArray = array();

                     foreach($value['question\_sets'] as $item){

                        // print\_r($req->test\_type);

                        if($item["test\_id"] == $req->test\_type){

                            $q\_setArray[]= $item;

                        }

                    }

                    $filterData[] =array\_merge($value,["question\_sets"=>$q\_setArray]) ;

                }

                return response()->json([

                    'status' => true,

                    'data' => $filterData,

               ]);

            }catch(\Exception $e) {

                return response()->json([

                    'status' => false,

                    'data' => (object)[],

                    'msg' => $e->getMessage()

               ]);

            }

        }else {

            return response()->json([

                'status' => false,

                'data' => (object)[],

                'errors' => $valid->errors()->all() ?? []

            ]);

        }

    }

**2.3 Create Question Set :**

public function createQuestionSetWithQuestions(Request $req){

        $rules = [

            'subject' => 'required',

            'test\_type' => 'required',

            'length' => 'required',

            'time' => 'required',

            'title' => 'required',

            'question\_array' => 'required',

        ];

        $messages = [

            'subject.required' => 'subject is required!',

            'test\_type.required' => 'test\_type is required!',

            'length.required' => 'length is required!',

            'time.required' => 'time is required!',

            'title.required' => 'title is required!',

            'question\_array.required' => 'question\_array is required!',

        ];

        $valid = Validator::make($req->all(), $rules, $messages);

        if($valid->passes()) {

            try{

                $set = new Question\_set() ;

                $set->subject\_id = $req->subject;

                $set->test\_id = $req->test\_type;

                $set->q\_length = $req->length;

                $set->date = $req->date ?? null;

                $set->time = $req->time;

                $set->title = $req->title;

                $set->save();

                $data = $req->toArray();

                foreach($data['question\_array'] as $q){

                    $question = new Question() ;

                    $question->question = $q['question'];

                    $question->question\_set\_id = $set->id;

                    $question->op1 = $q['op1'];

                    $question->op2 = $q['op2'];

                    $question->op3 = $q['op3'];

                    $question->op4 = $q['op4'];

                    $question->answer = $q['answer'];

                    $question->save();

                }

                return response()->json([

                    'status' => true,

                    'data' => "Succesfully Created",

                    // 'data' => $req->all(),

               ]);

            }catch(\Exception $e) {

                return response()->json([

                    'status' => false,

                    'data' => "Failed to make Question set",

                    'msg' => $e->getMessage()

               ]);

            }

        }else {

            return response()->json([

                'status' => false,

                'data' => (object)[],

                'errors' => $valid->errors()->all() ?? []

            ]);

        }

    }

**2.4 Upload Answers:**

public function uploadAnswers(Request $req){

        $rules = [

            'answer\_array' => 'required',

        ];

        $messages = [

            'answer\_array.required' => 'answer\_array is required!',

        ];

        $valid = Validator::make($req->all(), $rules, $messages);

        // print\_r($data['question\_array']);

        // exit;

        if($valid->passes()) {

            try{

                $data = $req->toArray();

                $testSession = auth()->user()->id.date("Ymdhis");

                foreach($data['answer\_array'] as $a){

                    $user\_answer = new User\_answer() ;

                    $user\_answer->question\_id = $a['id'];

                    $user\_answer->question\_set\_id = $a['question\_set\_id'];

                    $user\_answer->user\_id = auth()->user()->id;

                    $user\_answer->answer = $a['selectedAnswer'];

                    $user\_answer->test\_session = $testSession;

                    $user\_answer->save();

                }

                return response()->json([

                    'status' => true,

                    'data' => "Succesfully Created",

               ]);

            }catch(\Exception $e) {

                return response()->json([

                    'status' => false,

                    'data' => "Failed to make user answers",

                    'msg' => $e->getMessage()

               ]);

            }

        }else {

            return response()->json([

                'status' => false,

                'data' => (object)[],

                'errors' => $valid->errors()->all() ?? []

            ]);

        }

    }

**2.5 GetQuestions :**

public function getQuestions(Request $req){

        $rules = [

            'set\_id' => 'required',

        ];

        $messages = [

            'set\_id.required' => 'set\_id is required!',

        ];

        $valid = Validator::make($req->all(), $rules, $messages);

        if($valid->passes()) {

            try{

                $data\_set = Question::select()->where("question\_set\_id","=",$req->set\_id)->get();

                return response()->json([

                    'status' => true,

                    'data' => (object)$data\_set,

               ]);

            }catch(\Exception $e) {

                return response()->json([

                    'status' => false,

                    'data' => (object)[],

                    'msg' => $e->getMessage()

               ]);

            }

        }else {

            return response()->json([

                'status' => false,

                'data' => (object)[],

                'errors' => $valid->errors()->all() ?? []

            ]);

        }

    }

**2.6 GetSubjects :**

public function getSubjects () {

        try {

            $datas = Subject::select("id","title")->get();

           return response()->json([

                'error' => false,

                'data' => $datas

           ]);

        } catch(\Exception $e) {

            return response()->json([

                'error' => true,

                'data' => (object)[],

                'msg' => $e->getMessage()

           ]);

        }

    }

**11.3 Android Frame**

**1. AndroidManifest.xml**

*<?*xml version="1.0" encoding="utf-8"*?>*<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 package="com.astergo.testobit">  
  
 <application  
 android:allowBackup="true"  
 android:icon="@mipmap/ic\_launcher"  
 android:label="@string/app\_name"  
 android:roundIcon="@mipmap/ic\_launcher"  
 android:supportsRtl="true"  
 android:usesCleartextTraffic="true"  
 android:theme="@style/Theme.TestObit">  
 <activity android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|uiMode|screenSize|smallestScreenSize" android:name=".MainActivity" android:theme="@style/Theme.AppCompat.NoActionBar">  
 <intent-filter>  
 <action android:name="android.intent.action.MAIN" />  
  
 <category android:name="android.intent.category.LAUNCHER" />  
 </intent-filter>  
 </activity>  
 </application>  
  
 <uses-permission android:name="android.permission.INTERNET"  
 tools:ignore="ManifestOrder" />  
 <uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />  
 <uses-permission  
 android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"  
 tools:ignore="ScopedStorage" />  
  
  
</manifest>

**2.activity\_main.xml**

<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
  
 tools:context=".MainActivity"  
 >  
  
<WebView  
 android:id="@+id/webview"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true" />  
  
</RelativeLayout>

**3.MainActivity.java**

package com.astergo.testobit;  
  
import androidx.appcompat.app.AppCompatActivity;  
  
import android.Manifest;  
import android.app.DownloadManager;  
import android.app.ProgressDialog;  
import android.content.pm.PackageManager;  
import android.os.Bundle;  
import android.util.Log;  
import android.webkit.CookieManager;  
import android.webkit.DownloadListener;  
import android.webkit.URLUtil;  
import android.webkit.WebChromeClient;  
import android.webkit.WebSettings;  
import android.webkit.WebView;  
  
  
*//import class for Uploading part start*import android.app.Activity;  
import android.content.Context;  
import android.content.Intent;  
import android.net.Uri;  
import android.os.Build;  
import android.os.Environment;  
import android.os.Parcelable;  
import android.provider.MediaStore;  
import android.webkit.ValueCallback;  
import android.widget.Toast;  
import java.io.File;  
import java.io.IOException;  
  
*//import class for Uploading part End*public class MainActivity extends AppCompatActivity {  
 WebView webView;  
  
 public Context context;  
  
 private static final String *TAG* = com.astergo.testobit.MainActivity.class.getSimpleName();  
  
 private static final int *FILECHOOSER\_RESULTCODE* = 1;  
 private ValueCallback<Uri> mUploadMessage;  
 private Uri mCapturedImageURI = null;  
  
 *// the same for Android 5.0 methods only* private ValueCallback<Uri[]> mFilePathCallback;  
 private String mCameraPhotoPath;  
  
  
  
  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 final ProgressDialog progressDialog = new ProgressDialog(this);  
 progressDialog.setMessage("Loading Data...");  
 progressDialog.setCancelable(false);  
  
 WebView webView = findViewById(R.id.*webview*);  
 webView.setVerticalScrollBarEnabled(true);  
  
 WebSettings webSetting = webView.getSettings();  
 webSetting.setBuiltInZoomControls(false);  
 webSetting.setJavaScriptEnabled(true);  
 *//new code* webSetting.setDomStorageEnabled(true);  
  
  
 *//webView.setWebViewClient(new WebViewClient());* webView.setWebChromeClient(new WebChromeClient(){  
  
  
  
 *// for Lollipop, all in one* public boolean onShowFileChooser(  
 WebView webView, ValueCallback<Uri[]> filePathCallback,  
 WebChromeClient.FileChooserParams fileChooserParams) {  
 if (mFilePathCallback != null) {  
 mFilePathCallback.onReceiveValue(null);  
 }  
 mFilePathCallback = filePathCallback;  
  
 Intent takePictureIntent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 if (takePictureIntent.resolveActivity(getPackageManager()) != null) {  
  
 *// create the file where the photo should go* File photoFile = null;  
 try {  
 photoFile = createImageFile();  
 takePictureIntent.putExtra("PhotoPath", mCameraPhotoPath);  
 } catch (IOException ex) {  
 *// Error occurred while creating the File* Log.*e*(*TAG*, "Unable to create Image File", ex);  
 }  
  
 *// continue only if the file was successfully created* if (photoFile != null) {  
 mCameraPhotoPath = "file:" + photoFile.getAbsolutePath();  
 takePictureIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*,  
 Uri.*fromFile*(photoFile));  
 } else {  
 takePictureIntent = null;  
 }  
 }  
  
 Intent contentSelectionIntent = new Intent(Intent.*ACTION\_GET\_CONTENT*);  
 contentSelectionIntent.addCategory(Intent.*CATEGORY\_OPENABLE*);  
 contentSelectionIntent.setType("image/\*");  
  
 Intent[] intentArray;  
 if (takePictureIntent != null) {  
 intentArray = new Intent[]{takePictureIntent};  
 } else {  
 intentArray = new Intent[0];  
 }  
  
 Intent chooserIntent = new Intent(Intent.*ACTION\_CHOOSER*);  
 chooserIntent.putExtra(Intent.*EXTRA\_INTENT*, contentSelectionIntent);  
 chooserIntent.putExtra(Intent.*EXTRA\_TITLE*, getString(R.string.*image\_chooser*));  
 chooserIntent.putExtra(Intent.*EXTRA\_INITIAL\_INTENTS*, intentArray);  
  
 startActivityForResult(chooserIntent, *FILECHOOSER\_RESULTCODE*);  
  
 return true;  
 }  
  
 *// creating image files (Lollipop only)* private File createImageFile() throws IOException {  
  
 File imageStorageDir = new File(Environment.*getExternalStoragePublicDirectory*(Environment.*DIRECTORY\_PICTURES*), "DirectoryNameHere");  
  
 if (!imageStorageDir.exists()) {  
 imageStorageDir.mkdirs();  
 }  
  
 *// create an image file name* imageStorageDir = new File(imageStorageDir + File.*separator* + "IMG\_" + String.*valueOf*(System.*currentTimeMillis*()) + ".jpg");  
 return imageStorageDir;  
 }  
  
 *// openFileChooser for Android 3.0+* public void openFileChooser(ValueCallback<Uri> uploadMsg, String acceptType) {  
 mUploadMessage = uploadMsg;  
  
 try {  
 File imageStorageDir = new File(Environment.*getExternalStoragePublicDirectory*(Environment.*DIRECTORY\_PICTURES*), "DirectoryNameHere");  
  
 if (!imageStorageDir.exists()) {  
 imageStorageDir.mkdirs();  
 }  
  
 File file = new File(imageStorageDir + File.*separator* + "IMG\_" + String.*valueOf*(System.*currentTimeMillis*()) + ".jpg");  
  
 mCapturedImageURI = Uri.*fromFile*(file); *// save to the private variable* final Intent captureIntent = new Intent(android.provider.MediaStore.*ACTION\_IMAGE\_CAPTURE*);  
 captureIntent.putExtra(MediaStore.*EXTRA\_OUTPUT*, mCapturedImageURI);  
 *// captureIntent.putExtra(MediaStore.EXTRA\_SCREEN\_ORIENTATION, ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT);* Intent i = new Intent(Intent.*ACTION\_GET\_CONTENT*);  
 i.addCategory(Intent.*CATEGORY\_OPENABLE*);  
 i.setType("image/\*");  
  
 Intent chooserIntent = Intent.*createChooser*(i, getString(R.string.*image\_chooser*));  
 chooserIntent.putExtra(Intent.*EXTRA\_INITIAL\_INTENTS*, new Parcelable[]{captureIntent});  
  
 startActivityForResult(chooserIntent, *FILECHOOSER\_RESULTCODE*);  
 } catch (Exception e) {  
 Toast.*makeText*(getBaseContext(), "Camera Exception:" + e, Toast.*LENGTH\_LONG*).show();  
 }  
  
 }  
  
 *// openFileChooser for Android < 3.0* public void openFileChooser(ValueCallback<Uri> uploadMsg) {  
 openFileChooser(uploadMsg, "");  
 }  
  
 *// openFileChooser for other Android versions  
 /\* may not work on KitKat due to lack of implementation of openFileChooser() or onShowFileChooser()  
 https://code.google.com/p/android/issues/detail?id=62220  
 however newer versions of KitKat fixed it on some devices \*/* public void openFileChooser(ValueCallback<Uri> uploadMsg, String acceptType, String capture) {  
 openFileChooser(uploadMsg, acceptType);  
 }  
  
  
 });  
 webView.loadUrl("file:///android\_asset/index.html");  
  
  
 *//external meadia permission* if(Build.VERSION.*SDK\_INT*>= Build.VERSION\_CODES.*M*){  
 if(checkSelfPermission(Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*) == PackageManager.*PERMISSION\_DENIED*){  
  
 Log.*d*("permission","permission denied to WRITE\_EXTERNAL\_STORAGE - requesting it");  
 String[] permissions = {Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*};  
 requestPermissions(permissions,1);  
 }  
  
 }  
 *//handle downloading* webView.setDownloadListener(new DownloadListener() {  
 @Override  
 public void onDownloadStart(String url, String userAgent, String contentDisposition, String mimeType, long contentLength) {  
  
 DownloadManager.Request request = new DownloadManager.Request(Uri.*parse*(url));  
 request.setMimeType(mimeType);  
 String cookies = CookieManager.*getInstance*().getCookie(url);  
 request.addRequestHeader("cookie",cookies);  
 request.addRequestHeader("User-Agent",userAgent);  
 request.setDescription("Downloading file....");  
 request.setTitle(URLUtil.*guessFileName*(url,contentDisposition,mimeType));  
 request.allowScanningByMediaScanner();  
 request.setNotificationVisibility(DownloadManager.Request.*VISIBILITY\_VISIBLE\_NOTIFY\_COMPLETED*);  
 request.setDestinationInExternalPublicDir(Environment.*DIRECTORY\_DOWNLOADS*,URLUtil.*guessFileName*(url, contentDisposition, mimeType));  
 DownloadManager dm = (DownloadManager) getSystemService(*DOWNLOAD\_SERVICE*);  
 dm.enqueue(request);  
 Toast.*makeText*(getApplicationContext(),"Downloading File",Toast.*LENGTH\_SHORT*).show();  
  
  
 }  
 });  
  
  
  
 }  
  
  
  
  
 *// return here when file selected from camera or from SD Card* @Override  
 public void onActivityResult(int requestCode, int resultCode, Intent data) {  
  
 *// code for all versions except of Lollipop* if (Build.VERSION.*SDK\_INT* < Build.VERSION\_CODES.*LOLLIPOP*) {  
  
 if (requestCode == *FILECHOOSER\_RESULTCODE*) {  
 if (null == this.mUploadMessage) {  
 return;  
 }  
  
 Uri result = null;  
  
 try {  
 if (resultCode != *RESULT\_OK*) {  
 result = null;  
 } else {  
 *// retrieve from the private variable if the intent is null* result = data == null ? mCapturedImageURI : data.getData();  
 }  
 } catch (Exception e) {  
 Toast.*makeText*(getApplicationContext(), "activity :" + e, Toast.*LENGTH\_LONG*).show();  
 }  
  
 mUploadMessage.onReceiveValue(result);  
 mUploadMessage = null;  
 }  
  
 } *// end of code for all versions except of Lollipop  
  
 // start of code for Lollipop only* if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*LOLLIPOP*) {  
  
 if (requestCode != *FILECHOOSER\_RESULTCODE* || mFilePathCallback == null) {  
 super.onActivityResult(requestCode, resultCode, data);  
 return;  
 }  
  
 Uri[] results = null;  
  
 *// check that the response is a good one* if (resultCode == Activity.*RESULT\_OK*) {  
 if (data == null || data.getData() == null) {  
 *// if there is not data, then we may have taken a photo* if (mCameraPhotoPath != null) {  
 results = new Uri[]{Uri.*parse*(mCameraPhotoPath)};  
 }  
 } else {  
 String dataString = data.getDataString();  
 if (dataString != null) {  
 results = new Uri[]{Uri.*parse*(dataString)};  
 }  
 }  
 }  
  
 mFilePathCallback.onReceiveValue(results);  
 mFilePathCallback = null;  
  
 } *// end of code for Lollipop only* }  
  
  
 @Override  
 public void onBackPressed() {  
 if(webView!= null && webView.canGoBack()){  
 webView.goBack();*// if there is previous page open it* }else{  
 *//super.onBackPressed();//if there is no previous page, close app* }  
 }  
}

**12. Screenshots**

**13. Aim Of Project**

**14. Conclusion**

**15. Bibliography**