

MINI PROJECT

(2020-21)

Building and Deployment of Web application

ONLINE EXAMINATION SYSTEM

PROJECT REPORT



Institute of Engineering & Technology

Submitted by: -

Sachi Tripathi (181500598)

Shikha Parashar (181500661)

Satyam Kumar Jha (181500627)

Supervised By: -

Dr. Manoj Varshney

Technical Trainer

Department of Computer Engineering & Applications

GLA University

Mathura- 281406,

INDIA



Department of computer Engineering and Applications

GLA University, Mathura

17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,

DECLARATION

I/we hereby declare that the work which is being presented in the B.Tech. Project **“Online Examination System”**, in partial fulfilment of the requirements for the award of the *Bachelor of Technology* in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of GLA University, Mathura, is an authentic record of my/our own work carried under the supervision of **Dr. Manoj Varshney**.

The contents of this project report, in full or in parts, have not been submitted to any other Institute or University for the award of any degree.

Sign _____

Name: Shikha Parashar

University Roll No.: 181500661

Sign _____

Name: Sachi Tripathi

University Roll No.: 181500598

Sign _____

Name: Satyam Kumar Jha

University Roll No.: 181500627

Date:



Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuha,

CERTIFICATE

This is to certify that the project entitled “**Online Examination System**”, carried out in Mini Project – II Lab, is a bonafide work by Satyam Kumar Jha (181500627), Sachi Tripathi (181500598) and Shikha Parashar (181500661) and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Signature of Supervisor:

Name of Supervisor: Dr. Manoj Varshney

Date:

ACKNOWLEDGEMENT

First and foremost, praises and thanks to the God, the Almighty, for His showers of blessings throughout our mini project to complete the project successfully.

I/we would like to express our deep and sincere gratitude to our college faculties for giving us this opportunity to do a mini project. I/We am extremely grateful to my mentor, Dr. Manoj Varshney, for his invaluable guidance throughout this mini project. His dynamism, vision, sincerity and motivation have deeply inspired us. He has guided us so well. It was a great privilege and honor to work and study under his guidance. I/we are extremely grateful for what he has offered us. I would also like to thank him for his empathy. I/we am extremely thankful to our friends and family for their acceptance and patience during this mini project.

I/We are extremely grateful to our parents for their love, prayers, caring and sacrifices for educating and preparing us for my future.

Sign _____

Name: Shikha Parashar

University Roll No.: 181500661

Sign _____

Name: Sachi Tripathi

University Roll No.: 181500598

Sign _____

Name: Satyam Kumar Jha

University Roll No.: 181500627

ABSTRACT

In this we will build a simple web application which will be based on Online Examination System using web development libraries such as ReactJS, Node JS and Express JS.

As we all know examination is a test not just for the person who is giving the exam but also, for the person who is managing the examination keeping in mind that everything goes well. Online Examination system removes those little possible discrepancies or faults which takes place during offline examinations.

A major advantage of such a system is that student does not have to be physically present at the exam Centre, he can give his exam from anywhere using just a desktop, mobile and an Internet connection. Also, in case of offline exams, there is a delay in calculating the results as it has to be done manually in that case but in online examination system there is a possible way to make sure that the results are declared not much after the student had finished it. It also minimizes the error in calculating the results. This is the basic idea for choosing this project.

TABLE OF CONTENTS

DECLARATION.....	2
CERTIFICATE.....	3
ACKNOWLEDGEMENT.....	4
ABSTRACT.....	5
1. INTRODUCTION	1
1.1. GENERAL INTRODUCTION.....	1
1.2. OBJECTIVE	1
1.3. MOTIVATION.....	1
1.4. FEATURES.....	2
1.5. TECHONOLOGIES USED.....	2
1.5.1. Node JS.....	2
1.5.2. ReactJS	3
1.5.3. JavaScript	5
Why to Learn JavaScript.....	5
1.5.4. Front-end Design: HTML, CSS, Bootstrap	7
1.5.5. ExpressJS	7
Installing Express.....	8
1.6. SCOPE.....	9
2. SOFTWARE REQUIREMENT ANALYSIS	10
2.1. INTRODUCTION.....	10
2.1.1. Purpose	10
2.1.2. Document Convention:	10
2.1.3. Intended Audience:.....	11
2.1.4. Definitions, Acronyms, and Abbreviations.	11
2.2. PERSPECTIVE:	11
2.3. PRODUCT FUNCTIONS:	11
2.4. USER CLASSES AND CHARACTERISCTICS:.....	12
2.5. OPERATING ENIRNOMENT:.....	12
2.6. DESIGN CONSTRAINS:	12
2.7. ASSUMPTIONS AND DEPENDENCIES:	13
2.8. FUNCTIONAL REQUIREMENT:	13
2.8.1. User Interfaces:.....	13

2.9.	PERORMANCE REQUIREMENTS:	14
2.9.1.	Hardware Requirements:	14
2.9.2.	Software Requirements:	14
2.10.	COMMUNICATION PROTOCOLS AND INTERFACES:	15
3.	SOFTWARE DESIGN	16
3.1.	DATAFLOW DIAGRAM.....	16
3.1.1.	DFD Level 0.....	16
3.1.2.	DFD Level 1.....	17
3.1.3.	DFD Level 2.....	18
3.2.	UML DIAGRAMS.....	19
3.2.1.	Sequence Diagram	19
3.2.2.	Class Diagram	20
3.2.3.	UseCase Diagram	21
4.	TESTING	22
4.1.	TEST PLAN	22
5.	IMPLEMENTATION AND USER INTERFACES.....	23
5.1.	COMPONENTS/SCREENS.....	23
5.3.1.	Signup.js	23
5.3.2.	Login.js	25
5.3.3.	ExamList.js.....	27
5.3.4.	MCQ.js.....	29
5.2.	App.js file	33
5.3.	USER INTERFACES	34
5.3.1.	Registration Page	34
5.3.2.	LOGIN PAGE	35
5.3.3.	Dashboard page	35
5.3.4.	Take Exam Page	36
5.3.5.	Questions Page	36
5.3.6.	Results Page	37
5.4.	DEPLOYMENT	37
6.	CONCLUSION AND FUTURE WORK.....	39
6.1.	CONCLUSION.....	39
6.2.	FUTURE WORK	39
7.	REFERENCES	40

APPENDIX.....	41
----------------------	-----------

LIST OF FIGURES

Figure 1 Features Node JS	3
Figure 2 Features React JS.....	5
Figure 3 Features JavaScript.....	6
Figure 4 Features Express JS	8
Figure 5 DFD level 0	16
Figure 6 DFD level 1	17
Figure 7 DFD level 2	18
Figure 8 Sequence Diagram.....	19
Figure 9 Class Diagram	20
Figure 10 Use Case Diagram	21
Figure 11 signUpNow() method.....	23
Figure 12 Registration page if entered Wrong email.....	24
Figure 13 SignUp.js render() method	25
Figure 14 Login Page when entered wrong credentials.....	26
Figure 15 Login.js Code	26
Figure 16 ExamList.js state variables	27
Figure 17 renderExamInfo() method in ExamList.js	28
Figure 18 renderExamList() in ExamList.js	28
Figure 19 MCQ.js state variables.....	29
Figure 20 MCQ.js next() method.....	30
Figure 21 MCQ.js timer() method	31
Figure 22 MCQ.js render() method	32
Figure 23 App.js file	33
Figure 24 Registration Page UI.....	34

Figure 25 Login Page UI.....	35
Figure 26 Dashboard UI.....	35
Figure 27 Exam Page UI.....	36
Figure 28 Question Panel UI.....	36
Figure 29 Result Page UI.....	37

1. INTRODUCTION

1.1. GENERAL INTRODUCTION

It is an online examination system to test the skills and knowledge acquired by the students during the tenure of their studies. As we are heading towards digital world, this can be a better way of taking examinations as it is more convenient than older ways.

With the help of this project, we have tried to build a web-based application for the knowledge and to test the acquired knowledge of the people, where someone can give their exam and get examined of what they have prepared while studying for the examination.

This mainly focuses on building a web-based examination portal. It aims to build a system which can offer a convenient way of taking examinations. This has multiple benefits like students don't have to actually visit examination centres or they need not to be physically present over there. They just need a device having a trustworthy internet connection. Also, it is very convenient for the officials to conduct exams in online mode as it relieves them of so many different things which they need to manage while conducting examinations in offline mode.

1.2. OBJECTIVE

The main objective of this project is to create a web application through which people can give online examinations. It will be a web application developed using Node.JS platform and other supporting technologies.

1.3. MOTIVATION

“Knowing Is Not Enough; We Must Apply. Wishing Is Not Enough; We Must Do.”

The motivation behind the building of this web application is that we all know how important it is for the students to test the skills acquired by them otherwise they will be unable to know how much knowledge they have gained during the tenure of their

education. Since in the current scenario, people are getting more comfortable with doing things virtually, so we came up with the idea of having an online platform for taking examinations.

1.4. FEATURES

- a) The application designed is interactive and user friendly.
- b) Simple and easy to take examination.
- c) Easy to modify

1.5. TECHNOLOGIES USED

1.5.1. Node JS

Node.js is a server-side platform built on Google Chrome's JavaScript Engine (V8 Engine). Node.js was developed by Ryan Dahl in 2009 and its latest version is v0.10.36.

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Features of NodeJS:

Following are some of the important features that make Node.js the first choice of software architects.

- **Asynchronous and Event Driven** – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API

after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

- **Very Fast** – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- **Single Threaded but Highly Scalable** – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- **No Buffering** – Node.js applications never buffer any data. These applications simply output the data in chunks.
- **License** – Node.js is released under the [MIT license](#)

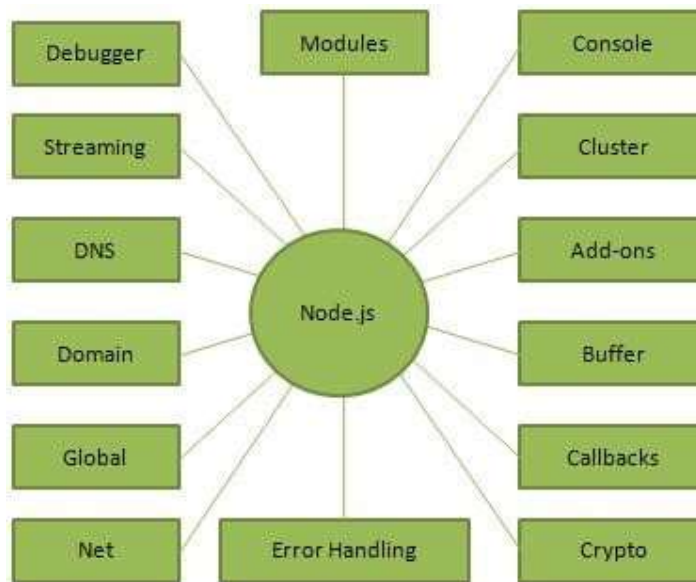


Figure 1 Features Node JS

1.5.2. ReactJS

React is a JavaScript library for building user interfaces. React is a library for building composable user interfaces. It encourages the creation of reusable UI components,

which present data that changes over time. Lots of people use React as the V in MVC. React abstracts away the DOM from you, offering a simpler programming model and better performance. React can also render on the server using Node, and it can power native apps using React Native. React implements one-way reactive data flow, which reduces the boilerplate and is easier to reason about than traditional data binding.

- **Declarative:** React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable, simpler to understand, and easier to debug.
- **Component-Based:** Build encapsulated components that manage their own state, then compose them to make complex UIs. Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM.
- **Learn Once, Write Anywhere:** We don't make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code. React can also render on the server using Node and power mobile apps using [React Native](#).

React Features:

- **JSX** – JSX is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
- **Components** – React is all about components. You need to think of everything as a component. This will help you maintain the code when working on larger scale projects.
- **Unidirectional data flow and Flux** – React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.
- **License** – React is licensed under the Facebook Inc. Documentation is licensed under CC BY 4.0.

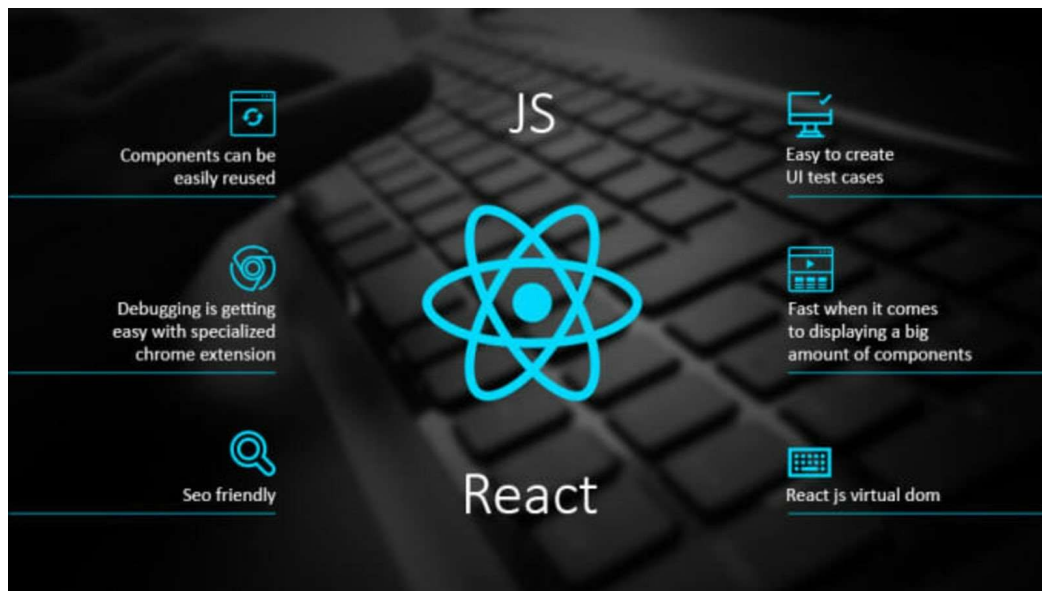


Figure 2 Features React JS

1.5.3. JavaScript

JavaScript is a lightweight, interpreted **programming** language. It is designed for creating network-centric applications. It is complimentary to and integrated with Java. **JavaScript** is very easy to implement because it is integrated with HTML. It is open and cross-platform.

Why to Learn JavaScript

JavaScript is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning JavaScript:

- JavaScript is the most popular **programming language** in the world and that makes it a programmer's great choice. Once you learnt JavaScript, it helps you developing great front-end as well as back-end software using different JavaScript based frameworks like jQuery, Node.JS etc.
- JavaScript is everywhere, it comes installed on every modern web browser and so to learn JavaScript you really do not need any special environment setup. For example, Chrome, Mozilla Firefox, Safari and every browser you know as of today, supports JavaScript.

- JavaScript helps you create really beautiful and crazy fast websites. You can develop your website with a console like look and feel and give your users the best Graphical User Experience.
- JavaScript usage has now extended to mobile app development, desktop app development, and game development. This opens many opportunities for you as JavaScript Programmer.
- Due to high demand, there is tons of job growth and high pay for those who know JavaScript. You can navigate over to different job sites to see what having JavaScript skills looks like in the job market.
- Great thing about JavaScript is that you will find tons of frameworks and Libraries already developed which can be used directly in your software development to reduce your time to market.

There could be 1000s of good reasons to learn JavaScript Programming. But one thing for sure, to learn any **programming language**, not only Javascript, you just need to code, and code and finally code until you become expert.

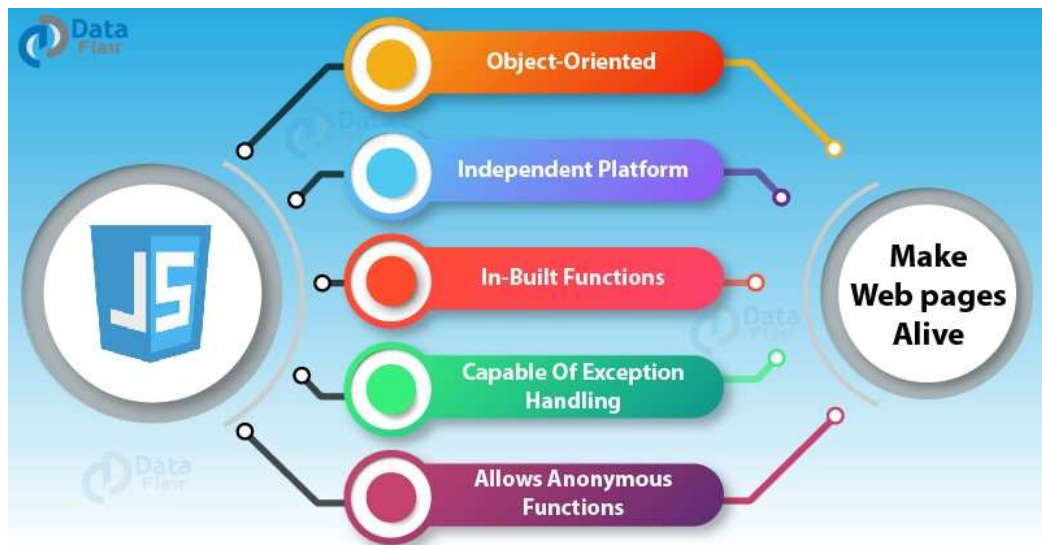


Figure 3 Features JavaScript

1.5.4. Front-end Design: HTML, CSS, Bootstrap

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, 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, 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.

1.5.5. ExpressJS

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework –

- Allows to set up middle wares to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.

- Allows to dynamically render HTML Pages based on passing arguments to templates.

Installing Express

Firstly, install the Express framework globally using NPM so that it can be used to create a web application using node terminal.

\$ npm install express –save

The above command saves the installation locally in the **node_modules** directory and creates a directory **express** inside **node_modules**. You should install the following important modules along with **express** –

- **body-parser** – This is a node.js middleware for handling JSON, Raw, Text and URL encoded form data.
- **cookie-parser** – Parse Cookie header and populate req. cookies with an object keyed by the cookie names.
- **multer** – This is a node.js middleware for handling multipart/form-data.

Express.js

- Adds functionality to Connect
- Built on top of Connect Middleware
 - Request / Response enhancements
 - Routing
 - View Support
 - HTML Helpers
 - Content Negotiation
- Exposes Connect Middleware

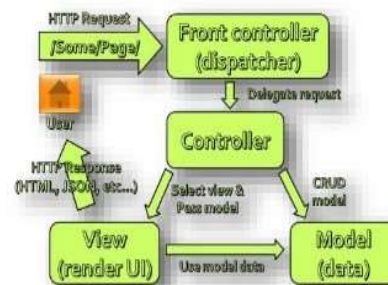


Figure 4 Features Express JS

1.6. SCOPE

The project that has been created is a fully automated version of kind of an Examination System. The aim of this project is to provide people with an interactive environment where they can improve their knowledge and test their knowledge by advancing through the questions. This web app can be a booster in youth by including more features like general knowledge, subject description, type of subjects, and many more and finally giving a complete virtual feel of an examination to the student which allow the students to give exams in different domains and other fields.

2. SOFTWARE REQUIREMENT ANALYSIS

2.1. INTRODUCTION

The aim of this part is to gather and analyze and give an in-depth insight of the complete **ONLINE-EXAMINATION-SYSTEM PROJECT** by defining the problem statement in detail. Nevertheless, it also concentrates on the capabilities required by stakeholders and their needs while defining high-level product features. The detailed requirements of **ONLINE-EXAMINATION-SYSTEM PROJECT** are provided in this document.

2.1.1. Purpose

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to consumers. Also, we shall predict and sort out how we hope this product will be used in order to gain a better understanding of the project, outline concepts that may be developed later, and document ideas that are being considered, but may be discarded as the product develops.

In short, the purpose of this report document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how our client, team and audience see the product and its functionality. Nonetheless, it helps any designer and developer to assist in software delivery lifecycle (SDLC) processes.

2.1.2. Document Convention:

In this text, it will use font small 2 and overstriking for primary title, font small 3 for secondary title and font 4 for the content. And it will use the italic when mentions the name of the application **ONLINE-EXAMINATION-SYSTEM**.

2.1.3. Intended Audience:

This SRS about ONLINE-EXAMINATION-SYSTEM is for developers, mentors, users and testers. The article mainly introduces the overall description, external interface requirements, system features and other non-functional requirements. I suppose mentor to read the whole article carefully and user pay attention to overall description especially. Users and testers read the system features carefully.

2.1.4. Definitions, Acronyms, and Abbreviations.

Configuration	It means a product which is available / Selected from a catalogue can be customized.
FAQ	Frequently Asked Questions
CRM	Customer Relationship Management
RAID 5	Redundant Array of Inexpensive Disk/Drives

2.2. PERSPECTIVE:

A web application preferably using the MERN (MongoDB, ExpressJS, ReactJS, NodeJS) stack.

2.3. PRODUCT FUNCTIONS:

Help the users to use this application to take exams and get their results according to the answers given by them.

This is simply a web application which will provide the user with an environment in which a registration page will appear as soon as the user land on the portal. It shows up when a user either tries to take the exam without logging in or logs out after giving the exam.

After registering, a login page will appear. When the user clicks on login button after entering his set email and password, he will be directed to the dashboard where option of giving exam is there. There the user can give exam in order to check his knowledge and according to his performance, he will be awarded the scores. After finishing the

exam that he/she can logout from the portal and he will return to the homepage and can login again for taking another exam.

❖ RESPONSE TIME: < 1 Second.

2.4. USER CLASSES AND CHARACTERISTICS:

Our application mainly those who are studying right now and are comfortable with virtual world. Mostly they are School and College going students, but it doesn't rule out some other people want to use this application to get knowledge and test their acquired knowledge.

The web application users interact with this application through a web browser. Other web applications are those applications like bots, third party application, can also use this application through its web Api. All requests should meet the specifications of application Api.

2.5. OPERATING ENVIRONMENT:

Our software is a multi-functional software system based on the windows platform. It is compatible and can run on 64 - bit laptop or ordinary desktop.

2.6. DESIGN CONSTRAINTS:

Our application must accept the command and then start the exam. We must consider about the arrangement and beautification of the interface; Prioritization of processing operations and it deepens the difficulty of coding and testing. This application needs users to enroll in the application and then choose the subject he/she wishes. Every time, the user logs in he/she will be shown a list of subjects, after selecting the desired subject, the user has to choose from the list of exams and finally calculate result according to the answers given by the user. So, the application must analyze the answers correctly so that the accurate result can be calculated. It may raise the error rate.

2.7. ASSUMPTIONS AND DEPENDENCIES:

The people who manage the application should know about the knowledge of the subjects in order to update the list of subjects and number of questions on regular basis.

ONLINE-EXAMINATION-SYSTEM is a web app having *NodeJS* for backend interactions or third-party application interactions. It also uses *ReactJS* and *JavaScript* for frontend interactions.

List- ReactJS, NodeJS, ExpressJS, JavaScript, HTML, and CSS.

2.8. FUNCTIONAL REQUIREMENT:

2.8.1. User Interfaces:

UI-1: As soon as the user land on the portal, registration page is displayed. It shows up when a user either tries to take the exam without logging in or logs out after giving the exam. This form has three fields and a button to register. First field is to enter Username, second is for email and third one is to set the desired password. If the user already has an account, he/she can directly login through one more button on the same page.

UI-2: This page is displayed to the user either after completing the registration or by clicking on the login button on registration page. It has a login for with two fields one for entering email and another for entering the password with a button to login and one more to go back to signup/registration page.

UI-3: Once the user is done with all the process of registration and logging in he lands up to the dashboard. It displays a list of thumbnails mentioning list of subjects with a button to proceed further. On the top-right corner there is a button to logout from the dashboard.

UI-4: This page displays the Questions which student has to answer. It displays a list of multiple-choice questions one by one to the student which he/she can answer in maximum one minute. There is a button at bottom of the question to jump to the next question and one at top-right to logout.

UI-5: This is the page which shows the result of the exam to the user. It shows the percentage of marks scored to the user and whether he/she is pass or fail in the test. There are two buttons to either go back to the dashboard or to logout of the system.

2.9. PERORMANCE REQUIREMENTS:

Since the system use NodeJS server client architecture, and use the large source information from distant server, it fetches data through internet. Internet bandwidth is major performance parameter. As System uses GitHub repository as source of massive data, multiple request and response is needed to handle in QuickTime for instant result back to user. Large system queries are handled by the fast-operating systems for both the mobile and the web-based process. Worker are used to provide faster HTTP Request handling.

2.9.1. Hardware Requirements:

The requests of the hardware for the web application are as followed:

- 64 bits laptop or desktop.
- TCP protocol.

2.9.2. Software Requirements:

To access a web portal of this application, its only need a PC/Laptop/Mobile with an integrated and updated web browser.

Desktop browser: Safari, Chrome, Firefox, Opera, IE9+.

Mobile browsers: Android, Chrome Mobile, iOS Safari.

On the server side: a PC/Web Server which meets these specifications:

- Window Operating System

- At least 2 GB RAM and 150 GB Free Space
- Redis Server Installed
- Python Compiler Installed

2.10. COMMUNICATION PROTOCOLS AND INTERFACES:

- A. TCP protocol.
- B. Secure transmission and encryption techniques.
- C. File transfer rate.

3. SOFTWARE DESIGN

Function Oriented Design for procedural approach and different diagram to show the designing of the application.

3.1. DATAFLOW DIAGRAM

3.1.1. DFD Level 0

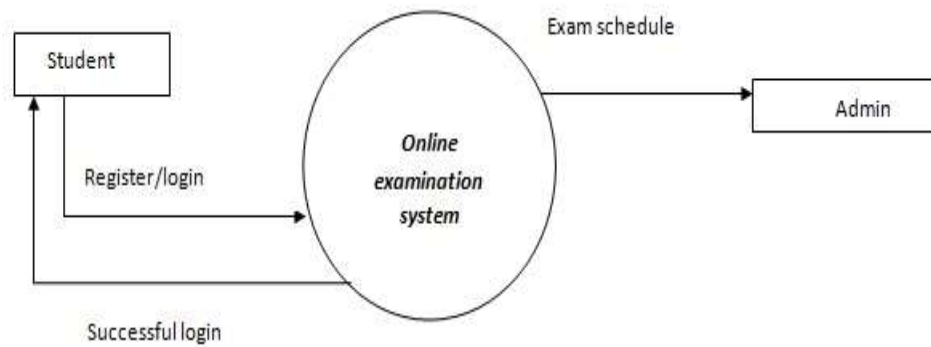


Figure 5 DFD level 0

3.1.2. DFD Level 1

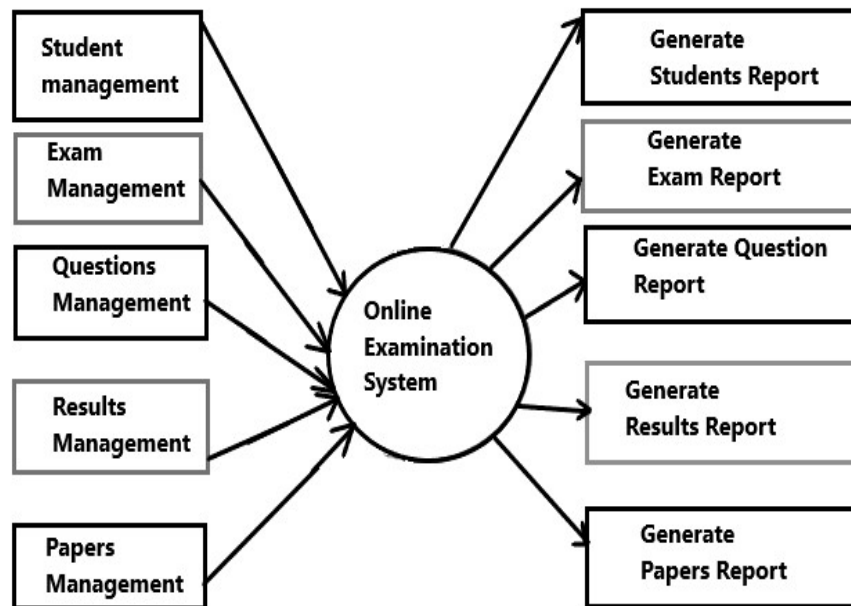


Figure 6 DFD level 1

3.1.3. DFD Level 2

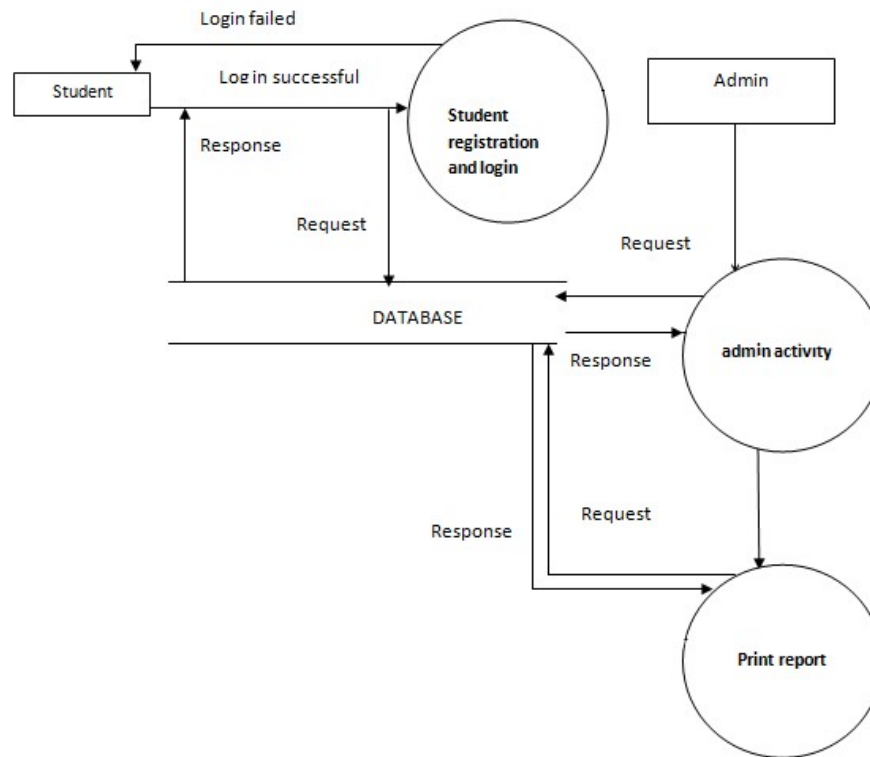


Figure 7 DFD level 2

3.2. UML DIAGRAMS

3.2.1. Sequence Diagram

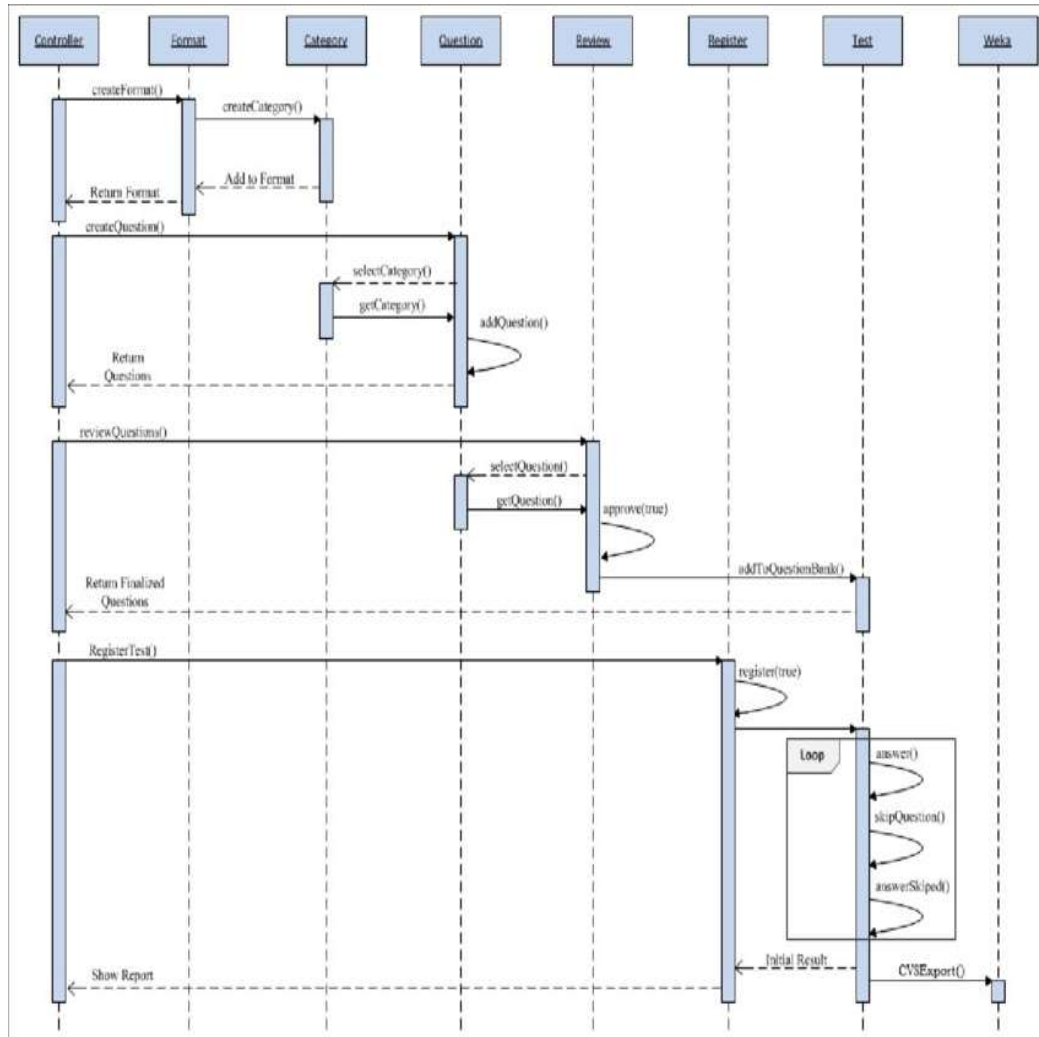


Figure 8 Sequence Diagram

3.2.2. Class Diagram

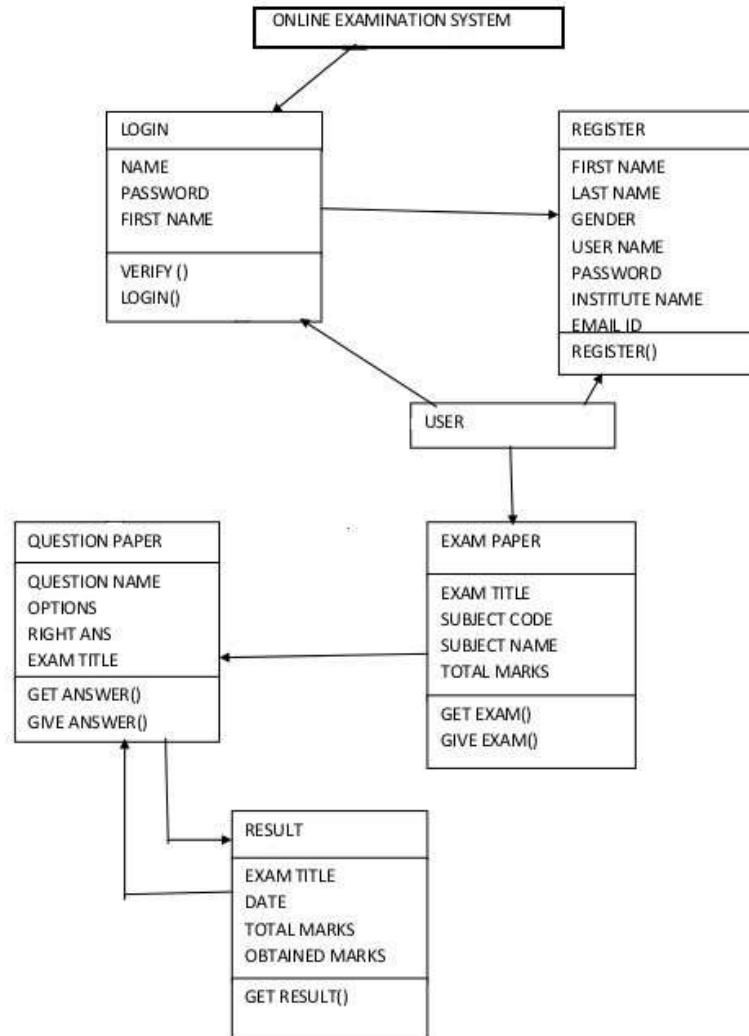


Figure 9 Class Diagram

3.2.3. UseCase Diagram



Figure 10 Use Case Diagram

4. TESTING

4.1. TEST PLAN

UNIT Testing: - Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.

Unit testing is often automated but it can also be done manually. A unit test is an automated piece of code that invokes a unit of work in the system and then checks a single assumption about the behavior of that unit of work.

In this application, a manually written unit test script method is used for testing function those perform unit amount of work and provides functionality.

5. IMPLEMENTATION AND USER INTERFACES

5.1. COMPONENTS/SCREENS

We have used total four components while developing the interfaces for our application which is described below:

5.3.1. Signup.js

This component is for displaying and managing the registration page as displayed to the user as soon as he tries to enter the application. It is a class component and we have use react state for validating the registration form.

```
src > Screens > SignUp > JS SignUps > ...
1  import React, { Component } from "react";
2
3  class SignUp extends Component {
4    constructor() {
5      super();
6      this.state = {
7        username: "",
8        email: "",
9        password: "",
10     };
11     this.signUpNow = this.signUpNow.bind(this);
12   }
13
14   signUpNow() {
15     let { toggleToSignIn } = this.props;
16     const { username, email, password } = this.state;
17     if (!email.match(/^[S+@S+\S+$/)) {
18       alert("please enter correct email");
19     } else if (!password.match(/(?=.*\d)(?=.*[a-z]).{8,}/)) {
20       alert(
21         "Please enter atleast 8 characters and contain atleast one character and one number"
22       );
23     } else {
24       let signUpObj = { username, email, password };
25       localStorage.setItem("userInfo", JSON.stringify(signUpObj));
26       toggleToSignIn(false);
27       console.log(signUpObj, "*****");
28     }
29   }
30 }
```

Figure 11 signUpNow() method

It has one user defined function named signUpNow (). This function is for form validation purpose and for storing the information of already registered user in the local storage.

```
let signUpObj = { username, email, password };  
localStorage.setItem("userInfo", JSON.stringify(signUpObj));
```

Here, we have stored the current state in respective variables and matched it with the regex pattern so that no one can enter incorrect email id or password which does not have specified characteristics like '@' in case of email and a special symbol or number in case of password. User cannot leave any of these fields blank. If he/she tries to do so then a alert box with a message to enter the value for the fields will be displayed.

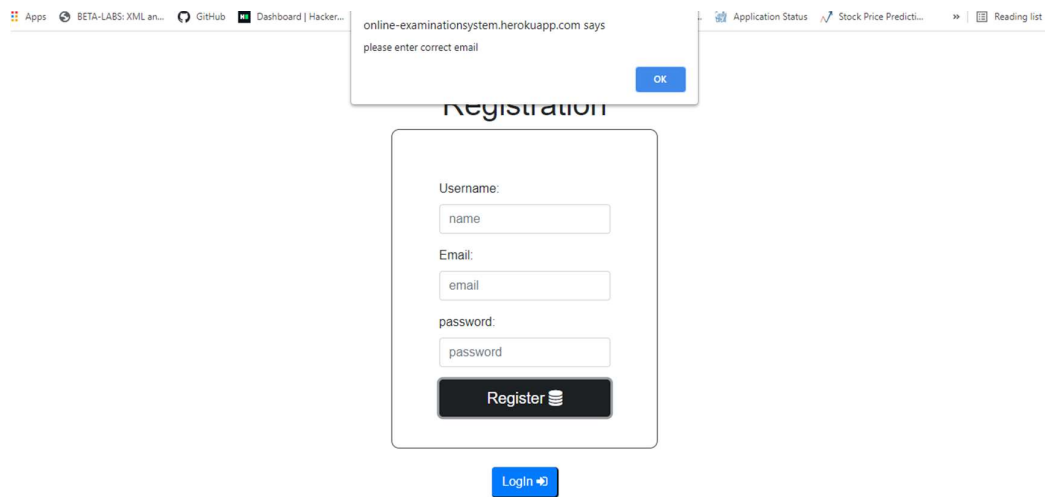


Figure 12 Registration page if entered Wrong email

Here is the code for render () method which is used for adding JSX components.

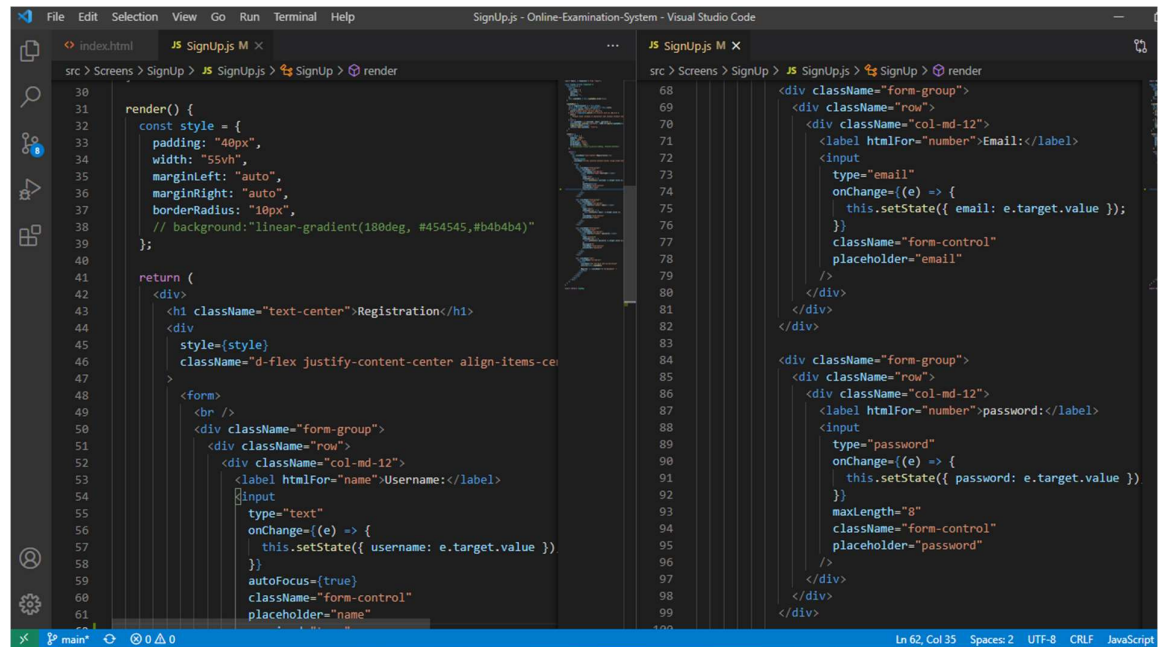


Figure 13 SignUp.js render() method

5.3.2. Login.js

After completing the registration process, we head up to login component. This is a login form having two fields, one for entering email and other for entering password previously set during the registration.

Similar to the signup.js component, it is also a class component with two parameters email and password and a method logInNow() to check the credentials the user has entered which is the basically the feature of a login page.

In the function logInNow () method, we have stored the current state in the respective variables and then checked them if they match with those already present in local storage or not.

```
if (email === userInfo.email && password === userInfo.password) {
  sessionStorage.setItem("userInfo", JSON.stringify(userInfo));
  this.props.changeUserState();
} else {
  console.log("enter correct details");
}
```

If the user tries to enter wrong information, then he/she will be prompted to enter it correctly and if one tries to leave the fields blank, he/she will be asked to enter it as all the fields are required.

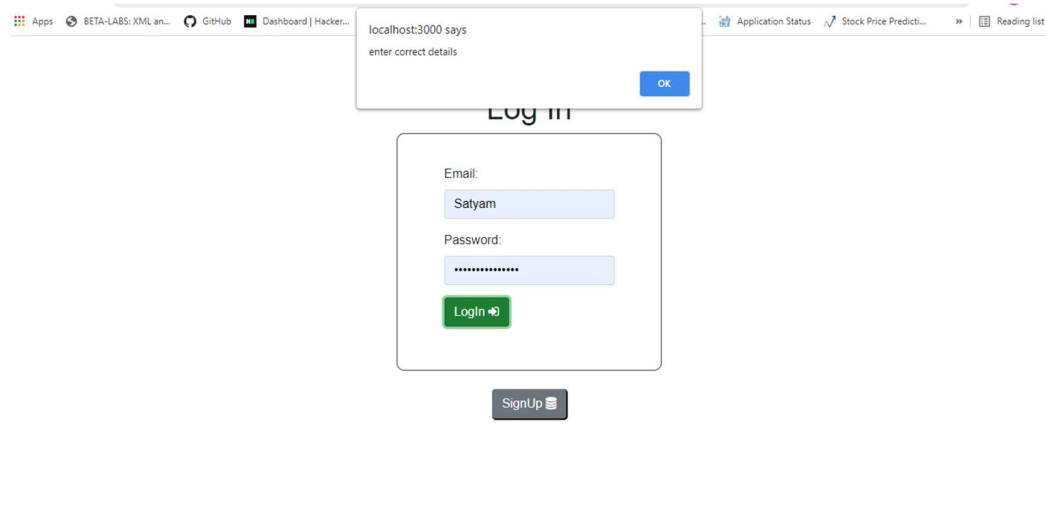


Figure 14 Login Page when entered wrong credentials

This is how the code looks like.

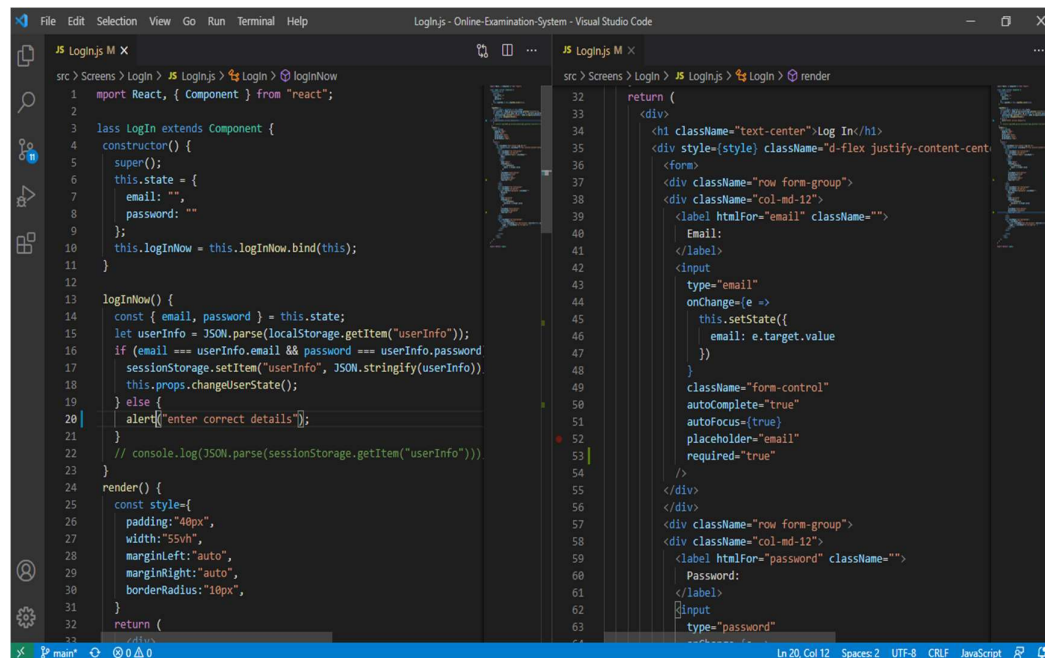


Figure 15 Login.js Code

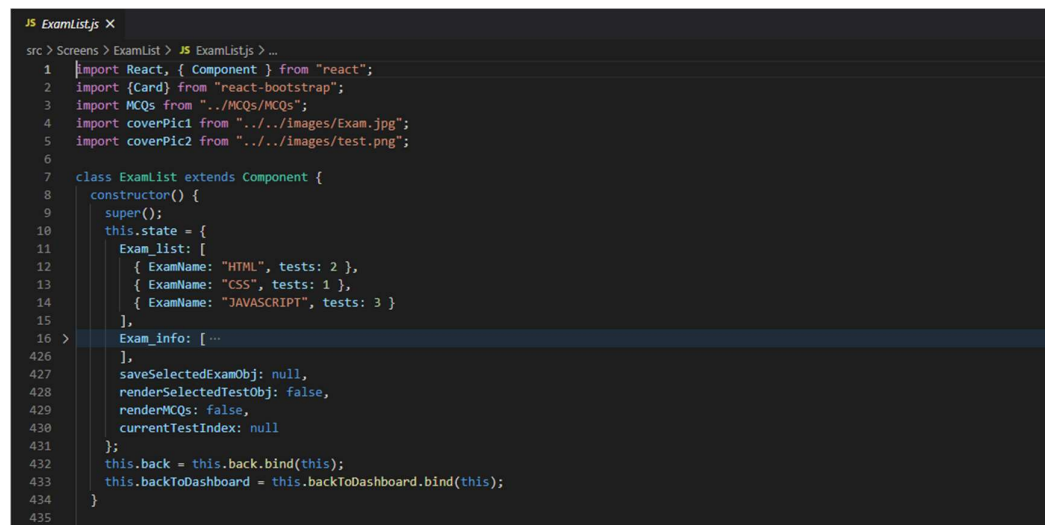
5.3.3. ExamList.js

This component is for displaying the thumbnails to enter the exam and to store the details of questions which would be displayed to the user. It is also a stateful component in which one variable is to store list of subjects to be displayed and other for storing the list of questions for each test. For now, we have given 5-5 questions for a particular test which could be increased later.

Exam_list is the variable which stores the list of subjects and Exam_info stores the list of questions. Here we have some helper methods which is to have a check on which exam will be shown to the user at a time.

renderExamInfo() is for rendering the list of subject to the user and renderExamList() method is for displaying the list of exams for a particular subject to the user which can be increased later.

This is how the code looks like for this much part



```

1  import React, { Component } from "react";
2  import {Card} from "react-bootstrap";
3  import MCQs from "../MCQs/MCQs";
4  import coverPic1 from "../../images/Exam.jpg";
5  import coverPic2 from "../../images/test.png";
6
7  class ExamList extends Component {
8    constructor() {
9      super();
10     this.state = {
11       Exam_list: [
12         { ExamName: "HTML", tests: 2 },
13         { ExamName: "CSS", tests: 1 },
14         { ExamName: "JAVASCRIPT", tests: 3 }
15       ],
16       Exam_info: [ ...
426     ],
427     saveSelectedExamObj: null,
428     renderSelectedTestObj: false,
429     renderMCQs: false,
430     currentTestIndex: null
431   };
432   this.back = this.back.bind(this);
433   this.backToDashboard = this.backToDashboard.bind(this);
434 }
435

```

Figure 16 ExamList.js state variables

```

JS ExamList.js X
src > Screens > ExamList > JS ExamList.js > ...
454 renderExamInfo() {
455   const { saveSelectedExamObj } = this.state;
456   return (
457     <div>
458       <h2>{saveSelectedExamObj.ExamName}</h2>
459
460       <div className="row">
461         {saveSelectedExamObj.tests.map((test, i) => {
462           return (
463             <div
464               className="col-md-4"
465               key={` ${saveSelectedExamObj.ExamName}_${test.name}`}
466             >
467               <Card style={{ width: '18rem' }}>
468                 <Card.Img variant="top" src={coverPic2} alt="Card image cap" />
469                 <Card.Body>
470                   <Card.Title>{test.name}</Card.Title>
471                   <Card.Text>
472                     <p className="card-text">
473                       Total Questions: {test.questions}
474                     </p>
475                     <p>Total Time: {test.time / 60} Minutes</p>
476                   </Card.Text>
477                 </Card.Body>
478                 <Card.Body>
479                   <button
480                     className="btn btn-success"
481                     onClick={() => {
482                       this.setState({
483                         renderMCQs: true,
484                         currentTestIndex: i,
485                         renderSelectedTestObj: false

```

Figure 17 renderExamInfo() method in ExamList.js

```

JS ExamList.js X
src > Screens > ExamList > JS ExamList.js > ...
506 renderExamList() {
507   const { Exam_list } = this.state;
508   return (
509     <div>
510       <h1 className="text-center">Dashboard</h1>
511
512       <div className="row">
513         {Exam_list.map((qList, index) => {
514           return (
515             <div className="col-md-4" key={` ${qList}_${index}`}>
516               <Card style={{ width: "18rem" }}>
517                 <Card.Img variant="top" src={coverPic1} alt="Card image cap"/>
518                 <Card.Body>
519                   <Card.Title>{qList.ExamName}</Card.Title>
520                   <Card.Text>
521                     Test your skills of {qList.ExamName} by taking this small
522                     Exam. It has {qList.tests} tests.
523                   </Card.Text>
524                   <button
525                     className="btn btn-warning"
526                     onClick={this.updateExamInfoState.bind(this, index)}
527                   >
528                     Go Ahead <i className="fa fa-paper-plane" />
529                   </button>
530                 </Card.Body>
531               </Card>
532             </div>
533           );
534         })}
535       </div>
536     </div>
537   );

```

Figure 18 renderExamList() in ExamList.js

5.3.4. MCQ.js

Till now we have only shown the list of subjects and exams to the user. Using this component, we are doing the actual work of showing the question and displaying the result to the user according to the answers given by the user. It is also a class Component like all other components.

```

JS MCQs.js X
src > Screens > MCQs > JS MCQs.js > ...
1  import React, { Component } from "react";
2
3  class MCQs extends Component {
4    static defaultProps = {
5      currentQuesObj: {},
6      currentTestIndex: 0
7    };
8    constructor(props) {
9      super(props);
10     const { currentQuesObj, currentTestIndex } = this.props;
11     this.state = {
12       question: currentQuesObj.tests[currentTestIndex].Exam_questions[0].Exam,
13       opt1: currentQuesObj.tests[currentTestIndex].Exam_questions[0].option1,
14       opt2: currentQuesObj.tests[currentTestIndex].Exam_questions[0].option2,
15       opt3: currentQuesObj.tests[currentTestIndex].Exam_questions[0].option3,
16       opt4: currentQuesObj.tests[currentTestIndex].Exam_questions[0].option4,
17       i: 0,
18       correct: 0,
19       score: 0,
20       min: null,
21       sec: null
22     };
23     this.minute = Math.ceil(currentQuesObj.tests[currentTestIndex].time / 60);
24     this.second = 1;
25     this.timeStart = null;
26     this.next = this.next.bind(this);
27     this.timer = this.timer.bind(this);
28   }
29

```

Figure 19 MCQ.js state variables

Here the next() method is managing the part where user click the button to answer the question and move to the next question. In the same method we are also calculating the result simultaneously and displaying it to the user once he/she finishes his/her exam.

```

30 next() {
31   const { currentQuesObj, currentTestIndex } = this.props;
32   var { i, correct, score } = this.state;
33
34   var Exam_questions = currentQuesObj.tests[currentTestIndex].Exam_questions;
35
36   var radioBtn = document.querySelector("input[name='option']:checked");
37   if (radioBtn == null) {
38     alert("select value");
39   } else {
40     if (Exam_questions[i].answer.match(radioBtn.value)) {
41       console.log("Exam_questions[i].answer**", Exam_questions[i].answer);
42       this.setState({ correct: ++correct });
43     }
44
45     if (Exam_questions.length - 1 === i) {
46       document.getElementById("ExamContainer").style.display = "none";
47       document.getElementById("resultContainer").style.display = "block";
48       score = correct * (100 / Exam_questions.length).toFixed(2);
49       console.log(Exam_questions.length);
50       this.setState({ score });
51     } else {
52       i++;
53       const question = Exam_questions[i].Exam;
54       const option1 = Exam_questions[i].option1;
55       const option2 = Exam_questions[i].option2;
56       const option3 = Exam_questions[i].option3;
57       const option4 = Exam_questions[i].option4;
58       const answer = Exam_questions[i].answer;
59       this.setState({
60         question,
61         opt1: option1,
62         opt2: option2,

```

Figure 20 MCQs.js next() method

timer() is for managing the maximum time which a user can take in order to answer the particular question.

If he exceeds the time limit, his exam will be finished at the exact moment and result will be calculated as soon as the time limit exceeds.

Code snippet for this method is here,

```

if (this.minute < 0) {
  clearInterval(this.timeStart);
  const { currentQuesObj, currentTestIndex } = this.props;
  var Exam_questions =
    currentQuesObj.tests[currentTestIndex].Exam_questions;
  var { score, correct } = this.state;
  this.setState({
    min: 0,
    sec: 0
  });
  score = correct * (100 / Exam_questions.length).toFixed(2);
  this.setState({
    score
  });
}

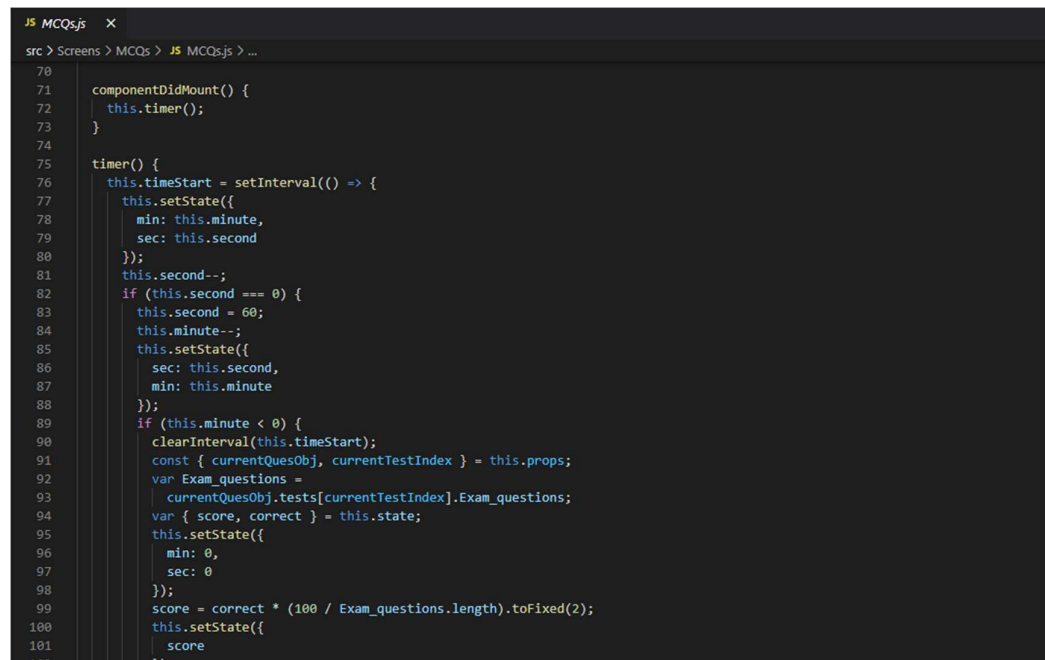
```



```

        document.getElementById("ExamContainer").style.display = "none";
        document.getElementById("resultContainer").style.display = "block";
    }

```



```

70
71 componentDidMount() {
72     this.timer();
73 }
74
75 timer() {
76     this.timeStart = setInterval(() => {
77         this.setState({
78             min: this.minute,
79             sec: this.second
80         });
81         this.second--;
82         if (this.second === 0) {
83             this.second = 60;
84             this.minute--;
85             this.setState({
86                 sec: this.second,
87                 min: this.minute
88             });
89             if (this.minute < 0) {
90                 clearInterval(this.timeStart);
91                 const { currentQuesObj, currentTestIndex } = this.props;
92                 var Exam_questions =
93                     currentQuesObj.tests[currentTestIndex].Exam_questions;
94                 var { score, correct } = this.state;
95                 this.setState({
96                     min: 0,
97                     sec: 0
98                 });
99                 score = correct * (100 / Exam_questions.length).toFixed(2);
100                 this.setState({
101                     score
102                 });

```

Figure 21 MCQ.js timer() method

render () method tis simply to display the question to the user in the form we wish it to be displayed to the user.

```

114 return (
115   <div>
116     <div className="col-md-12">
117       <div className="col" id="content">
118         <div id="ExamContainer">
119           <div className="modal-header">
120             <h5>
121               <i className="fa fa-question-circle" />
122               <span> </span>
123               <span className="label label-warning">{question
124             </h5>
125             <h5>
126               {min} : {sec}
127             </h5>
128           </div>
129           <div className="modal-body">
130             <div className="Exam" id="Exam" data-toggle="butt
131             <label className="btn btn-lg btn-info btn-block
132             <span className="btn-label">
133               <input type="radio" name="option" value="1"
134               <br />
135               <i className="fa fa-arrow-right" />
136             </span>
137             <span>{opt1}</span>
138             </label>
139             <label className="btn btn-lg btn-info btn-block
140             <span className="btn-label">
141               <input type="radio" name="option" value="2"
142               <br />
143               <i className="fa fa-arrow-right" />
144             </span>
145             <span>{opt2}</span>
146             </label>

```

```

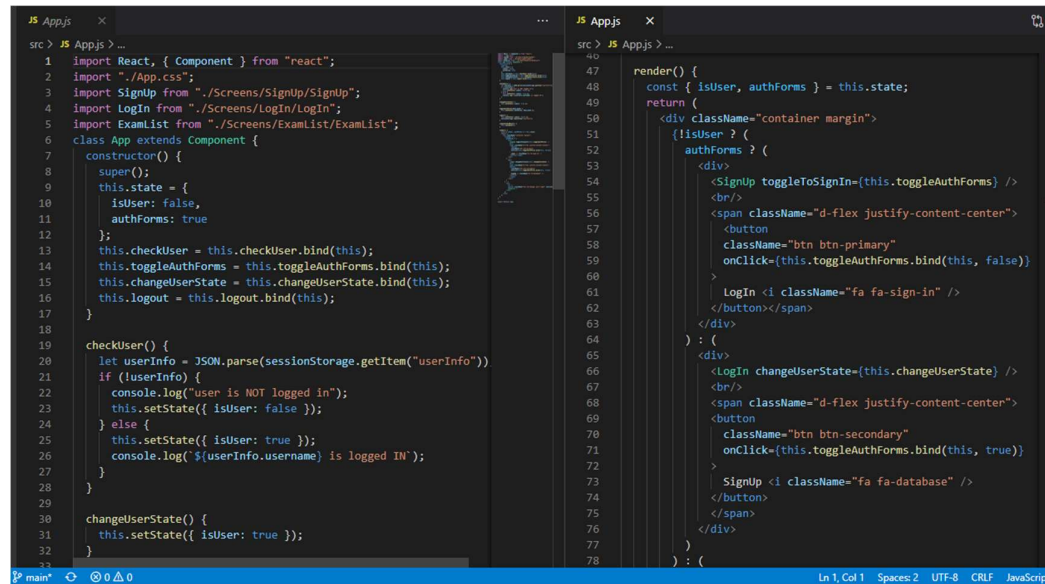
182   }
183   >
184   Goto Dashboard <i className="fa fa-undo" />
185   </button>
186   </div>
187   <div className="modal-body">
188     <h3>{currentQuesObj.tests[currentTestIndex].nam
189     <p>
190       Time:
191       {currentQuesObj.tests[currentTestIndex].time
192     </p>
193     <p>Questions: {currentQuesObj.tests[currentTest
194     {score < 70 ? {
195       <h3>You are fail with grades {score}%</h3>
196     } : {
197       <h3>You are pass with grades {score}%</h3>
198     }}
199     <hr />
200     <p className="badge badge-warning text-center">
201     <br />
202   </div>
203   </div>
204   </div>
205   </div>
206   </div>
207   </div>
208   </div>
209   </div>
210   </div>
211   </div>
212   export default MCQs;
213

```

Figure 22 MCQ.js render() method

5.2. App.js file

In this file we have rendered all the components described above. We can say that it is acting as a window or door for all the components to get displayed on the screen. It also has some other helper functions to check whether user is logged in or not etc.



```

1  import React, { Component } from "react";
2  import "../App.css";
3  import SignUp from "../Screens/SignUp/SignUp";
4  import Login from "../Screens/Login/Login";
5  import ExamList from "../Screens/ExamList/ExamList";
6  class App extends Component {
7    constructor() {
8      super();
9      this.state = {
10        isUser: false,
11        authForms: true
12      };
13      this.checkUser = this.checkUser.bind(this);
14      this.toggleAuthForms = this.toggleAuthForms.bind(this);
15      this.changeUserState = this.changeUserState.bind(this);
16      this.logout = this.logout.bind(this);
17    }
18
19    checkUser() {
20      let userInfo = JSON.parse(sessionStorage.getItem("userInfo"));
21      if (userInfo) {
22        console.log("user is NOT logged in");
23        this.setState({ isUser: false });
24      } else {
25        this.setState({ isUser: true });
26        console.log(`${userInfo.username} is logged IN`);
27      }
28    }
29
30    changeUserState() {
31      this.setState({ isUser: true });
32    }
33
34    render() {
35      const { isUser, authForms } = this.state;
36      return (
37        <div className="container margin">
38          {isUser ? (
39            authForms ? (
40              <div>
41                <SignUp toggleToSignIn={this.toggleAuthForms} />
42                <br/>
43                <span className="d-flex justify-content-center">
44                  <button
45                    className="btn btn-primary"
46                    onClick={this.toggleAuthForms.bind(this, false)}
47                  >
48                    Login <i className="fa fa-sign-in" />
49                  </button></span>
50              </div>
51            ) : (
52              <div>
53                <Login changeUserState={this.changeUserState} />
54                <br/>
55                <span className="d-flex justify-content-center">
56                  <button
57                    className="btn btn-secondary"
58                    onClick={this.toggleAuthForms.bind(this, true)}
59                  >
60                    SignUp <i className="fa fa-database" />
61                  </button>
62                </span>
63              </div>
64            )
65          ) : (
66            <ExamList />
67          )}
68        </div>
69      );
70    }
71  }
72  export default App;

```

Figure 23 App.js file

5.3. USER INTERFACES

5.3.1. Registration Page

Registration

Username:

Email:

password:

[Register](#)

[Login](#)

Figure 24 Registration Page UI

5.3.2. LOGIN PAGE

Log In

Email:

Password:

[Login](#) ➔


[SignUp](#) ➔

Figure 25 Login Page UI

5.3.3. Dashboard page

[Logout](#) ➔


Dashboard



HTML Exam

Test your skills of HTML by taking this small Exam. It has 2 tests.


[Go Ahead](#) ➔



CSS Exam

Test your skills of CSS by taking this small Exam. It has 1 tests.

[Go Ahead](#) ➔



JAVASCRIPT Exam

Test your skills of JAVASCRIPT by taking this small Exam. It has 3 tests.

[Go Ahead](#) ➔

Figure 26 Dashboard UI

5.3.4. Take Exam Page

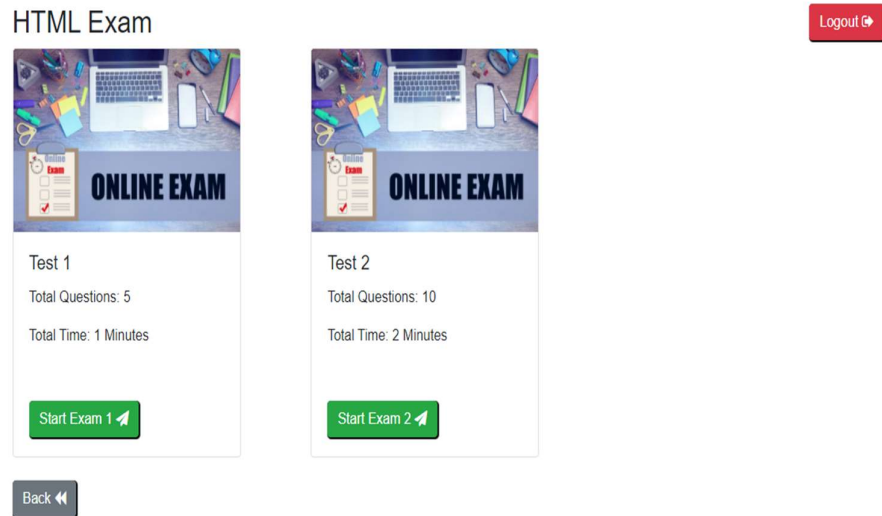


Figure 27 Exam Page UI

5.3.5. Questions Page

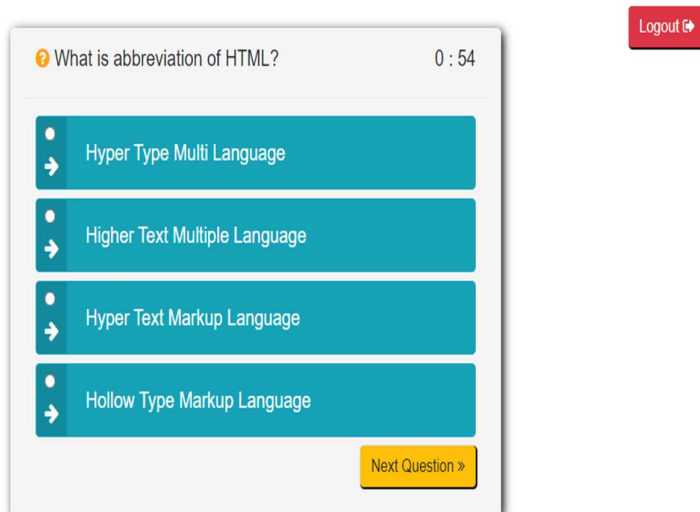


Figure 28 Question Panel UI

5.3.6. Results Page

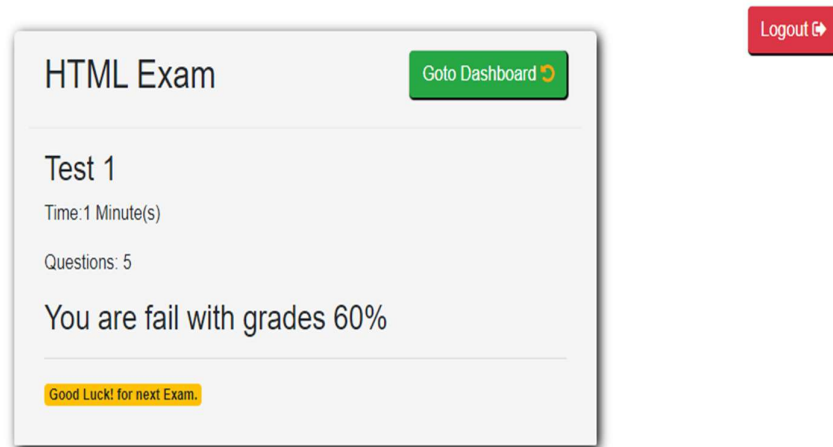


Figure 29 Result Page UI

5.4. DEPLOYMENT

The flask app is deployed on Heroku. Heroku is a cloud platform as a service (PaaS) supporting several programming languages. One of the first cloud platforms, Heroku has been in development since June 2007, when it supported only the Ruby programming language, but now supports Java, Node.js, Scala, Clojure, Python, PHP, and Go. For this reason, Heroku is said to be a polyglot platform as it has features for a developer to build, run and scale applications in a similar manner across most languages.

To deploy this, react app on Heroku. We had used the website of Heroku, after logging in to the website mentioned below, we proceeded with creating a new app and connecting it with our GitHub repository.

<https://dashboard.heroku.com/apps>

After this we proceeded with the user interface of Heroku platform and after its execution and connection process was finished, we deployed our application at

<https://online-examinationsystem.herokuapp.com/>

How to Run the app on local:

1. Open the Terminal.
2. Clone the repository by entering

```
$ git clone https://github.com/satyamjha1710/Online-Examination-System.git.
```

3. Ensure that NodeJS is installed on the system.
4. change the directory to repository name using

```
$ cd [Repository name].
```

5. Enter the cloned repository directory and execute

```
npm install
```

6. Now enter one command

```
npm install react-scripts
```

7. Now, execute the following command: `npm start` and it will point to the localhost server with the port 3000.
8. Enter the IP Address: `http://localhost:3000` on a web browser and use the application.

6. CONCLUSION AND FUTURE WORK

6.1. CONCLUSION

The main purpose of our project is to develop an application that offers new aspects of learning and improving knowledge in educational area. Most of the available apps are entertainment-based, which mostly do not contribute to the academic enhancement of the students. The theme of our application is to provide user to practice for understanding exams and information, so in this app we focus education and learning. This application is useful for enhancing knowledge among students.

This application some set of multiple-choice questions for a particular exam and that too for a particular subject, which the student/user has to answer in order to get his/her result. The result is being calculated in percentage according to the number of answers correctly given by the student/user. We have found that the data collection process i.e., collecting the questions and their answers is hard and time-consuming, but it can be managed by a team work. We hope that other developers will take advantage from our experience/from our development.

6.2. FUTURE WORK

We are planning to keep managing the project and improving it based on user feedback. Here is our to do list for future

- ❖ We will add some more categories in our app.
- ❖ We'll try to make it more user friendly than it is now.
- ❖ We'll try to improve its quality.
- ❖ We'll work on another feature in our app to add a module so that everybody attempts the exam at the same time.

7. REFERENCES

1. <https://reactjs.org/>
2. <https://nodejs.org/>
3. <https://expressjs.com/>
4. <https://www.npmjs.com/>
5. <https://www.heroku.com/>
6. <https://www.w3schools.com/nodejs/>
7. <https://www.tutorialspoint.com/>
8. <https://www.coursera.org/learn/javascript>
9. <https://www.coursera.org/specializations/web-design>

APPENDIX

Source Code –

- Online-Examination-System app:
<https://github.com/satyamjha1710/Online-Examination-System>
- Deployment Link:
<https://online-examinationsystem.herokuapp.com/>