**Test Player**

**A PROJECT REPORT Submitted**

**in partial fulfillment of the Requirements**

**for the Degree of**

**MASTER OF COMPUTER APPLICATION**

by

**Manthan Chauhan**

**University Roll. No.**

**1900290149058**

**Submitted to**

**Ms. Neelam Rawat**

**(Associate Professor)**

**KIET Group of Institutions, Ghaziabad**



to the

**Department of Computer Applications,**

**Dr A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY**

**LUCKNOW**

**(Formerly Uttar Pradesh Technical University, Lucknow)**

**(July 2021)**

**DECLARATION**

I hereby declare that the work presented in this report entitled “Test Player”, was carried out by me. I have not submitted the matter embodied in this report for the award of any other degree or diploma of any other University or Institute.

I have given due credit to the original authors/sources for all the words, ideas, diagrams, graphics, computer programs, experiments, results, that are not my original contribution. I have used quotation marks to identify verbatim sentences and given credit to the original authors/sources.

I affirm that no portion of my work is plagiarized, and the experiments and results reported in the report are not manipulated. In the event of a complaint of plagiarism and the manipulation of the experiments and results, I shall be fully responsible and answerable.

Name : Manthan Chauhan

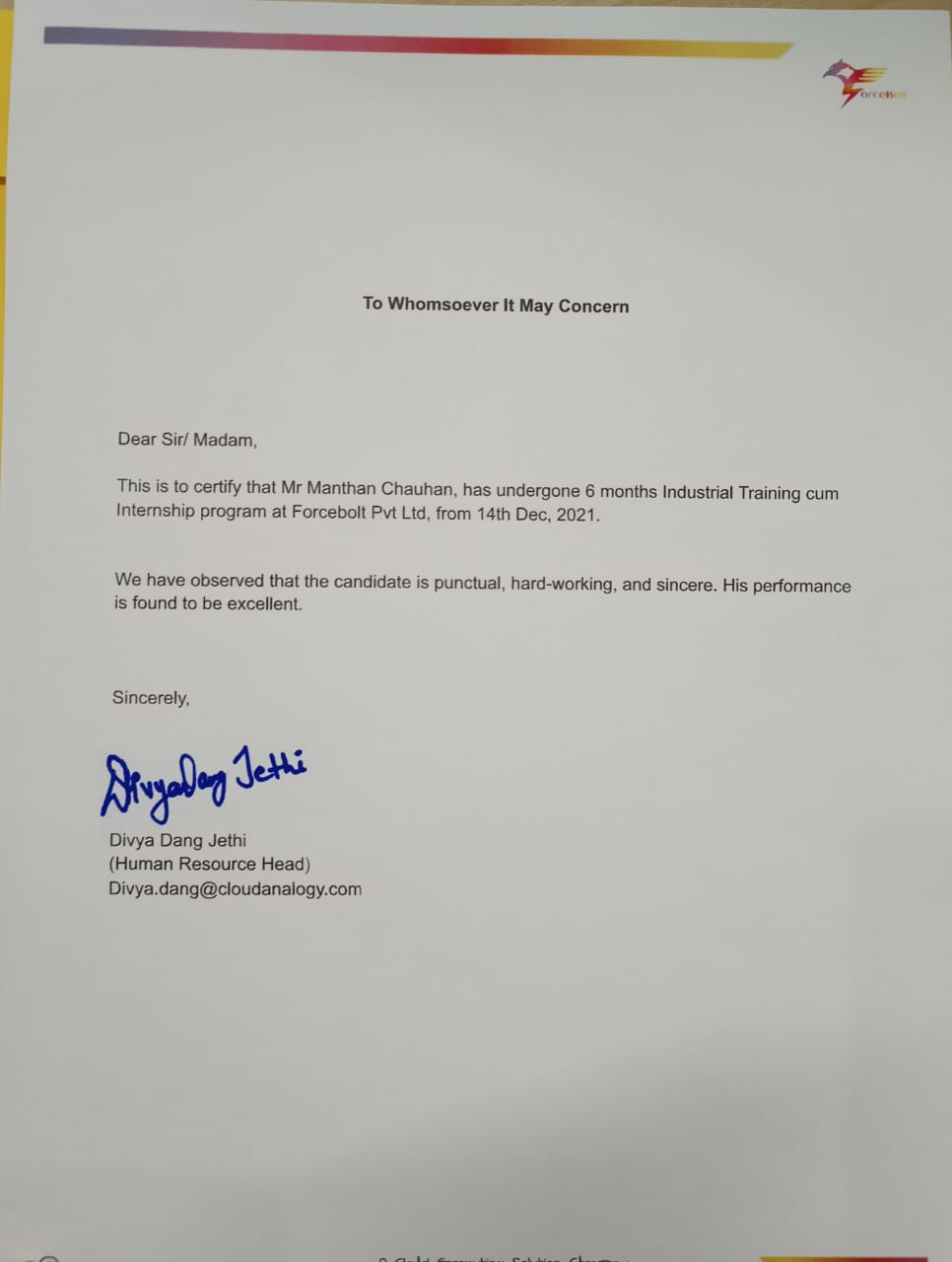
Roll. No. : 1900290149016

Branch : Master of Computer Applications

Manthan Chauhan

**(Candidate Signature)**

**TRAINGING LETTER**

****

**CERTIFICATE**

Certified that **Manthan Chauhan** (**1900290149058**) has carried out the project work presented in this report entitled “**Test Player**” for the award of **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University, Lucknow under my supervision. The report embodies result of original work, and studies are carried out by the student himself and the contents of the report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University.

**Ms. Neelam Rawat External Examiner**

Associate Professor

Dept. of Computer Applications

KIET Group of Institutions, Ghaziabad

**Dr. Ajay Kr Shrivastava**

Professor & Head

Department of Computer Application

KIET Group of Institutions, Ghaziabad

Date

**ACKNOWLEDGEMENT**

Success in life is never attained single handedly. My deepest gratitude goes to my thesis supervisor, **Ms. Neelam Rawat** for her guidance, help and encouragement throughout my research work. Their enlightening ideas, comments, and suggestions.

Words are not enough to express my gratitude to Dr. Ajay Kumar Shrivastava, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help at various occasions.

Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions.

Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

**Manthan Chauhan**

**1900290149058**

**Table of Contents**

|  |  |
| --- | --- |
| Declaration | ii |
| Certificate | iii |
| Acknowledgements | iv |
| Table of Contents | v |
|  |  |
|  |  |

**Chapter 1: INTRODUCTION**  1

1.1 Test Player 1

1.2 Test Player Description 1

1.3 Test Player Scope 1

1.4 Advantages of Test Player 1

1.5 Disadvantages of Test Player 1

**Chapter 2: LITERATURE REVIEW 2**

**Chapter 3: TECHNICAL FEASIBILITY**

3.1 Feasibility Study 3

3.2 Technical Feasibility 3

3.3 Technology Description 3

3.4 Technology used in the Test Player 3

3.4.1 React JS 3

3.4.1.1 React JS Library 3

3.4.1.2 React JS Hooks 5

3.4.1.3 Reusable Components 6

3.4.2 HTML 6

3.4.3 CSS 7

3.4.4 Redux 7

3.4.5 Firebase 8

3.4.6 Cloud Messaging 9

**Chapter 4: FRONTEND DESIGN**

4.1 Instruction Screen 10

4.2 Test Panel Screen 11

4.3 Leave Popup 12

4.4 Remaining Time Alert 12

4.5 Test Summary Screen 13

4.6 Loading Screen 13

4.7 Result Screen 14

4.8 Failure Screen 14

**Chapter 5: CODING**

4.1 Instruction Page 15-23

4.2 Test Panel Page 23-33

4.3 Leave Popup Page 33-34

4.4 Remaining Time Alert Page 34

4.5 Test Summary Page 34-37

4.6 Loading Page 37-38

4.7 Result Page 38-43

4.8 Failure Page 43-44

**Chapter 7: LIMITATIONS**

7.1 Limitations 44-45

7.2 Future Scope 45

7.3 Future Enhancement 45

**Chapter 11: CONCLUSION** 46

**Chapter 12: REFERENCE** 47

**CHAPTER 1**

**INTRODUCTION**

**1.1 Test Player**

Test player is an on-line test simulator is to take online examination, test in an efficient manner and no time wasting for manually checking of the test paper. The main objective of this web based online examination system is to efficiently evaluate the student thoroughly through a fully automated system that not only saves lot of time but also gives fast and accurate results. For students they give papers according to their convenience from any location by using internet and time and there is no need of using extra thing like paper, pen etc.  
  
**1.1 Test Player Description**

Test player helps students to offer a quick and easy way to appear for the test. It also provides the results immediately after the examination with 100% accuracy and security. Student can enter to perform exam only with their valid username and password. This examination contains multiple choice questions and appropriate number of options. There are no limitations on number of options and it can be randomized so same set of question will not appear to all student so it prevent manipulation. More than one option can be correct but the user can select only one option. This provides time limit. The user can see their results after completing the exam. This helps the students to write the exam from far distance and which can provide security and simplicity and other beneficial features to the user.

**1.3 Test Player Scope**

Online Examination System is widely used as compared to other exams. Online examination system can be used in private institutes as well as educational institution. As it is user friendly web base application it can be used anywhere and anytime. Every software may have some cases of bugs, errors, security related problems or system faults. There are many problems or system faults for example; computer collapse or crashes due to power supply problem will invalidate efforts of number of students. There are large numbers of chances in which software may produce wrong results or may display invalid data. These bugs must be identified and solved for improving quality of software. So in future we can develop more secure software by using advanced technologies.

**1.4 Advantages Of Test Player**

Online Examination System is a computerized system which gives instant results and also saves time. It fully automates the previous manual process of taking written exams. It is implemented by web based online examination software or through Intranet variance. It decreases the need of supervision during the exam is being examine or taken using web based Online Examination System gives a high level of clarity as opposite of traditional method. Most of Online Examination System gives the result and instantly. In high school, Online Examination. System is able to reduce the workload of teachers by using automated test paper exams and marking schemes. Students can study independently for example at home or any place. The amount of time given at a particular question gives you the ability of “Quick Learning Quick Thinking“. The data in Online Examination System is regenerated repeatedly so that students have access to new data.

**1.5. Disadvantages Of Test Player**

The question and answers may be tricky and confusing as there are multiple choices for answers. Lack of interaction between the teacher and student. Network problems may arise in Online Examination System, test may be postponed or cancel. As India is not a fully developed country problem may arise in rural areas where there is deficiency of computers or where computers are not yet discovered. Hackers can easily manage to achieve confidential data through fake online application or websites. While attempting online exams we can easily manage to open another window on existing computer. Teaching techniques of an instructor or teacher may vary from online exam questions.

**CHAPTER 2**

**LITERATURE REVIEW**

The entire process of allocating test and analyzing their scores after the test was operated manually for example examining test and distributing score which used to take a lot of time when online examination was not in existence. To take written exam of number of students we need more invigilators to watch examination candidates for the purpose of reducing or preventing cheating. It is developed for reducing large number of relevant resources. Online Examination System is developed base on web and network. It recognizes the concept of design system that characterizes the main purpose of the system, examining the algorithm of creating question paper and also about the security of the system. The method of online examination was for important activities like to evaluate performance of student in existing institutes. The exam question quality of online examination will regulate or decide the quality of students growing in the institutes. Present technologies as familiar with advanced techniques like adding question bank in database. It also helps to originate or generate the different questions in sets without repeating and reducing the duplication of questions. Online Examination System characterize the use of shuffling algorithm is a Generator Question System (GQS) as one of the techniques used to overcome randomization issues of collecting sets of exam paper. **CHAPTER 3**

**TECHNICAL FEASIBILITY**

**3.1 Feasibility Study**

The feasibility study of any system is mainly intended to study and analyze the proposed system and to decide whether the system under consideration will be viable or not after implementation. That is it determines the usability of the project after deployment. To come to result a set of query is answered keeping the efficiency of the software and its impact on the domain for which it was developed. It main emphasis is on the following three questions elucidated below as:

What are the user’s requirements and how does a candidate system meet them?

What resources are available for the proposed systems? Is it worth solving the problem?

What is the likely impact of the proposed system on the organization? I.e. how does the proposed system fit with in the organization?

Thus since the feasibility study may lead to commitment of large resources, it becomes necessary that it should be conducted competently and no fundamental errors of judgment are made. Different types of feasibility study and the way we performed on our project “On Line Examination**”** .

**3.2 Technical Feasibility**

In technical feasibility, we study all technical issues regarding the proposed system. It is mainly concerned with the specifications of the equipments and the software, which successfully satisfies the end-user’s requirement. The technical needs of the system may vary accordingly but include:

* The feasibility to produce outputs in a given time.
* Response time under certain conditions.
* Ability to process a certain volume of the transaction at a particular speed.
* Facility to communicate data.

Under this analysis process questions like (i) does the compatible platform exist within our domain or can we procure it? (ii) Does the proposed equipment have the technical capacity to hold the data required using the new system?.

Both at the development site and at server where we will be hiring the space for the website, and also the database would it be possible to upgrade the system after it is developed and implemented, if necessary? And would the recommended technology guarantee the reliability, accuracy and data security? This analysis process requires more emphasis on system configuration given more importance rather than the actual hardware specifications.

. The configuration of the existing systems is:

* Processor   : Pentium III, 500 MHz (or above)
* Memory    : 128 MB (or above)
* Secondary storage  : 20 GB (or above)

**3.3 Technology Description**

HTML, CSS, React JS, Redux, Firebase, Cloud Messaging.

**3.4 Technology used in the Test Player**

**3.4.1 React JS**

**3.4.1.1 React JS Library**

**React** (also known as **React.js** or **ReactJS**) is a free and 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.

**3.4.1.2 React JS Hooks**

Hooks are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class. Hooks are the functions which "hook into" React state and lifecycle features from function components. It does not work inside classes.

Hooks are backward-compatible,

which means it does not contain any breaking changes. Also, it does not replace your knowledge of React concepts.

## When to use a Hooks

If you write a function component, and then you want to add some state to it, previously you do this by converting it to a class. But now you can do it by using a Hook inside the existing function component.

Basic Hooks

* useState
* useEffect
* useContext

Additional Hooks

* useReducer
* useCallback
* useMemo
* useRef
* useImperativeHandle
* useLayoutEffect
* useDebugValue

**3.4.1.3 Reusable Components**

In React, a reusable component is a piece of UI that can be used in various parts of an application to build more than one UI instance. For instance, we can have a button component display with different colors in several parts of our application. Although it is the same button component when we provide it with a dataset (e.g color, or a function), it modifies itself and outputs a UI instance of the element.

This pattern of creating React components is necessary for scaling. It helps save time by ensuring less code is written, development is faster, the codebase is simpler, and maintenance is stress-free.

In this tutorial, we will build reusable React components that can be used throughout your projects to maintain consistency in logic and presentation. We’ll use Hooks for managing and manipulating state data.

Reusable components are those React components that can be used multiple times in your application. As a result, they need to be generic enough so that it’s free from complex business logic. If a component contains any complex logic inside it, not only does it become difficult to reuse, it also becomes less maintainable. React hooks are the perfect fit for reusable component logic.

**3.4.2 HTML**

The HyperText Markup Language, or HTML is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading style sheets (CSS) and scripting languages such as JavaScript.

Web browser 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.

HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by *tags*, written using angle brackets. Tags such as <**img** /> and <**input** /> directly introduce content into the page. Other tags such as <**p**> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997.

**3.4.3 CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language such as 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, colors, 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 which reduces complexity and repetition in the structural content as well as enabling the .css file to be cached to improve the page load speed between the pages that share the file and its formatting.

Separation of formatting and content also makes it feasible to present the same markup page in different styles for different rendering methods, such as on-screen, in print, by voice (via speech-based browser or screen reader), and on Braille-based tactile devices. CSS also has rules for alternate formatting if the content is accessed on a mobile device.

The name *cascading* comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

The CSS specifications are maintained by the World Wide Web Consortium (W3C). Internet media type (MIME type) text/css is registered for use with CSS by RFC 2318 (March 1998). The W3C operates a free CSS validation service for CSS documents.

In addition to HTML, other markup languages support the use of CSS including XHTML, plain XML, SVG, and XUL.

**3.4.4 Redux**

Redux is an open-source JavaScript library for managing application state. It is most commonly used with libraries such as React or Angular for building user interfaces. Similar to (and inspired by) Facebook's Flux architecture, it was created by Dan Abramov and Andrew ClarkSanity.

**History:**

Redux was created by Dan Abramov and Andrew Clark in 2015. Abramov began writing the first Redux implementation while preparing for a conference talkat React Europeon hot reloading. Abramov remarks, "I was trying to make a proof of concept of Flux where I could change the logic. And it would let me time travel. And it would let me reapply the future actions on the code change."

Abramov was struck by the similarity of the Flux pattern with a reducing function. "I was thinking about Flux as a reduce operation over time... your stores, they accumulate state in response to these actions. I was thinking of taking this further. What if your Flux store was not a store but a reducer function?"

Abramov reached out to Andrew Clark (author of the Flux implementation Flummox) as a collaborator. Among other things, he credits Clark with making the Redux ecosystem of tools possible, helping to come up with a coherent API, implementing extension points such as middleware and store enhancers.

In February 2019, useReducer was introduced as a React hook in the 16.8 release. It provides an API that is consistent with Redux, enabling developers to create Redux-like stores that are local to component states.

**3.4.5 Firebase**

Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011.In 2014, Google acquired the platform and it is now their flagship offering for app development.

**History:**

Firebase evolved from Envolve, a prior startup founded by James Tamplin and Andrew Lee in 2011. Envolve provided developers an API that enables the integration of online chat functionality into their websites. After releasing the chat service, Tamplin and Lee found that it was being used to pass application data that were not chat messages. Developers were using Envolve to sync application data such as game state in real time across their users. Tamplin and Lee decided to separate the chat system and the real-time architecture that powered it. They founded Firebase as a separate company in September 2011and it launched to the public in April 2012.

Firebase's first product was the Firebase Realtime Database, an API that synchronizes application data across iOS, Android, and Web devices, and stores it on Firebase's cloud. The product assists software developers in building real-time, collaborative applications.

In May 2012, a month after the beta launch, Firebase raised $1.1 million in seed funding from venture capitalists Flybridge Capital Partners, Greylock Partners, Founder Collective, and New Enterprise Associates. In June 2013, the company further raised $5.6 million in Series A funding from Union Square Ventures and Flybridge Capital Partners.

In 2014, Firebase launched two products. Firebase Hosting and Firebase Authentication. This positioned the company as a mobile backend as a service.

In October 2014, Firebase was acquired by Google. A year later, in October 2015, Google acquired Divshot, an HTML5 web-hosting platform, to merge it with the Firebase team.

In May 2016, at Google I/O, the company's annual developer conference, Firebase introduced Firebase Analytics and announced that it was expanding its services to become a unified backend-as-a-service (BaaS) platform for mobile developers. Firebase now integrates with various other Google services, including Google Cloud Platform, AdMob, and Google Ads to offer broader products and scale for developers.Google Cloud Messaging, the Google service to send push notifications to Android devices, was superseded by a Firebase product, Firebase Cloud Messaging, which added the functionality to deliver push notifications to both iOS and web devices. In January 2017, Google acquired Fabric and Crashlytics from Twitter to add those services to Firebase.

In October 2017, Firebase has launched Cloud Firestore, a real-time document database as the successor product to the original Firebase Realtime Database.

**3.4.6 Cloud Messaging**

Firebase Cloud Messaging (FCM), formerly known as Google Cloud Messaging (GCM), is a cross-platform cloud solution for messages and notifications for Android, iOS, and web applications, which as of 2021 can be used at no cost. Firebase Cloud Messaging allows third-party application developers to send notifications or messages from servers hosted by FCM to users of the platform or end users.

The service is provided by Firebase, a subsidiary of Google. On October 21, 2014, Firebase announced it had been acquired by Google for an undisclosed amount. The official Google Cloud Messaging website points to Firebase Cloud Messaging (FCM) as the new version of GCM. Firebase is a mobile platform which supports users in developing mobile and web applications. Firebase Cloud Messaging is one of many products which are part of the Firebase platform. On the platform users can integrate and combine different Firebase features in both web and mobile applications.

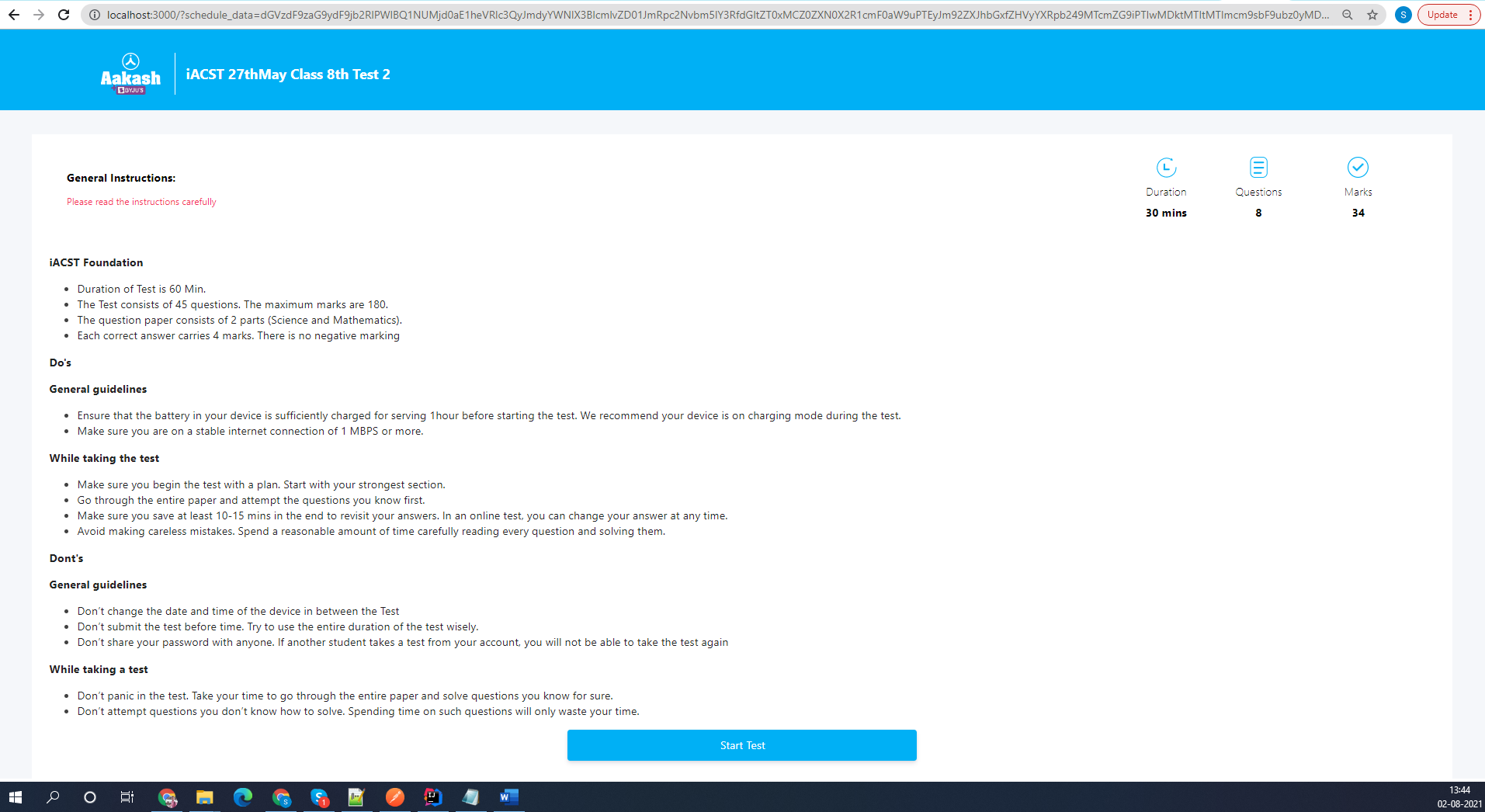
### **Implementation**

The implementation process has two key components. First, a secure environment to send and receive messages is required for FCM or other application servers to facilitate message transaction. Second, a client application of possible types, iOS, Android or web (javaScript), which is also compatible with the selected platform service is needed.

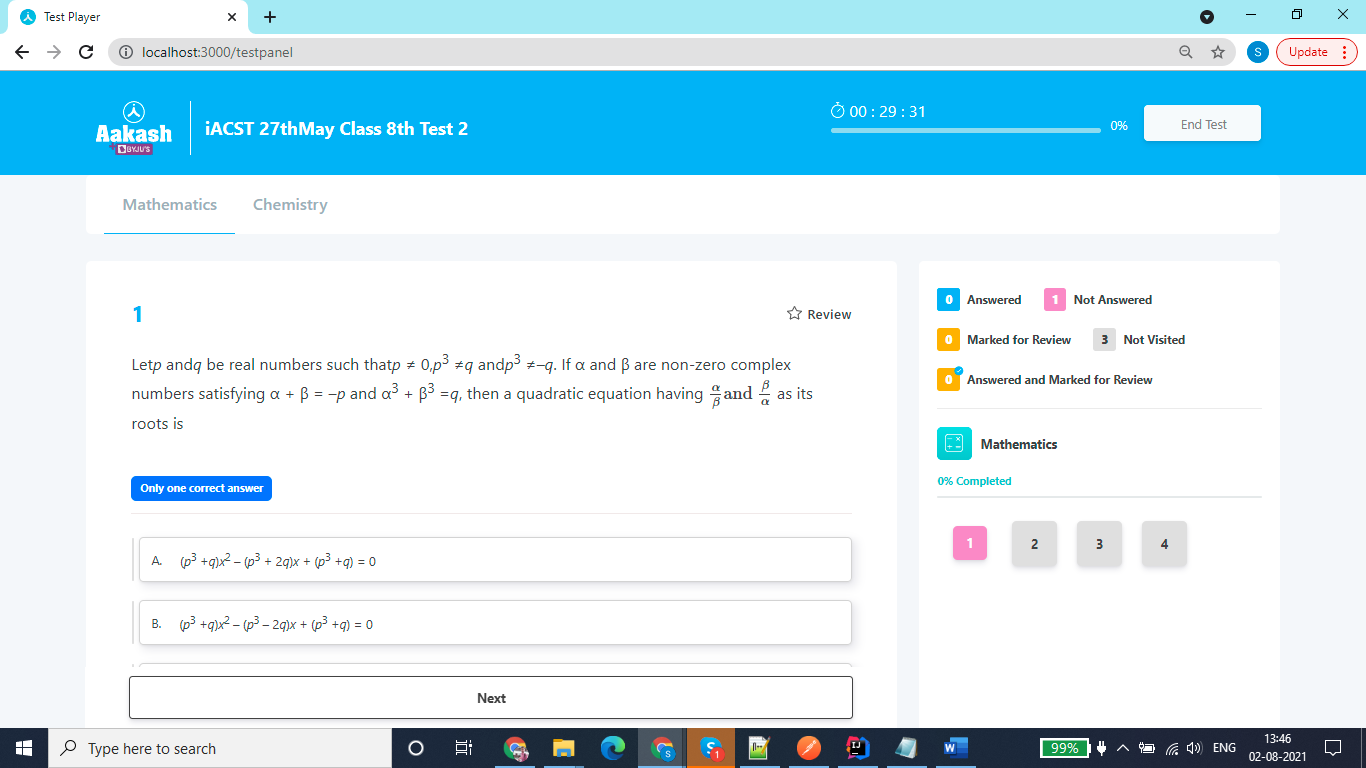
The implementation path for FCM is initiated with the FCM SDK setup following the instructions prescribed for the decided platform. Following setup, the client application must be developed. On the client app, add message handling, topic subscription logic and other required features. During this step, test messages can also be sent from the Notifications composer. The application server is developed next to build the sending logic. The base server environment is created without code.

**CHAPTER 4**

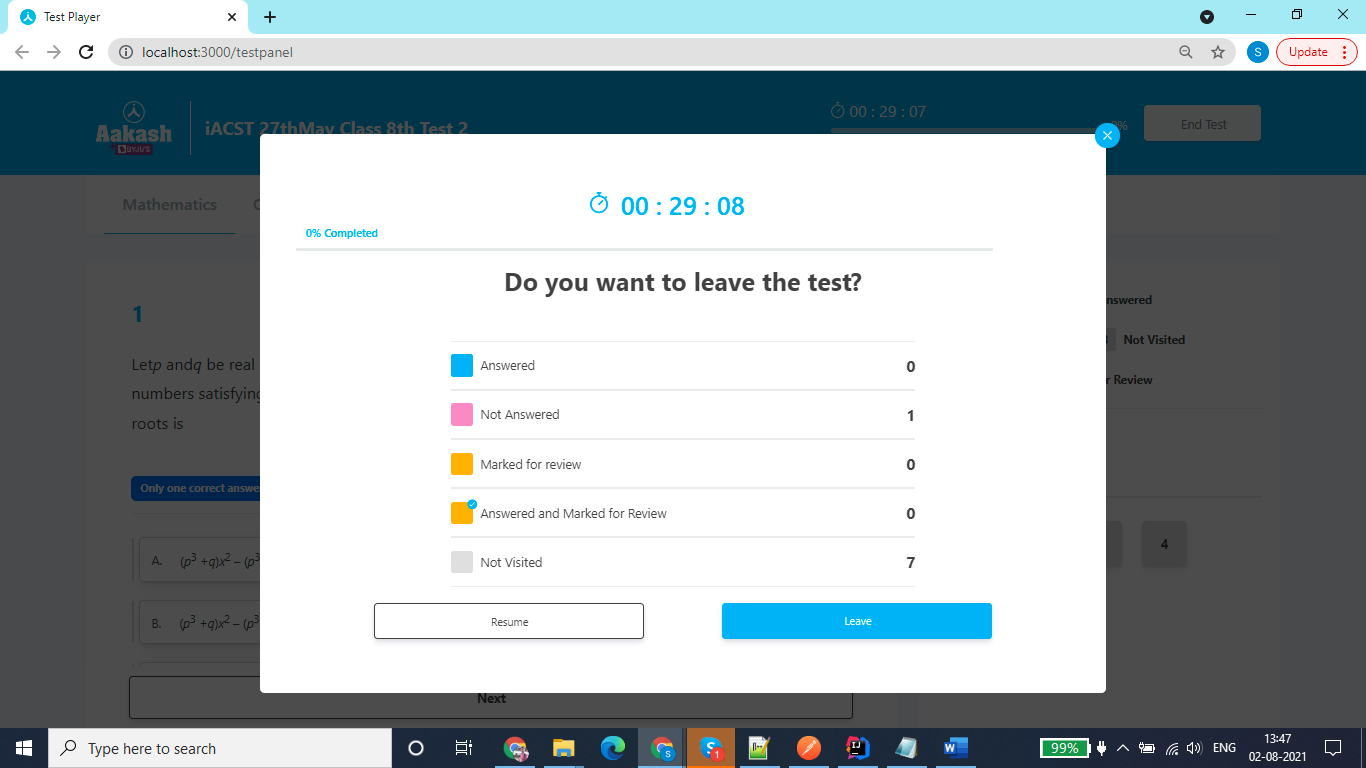
**FRONTEND DESIGN**

**4.1 Instruction Screen**

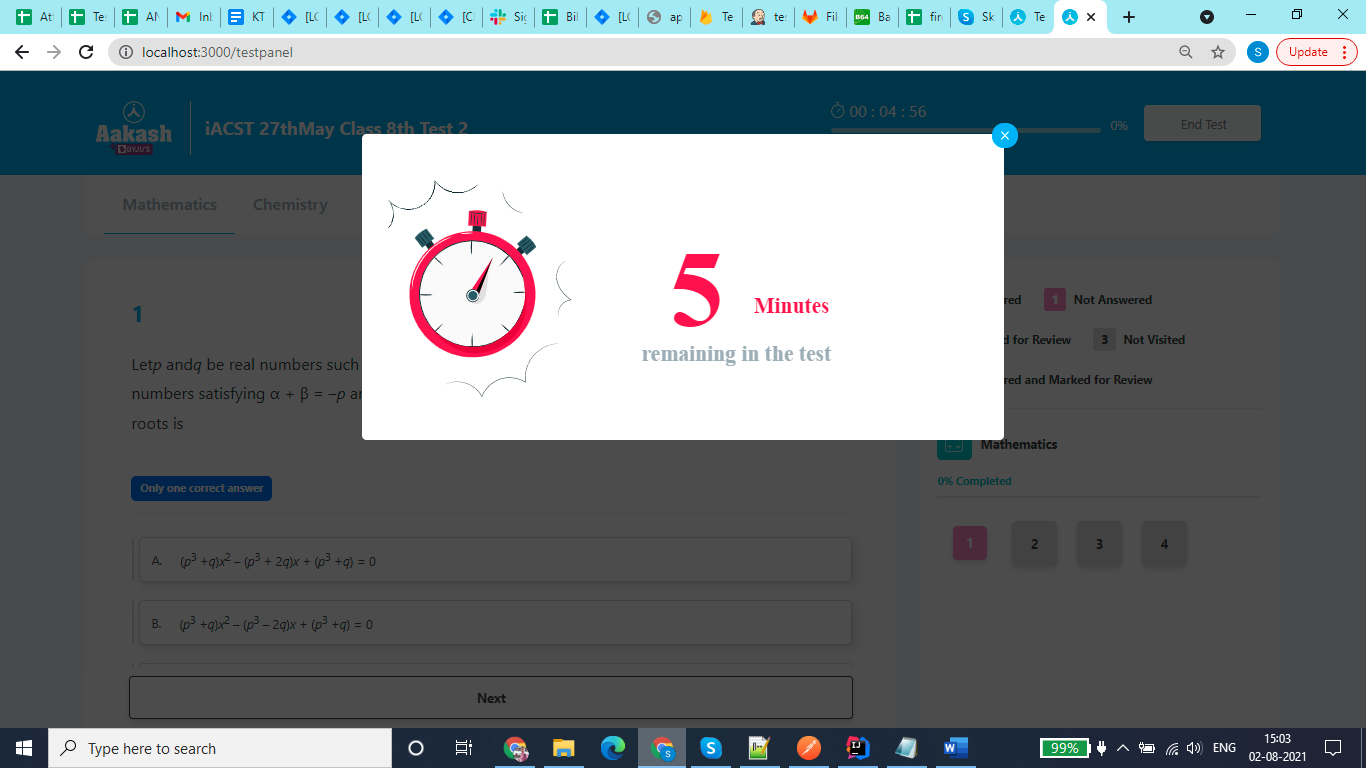
**4.2 Test Panel Screen**

****

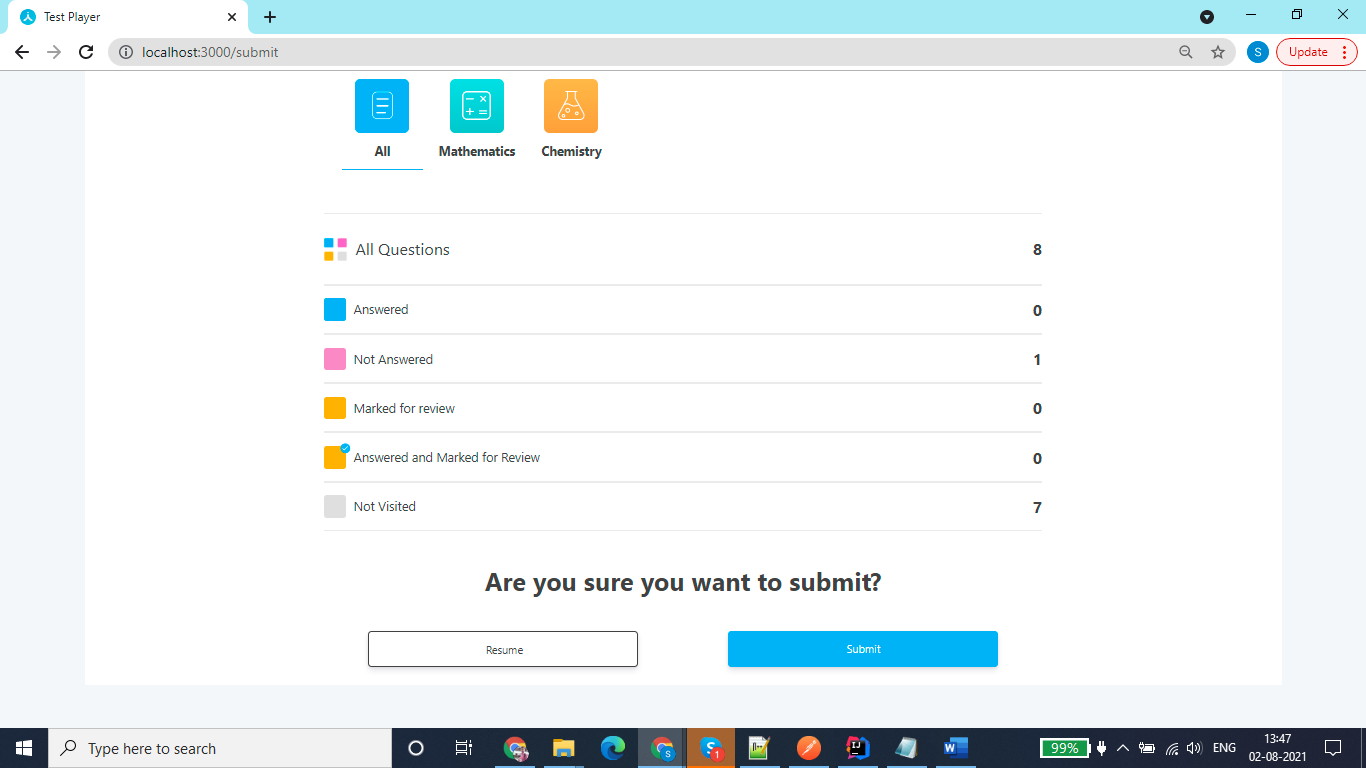
**4.3 Leave Popup**

****

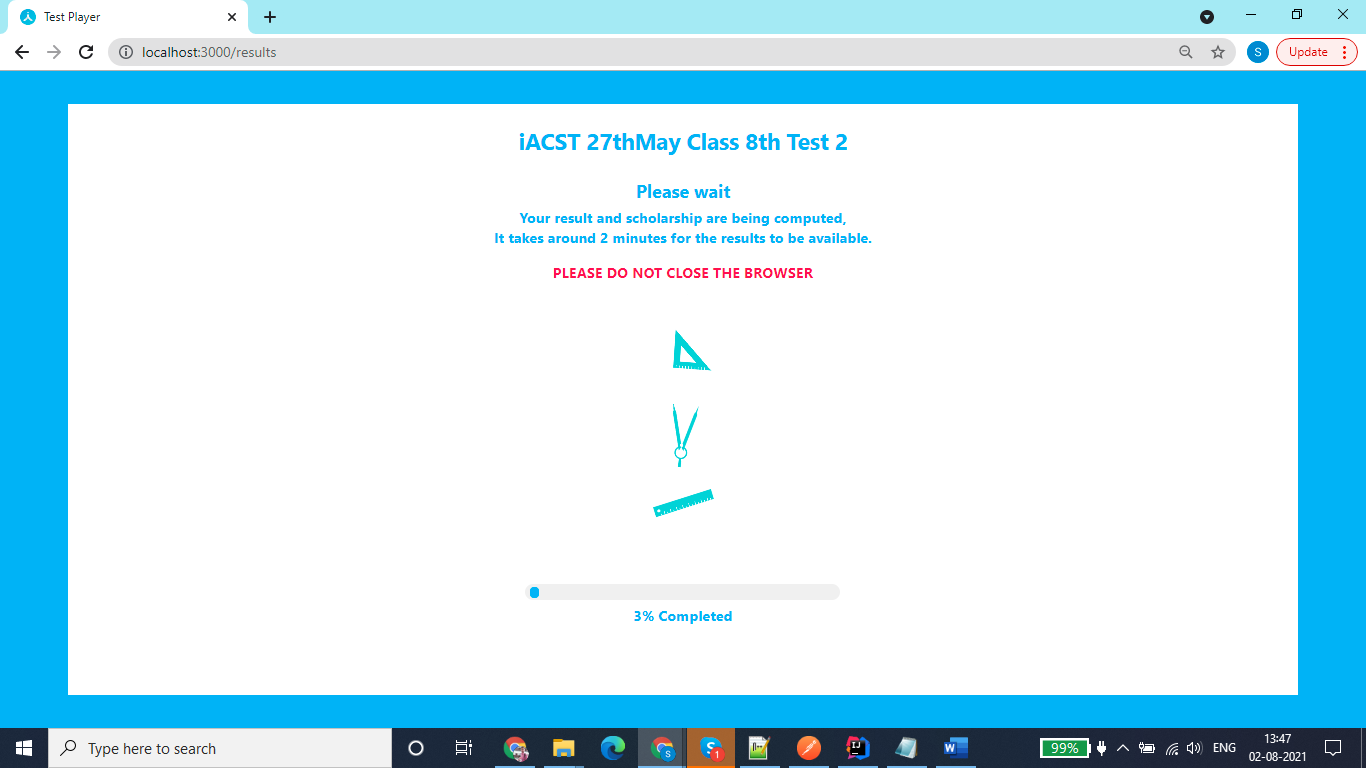
**4.4 Remaining Time Alert**



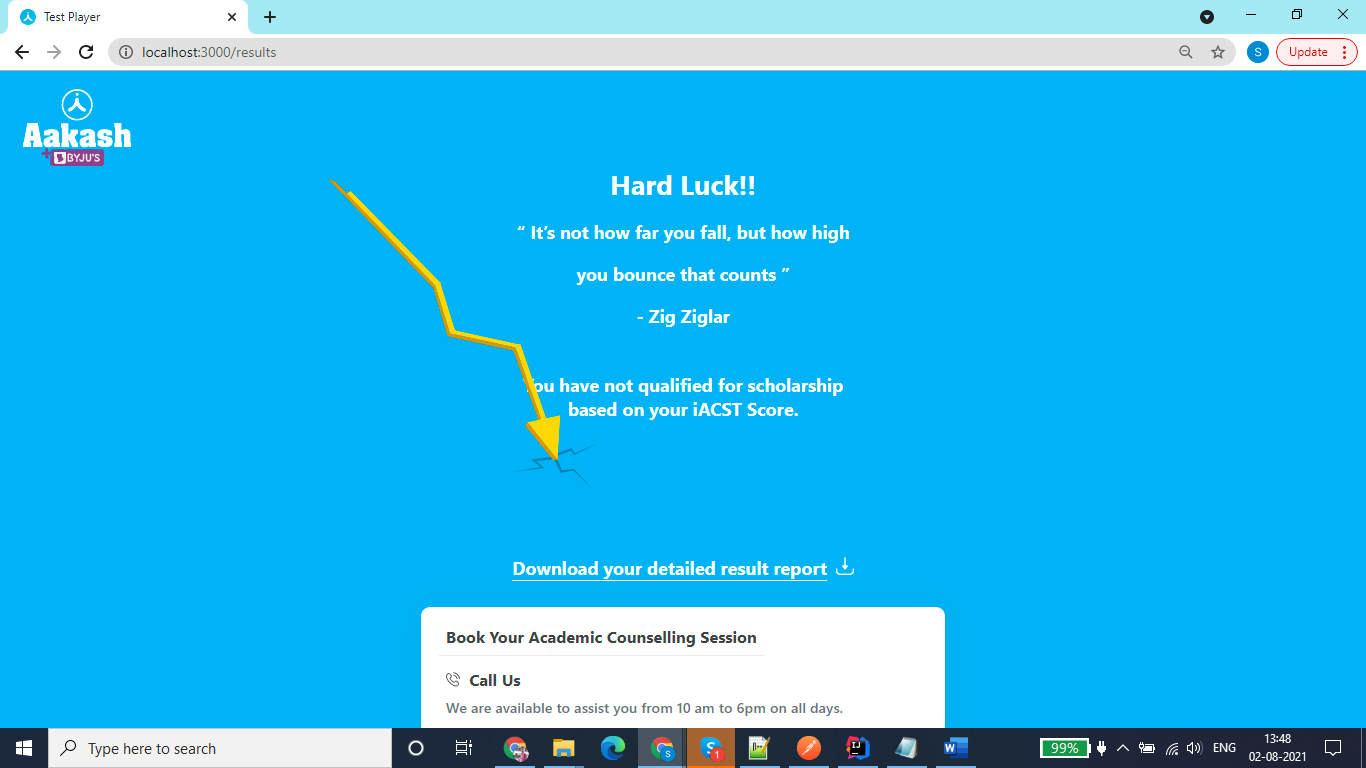
**4.5 Test Summary Screen**



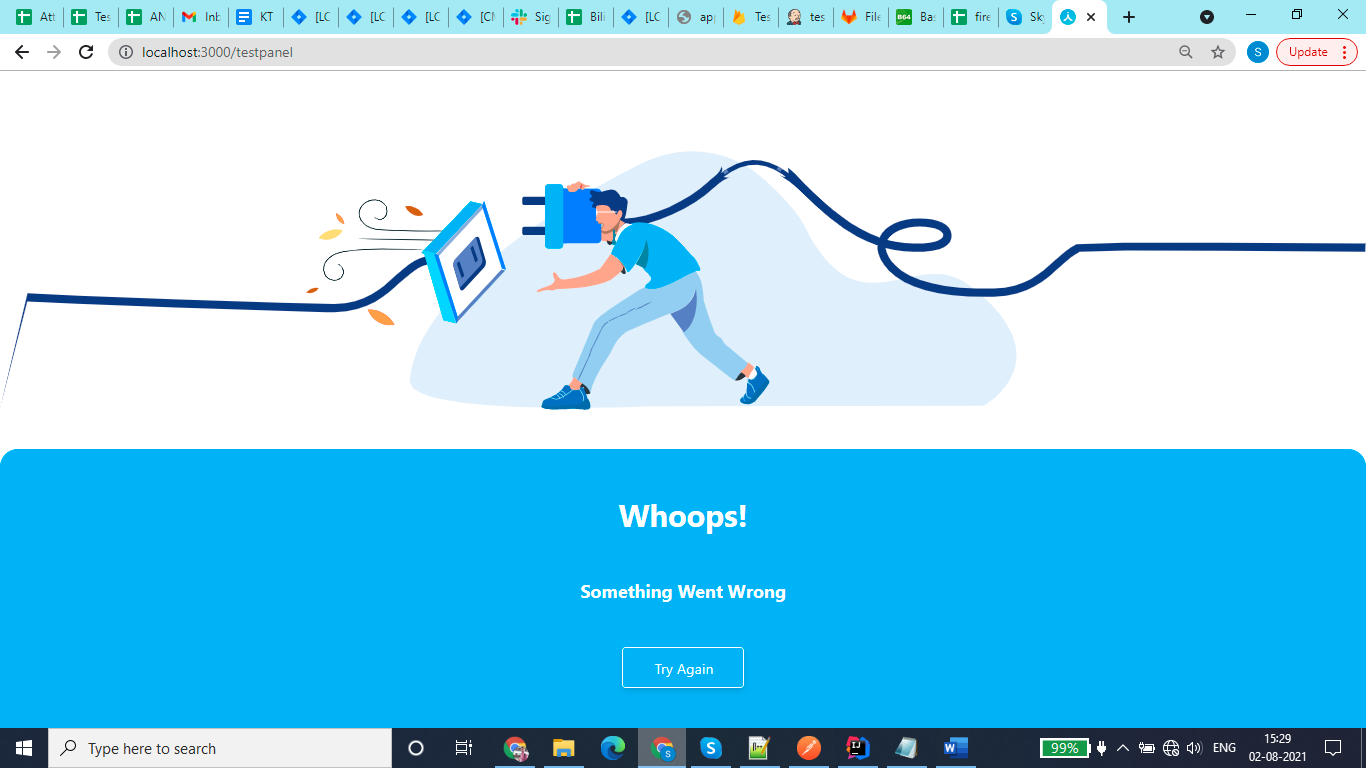
**4.6 Loading Screen**

****

**4.7 Result Screen**

****

**4.8 Failure Screen**

****

**Chapter 5**

**Coding**

**5.1 Instruction page**

import ***React***, { useEffect, useState } from "react";  
import { Redirect } from "react-router-dom";  
import { withRouter } from 'react-router-dom';  
import './InstructionsPage.css';  
import Duration from '../../components/Duration/Duration';  
import NoOfQuestions from '../../components/NoOfQuestions/NoOfQuestions';  
import Marks from '../../components/Marks/Marks';  
import InstructionList from "../../components/InstructionList/InstructionList";  
import { ***connect*** } from 'react-redux';  
import {  
 ***INSTRUCTION***,  
 ***SET\_TEST\_STATUS***,  
 ***HIDE\_TIMER***,  
 ***UPDATE\_TIME***,  
 ***FETCH\_QUESTION\_SET***,  
 ***IS\_TEST\_DATA\_LOADED***,  
 ***UPDATE\_GLOBAL\_TEST\_STATUS***,  
 ***UPDATE\_GLOBAL\_QUERY\_PARAMS***,  
 ***UPDATE\_EXISTING\_TEST\_STATUS***,  
 ***UPDATE\_DEVICE\_ID***,  
 ***UPDATE\_RESUMED\_DATA***,  
 ***UPDATE\_PACKAGE\_ID***,  
 ***UPDATE\_BASE\_URL***,  
 ***UPDATE\_DISCONNECT\_TIME***,  
 ***UPDATE\_TEST\_DURATION***,  
 ***UPDATE\_CALCULATED\_TIME\_LEFT***,  
} from '../../constants/actionTypes';  
import Button from '../../components/Button/Button';  
import Spinner from '../../components/Spinner/Spinner';  
import ***Api*** from '../../data';  
import { useHistory } from 'react-router-dom';  
import PopUpWrapper from '../../components/PopUpWrapper/PopUpWrapper'  
import DeviceDetector from "device-detector-js";  
import ***axios*** from 'axios'  
import { ***store***,***persistor*** } from '../../redux/store';  
import InternetIssue from '../../components/InternetIssue/InternetIssue';  
import TestStatus from './testStatus';  
  
const mapStateToProps = state => ({  
 testStatus : state.testStatus,  
 ...state  
})  
  
const mapDispatchToProps = dispatch => ({  
 onQuestionLoad: payload => {dispatch({ type: ***FETCH\_QUESTION\_SET***, payload })},  
 onLoad: payload => dispatch({ type: ***INSTRUCTION***, payload }),  
 setTestStatus: payload => {dispatch({ type: ***SET\_TEST\_STATUS***, payload });},  
 // onChange: payload => dispatch({ type: UPDATE\_TIME, payload }),  
 resetTimer: payload => dispatch({ type: ***UPDATE\_TIME***, payload }),  
 setTestDataLoadedStatus: payload => dispatch({ type: ***IS\_TEST\_DATA\_LOADED***, payload }),  
 setGlobalTestStatus: payload => dispatch({type:***UPDATE\_GLOBAL\_TEST\_STATUS***,payload}),  
 setQueryParam: payload => dispatch({type:***UPDATE\_GLOBAL\_QUERY\_PARAMS***,payload}),  
 setIsExistingTestParam: payload => dispatch({type:***UPDATE\_EXISTING\_TEST\_STATUS***,payload}),  
 setDeviceId : payload => dispatch({type:***UPDATE\_DEVICE\_ID***,payload}),  
 setResumedData: payload => dispatch({type:***UPDATE\_RESUMED\_DATA***,payload}),  
 setPackageId: payload => dispatch({type:***UPDATE\_PACKAGE\_ID***,payload}),  
 setBaseUrl: payload => dispatch({type:***UPDATE\_BASE\_URL***,payload}),  
 hideTimer: payload => dispatch({ type: ***HIDE\_TIMER***, payload }),  
 setDisconnectTime : payload => dispatch({type:***UPDATE\_DISCONNECT\_TIME***, payload}),  
 setTestDuration : payload => dispatch({type:***UPDATE\_TEST\_DURATION***,payload}),  
 updateCalculatedTimeLeft : payload => dispatch({type: ***UPDATE\_CALCULATED\_TIME\_LEFT*** , payload}),  
});  
  
  
const InstructionsPage = (props) => {  
  
  
 const [isError,setIsError] = useState(false)  
 const [isRedirect,setisRedirect] = useState(false)  
 const [start\_name,setStart\_Name] = useState('');  
 const [param , setParam] = useState('MAN\_117430050007\_8')  
 const [allTestStatus, setAllTestStatus ] = useState()  
 const [deviceId, setDeviceId] = useState()  
 const [testId, setTestId] = useState()  
 const [errorMsg, setErrorMsg] = useState('')  
 const [isTestCompleted, setIsTestCompleted] = useState(false)  
 const SITE\_KEY = "6Ld1q4EbAAAAANVVdaWsziOPrAFC7ugZm5YVs33w";  
 const [IsSecure, setIsSecure] = useState(true);  
  
 const startTest = () => {  
  
  
  
  
 const deviceDetector = new DeviceDetector();  
 const device = deviceDetector.parse(***navigator***.userAgent);  
 props.setTestDataLoadedStatus(false)  
 props.setGlobalTestStatus(param)  
 if(start\_name==='Resume Test'){  
 props.handleFirebaseEventLogs("tap\_resume\_test\_web","User comes back to the test after being logged out and clicks on the resume test button",  
 {  
 "success" : true,  
 "dob" : props.queryparam.dob,  
 "class" : props.instructions.testData.class,  
 "test\_name" : props.instructions.testData.titleEn,  
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : props.instructions.testData.id,  
 "grace\_period" : props.queryparam.grace\_period,  
 "disconnection\_time" : props.queryparam.disconnection\_time,  
 "duration\_of\_test" : props.instructions.testData.allowedDuration,  
 "total\_questions" : props.instructions.testData.totalQuestion,  
 "max\_marks" : props.instructions.testData.maxMarks  
 })  
  
 //GET call of teststatus  
 props.setIsExistingTestParam(true)  
 let deviceInfo = {  
 "deviceSource": device.device.type,  
 "deviceName": device.os.name,  
 "deviceId": props.deviceId,  
 "deviceToken" : props.deviceId  
 }  
 ***Api***.resumeTest(allTestStatus[param]['userTestId'],deviceInfo)  
 .then(res=>res.data)  
 .then(res => {  
 if(res.message === 'Success') {  
 props.setResumedData(res.data.consumedTime/60)  
 props.setTestStatus(res)  
 props.hideTimer(false)  
 setisRedirect(true)  
 props.setDisconnectTime(props.queryparam.disconnect\_time\*60)  
 }  
 })  
 .catch((error)=>{  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage)  
 setIsError(true)  
 setisRedirect(false)  
 }  
 else{  
 setIsError(true)  
 setisRedirect(false)  
 }  
 })  
 }  
 if(start\_name==='Start Test'){  
 //POST call of teststatus  
 // console.log("----",props.instructions)  
 props.handleFirebaseEventLogs("tap\_start\_test\_web","Usertaps on start test button",{  
 "success" : true,  
 "dob" : props.queryparam.dob,  
 "class" : props.instructions.testData.class,  
 "test\_name" : props.instructions.testData.titleEn,  
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : props.instructions.testData.id,  
 "grace\_period" : props.queryparam.grace\_period,  
 "disconnection\_time" : props.queryparam.disconnection\_time,  
 "duration\_of\_test" : props.instructions.testData.allowedDuration,  
 "total\_questions" : props.instructions.testData.totalQuestion,  
 "max\_marks" : props.instructions.testData.maxMarks  
 })  
  
 let deviceInfo = {  
 "deviceSource": device.device.type,  
 "deviceName": device.os.name,  
 "deviceId": props.deviceId,  
 "deviceToken" : props.deviceId  
 }  
 let data = {  
 "testId":testId,  
 "testShortCode": props.queryparam.test\_short\_code,  
 "testRollNo":props.queryparam.roll\_no,  
 "packageId": props.packageId,  
 "idealTime":props.queryparam.disconnect\_time\*60,  
 "resetCount":props.queryparam.test\_attempt,  
 "totalTimeOfTest": props.queryparam.overall\_duration\*60,  
 "gracePeriod":props.queryparam.grace\_period\*60,  
 "setId":"1",  
 "examNameType":props.queryparam.exam\_name\_type,  
 "testGroup":props.queryparam.test\_group,  
 "testTitle":props.instructions.testData.titleEn  
 // "userId":"1"  
 }  
 ***Api***.startNewTest(data,deviceInfo)  
 .then(res=>res.data)  
 .then(res => {   
 if (res.message == "Success") {  
 props.setResumedData(0)  
 props.setTestStatus(res)  
 props.hideTimer(false)  
 setisRedirect(true)  
 props.setDisconnectTime(props.queryparam.disconnect\_time\*60)  
 }  
 })  
 .catch((error)=>{  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage)  
 setIsError(true)  
 setisRedirect(false)  
 }  
 else{  
 setIsError(true)  
 setisRedirect(false)  
 }  
 })  
 }  
 }  
 const getUrlParameter = (sParam)=> {  
 let sPageURL = ***window***.location.search.substring(1),  
 sURLVariables = sPageURL.split('&'),  
 sParameterName,  
 i;  
   
 for (i = 0; i < sURLVariables.length; i++) {  
 sParameterName = sURLVariables[i].split('=');  
   
 if (sParameterName[0] === sParam) {  
 return typeof sParameterName[1] === undefined ? true : decodeURIComponent(sParameterName[1]);  
 }  
 }  
 return false;  
 }  
  
 const handleOnClick = e => {  
 e.preventDefault();  
 // setLoading(true);  
  
 }  
  
const handleVerifyCaptcha = ()=>{  
 ***window***.grecaptcha.ready(() => {  
 ***window***.grecaptcha.execute(SITE\_KEY, { action: 'submit' }).then(token => {  
 ***Api***.verifyCaptcha(token)  
 .then((response)=>{  
 ***console***.log("ReCaptacha Script loaded!");  
 ***console***.log(***JSON***.stringify(response.data));  
 if(response.data.score <= 0.5){  
 setIsSecure(false)  
 }  
 })  
 .catch((error)=>{  
 ***console***.log(error);  
 });  
 });  
 });  
}  
  
  
 useEffect(() => {  
  
 props.updateCalculatedTimeLeft(0)  
  
//start captcha script  
  
const loadScriptByURL = (id, url, callback) => {  
  
 const isScriptExist = ***document***.getElementById(id);  
   
 if (!isScriptExist) {  
 var script = ***document***.createElement("script");  
 script.type = "text/javascript";  
 script.src = url;  
 script.id = id;  
 script.onload = ()=>{  
 if (callback) callback();  
 };  
 ***document***.body.appendChild(script);  
 }  
   
 if (isScriptExist && callback) callback();  
  
 }  
   
 loadScriptByURL("recaptcha-key", `https://www.google.com/recaptcha/api.js?render=${SITE\_KEY}`, handleVerifyCaptcha);  
  
//end captcha script  
  
  
 props.hideTimer(true)  
 props.onQuestionLoad([])  
 let url\_string = ***window***.location.href;  
 let url = new ***URL***(url\_string);  
 let obj;  
 let hash\_base64 = getUrlParameter('schedule\_data') ? getUrlParameter('schedule\_data') : 'dGVzdF9zaG9ydF9jb2RlPUlBQ1NUJmdyYWNlX3BlcmlvZD0xMjAmZGlzY29ubmVjdF90aW1lPTEmdGVzdF9kdXJhdGlvbj0xMjAmb3ZlcmFsbF9kdXJhdGlvbj0yNDAmZG9iPTIwMDktMTItMTImcm9sbF9ubz0yMDEyMTEwNTAxMDYmdGVzdF9hdHRlbXB0PTc1Njc4OSZ0ZXN0X3N0YXJ0X2RhdGU9MjAyMS0wNC0wOSZleGFtX25hbWVfdHlwZT1PVEhFUiZ0ZXN0X2dyb3VwPQ==';  
 let instruction\_url = `${***window***.location.protocol}//${***window***.location.host}/?schedule\_data=${getUrlParameter('schedule\_data')}`;  
 props.setBaseUrl(instruction\_url)  
 if(hash\_base64!==undefined && hash\_base64!=null && hash\_base64!=''){  
 obj = atob(hash\_base64)  
 }  
 let searchParams = new ***URLSearchParams***(obj);  
  
 let obj1 = {}  
 for (let p of searchParams) {  
 let p1 = p[0]   
 let p2 = p[1]  
 obj1[p1] = p2  
 }  
 ***localStorage***.setItem("roll\_no", ***JSON***.stringify(obj1.roll\_no));  
 props.setQueryParam(obj1)  
 //API call to get the auth token  
 ***Api***.lcmsAuthentication(obj1.roll\_no,obj1.dob)  
 .then((res)=> {  
 ***Api***.setAuthToken(res.data.auth\_token)  
 ***Api***.getInstructionDataByShortCode(obj1.test\_short\_code)  
 .then(res => res.data)  
 .then(res => {  
 if( res.data.hasOwnProperty('instruction') &&  
 res.data.hasOwnProperty('title') &&  
 res.data.hasOwnProperty('titleEn') &&   
 res.data.hasOwnProperty('totalQuestion') &&   
 res.data.hasOwnProperty('syllabus') &&   
 res.data.hasOwnProperty('maxMarks') &&  
 res.data.hasOwnProperty('packageId')   
 )  
 {  
 // props.onChange((res.data.allowedDuration)\*60)  
 props.setTestDuration((res.data.allowedDuration))  
 props.setPackageId(res.data.packageId)  
 let param = obj1.test\_short\_code + '\_' + obj1.roll\_no + '\_' + obj1.test\_attempt  
 setParam(param)  
 setTestId(res.data.id)  
 ***Api***.getTestStatus(props.queryparam.test\_short\_code ,props.queryparam.roll\_no ,param,obj1.test\_attempt)  
 .then((r)=>r.data)  
 .then((r)=>{  
 ***localStorage***.setItem('percent', '0')  
 if(r.code==='NF\_RECORD'){  
 setStart\_Name('Start Test')  
 props.onLoad(res.data)  
 if(res.data.hasOwnProperty('allowedDuration')){  
 //res.data.allowedDuration  
 }  
 }  
 if(r.code==='0' && r.data[param].state==='Completed'){  
 setIsTestCompleted(true)  
 setisRedirect(true)  
 }  
 if(r.code==='0' && r.data[param].state==='Started'){  
 setStart\_Name('Resume Test')  
 setAllTestStatus(r.data)  
 props.onLoad(res.data)  
 }  
 props.handleFirebaseEventLogs("instruction\_landing\_web","User lands on the instruction page of the test",{  
 "success" : true,  
 "dob" : props.queryparam.dob,  
 "class" : res.data.class,  
 "test\_name" : res.data.titleEn,  
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : res.data.id,  
 "grace\_period" : props.queryparam.grace\_period,  
 "disconnection\_time" : props.queryparam.disconnection\_time,  
 "duration\_of\_test" : res.data.allowedDuration,  
 "total\_questions" : res.data.totalQuestion,  
 "max\_marks" : res.data.maxMarks  
 })  
  
 })  
 }  
 })  
 .catch((error)=>{  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage)  
 setIsError(true)  
 setisRedirect(false)  
 }  
 else{  
 setIsError(true)  
 setisRedirect(false)  
 }  
 })  
 })  
 .catch((error)=>{  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage)  
 setIsError(true)  
 setisRedirect(false)  
 }  
 else{  
 setIsError(true)  
 setisRedirect(false)  
 }  
 })  
  
 }, [])  
  
 const isInPhoneOrTablet = ()=>{  
 if(***window***.matchMedia("(max-width: 768px)").matches){  
 return true  
 }  
 return false  
 }  
 if(props.testStatus.testStatus && isRedirect===true && !isTestCompleted){  
 return <TestStatus data={{  
 test\_short\_code:props.queryparam.test\_short\_code,  
 roll\_no:props.queryparam.roll\_no,  
 param:param,  
 title:props.instructions.testData.titleEn  
 }}/>  
 }  
 if(isRedirect===true && isTestCompleted===true){  
 return <Redirect to="/results" />  
 }  
 if (props.instructions.testData && !isRedirect && !isError) {  
 return <div className="instructions">  
 <div className="instruction-container">  
 <div className="instruction-header">  
 <div className="instruction-header-title-section">  
 <div className="instruction-header-title">  
 General Instructions:  
 </div>  
 <div className="instruction-header-subtitle">  
 Please read the instructions carefully  
 </div>  
 </div>  
 <div className="instruction-test-contraints">  
 <Duration time={props.instructions.testData} />  
 <NoOfQuestions number={props.instructions.testData} />  
 <Marks mark={props.instructions.testData} />  
 </div>  
 </div>  
 <InstructionList instructions={props.instructions.testData} />  
 <div className="start-button-wrapper">  
 {IsSecure===true ? <Button name={start\_name} style = { isInPhoneOrTablet()===true ? {background : '#00b0f5',width : '300px', height : '45px', color : '#fff', position : 'fixed', right : '0',margin:'auto', bottom : '1%',left:'0'} : {background:'#00b0f5', width:'500px', height : '45px', color : '#fff', margin : 'auto', left:'0', right:'0',bottom : '2%'}} onClick={startTest} /> : <p style = {{color: 'red', fontSize:'25px',fontWeight:'500'}}>Seems like you are a boat..!!</p>}  
 </div>  
 </div>  
 </div>  
 }  
   
 if(isError && !isRedirect){  
 return <InternetIssue message={errorMsg!=='' ? errorMsg : 'Something Went Wrong'}/>  
  
 }  
 return <Spinner message = {'loading'}/>  
  
}  
  
export default withRouter(***connect***(mapStateToProps, mapDispatchToProps)(InstructionsPage));

**5.2 Test Panel Page**

import ***React***, { useState, useEffect } from "react";  
import './TestPanel.css';  
import { withRouter } from "react-router";  
import Question from '../../components/Question/Question';  
import QuestionHeader from '../../components/QuestionHeader/QuestionHeader'  
import {  
 ***HIDE\_TIMER***,  
 ***FETCH\_QUESTION\_SET***,  
 ***UPDATE\_TIME***,  
 ***IS\_TEST\_DATA\_LOADED***,  
 ***UPDATE\_DISCONNECT\_TIME***,  
 ***UPDATE\_CALCULATED\_TIME\_LEFT***,  
 ***UPDATE\_CURRENT\_QUESTION***,  
 ***UPDATE\_HEADER\_VISIBLE***} from '../../constants/actionTypes';  
import ***Api*** from '../../data';  
import { ***connect*** } from 'react-redux';  
import Spinner from '../../components/Spinner/Spinner';  
import { useHistory, useLocation, Link, Redirect} from 'react-router-dom';  
import DeviceDetector from "device-detector-js";  
import PopUpWrapper from '../../components/PopUpWrapper/PopUpWrapper';  
import online from '../../../src/assets/online.svg';  
import offline from '../../../src/assets/offline.svg';  
import CloseTestConfirmation from '../../../src/components/CloseTestConfirmation/CloseTestConfirmation';  
import { ***store***,***persistor*** } from '../../redux/store';  
import PreLoadingScreen from '../../components/PreLoadingScreen/PreLoadingScreen';  
import InternetIssue from '../../components/InternetIssue/InternetIssue';  
import LoaderNew from "../../components/LoaderNew/LoaderNew";  
  
  
const mapStateToProps = (state) => {  
 return{  
 testData:state.testpanel,  
 time: state.timer,  
 userTestid: state.testStatus,  
 ...state,  
  
}}  
  
const mapDispatchToProps = dispatch => {  
 return{  
 onQuestionLoad: payload => dispatch({ type: ***FETCH\_QUESTION\_SET***, payload }),  
 hideTimer: payload => dispatch({ type: ***HIDE\_TIMER***, payload }),  
 setTestDataLoadedStatus: payload => dispatch({ type: ***IS\_TEST\_DATA\_LOADED***, payload }),  
 setDisconnectTime : payload => dispatch({type:***UPDATE\_DISCONNECT\_TIME***, payload}),  
 updateCalculatedTimeLeft : payload => dispatch({type: ***UPDATE\_CALCULATED\_TIME\_LEFT*** , payload}),  
 UpdateCurrentQuestion : payload => dispatch({type : ***UPDATE\_CURRENT\_QUESTION***, payload}),  
 setHeaderVisible:payload => dispatch({ type: ***UPDATE\_HEADER\_VISIBLE***,payload}),  
 updateTimer: payload => {   
 return new ***Promise***((resolve, reject) => {  
 dispatch({ type: ***UPDATE\_TIME***, payload });  
 resolve()  
 })  
 }  
  
}}  
  
  
  
const TestPanel = (props) => {  
  
 const [isError,setIsError] = useState(false);  
 const [showGoBackPopUp,setShowGoBackPopUp] = useState(false);  
 const [open, toggleClass] = useState(false)  
 const [connection, setConnection] = useState('')  
 const [done,setdone] = useState(0)  
 const [errorMsg, setErrorMsg] = useState('')  
 const [DontShowPop ,setDontShowPop] = useState(false)  
 const [timer, setTimer] = useState();  
 const [isRedirect,setisRedirect] = useState(false)  
 const [question, setQuestion] = useState();  
 const history = useHistory();  
 const location = useLocation();  
 const [isOffline,setIsOffline]=useState(false);  
  
 let timerAddress = timer  
  
 const handleClick = (index, data) => {  
 for (let i = 0; i < data.questionList.length; i++) {  
 if (data.questionList[i].question.isCurrentlyOpened === true) {  
 props.handleFirebaseEventLogs("tap\_section\_web","User taps on any section in the test",  
 {  
 "success" : true,  
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : props.instructions.testData.id,  
 "question\_id" : data.questionList[i].question.questionid,  
 "question\_type" : data.questionList[i].question.questionType,  
 "options\_selected" : data.questionList[i].question.answerbyUser  
 }  
 )  
   
 props.UpdateCurrentQuestion({...data.questionList[i].question, ...{"subject":data.subject}})  
 }  
 }  
  
 const \_index = parseInt(index, 0);  
 if (\_index !== props.active) {  
 props.setActive(\_index)  
 }  
 }  
 const handleType = (questionType, answerbyUser)=>{  
  
 if(questionType==='SCMCQ'){  
 ***console***.log("----answerbyuser--SCMCQ--",answerbyUser)  
 return ***JSON***.parse(answerbyUser)[0]  
 }  
 if(questionType==='MATRIX'){  
 ***console***.log("----answerbyuser---MATRIX-",answerbyUser)  
 return answerbyUser  
 }  
 if(questionType==='SAN' || questionType==='SAT'){  
 if(***Array***.isArray(***JSON***.parse(answerbyUser))){  
 ***console***.log("----answerbyuser---SAN--SAT--",answerbyUser)  
 return ***JSON***.parse(answerbyUser)[0]  
 }  
 }  
 if(questionType==='MCMCQ'){  
 ***console***.log("----answerbyuser----MCMCQ--",answerbyUser)  
 return ***JSON***.parse(answerbyUser)  
 }  
 return ''  
   
 }  
 const handleResponse = (res,res1) => {  
 let latestStatus = {}  
 res1.map((item) => {  
 latestStatus[item['questionId']] = item  
 })  
 let sections = []  
 res.sectionList.map((item) => {  
 sections.push({  
 "sectionid": item.id,  
 "sectionname": item.title.en  
 })  
 })  
 let questions = [];  
 sections.map((parent, i) => {  
 questions.push({  
 "subject": parent.sectionname,  
 "questionList": []  
 });  
 res.SectionQuestion.map((item, pos) => {  
  
 if (item.sectionId === parent.sectionid) {  
  
 let text = 'text';  
 if (item.question.defaultQuestionLanguage !== null) {  
 text = ('text' in item.question.defaultQuestionLanguage.question) ? item.question.defaultQuestionLanguage.question.text : item.question.defaultQuestionLanguage.question.question.text;  
 }  
 let answer = {}  
 if (item.question.defaultQuestionLanguage !== null) {  
 answer = 'question' in item.question.defaultQuestionLanguage ? item.question.defaultQuestionLanguage.question : {}  
 }  
 questions[i].questionList.push({  
 "qid": pos + 1,  
 "question": {  
 "questionid": item.questionId,  
 "text": text,  
 "questionType": item.question.questionType,  
 "answers": answer,  
 "isAnswered": latestStatus.hasOwnProperty(item.questionId) ? latestStatus[item.questionId].attempted===1 ? true : false : false,  
 "answerbyUser":latestStatus.hasOwnProperty(item.questionId) ? handleType(item.question.questionType, latestStatus[item.questionId].answer) : [""],  
 "isMarkedForReview": latestStatus.hasOwnProperty(item.questionId) ? latestStatus[item.questionId].markedForReview=== 1 ? true : false : false,  
 "isVisited":latestStatus.hasOwnProperty(item.questionId) ? true : false,  
 "answeredAndMarkedForReview": false,  
 "isCurrentlyOpened": questions[i].questionList.length === 0 ? true : false,  
 "timeSpent":0,  
 "lod": item.question.lod1  
 }  
 })  
  
 }  
  
 })  
 })  
 props.onQuestionLoad(questions)  
 }  
  
  
  
 const countDown=async()=> {  
 if(props.disconnect\_timer===0){  
 await ***Api***.logActivityType(props,"Auto\_Logout")  
 saveTotalData('Auto logged out due to exceeding the idle time',false)  
 ***window***.location.href = props.base\_url  
 ***persistor***.pause()  
 ***persistor***.flush().then(() => { return ***persistor***.purge() })  
 clearInterval(timerAddress)  
 }  
 if(props.disconnect\_timer){  
 props.setDisconnectTime(props.disconnect\_timer-1)  
 }  
 }  
  
  
 useEffect(() => {  
 ***navigator***.serviceWorker.onmessage=(e)=> {  
 if(e.data.isLogout){  
 setErrorMsg('You are being logged out of the test')  
 setIsError(true)  
 saveTotalData('logged out due to multiple device',false)  
 setTimeout(()=>{  
 ***window***.location.href = props.base\_url  
 },2000)  
 ***persistor***.pause()  
 ***persistor***.flush().then(() => { return ***persistor***.purge() })  
 }  
 }  
 props.handleFirebaseEventLogs("landing\_screen\_test\_web","User lands on the 1st question of the test",  
 {  
 "success" : true,  
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : props.instructions.testData.id,  
 "duration\_of\_test" : props.instructions.testData.allowedDuration,  
 "total\_questions" : props.instructions.testData.totalQuestion,  
 "max\_marks" : props.instructions.testData.maxMarks  
 })  
 if(props.testpanel.questionset.length>0 && props.testDataConfirmation.isTestDataLoaded &&!isError){  
 timerAddress = setInterval(countDown, 1000)  
 setTimer(timerAddress)  
 }  
 //condition for the showing the pop  
 if(props.testpanel.questionset.length>0 && props.testDataConfirmation.isTestDataLoaded && props.timer.t>0 && !DontShowPop && props.location.key===undefined){  
 setShowGoBackPopUp(true)  
 }  
 if(props.location.key!==undefined){  
 props.hideTimer(false)  
 }  
 ***window***.addEventListener('offline', function(e) {  
 setIsOffline(true);  
 setConnection('offline')  
 }, false)  
   
 ***window***.addEventListener('online', function(e) {  
 setConnection('online')  
 saveTotalData('internet got back after a disconnection',false)  
 setDontShowPop(true)  
 }, false)  
  
 if(props.pushNotificationEvent.hasOwnProperty('isLogout') && props.pushNotificationEvent.hasOwnProperty('detailMessage') && props.pushNotificationEvent.isLogout==="true"){  
 setErrorMsg(props.pushNotificationEvent.detailMessage)  
 setIsError(true)  
 saveTotalData('logged out due to multiple device',false)  
 setTimeout(()=>{  
 ***window***.location.href = props.base\_url  
 },2000)  
 ***persistor***.pause()  
 ***persistor***.flush().then(() => { return ***persistor***.purge() })  
 }  
 if(!showGoBackPopUp && !props.testDataConfirmation.isTestDataLoaded){  
 props.setHeaderVisible(false);  
 // let test\_attempt=props.globalTestStatus.split('\_');  
 // await Api.getTestStatus(props.queryparam.test\_short\_code ,props.queryparam.roll\_no ,props.globalTestStatus,test\_attempt[test\_attempt.length-1])  
 // .then((r)=>r.data)  
 // .then((r)=> {  
 // if (r.code === '0' && r.data[props.globalTestStatus].state === 'Completed') {  
 // setisRedirect(true);  
 // }  
 // })  
 setTimeout(() => {  
 props.handleFirebaseEventLogs("pre\_load\_screen\_web","When pre load screens comes up before loading of the test",  
 {  
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : props.instructions.testData.id  
 }  
 )  
 setdone(5)  
 setTimeout(() => {  
 setdone(50)  
 setTimeout(() => {  
 setdone(75)  
 setTimeout(() => {  
 setdone(85)  
 setTimeout(() => {  
 setdone(100)  
 if(props.isExistingTest){  
 if(props.test\_duration - props.resumedData<=0){  
 // alert("Consumed time is greater or equal to test duration")  
 ***console***.log("----props.test\_duration - props.resumedData---",props.test\_duration , props.resumedData)  
 setisRedirect(true)  
 return  
 }  
 props.updateTimer(6\*60).then(()=>{  
 props.updateCalculatedTimeLeft((props.test\_duration - props.resumedData)\*60)  
 props.setDisconnectTime(props.queryparam.disconnect\_time\*60)  
 })  
  
 }  
 else{  
 props.updateTimer(6\*60).then(()=>{  
 props.updateCalculatedTimeLeft(props.test\_duration\*60)  
 props.setDisconnectTime(props.queryparam.disconnect\_time\*60)  
 })  
   
 }  
 props.setTestDataLoadedStatus(true)  
 props.setHeaderVisible(true);  
 }, 1000)  
 }, 500)  
 }, 2000)  
 }, 2000)  
 }, 1000)  
 }  
 let ls = props.testData.hasOwnProperty('questionset') ? [...props.testData.questionset] : null  
 if (ls !== null && ls !== undefined && ***Array***.isArray(ls) && ls.length>0) {  
 props.onQuestionLoad(ls);  
 }  
 else {  
 if(props.testStatus.testStatus.data.testId){  
 ***Api***.getTestQuestionAnswerData(props.testStatus.testStatus.data.testId)  
 .then(res => res.data)  
 .then(res => {  
 if(props.isExistingTest===true){  
 ***Api***.getTestState(props.testStatus.testStatus.data.userTestId)  
 .then(res=>res.data)  
 .then(res1=>{  
 handleResponse(res,res1)  
 })  
 .catch((error)=>{  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage)  
 setIsError(true)  
 }  
 else{  
 setIsError(true)  
 }  
 })  
 }  
 else{  
 handleResponse(res,[])  
 }  
 setDontShowPop(true)  
 })  
 .catch((error)=>{  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage)  
 setIsError(true)  
 }  
 else{  
 setIsError(true)  
 }  
 })  
 }  
 }  
 //for getting the back button press event  
 ***window***.addEventListener('popstate', (event) => {  
 if (event.state) {  
 setIsOffline(false);  
 setShowGoBackPopUp(true)  
 }  
 }, false)  
 ***window***.history.pushState({name: "browserBack"}, "on browser back click", ***window***.location.href)  
 ***window***.history.pushState({name: "browserBack"}, "on browser back click", ***window***.location.href)  
 return () => {  
 clearInterval(timerAddress);  
 };  
 }, [props.testDataConfirmation.isTestDataLoaded,props.pushNotificationEvent,props.disconnect\_timer])  
  
  
 const saveTotalData = (activityType,isFinalSubmit)=>{  
  
 ***Api***.submitTest(props,isFinalSubmit)  
 .then(res => res.data)  
 .then(res => {  
 if (res.finalSubmit === isFinalSubmit) {  
 // Api.logActivityType(props,activityType)  
 // .then(res=> res.data)  
 // .then(res=>{  
 // if(res.message==='Success'){  
 //  
 // }  
 // else{  
 // setIsError(true)  
 // setErrorMsg(res.message)  
 // }  
 // })  
 }  
 else {  
 setIsError(true)  
 setErrorMsg(res.message)  
  
 // alert(res.message)  
 // setshowError(true)  
 }  
 })  
 .catch((error)=>{  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage)  
 setIsError(true)  
 }  
 else{  
 setIsError(true)  
 }  
 })  
 }  
  
  
 const isInPhoneOrTablet = ()=>{  
 if(***window***.matchMedia("(max-width: 768px)").matches){  
 return true  
 }  
 return false  
 }  
 const handleSubmit = (props,isFinalSubmit) => {  
 // props.handleFirebaseEventLogs("tap\_submit\_last\_question\_web","User taps on the submit button on the last question of the test")  
 saveTotalData('Leave Test',isFinalSubmit)  
 ***persistor***.pause()  
 ***persistor***.flush().then(() => { return ***persistor***.purge() })  
 ***window***.location.href = props.base\_url  
 }  
  
if(isRedirect===true){  
 return <Redirect to="/results" />  
}  
   
 if(props.testpanel.questionset.length>0 && props.testDataConfirmation.isTestDataLoaded &&!isError) {  
 return <div className={open && isInPhoneOrTablet() ? "test-panel-wrapper sidenavopen" :"test-panel-wrapper"}>  
 {connection==='offline' ?   
 <InternetIssue message={'Something Went Wrong'}/>  
 : null}  
 {connection==='online' ?  
 <PopUpWrapper isToast= {true} toastType = {'success'} width = {'281px'} height = {'81px'} top = {'auto'} right = {'auto'} closeButtonBackground = {'#000'}>  
 <div className="status-container"><img src = {offline}/>  
 <div>  
 <div className="status-text">  
 <span className="status-type">Connected!</span>  
 <span className="status-message">You are back online</span>  
 </div>  
 </div>  
 </div>  
 </PopUpWrapper> : null}  
   
 <div className="test-panel">  
 <QuestionHeader header={props.testpanel.questionset} handleClick={handleClick} active={props.active} />  
 <div className="tab-container-wrapper">  
 <Question question={props.testpanel.questionset} active={props.active} setActive = {props.setActive} open = {open} toggleClass ={toggleClass} handleFirebaseEventLogs = {props.handleFirebaseEventLogs} />  
 </div>  
 </div>  
 {showGoBackPopUp===true ? <PopUpWrapper style ={isInPhoneOrTablet() ? {width:'90%', transition : 'none'} :{width:'62%',transition : 'none'}} callback = {setShowGoBackPopUp} callback2 = {setDontShowPop} isToast = {false} > <CloseTestConfirmation handleFirebaseEventLogs = {props.handleFirebaseEventLogs} setDontShowPop = {setDontShowPop} handleSubmit = {handleSubmit} setShowGoBackPopUp = {setShowGoBackPopUp} showGoBackPopUp = {showGoBackPopUp}/></PopUpWrapper> : null}  
 </div>  
   
 }   
  
 if(isError===true && showGoBackPopUp===false){  
 return <InternetIssue redirectToInstruction message={errorMsg!=='' ? errorMsg : 'Something Went Wrong'}/>  
 }  
 if(!props.testDataConfirmation.isTestDataLoaded){  
 return <LoaderNew done={done} title1={'testData' in props.instructions?props.instructions.testData.titleEn:null} title2={'while we are loading your test'} title3={null} title4={'PLEASE DO NOT CLOSE THE BROWSER'}/>  
 }  
  
 return <Spinner message = {'Loading'}/>  
}  
  
  
  
export default withRouter(***connect***(mapStateToProps, mapDispatchToProps)(TestPanel));

**5.3 Leave popup page**

import { React, useState } from "react";  
import './PopUp.css'  
import close from '../../assets/close.svg'  
const PopUp = (props)=>{  
  
 const isFunction = (functionToCheck)=> {  
 return functionToCheck && {}.toString.call(functionToCheck) === '[object Function]';  
 }  
 return <div class="overlay" style = {props.isToast ? {background:'none', top : props.top, right :props.right} : null}>  
 <div class="popup" style = {props.toastType === 'danger' ? {background: 'red',color : '#fff', fontWeight : 'bold', width : props.width, height: props.height} : props.toastType ==='success' ? {background : 'green',color : '#fff', fontWeight : 'bold', width:props.width, height: props.height } : props.style}>  
 <div class="content">  
 {props.children}  
 </div>  
 {!props.isToast ? <img className="close" style={{background:props.closeButtonBackground}} src={close}   
 onClick={()=>{props.setShowPopUp(false);  
 if(isFunction(props.callback)) props.callback(false);  
 if(props.callback2) props.callback2(true) }}/> : null}  
 </div>  
 </div>   
  
  
}  
export default PopUp;

**5.4 Remaining Time Alert Page**

import ***React*** from "react";  
import './Palette.css';  
  
const Palette = (props)=>{  
 return <div className={props.className}><span>{props.value}</span><label>{props.name}</label></div>  
}  
  
export default Palette;

**4.5 Test Summary Page**

import ***React***, { useEffect, useState } from "react";  
import './CloseTestConfimation.css';  
import { Link, Redirect } from "react-router-dom";  
import QuestionHeader from '../QuestionHeader/QuestionHeader';  
import QuestionSummary from '../QuestionSummary/QuestionSummary'  
import { ***connect*** } from "react-redux";  
import Button from '../Button/Button'  
import { useHistory } from 'react-router-dom';  
import ***Api*** from '../../data';  
import { ***SUBMIT\_TEST***,***UPDATE\_TIME***,***HIDE\_TIMER*** } from "../../constants/actionTypes";  
import PopUpWrapper from '../PopUpWrapper/PopUpWrapper'  
import ProgressBar from '../ProgressBar/ProgressBar';  
import Timer from '../../components/Timer/Timer';  
  
const mapStateToProps = (state) => {  
 return{  
 ...state.testpanel,  
 time: state.timer,  
 usertestid: state.testStatus,  
 testData:state.testpanel,  
 completion: state.completion,  
 ...state  
}}  
  
const mapDispatchToProps = dispatch => {  
 return{  
 onSubmitTest: payload => dispatch({ type: ***SUBMIT\_TEST***, payload }),  
 hideTimer: payload => dispatch({ type: ***HIDE\_TIMER***, payload }),  
}};  
  
const CloseTestConfimation = (props) => {  
  
 const [timer, setTimer] = useState();  
 const [active, setActive] = useState(0);  
 const [showError, setshowError] = useState(false);  
 const [tData, settData] = useState([])  
  
  
 useEffect(()=>{  
 let timerAddress = timer  
 if(props.t===0){  
 clearInterval(timerAddress);  
 }  
  
   
 let tData = ***Array***.isArray(props.testData.questionset) ? [...props.testData.questionset] : null  
 if(***Array***.isArray(tData)){  
 let answered = 0, notAnswered = 0, visited = 0, markedForReview = 0, ansRev = 0, notVisited = 0, total = 0;  
 for (let i = 0; i < tData.length; i++) {  
 for (let j = 0; j < tData[i].questionList.length; j++) {  
 if (tData[i].questionList[j].question.isAnswered) {  
 answered++;  
 }  
 if (tData[i].questionList[j].question.isVisited) {  
 visited++;  
 }  
 // if (tData[i].questionList[j].question.isMarkedForReview){  
 // console.log("--c----",tData[i].questionList[j].question.isMarkedForReview,  
 // tData[i].questionList[j],  
 // tData[i].questionList[j].question.isAnswered)  
 // }  
 if (tData[i].questionList[j].question.isMarkedForReview && !tData[i].questionList[j].question.isAnswered) {  
 markedForReview++;  
 }  
 if (tData[i].questionList[j].question.isMarkedForReview && tData[i].questionList[j].question.isAnswered) {  
 ansRev++;  
 }  
 if (!tData[i].questionList[j].question.isAnswered && tData[i].questionList[j].question.isVisited && !tData[i].questionList[j].question.isMarkedForReview) {  
 notAnswered++;  
 }  
 total++;  
 }  
   
 }  
 notVisited = total - visited;  
 tData.unshift({  
 "subject": "All",  
 "summary": []  
 })  
 tData[0].summary.push({  
 "total": total,  
 "answered": answered,  
 "notanswered": notAnswered,  
 "markedForReview": markedForReview,  
 "notvisited": notVisited,  
 "answeredAndMarkedForReview": ansRev  
 })  
 settData(tData)  
 }  
 else{  
 setshowError(true)  
 }  
 },[])  
  
  
const isInPhoneOrTablet = ()=>{  
 if(***window***.matchMedia("(max-width: 768px)").matches){  
 // console.log("width check",window.matchMedia("(max-width: 768px)").matches)  
 return true  
 }  
 return false  
}  
 return <div className="close-test-confirmation-wrapper ">  
  
{***Array***.isArray(tData) ?   
 <div className="close-test-confirmation-section">  
 {/\* <QuestionHeader header={tData} icon={true} active={active} /> \*/}  
 <Timer setTimer = {setTimer} style = {{textAlign : 'center',color :'#00b0f5',fontSize: '28px',fontWeight:'500'}} class = {isInPhoneOrTablet() ? 'timer\_watch\_small' :'timer\_watch'} />  
 <span className="Completed">{props.completion.completion}% Completed</span>  
 <ProgressBar color= {'lightgrey'} done={props.completion.completion} opacity = {'1'} barcolor = {'#00b0f5'} barwidth = {'90%'} bgcolor = {'#e6e9ea'} barheight = '3px' />   
 <span className="confirmation-title" style={isInPhoneOrTablet()? {fontSize:'18px'} : {fontSize : '28px'}}>Do you want to leave the test?</span>  
 <div className="tab-container-wrapper-submit">  
 <QuestionSummary isCloseTestConfimationPopUp = {true} summary={tData} active={active} />  
 </div>  
 {props.time.t !== 0 ?  
 <div>  
 <div className="close-test-confirmation-submit-buttons" style = {isInPhoneOrTablet() && props.showGoBackPopUp ? {display:'flex'} : null}>  
 <Link to={{  
 pathname: "/testpanel",  
 state: { property\_id: "DontShowPopUp" }  
 }}  
 className='submit-btn'>  
 <Button onClick = {()=>{props.setShowGoBackPopUp(false);props.setDontShowPop(true)}} style = {{border:'1px solid #414141', backgroundColor:'#fff', color:'#414141', left : '0', right : '0',bottom:'20', fontSize:'12px',height: '40px', width :'100%', marginRight: '25px'}} name={"Resume"} />  
 </Link>  
 <Link to="/" className='submit-btn'>  
 <Button onClick = {()=>{props.handleSubmit(props,false);props.setShowGoBackPopUp(false);props.handleFirebaseEventLogs("tap\_leave\_test\_popup\_web","User taps on the leave button on the popup once the user taps on refresh, back or close the browser buttons",  
 {  
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : props.instructions.testData.id  
 }  
 )}} name={"Leave"} style = {{background:'#01b0f5', left : '0', right : '0',bottom:'20', color : '#fff', fontSize : '12px', height: '40px', width : '100%'}} />  
 </Link>  
 </div>  
 </div>  
 :  
 <Link to="/success">``  
 <Button name={"OK"} background={"#01b0f5"} color={"#fff"} fontSize={'12px'} height={'40px'} width={"300px"} margin ={"auto"}/>  
 </Link>  
 }  
 </div>:  
 showError ? <PopUpWrapper > Oops !! something went wrong ! Please Try again</PopUpWrapper> : null  
 }  
 </div>  
}  
export default ***connect***(mapStateToProps,mapDispatchToProps)(CloseTestConfimation);

**4.6 Loading Page**

import ***React***,{useEffect,useState} from "react";  
import ***LoaderData*** from "./loader.js";  
import './Loader.css';  
const Loder=(props)=>{  
 let count = 0,interval=0;  
 const [index,setIndex]=useState(0);  
 useEffect(()=>{  
 setInterval( () => {  
 count === 0 ? interval = count : count === 1 ?interval = 3 :interval = 6;  
 setIndex(interval);  
 if ( count >= 0 && count <= 2 ) {  
 count++;  
 } else {  
 count = 0;  
 }  
 },1000)  
 },[])  
 return(  
 <div className='loader\_new' style={{display:"flex",alignItems:'center',justifyContent:'center'}}>  
 <div style={{height:'90vh',width:'90%',backgroundColor:'#fff'}}>  
 <div className='title'>  
 <p style={{color:'#00b0f5',fontSize:'25px'}}>{props.title1}</p>  
 <p style={{color:'#00b0f5',fontSize:'20px',margin:'0px'}}>Please wait</p>  
 <p style={{color:'#00b0f5',marginTop:'5px'}}>{props.title2}<br/>{props.title3}</p>  
 <p style={{color:'#f5365c'}}>{props.title4}</p>  
 </div>  
 <div className='main\_container'>  
 <img className='image' src={`${***LoaderData***[index].p}`}></img>  
 <img className='image' src={`${***LoaderData***[index+1].p}`}></img>  
 <img className='image' src={`${***LoaderData***[index+2].p}`}></img>  
 </div>  
 <div className='progress\_container'>  
 <div className='progress-bar'>  
 <div className='loading' style={{width:`${props.done}%`}}>  
 </div>  
 </div>  
 </div>  
 <div className='title' style={{color:'#00b0f5',marginTop:'5px'}}>{props.done}% Completed</div>  
 </div>  
 </div>  
)  
}  
export default Loder;

**4.7 Result Page**

import ***React***, { useEffect,useState } from "react";  
import group from "../../assets/champion.svg"  
import download from "../../assets/download.svg"  
import call from "../../assets/phone-call-1.svg"  
import star\_unfilled from "../../assets/star\_unfilled.svg"  
import phone\_rating from "../../assets/ratings.svg"  
import Spinner from '../../components/Spinner/Spinner';  
import not\_qualified\_banner from '../../assets/not-qualified.svg'  
import PreLoadingScreen from '../../components/PreLoadingScreen/PreLoadingScreen';  
import ***Api*** from '../../data';  
import { useHistory } from 'react-router-dom';  
import { Link, Redirect } from "react-router-dom";  
import './Results.css';  
import { object } from "underscore";  
import { withRouter } from 'react-router-dom';  
import { ***connect*** } from 'react-redux';  
import InternetIssue from '../../components/InternetIssue/InternetIssue';  
import {***HIDE\_TIMER***} from '../../constants/actionTypes';  
import LoaderNew from '../../components/LoaderNew/LoaderNew';  
import logo from '../../assets/aakash-byjus-logo.svg';  
  
  
const mapStateToProps = state => ({  
 ...state  
})  
  
const mapDispatchToProps = dispatch => ({  
 hideTimer: payload => dispatch({ type: ***HIDE\_TIMER***, payload })  
});  
const Results = (props)=>{  
 const [resultData , setResultData] = useState('')  
 const [errorMsg, setErrorMsg] = useState('')  
 const [isError,setIsError] = useState(false);  
 const [isRedirect,setisRedirect] = useState(false)  
 const history = useHistory();  
 const [loadingPercent, setLoadingPercent] = useState(0);  
  
 const getResult = (call\_count)=>{  
 setInterval(() => {  
 ***localStorage***.setItem(  
 'percent',  
 ***String***(***Number***(***localStorage***.getItem('percent')) + 1)  
 );  
 if (***Number***(***localStorage***.getItem('percent')) > 100)  
 ***localStorage***.setItem('percent', '100');  
 setLoadingPercent(***Number***(***localStorage***.getItem('percent')));  
 }, 7000);  
 if(call\_count===3){  
 setIsError(true)  
 return   
 }  
 ***Api***.getoAuthToken()  
 .then(res => res.data)  
 .then(res => {  
 ***Api***.getResultData(res.access\_token , props.queryparam.roll\_no!==undefined && props.queryparam.roll\_no!==null ? props.queryparam.roll\_no : ***JSON***.parse(***localStorage***.getItem("roll\_no")))  
 .then((res)=>res.data)  
 .then((res)=>{  
 if(res.result\_status===null ){  
 setTimeout(()=>{  
 getResult(call\_count + 1)  
 },30000)  
 }  
 else{  
 setLoadingPercent(100);  
 ***localStorage***.setItem('percent', '101');  
 setTimeout(() => {  
 setResultData(res);  
 }, 2000);  
 }  
   
 })  
 .catch(error=>{  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage || error.response.data.message)  
 setIsError(true)  
 }  
 else{  
 setIsError(true)  
 }  
 })  
 })  
 .catch(error => {  
 if(error.response != undefined && error.response.status>=400){  
 setErrorMsg(error.response.data.detailMessage || error.response.data.message)  
 setIsError(true)  
 }  
 else{  
 setIsError(true)  
 }  
 })  
 }  
 useEffect(() => {  
 ***window***.addEventListener('load', () => {  
 ***localStorage***.setItem('percent', '0');  
 });  
 setInterval(() => {  
 ***localStorage***.setItem(  
 'percent',  
 ***String***(***Number***(***localStorage***.getItem('percent')) + 1)  
 );  
 if (***Number***(***localStorage***.getItem('percent')) > 100)  
 ***localStorage***.setItem('percent', '100');  
 setLoadingPercent(***Number***(***localStorage***.getItem('percent')));  
 }, 2000);  
 }, []);  
  
 useEffect(() => {  
 props.hideTimer(true)  
  
 props.handleFirebaseEventLogs("test\_result\_web","When user lands on the test result screen",  
 {  
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : 'testData' in props.instructions ? props.instructions.testData.id : "directly landed from instruction page to result page so test id is not available",  
 }  
 )  
 if (loadingPercent == 31) {  
 getResult(0);  
 }  
  
 ***window***.addEventListener('popstate', (event) => {  
 if (event.state) {  
 ***window***.location.href = "https://iacst.aakash.ac.in/exam/login"  
 }  
 }, false)  
 ***window***.history.pushState({name: "browserBack"}, "on browser back click", ***window***.location.href)  
 ***window***.history.pushState({name: "browserBack"}, "on browser back click", ***window***.location.href)  
  
  
 },[loadingPercent])  
  
 if(resultData!==undefined && resultData!== null && resultData!=='' && typeof(resultData)==='object' && resultData.result\_status !== null){  
 return <div className="results">  
 <div className="results-status">  
 {/\* this needed to updated \*/}  
 <img style={{height:'85px',display:'flex',marginLeft:'25px'}} src={logo} />  
 {resultData.result\_status.toLowerCase() === "" || resultData.result\_status.toLowerCase() == "qualified" ?  
 <div className="results-qualified">  
 <div className="results-congratulations">  
 <span className="congrate">Congratulations!</span>  
 <span className="not-qualified-msg">You have qualified for scholarship based on your iACST Score.</span>  
 </div>  
 <div className="results-champion">  
 <img className="results-champion-img" src={group}/>  
 </div>  
 </div>  
 :  
 <div className="results-failed">  
 <div className="results-congratulations">  
 <img className="not\_qualified\_banner" src = {not\_qualified\_banner}/>  
  
 <span className="congrate">Hard Luck!!</span>  
 <span className="not-qualified-msg">“ It’s not how far you fall, but how high </span>  
 <span className="not-qualified-msg">you bounce that counts ”</span>  
 <span className="not-qualified-msg">- Zig Ziglar</span>  
 <span className="not-qualified-msg-explain">You have not qualified for scholarship based on your iACST Score.</span>  
 </div>  
  
 </div>  
 }  
 </div>  
  
 <div className="results-download-reports">  
 <a className="results-download-reports-txt" onClick = {props.handleFirebaseEventLogs("tap\_download\_pdf\_web","User taps on the download pdf button on the test result screen",{  
   
 "roll\_no" : props.queryparam.roll\_no,  
 "test\_id" : 'testData' in props.instructions ? props.instructions.testData.id : "directly landed from instruction page to result page so test id is not available",  
   
 })} href={resultData.result\_url}>Download your detailed result report</a>  
 {/\* <span href= {resultData.result\_url}>Download your detailed result report</span> \*/}  
 <img className="results-download-reports-img" src={download}/>  
 </div>  
 <div className="results-contactus-rateus">  
 <div className="results-contactus-rateus-cotactus">  
 <span className="results-contactus-rateus-cotactus-title">Book Your Academic Counselling Session</span>  
 <div className="results-contactus-rateus-content">  
 <div className="results-contactus-rateus-call-us">  
 <img className="results-contactus-rateus-call-us-img" src={call}/>  
 <span className="results-contactus-rateus-call-us-text">Call Us</span>  
 </div>  
 <span className="results-contactus-rateus-call-us-timing">We are available to assist you from 10 am to 6pm on all days.</span>  
 <span className="results-contactus-rateus-call-us-number">{resultData.branch\_phone} or {resultData.generic\_phone\_no} </span>  
 </div>  
 </div>  
 {/\* <div className="results-contactus-rateus-rateus">  
 <img src={phone\_rating} className="results-contactus-rateus-rateus-phone-rating"/>  
 <span className="results-contactus-rateus-rateus-text">How was your iACST exam?</span>  
 <span className="results-contactus-rateus-rateus-description">Help us improve your exam experince.</span>  
 <div className="results-contactus-rateus-rateus-stars">  
 <img className="results-contactus-rateus-rateus-star" src={star\_unfilled}/>  
 <img className="results-contactus-rateus-rateus-star" src={star\_unfilled}/>  
 <img className="results-contactus-rateus-rateus-star" src={star\_unfilled}/>  
 <img className="results-contactus-rateus-rateus-star" src={star\_unfilled}/>  
 <img className="results-contactus-rateus-rateus-star" src={star\_unfilled}/>  
 </div>  
 </div> \*/}  
 </div>  
 </div>  
 }  
 if(resultData.result\_status === null){  
 ***localStorage***.setItem('percent', '0');  
 return <InternetIssue redirectToInstruction message={errorMsg!=='' ? errorMsg : 'Something Went Wrong'}/>  
 }  
 if(isError===true){  
 ***localStorage***.setItem('percent', '0');  
 return <InternetIssue redirectToInstruction message={errorMsg!=='' ? errorMsg : 'Something Went Wrong'}/>  
 }  
 return <LoaderNew done={loadingPercent} title1={'testData' in props.instructions?props.instructions.testData.titleEn:null}  
 title2={'Your result and scholarship are being computed,'} title3={'It takes around 2 minutes for the results to be available.'}  
 title4={'PLEASE DO NOT CLOSE THE BROWSER'}/>;  
}  
  
export default withRouter(***connect***(mapStateToProps, mapDispatchToProps)(Results));

**4.8 Failure Page**

import ***React***, { useEffect,useState } from "react";  
import group from "../../assets/champion.svg"  
import download from "../../assets/download.svg"  
import call from "../../assets/phone-call-1.svg"  
import star\_unfilled from "../../assets/star\_unfilled.svg"  
import phone\_rating from "../../assets/ratings.svg"  
import offline\_banner from "../../assets/offline-banner.svg"  
import Button from "../Button/Button"  
import ***Api*** from '../../data';  
import './InternetIssue.css';  
import autoMergeLevel1 from "redux-persist/es/stateReconciler/autoMergeLevel1";  
  
  
  
  
const InternetIssue = (props)=>{  
  
 useEffect(() => {  
 let el = ***document***.getElementById('II');  
 el.scrollIntoView();  
 ***console***.log("------redirectToInstruction------",props.redirectToInstruction)  
 ***console***.log("------redirectToInstruction------",props.base\_url)  
  
 if(props.redirectToInstruction){  
 // setTimeout(()=>{  
 // window.location.href = props.base\_url  
 // },2000)  
 ***console***.log("-----",props.base\_url)  
 }  
 // Api.getoAuthToken()  
 // .then(res => res.data)  
 // .then(res => {  
 // console.log("----",res)  
 // })  
 // .catch(error => {  
 // })  
  
 }, []);   
   
 return <div className="offline" id='II'>  
 <img className="offline-banner" src={offline\_banner}/>  
 <div className="offline-footer">  
 <span className="offline-footer-title">Whoops!</span>  
 <span className="offline-footer-text">{props.message}</span>  
 <Button name="Try Again" onClick = {()=>***window***.location.reload()} style = {{margin:'auto',marginTop:'0px',width:'135px',background: '#00b0f5',border: 'solid 1px #fff',color: '#fff'}}/>  
 </div>  
 </div>  
  
}  
  
export default InternetIssue;

**CHAPTER 7**

**7.1 LIMITATIONS**

#### **Challenges in Technology Adoption**

Implementing a new system may create a minor disruption and require a period of familiarization by the users. Any transition from traditional methodology to the online mode will require some investment from educational institutions to upgrade the systems. However, familiarizing both students and teachers with the transition is the ideal way forward. Such training would educate students and teachers about the new features and benefits of an online examination system.

#### **Infrastructural Barriers**

One of the major disadvantages of an online examination system surfaces in remote locations where access to electricity, stable internet connection and other basic system requirements are difficult to meet. Such barriers impede online exams. Therefore, while shifting to the online mode, institutions must consider whether all students can appear for the test and what arrangements can be made to ensure the same.

#### **Difficulty in Grading Long Answer-type**

Even though online examinations have eased the invigilators’ job, the problem arises mostly with long answer-type questions. Subjective answers require manual grading, for which examiners need to invest additional time. This is where the modern system overlaps with the traditional one. As technology in the education sector continues to evolve, the grading of subjective answers would become easier.

#### **Susceptible to Cheating**

One of the major disadvantages of an online examination system is cheating. Especially in high-stakes exams, students often resort to impersonation, making universities suspect test-takers’ identity. Students may also get external help via smartphones or smartwatches. The systems used by candidates also offer multiple cheating opportunities, such as connecting external storage devices, screen sharing, etc.

These cheating instances can be averted with Mercer | Mettl’s highly secure examination platform, featuring advanced AI-based proctoring features. Our three-step authentication process, secure browser and multiple proctoring features flag a host of malpractices, lending credibility to online exams.

#### **Transitioning to Open-Book Exams**

Usually, the format of online exams is perceived as akin to open-book exams. Unlike a traditional exam setting where external materials are not permitted inside the exam hall, there is no such provision in the digital format. In fact, it is difficult to stop students from referring to their study materials during an online assessment. Moreover, there is a lack of supervision during the exam session, which jeopardizes the credibility of these tests.

However, an advanced online platform such as Mercer | Mettl offers several benefits of an online examination system to address the challenge in online remote proctoring. This unique service curbs cheating by providing online proctoring services to keep candidates from indulging in untoward activities. Using this service, teachers can safeguard examination integrity. If a candidate deviates from their computer screen to use reference material or seeks support from another person or attempts to open multiple tabs on the browser, proctors will flag such a suspicious behavior and take appropriate action.

**7.2 FUTURE SCOPE**

Online Examination System is widely used as compared to other exams . Online examination system can be used in private institutes as well as educational institution. As it is user friendly web base application it can be used anywhere and anytime. Every software may have some cases of bugs, errors, security related problems or system faults. There are many problems or system faults for example; computer collapse or crashes due to power supply problem will invalidate efforts of number of students. There are large numbers of chances in which software may produce wrong results or may display invalid data. These bugs must be identified and solved for improving quality of software. So in future we can develop more secure software by using advanced technologies.

**7.3 Future Enhancement**

**-** to implement the functionality for bilingual support.

- to implement the functionality for proctoring.

**CHAPTER 11**

**CONCLUSION**

Online Examination System is significantly superior among the other exams. We have come to result that the problems can be solved by introducing new security systems using biometrics, we can identify the student’s true identity by analyzing digital signature or by finger print mechanism and also by providing web cameras in the examination hall. Although web cameras Sometimes gets failed, if supposed a candidate is giving exam and facing downwards in such case Iris recognition and face recognition must be used. We conclude that no mechanism is ideal. Each mechanism has some restriction on its own. Key concepts are to develop paperless environment and to convert all the documentation in digital form.

**CHAPTER 12**

**REFERENCES**

**Journal Research Papers**:

[1]. Shubham Bobde, Suraj Chaudhari, Suraj Chaudhari and Jagupati Golguri,”Web Based Online Examination System” ,Volume 2, April 2017. Vol-4 Issue-2 2018 IJARIIE-ISSN(O)-2395-4396 7475 www.ijariie.com 214

[2]. Deepankar Vishwas Kotwal, Shubham ,Rajendra Bhadke ,Aishwarya Sanjay Gunjal and Puspendu Biswas,”online examination system”, Volume: 0, Jan-2016.

[3]. Zhao Qiao-fang, Li Yong-fei,”Research & development of online examination system”.

[4]. IJCSI International Journal of ComputerScience Issues, Vol. 10, Issue 1, No 1,January 2013 ISSN (Print): 1694-0784 |ISSN (Online): 1694-0814www.IJCSI.org

[5]. Xu Qiaoxia, Liu Dongsheng. Research and Design of the Security of Network Examination System [J]. Computer Education, vol 5, pp. 40-42, 2010. [2] Song Luo, Jianbin Hu and Zhong Chen."Task based automatic examination system for sequenced test". 2009 International Conference on Electronic Computer Technolgoy, 2009, pp.18-21.

[6]. PENG JiJiang, YANG Guangzhong. "Design and realization of internet Exam system based on .NET". Science Technology and Engineering Vol(5),2005(20), pp. 1555-1560.

[7]. WebBased online Secured Exam; B.Persis Urbana Ivy,A.shalini, A.Yamuna/International Journal of Engineering Research and Applications (IJERA) ISSN:2248-9622 www.ijera.com Vol. 2, Issue 1,JanFeb 2012, pp.943-944943.