A Thesis/Project/Dissertation Report

on

Examination Control System

Submitted in partial fulfillment of the
requirement for the award of the degree of

Bachelor of Technology in
Computer Science Engineering

GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
Mr. Vivek Anand M
Assistant Professor

Submitted By

Swapnil Tyagi
18SCSE1140050

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING /
DEPARTMENT OF COMPUTERAPPLICATION
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
DECEMBER-2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"Examination Control System"** in partial fulfillment of the requirements for the award of the Bachelors of Technology submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of **Mr. Vivek Anand M,** Assistant Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Swapnil Tyagi – 18SCSE1140050

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Vivek Anand M

Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of Swapnil Tyagi-18SCSE1140050 has been held on _____ and his/her work is recommended for the award of Bachelors of Technology.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December, 2021

Place: Greater Noida

Abstract

Problem Stated

In the era of Digitalization and post covid situation, many colleges as well as university opted for online examination options for safety and wellness of their students. These examinations are held by examination control system and all details of paper, student verification etc is handled by them. These systems help to collect all answer and paper data and stores them as per specific student's id. This system will be made using strong web framework and will provide full-fledged, smooth handling of all examination using automation.

Problem Solution

Our system will be equipped with all anti-cheating features such as camera, eye detection, screen pinning, tabs switch, etc. We will build full secure system without any trespass of any data.

Tools and Technology

Techs used are Visual Studio Code editor and languages used are HTML, CSS, Javascript and web framework to support cross platform app.

Result

It will be a complete working and cross platform responsive website to conduct exam securely and efficiently.

Conclusion

This app can be expanded in future to provide more accurate anti-cheating and also be more accessible in low latency areas.

Table of Contents

List of Figures

Acronyms

B.Tech.	Bachelor of Technology
M.Tech.	Master of Technology
BCA	Bachelor of Computer Applications
MCA	Master of Computer Applications
B.Sc. (CS)	Bachelor of Science in Computer Science
M.Sc. (CS)	Master of Science in Computer Science
SCSE	School of Computing Science and Engineering

CHAPTER-1

Introduction

Since long time many government authorities as well as many placement companies were using online platform to conduct exams. Such online exams help authorities to maintain timing and schedule of examination held easily all over country due to lack of space as well as availability of students to migrate to new places to give exams.

In 2020, due to covid, many government examinations such as JEE were postponed due to safety and precautions. With this epidemic, all people were in dire need to find solution to take exams without any taking any risk and also ensure all guidelines as well as rules are followed.

1.1 Formulation of Problem

Many systems introduced have some flaws such as non-supporting OS platform, weak internet issues, seamless conduct of exams etc. Uploading of Answers is also issue in some cases where student couldn't upload or write answers on online available space.

1.2 Tools and Technology Used

Basic structure will be made using HTML, CSS and Javascript. Web framework like Node.js and Express.js will used to support the backend and database of app. Visual Studio Code will used to create the whole app with coding of each and every module.

CHAPTER – 2

Literature Survey

An online exam system is less laborious and overhead intensive examination experience since it automated and digitized the examination administration experience. The online examination system is useful all over the educational and corporate sector. Use of Random Number Generator Algorithm to shuffle questions. The advancement in the field of Deep Learning can help in the better assessment and monitoring of the online exams. Many countries like India, USA, etc., are using online exam management systems for conducting exams.

The currently used online exam management systems have following shortcomings:

Malpractices and Suspicious Activities

Due to the absence of supervision during the exams, identification of the rule violators is difficult. Students refer to their books, course materials and search engines while giving exams.

Identification and Verification

While giving the exam the identification and verification of the correct student is not done right at that moment that is, either not done, or done after the exam and as a result any unauthorized person can give exam in place of the authorized candidate. Live video feed and identity proof before examination has to be done to verify the candidate’s authority.

Infrastructural Barriers

Continuous internet connection is required while giving the exam. If connectivity is lost then either student has to give the whole exam again or resumes from same point after logging in again. This causes distraction and disturbance as a result student loses focus. Many exam systems only allow giving of exam on computers with webcam so the students without computer or laptop either have to go to the cybercafé or other places with public computers or they cannot give
exams.

Use of Human Evaluators and Proctors

Many online exam systems are using human proctors for monitoring the activities of the students while giving exams through snapshots taken by their (students') webcam. Moreover, some hire humans to evaluate the theoretical responses of the students. This not only increases the cost of
conducting exams but also somewhere affects the speed and accuracy of evaluation.

CHAPTER – 3

Methodology and System analysis

3.1 ANALYSIS OF EXISTING SYSTEM

Throughout the system analysis, an in-depth, study of end-user information is conducted, for producing functional requirement of the proposed system. Data about the existing examination system is collected through several fact-finding techniques such as website visit and document review, at the beginning of this stage. The data collected facilities information required during detailed analysis. A study on the current system is performed based on the collected data. As a result, user requirement of the proposed system is determined. At the end of this stage, requirement specification is produced as deliverable.

3.2 THE EXISTING SYSTEM

The existing system happens to be a non-computerized operating system where all operations are done manually by the people carrying paper and to take down the weather data by analysing. Due to current system, more paper-work is needed and wasted as well as online system have various faults.

3.3 PROBLEMS OF EXISTING SYSTEM

Due to old system, it is very difficult to match real world problems. Most of the problems include:

- Mistakes are made in verification.

- The process of collecting examination answer is slow.

- It leads to lack of reliability of system going offline on real time.

- The record keeping system is poor.

- Unnecessary time is wasted conveying information through the ladder of authority. Management at times seeks to get a copy of the student data from and this may take a lot of time to obtain it.

- It causes reduction of production flow. These are the major problems facing the existing system and would be corrected with the help of the proposed system.

3.4. OBJECTIVES OF THE PROPOSED SYSTEM

The proposed system is developed to manage activities in fast student data access. It helps to get student data in real time. The system should cover the following functions in order to support the weather data process for achieving the objectives:

- To allow the update of student and exam data in real time.

- To provide interface that allows check and update.

- To prevent interface that shows wrong data.

- Tools that generate reports that can be used for decision making.

- A tool that allows the management to modify the student and exam information such as new date or rescheduling of examination as well as student data.

3.5 JUSTIFICATION FOR THE NEW SYSTEM

It is the purpose of the new system to address all the problems plaguing the present system. This system will do the analysing and storing of information either automatically or interactively. It will make use of REACT and MONGODB. This will be like this: a report is generated conforming to particular information needed by the management via the monitor. This will require the input of necessary data and record of student and examination details and then a report is generated. The proposed system will also have some other features such as:

1. Accuracy in handling of data

2. The volume of paper work will be greatly reduced.

3. Fast rate of operation as in making the ordered weather available and delivered on time.

4. Flexibility (i.e. it can be accessed at any time)

5. Easy way to back up or duplicating data in CD’s in case of data loss

6. Better storage and faster retrieval system.

7. Errors in the reports will be greatly minimized.

Feasibility Analysis

The concept of this project is to make it easy to conduct and handle examination in online meduim. The application will be a mobile application as well as desktop i.e cross platform due to the current atmosphere where it is common to use specified apps for most activities.

Overall App Feasibility

• Considering the major features

It is cross platform.

Safe to use.

• Benefits of the product or service

Information of student and exam is easily available.

• Redefined Concept:

Old App problems

- There are many apps and website already in market but most of them (approx. 50%) are outdated because of outdated database and not working anymore.
- Apps which are in market use SMS method and some use fixed database that nearly needs to update every time which is costly and slower to work on.

Our Concept

- We will use an API based database which will automatically update the data.
- App will be cross platform which means can run on any device.

CHAPTER – 4

Functionality

REQUIRED TOOLS

- **Firebase Database**

  Firebase is a toolset to "build, improve, and grow your app", and the tools it gives you cover a large portion of the services that developers would normally have to build themselves, but don't really want to build, because they'd rather be focusing on the app experience itself. This includes things like analytics, authentication, databases, configuration, file storage, push messaging, and the list goes on. The services are hosted in the cloud, and scale with little to no effort on the part of the developer.

- **Used language- Java, JavaScript**

  It can be combined with native code written in other programming languages, making it very flexible and a good choice for adding new features to existing applications.

- **Visual studio Code**

  A code editor for writing the code and making the web app.

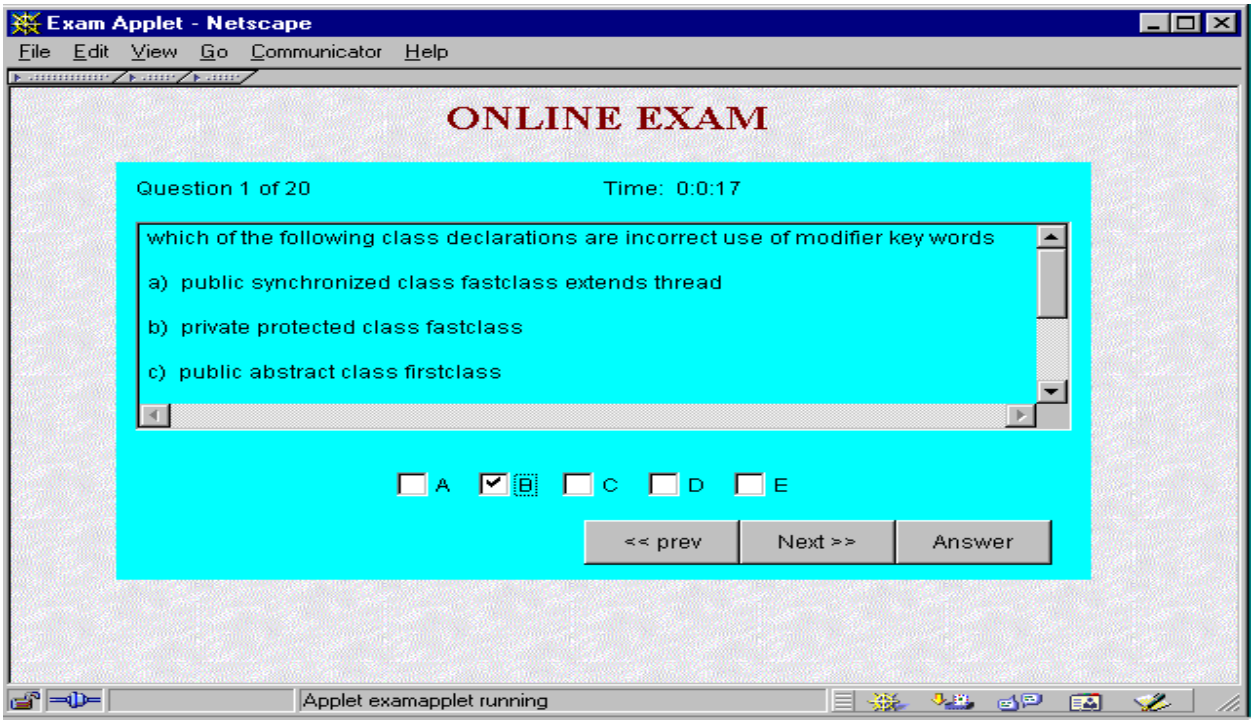
- **Android Studio**

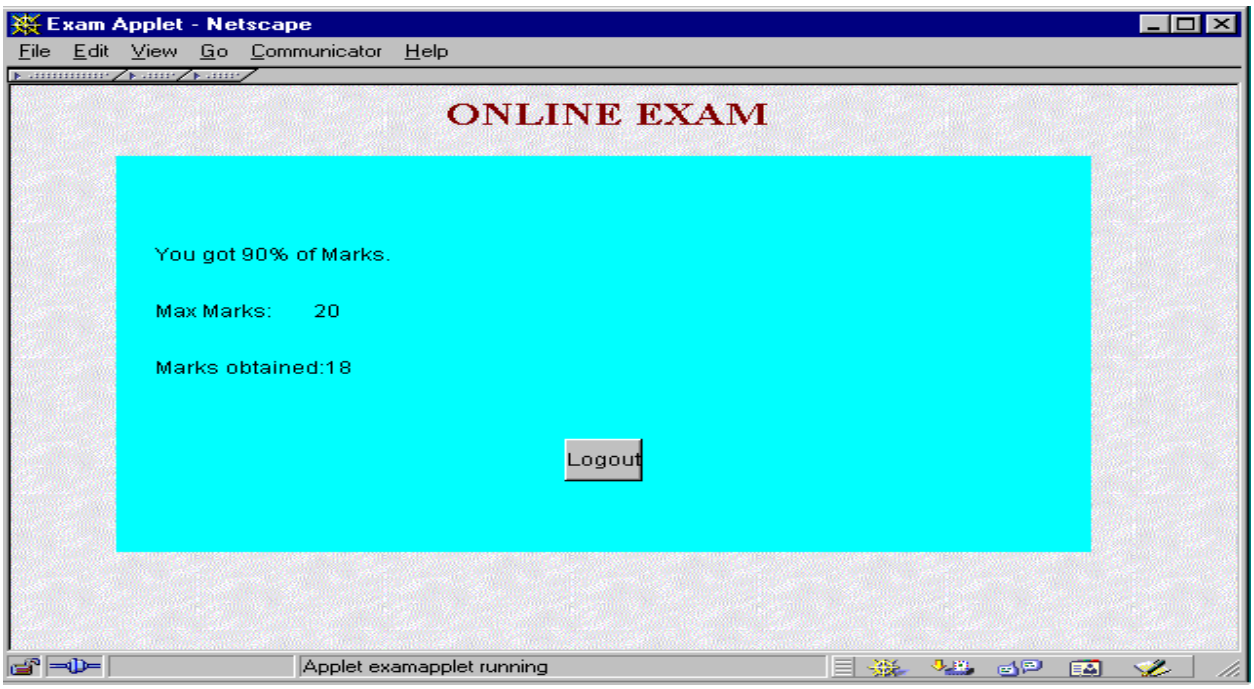
  It is widely used because it has below feature:

  - Gradle-based build support
  - Android-specific refactoring and quick fixes

- Lint tools to catch performance, usability, version compatibility and other problems
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Snippets of Design




Exam Applet - Netscape
File Edit View Go Communicator Help
ONLINE EXAM
You got 90% of Marks.
Max Marks: 20
Marks obtained:18
Logout
Applet examapplet running


ONLINE EXAM - Netscape
File Edit View Go Communicator Help
LOGIN ENTRY
LOGIN-ID
CHALA
PASSWORD
*****
LOGIN
BACK
Document: Done

CHAPTER – 5

System Requirements

SOFTWARE REQUIREMENTS SPECIFICATION

5.1 Hardware Requirements

Number Description

1 PC with 250 GB or more Hard disk.

2 PC with 2 GB RAM.

3 PC with Pentium 1 and above.

5.2 Software Requirements

Number Description Type

1 Operating System Windows XP / Windows

2 Language React, HTML, CSS, JS

3 Database MongoDB

4 IDE Visual Code

5 Browser Google Chrome

In this Section we will do Analysis of Technologies to use for implementing the project.

5.3 FRONT END

HTML

Hypertext Markup Language (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 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.

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 behaviour 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.

CSS

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML.CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.CSS is designed to enable the separation of presentation and content,

including layout, colours, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

CSS information can be provided from various sources. These sources can be the web browser, the user and the author. The information from the author can be further classified into inline, media type, importance, selector specificity, rule order, inheritance and property definition. CSS style information can be in a separate document or it can be embedded into an HTML document. Multiple style sheets can be imported. Different styles can be applied depending on the output device being used; for example, the screen version can be quite different from the printed version, so that authors can tailor the presentation appropriately for each medium. The style sheet with the highest priority controls the content display. Declarations not set in the highest priority source are passed on to a source of lower priority, such as the user agent style. The process is called cascading.

One of the goals of CSS is to allow users greater control over presentation. Someone who finds red italic headings difficult to read may apply a different style sheet. Depending on the browser and the web site, a user may choose from various style sheets provided by the designers, or may remove all added styles and view the site using the browser's default styling, or may override just the red italic heading style without altering other attributes.

JavaScript

JavaScript s a high-level, interpreted scripting language that conforms to the ECMA Script specification. JavaScript has curly-bracket syntax, dynamic typing, prototype-based object-orientation, and first-class functions. Alongside HTML and CSS, JavaScript is one of the core technologies of the World Wide Web. JavaScript enables interactive web pages and is an essential part of web applications. The vast majority of websites use it, and major web browsers have a dedicated JavaScript engine to execute it.

As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative (including object-oriented and prototype-based) programming styles. It has APIs for working with text, arrays, dates, regular expressions, and the DOM, but the language itself does not include any I/O, such as networking, storage, or graphics facilities. It relies upon the host environment in which it is embedded to provide these features.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases, and in non-web programs such as word processors and PDF software, and in runtime environments that make JavaScript available for writing mobile and desktop applications, including desktop widgets. The terms Vanilla JavaScript and Vanilla JS refer to JavaScript not extended by any frameworks or additional libraries. Scripts written in Vanilla JS are plain JavaScript code. Google's Chrome extensions, Opera's extensions, Apple's Safari 5 extensions, Apple's Dashboard Widgets, Microsoft's Gadgets, Yahoo! Widgets, Google Desktop Gadgets are implemented using JavaScript.

React JS

React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It's 'V' in MVC. React JS is an open-source, component-based front-end library responsible only for the view layer of the application. It is maintained by Facebook. React uses a declarative paradigm that makes it easier to reason about your application and aims to be both efficient and flexible. It designs simple views for each state in your application, and react will efficiently update and render just the right component when your data changes. The declarative view makes your code more predictable and easier to debug.

Features of React.js

There are unique features are available on React because that it is widely popular.

- **Use JSX:** It is faster than normal JavaScript as it performs optimizations while translating to regular JavaScript. It makes it easier for us to create templates.

- **Virtual DOM:** Virtual DOM exists which is like a lightweight copy of the actual DOM. So, for every object that exists in the original DOM, there is an object for that in React Virtual DOM. It is exactly the same, but it does not have the power to directly change the layout of the document. Manipulating DOM is slow, but manipulating Virtual DOM is fast as nothing gets drawn on the screen.

- **One-way Data Binding:** This feature gives you better control over your application.

- **Component:** A Component is one of the core building blocks of React. In other words, we can say that every application you will develop in React will be made up of pieces called components. Components make the task of

building UIs much easier. You can see a UI broken down into multiple individual pieces called components and work on them independently and merge them all in a parent component which will be your final UI.

- **Performance:** React.js use JSX, which is faster compared to normal JavaScript and HTML. Virtual DOM is a less time taking procedure to update webpages content.

BOOTSTRAP

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

Why we use Bootstrap?

- It is Faster and Easier way for Web-Development.
- It creates Platform-independent web-pages.
- It creates Responsive Web-pages.
- It designs the responsive web pages for mobile devices too.
- It is Free and open-source framework available on www.getbootstrap.com

GIT

Git is a free open-source **distributed version control system** designed to handle everything from small to very large projects with speed and efficiency.

Git relies on the **basis of distributed development** of a software where more than one developer may have access to the source code of a specific application and can modify changes to it which may be seen by other developers.

Every git working directory is **a full-fledged repository** with complete history and full version-tracking capabilities, independent of network access or a central server.

Git **allows a team of people to work together**, all using the same files. And it helps **the team cope up with the confusion** that tends to happen **when multiple people are editing the same files**.

Characteristics of Git

1. **Strong support for non-linear development**
2. Git supports rapid branching and merging, and includes specific tools for visualizing and navigating a non-linear development history.
3. A major assumption in Git is that a change will be merged more often than it is written.
4. **Branches** in Git **are very lightweight**.

Distributed development

1. Git **provides** each developer **a local copy** of the entire development history, and changes are copied from one such repository to another.

2. The changes can be merged in the same way as a locally developed branch very efficiently and effectively.

Compatibility with existing systems/protocol Git has a CVS server emulation, which enables the use of existing CVS clients and IDE plugins to access Git repositories.

Github

GitHub is a code hosting platform for version control and collaboration. It lets you and others work together on projects from anywhere. This tutorial teaches you GitHub essentials like repositories, branches, commits, and pull requests.

CHAPTER – 6

Modules

App module is divided in two parts:

1) Database

2) Functionality

Database Module is implemented using Firebase and firestore database storage. They provide both online and offline handling of data.

Functionality is implemented using javascript and connectivity with google service API.

Working Explanation

Languages Used

HTML

HTML stands for Hyper Text Markup Language. HTML is the standard markup language for creating Web pages. It describes the structure of a Web page and consists of a series of elements. HTML elements tell the browser how to display the content. Elements also label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.

Example:

```
<!DOCTYPE html>
<html>
      <head>
```

```
        <title>Page Title</title>
    </head>
    <body>
        <h1>My First Heading</h1>
        <p>My first paragraph. </p>
    </body>
</html>
```

CSS

CSS stands for Cascading Style Sheets. CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once. External stylesheets are stored in CSS files with extension .css.

JavaScript

JavaScript often abbreviated JS, is a programming language that is one of the core technologies of the World Wide Web, alongside HTML and CSS. Over 97% of websites use JavaScript on the client side for web page behavior, often incorporating third-party libraries. All major web browsers have a dedicated JavaScript engine to execute the code on the user's device.

API

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

CONCEPTS

What is PWA?

PWA stands for Progressive Web App. PWA are web applications that have been designed so they are capable, reliable, and installable. These three pillars transform them into an experience that feels like a platform-specific application. They have both capabilities of website and an app.

Database

Firebase’s firestore database with real time updating is used to display information. Also cached database for local assistance when user goes offline is also used to showcase database. It is not required to store any database while using this app.

What are service workers?

A service worker is a type of web worker. It's essentially a JavaScript file that runs separately from the main browser thread, intercepting network requests, caching or retrieving resources from the cache, and delivering push messages.

CHAPTER – 7

Source Code

Student.js

```
const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const Student = require("../model/Student");
const Test = require("../model/Test");
const User = require("../model/User");
require("dotenv").config();

/**
 * @method - GET
 * @param - /profile/:profileID
 * @description - Fetch student profile using profileID
 */

router.get("/profile/:profileID", auth, async (req, res) => {
  const profileID = req.params.profileID;

  try {
    await Student.findOne({
      _id: profileID,
    })
      .populate("profileInfo")
      .exec(function (err, obj) {
        if (err) {
          return res.status(400).json({ err });
        } else {
          return res.status(200).json({
            obj,
          });
        }
      });
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in fetching Student Data");
  }
});

/**
 * @method - GET
 * @param - /tests/:studentClass
 * @description - Fetch all the tests that student class assigned
 */

router.get("/tests/:studentClass", auth, async (req, res) => {
  const studentClass = req.params.studentClass;

  try {
    await Test.find(
      {
```

```
        className: studentClass,
      },
      "-assignedTo -submitBy -teacherId -__v"
    ).exec(function (err, obj) {
      if (err) {
        return res.status(400).json({ err });
      } else {
        return res.status(200).json({
          obj,
        });
      }
    });
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in fetching Test Data");
  }
});

/**
 * @method - GET
 * @param - /attempt-tests/:studentID
 * @description - Fetch all attempted tests of student
 */

router.get("/attempt-tests/:studentID", auth, async (req, res) => {
  const studentID = req.params.studentID;

  try {
    await Student.find({
      _id: studentID,
    }).exec(function (err, obj) {
      if (err) {
        return res.status(400).json({ err });
      } else {
        return res.status(200).json({
          obj,
        });
      }
    });
  } catch (err) {
    console.log(err.message);
    res.status(500).send(err);
  }
});

/**
 * @method - POST
 * @param - /results/:studentID
 * @description - Fetch student results using studentID
 */

router.post("/results/:studentID", auth, async (req, res) => {
  const studentID = req.params.studentID;
  const { testID } = req.body;

  try {
```

```
    await Test.find(
      {
        _id: testID,
      },
      "submitBy -_id",
      function (err, obj) {
        if (err) {
          return res.status(400).json({ err });
        } else {
          const result = obj[0].submitBy.filter((student, index) => {
            return student.id === studentID;
          });

          return res.status(200).json({
            result,
          });
        }
      }
    );
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in fetching Test Data");
  }
});

/**
 * @method - PUT
 * @param - /update-profile/:profileID
 * @description - Update student profile using profileID
 */

router.put("/update-profile/:profileID", auth, async (req, res) => {
  const profileID = req.params.profileID;
  const {
    firstName,
    lastName,
    email,
    password,
    phone,
    className,
    section,
  } = req.body;
  try {
    const testData = await User.findOneAndUpdate(
      { _id: profileID },
      { ...req.body },
      function (err, updatedData) {
        if (err) {
          return res.status(400).json({ message: "failed to update profile" });
        } else {
          console.log(updatedData);
          return res.status(200).json({
            message: "profile succesfully updated",
          });
        }
      }
```

```
    );
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in Updating Profile");
  }
});

/**
 * @method - PUT
 * @param - /submit-test/:testID
 * @description - Submit particular test using testID
 */

router.put("/submit-test/:testID", auth, async (req, res) => {
  const testID = req.params.testID;
  const submittedData = req.body.submitBy;
  const testName = req.body.testName;
  const date = Date.now();

  try {
    await Test.updateOne(
      { _id: testID },
      {
        $addToSet: { submitBy: [...submittedData] },
        attempted: true,
      },
      async function (err, updatedData) {
        if (err) {
          return res.status(400).json({ message: "failed to submit test" });
        } else {
          await Student.updateOne(
            { _id: submittedData[0].profileID },
            {
              $addToSet: {
                attemptedTest: [{ testName, date, ...submittedData }],
              },
            }
          );
          return res.status(200).json({
            message: "test submitted succesfully",
          });
        }
      }
    );
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in submitting test data");
  }
});

/**
 * @method - PUT
 * @param - /update-test-status/:testID
 * @description - Tracking how much time user spented on test using attemptedTime
 */
```

```
router.put("/update-test-status/:testID", auth, async (req, res) => {
  const testID = req.params.testID;
  const profileID = req.body.profileID;
  const testName = req.body.testName;
  const completed = req.body.completed;
  const attemptedTime = req.body.attemptedTime;
  const totalTime = req.body.totalTime;
  //console.log(...req.body);
  console.log(testID, profileID, testName, completed, attemptedTime, totalTime);
  if (testID) {
    try {
      let studentData = await Student.findById(profileID);

      let { testStatus } = studentData;
      let test = testStatus.filter((t) => t.testID === testID);
      if (test.length < 1) {
        studentData.testStatus.push({
          testID,
          attemptedTime,
          testName,
          completed,
          totalTime,
        });
        studentData.save();
        return res.status(200).json({
          studentData,
        });
      } else {
        await Student.findOneAndUpdate(
          { _id: profileID, "testStatus.testID": testID },
          {
            $set: {
              "testStatus.$.attemptedTime": attemptedTime,
            },
          },
          { new: true },
          (err, obj) => {
            return res.status(200).json({
              obj,
            });
          }
        );
      }
    } catch (err) {
      console.log(err.message);
      res.status(500).send("Error in updating test data");
    }
  } else {
    res.status(500).send("Undefined Test ID");
  }
});

module.exports = router;
```

Teacher.js

```
const express = require("express");
const router = express.Router();
const auth = require("../middleware/auth");
const Test = require("../model/Test");
const Teacher = require("../model/Teacher");
const User = require("../model/User");
require("dotenv").config();

/**
 * @method - GET
 * @param - /tests
 * @description - Fetching All the tests that teacher assigned using testID
 */

router.get("/tests/:profileID", auth, async (req, res) => {
  const profileID = req.params.profileID;
  console.log("teacher", profileID);
  try {
    await Test.find(
      {
        teacherId: profileID,
      },
      "submitBy className testName"
    ).exec(function (err, obj) {
      if (err) {
        return res.status(400).json({ err });
      } else {
        return res.status(200).json({
          obj,
        });
      }
    });
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in fetching Tests");
  }
});

/**
 * @method - GET
 * @param - /classes
 * @description - Fetching All the classes which are registered in Database
 */

router.get("/classes", auth, async (req, res) => {
  console.log("fetch classes");
  try {
    await User.find({}, "className -_id", function (err, obj) {
      if (err) {
        return res.status(400).json({ err });
      } else {
        return res.status(200).json({
```

```
          obj,
        });
      }
    });
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in fetching Tests");
  }
});

/**
 * @method - GET
 * @param - /profile/:profileID
 * @description - Fetching Teacher Profile from database
 */

router.get("/profile/:profileID", auth, async (req, res) => {
  const profileID = req.params.profileID;

  try {
    await Teacher.findOne({
      _id: profileID,
    })
      .populate("profileInfo")
      .exec(function (err, obj) {
        if (err) {
          return res.status(400).json({ err });
        } else {
          return res.status(200).json({
            obj,
          });
        }
      });
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in fetching Student Data");
  }
});

/**
 * @method - POST
 * @param - /create-test
 * @description - Creating Test for the students using teacherID
 */

router.post("/create-test", auth, async (req, res) => {
  const {
    teacherId,
    testName,
    category,
    minutes,
    rules,
    className,
    outOfMarks,
    answers,
    questions,
```

```
  } = req.body;
  console.log(questions, answers, rules);
  try {
    let createTest = await Test.findOne({
      testName,
      className,
      category,
    });
    if (createTest) {
      return res.status(400).json({
        msg: "Test Already Created",
      });
    }

    createTest = new Test({
      teacherId,
      testName,
      category,
      answers,
      minutes,
      className,
      rules,
      outOfMarks,
      questions,
    });

    let data = await createTest.save();

    const payload = {
      data,
    };

    res.status(200).json({
      payload,
    });
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in Saving");
  }
});

/**
 * @method - PUT
 * @param - /update-test/:testid
 * @description - Updating Test using testID
 */

router.put("/update-test/:testid", auth, async (req, res) => {
  const testID = req.params.testid;
  console.log(testID);
  const questionsData = req.body.questions;
  try {
    const testData = await Test.findOneAndUpdate(
      { _id: testID },
      { questions: questionsData },
      function (err, updatedData) {
```

```
        if (err) {
          return res.status(400).json({ message: "failed to update document" });
        } else {
          return res.status(200).json({
            message: "questions succesfully updated",
          });
        }
      }
    );
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in Updating");
  }
});

/**
 * @method - PUT
 * @param - /update-profile/:profileID
 * @description - Updating Teacher profile using profileID
 */

router.put("/update-profile/:profileID", auth, async (req, res) => {
  const profileID = req.params.profileID;
  const { firstName, lastName, email, password, phone } = req.body;
  try {
    const testData = await Teacher.findOneAndUpdate(
      { _id: profileID },
      { ...req.body },
      function (err, updatedData) {
        if (err) {
          return res.status(400).json({ message: "failed to update profile" });
        } else {
          console.log(updatedData);
          return res.status(200).json({
            message: "profile succesfully updated",
          });
        }
      }
    );
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in Updating Profile");
  }
});

/**
 * @method - PUT
 * @param - /assigend-to/:testID
 * @description - Fetching classes to which the test assigned using testID
 */

router.put("/assigend-to/:testID", auth, async (req, res) => {
  const testID = req.params.testID;
  const { className } = req.body;
  try {
    await Test.updateOne(
```

```
      { _id: testID },
      {
        $addToSet: { assignedTo: [...className] },
      },
      function (err, updatedData) {
        if (err) {
          return res
            .status(400)
            .json({ message: "failed to update assigendStudents" });
        } else {
          return res.status(200).json({
            updatedData,
          });
        }
      }
    );
  } catch (err) {
    res.status(500).send("Error in Updating");
  }
});

/**
 * @method - DELETE
 * @param - /delete-test/:testid
 * @description - Delete a particular test using testID
 */

router.delete("/delete-test/:testid", auth, async (req, res) => {
  const testID = req.params.testid;
  console.log(testID);
  try {
    const testData = await Test.findByIdAndDelete(testID, function (err) {
      if (err) {
        return res.status(400).json({ message: "failed to delete document" });
      } else {
        return res.status(200).json({
          message: "successfully deleted",
        });
      }
    });
  } catch (err) {
    console.log(err.message);
    res.status(500).send("Error in Deleting");
  }
});

module.exports = router;
```

CHAPTER 8
APPLICATION DESIGN / IMLEMENTATION

8.1 DESIGN STANDARD

The system is designed with several interaction cues on each web page that makes up the web application. These cues are well-defined such as to make several functionalities that the application exposes to collect, process and output data. Access to these functionalities is made possible by the well-designed user interface which embodies several technologies such as AJAX (Asynchronous JavaScript and XML) to process data. The application is built in a modular form where these functionalities are built into modules.

8.2 OUTPUT SPECIFICATION

The system is designed in such a way that it efficiently provides output to the user promptly and in a well-organized manner. The format for the several outputs is make available on the output web pages. Output can be relayed using the following page modules:

1. **Product_list.jsx**: This display output information for the list of weather delicacies which are currently available.
2. **Search_result.jsx**: This displays output information for the order report.
3. **Home_page.jsx**: This displays output information for the home page items details.

8.3 INPUT SPECIFICATION.

The system is designed to accept several input details efficiently through input forms and user clicks. The data captured through the user keystrokes and clicks are received by specific modules on the system and relayed to the back-end of the system for processing.

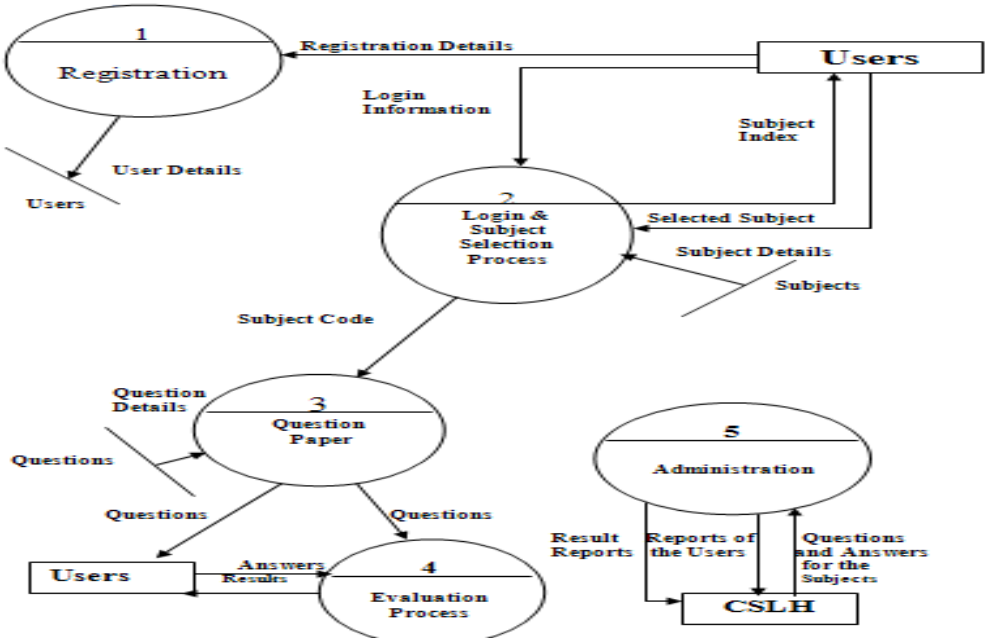
8.4 DATABASE SPECIFICATION

The database system used to implement the back-end of the system is MongoDB. Access to the system was made possible by a graphical interface with an ISAM engine. The database name is weather and the structure of the database are as follows:

1. User
2. Student
3. Teacher

DFD -1

1-level DFD: In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.



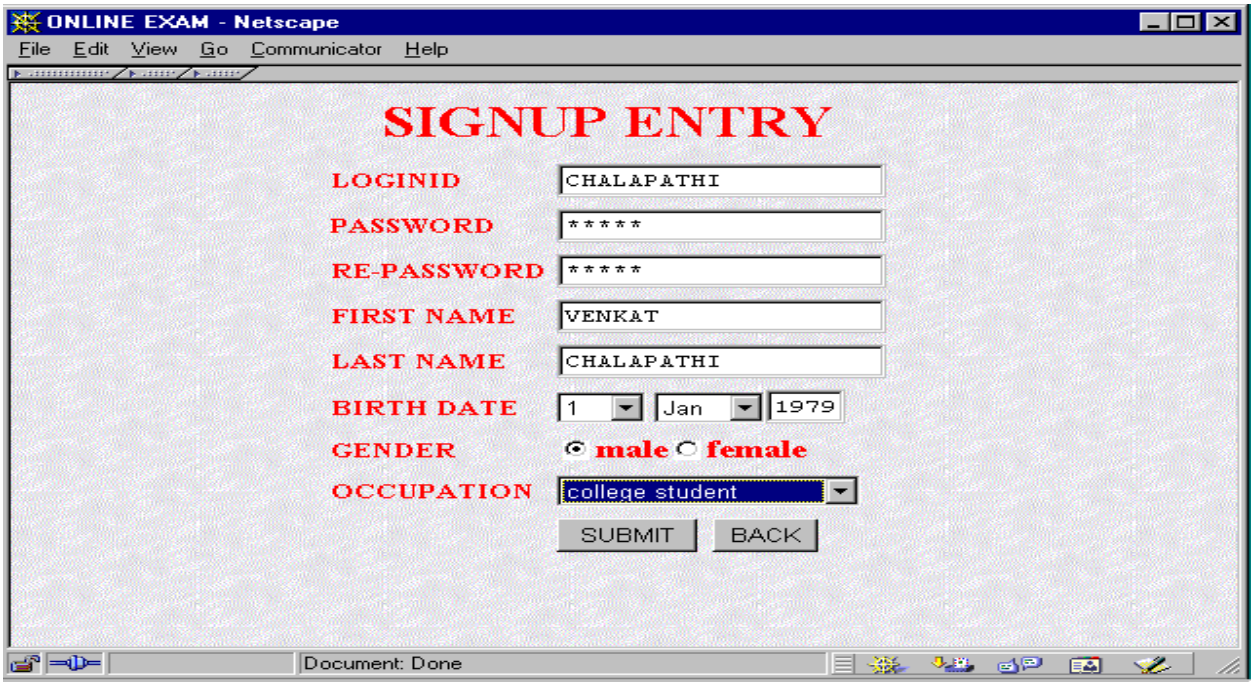
8.5 ER Diagram

An Entity may be an object with a physical existence – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course. An Entity is an object of Entity Type and set of all entities is called as entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as: Below shown are the ER diagrams that are used to construct this application Fig-1. This above simulation flow is with respect to customer point of view. And the restaurant manager or staff can keep on track of the orders by viewing the database or by the notification.

8.6 Application

The application starts by displaying the Home Page with a dashboard of delivery, dine out and Nightlife. When the user can navigate or can search for any data. If the user is ordering for first time i.e. then, he/she has to first ‘Register’ and then they can start viewing the deals. Else, if it’s not their first time then they have to ‘Login’

with all the credentials such as filling his/her first name, last name, phone number, Email Id, address and password. He/she has to choose their favorite dishes from the menu, then place their favorite dishes in the weather cart, this weather cart will help them to customize the orders like increasing the quantity, removing the weather items etc. Once he/she is done customizing their orders, they can checkout and will be redirected to the final order page including their personal details, their orders, total amount to be paid with appropriate payment method. Lastly, they can just pay the amount by selecting the payment method of their choice and simply log-out.



CHAPTER – 9

TESTING AND IMPLEMENTATION

The term implementation has different meanings ranging from the conversation of a basic application to a complete replacement of a computer system. The procedures however, are virtually the same. Implementation includes all those activities that take place to convert from old system to new. The new system may be totally new replacing an existing manual or automated system or it may be major modification to an existing system. The method of implementation and time scale to be adopted is found out initially. Proper implementation is essential to provide a reliable system to meet organization requirement.

9.1: UNIT TESTING

Introduction

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but it is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. It forms the basis for component testing. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

Benefits

The goal of unit testing is to isolate each part of the program and show that the individual parts are correct. A unit test provides a strict, written contract that the piece of code must satisfy. As a result, it affords several benefits.

1) **Find problems early:** Unit testing finds problems early in the development cycle. In test-driven development (TDD), which is frequently used in both extreme programming and scrum, unit tests are created before the code itself is written. When the tests pass, that code is considered complete. The same unit tests are run against that function frequently as the larger code base is developed either as the code is changed or via an automated process with the build. If the unit tests fail, it is considered to be a bug either in the changed code or the tests themselves. The unit tests then allow the location of the fault or failure to be easily traced. Since the unit tests alert the development team of the problem before handing the code off to testers or clients, it is still early in the development process.

2) **Facilitates Change:** Unit testing allows the programmer to refactor code or upgrade system libraries at a later date, and make sure the module still works correctly (e.g., in regression testing). The procedure is to write test cases for all functions and methods so that whenever a change causes a fault, it can be quickly identified. Unit tests detect changes which may break a design contract.

3) **Simplifies Integration:** Unit testing may reduce uncertainty in the units themselves and can be used in a bottom-up testing style approach. By testing the parts of a program first and then testing the sum of its parts, integration testing becomes much easier.

4) **Documentation:** Unit testing provides a sort of living documentation of the system. Developers looking to learn what functionality is provided by a unit, and how to use it, can look at the unit tests to gain a basic understanding of the unit's interface (API). Unit test cases embody characteristics that are critical to the success of the unit. These characteristics can indicate appropriate/inappropriate use of a unit as well as negative behaviours that are to be trapped by the unit. A unit test case, in and of itself, documents these critical characteristics, although many software development environments do not rely solely upon code to document the product in development.

9.2: INTEGRATION TESTING

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e., assemblages (or groups of units), are exercised through their interfaces using black-box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface. Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e., unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages. Software integration testing is performed according to the software development life cycle (SDLC) after module and functional tests. The cross-dependencies for software integration testing are: schedule for integration testing, strategy and selection of the tools used for integration, define the cyclomatic complexity of the software and software architecture, reusability of modules and life-cycle and versioning management. Some different types of integration testing are big-bang, top-down, and bottom-up, mixed (sandwich) and risky-hardest. Other Integration Patterns are: collaboration integration, backbone integration, layer integration, client-server integration, distributed services integration and high-frequency integration.

9.3: SYSTEM TESTING

System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

System testing is performed on the entire system in the context of a Functional Requirement Specification(s) (FRS) and/or a System Requirement Specification (SRS). System testing tests not only the design, but also the behavior and even the believed expectations of the customer. It is also intended to test up to and beyond the bounds defined in the software/hardware requirements specification(s).

CHAPTER – 10

Conclusion

We have finally deduced a way to handle examinations of any level efficiently through online with all security.

References

Read article based on Machine learning use in Examination System.

1) **Design of English hierarchical online test system based on machine learning**
   By *Xiahui Wang, Dan Zhang, Abhinav Asthana, Sudeep Asthana, Shaweta Khanna and Chaman Verma*
   From the journal of Intelligent Systems
   https://doi.org/10.1515/jisys-2020-0150

2) **Research on English Online Education Platform Based on Genetic Algorithm and Blockchain Technology**

   Volume 2020 |Article ID 8827084 | https://doi.org/10.1155/2020/8827084

   https://www.hindawi.com/journals/wcmc/2020/8827084/