# ONLINE EXAMINATION SYSTEM

## I. Abstract

**Objective:** The project mainly intends to provide the insight of the functionality and behavior of the Online Examination System. This project integrates computer and Internet technologies for the purpose of student assessment .It provides the options of the examinations available on the system according to the user privilege.

**Technology or Method:** Technology used to create the website is Django, which is a framework written in python which helps us to create websites faster and with high efficiency.

**Results and Findings:** Currently the website consist of variety of quizzes for students to appear and view their marks.

**Conclusions:** In this website, we are storing the performance of each and every student and how much they are improving themselves after every exam related to each subject they appear.

## II. Introduction

This website has been developed to overwrite the problems prevailing in the practicing manual system. This software is supported to eliminate and in some case reduce the hardships faced by existing system.

User needs to login to the website with his or her username and password. If they have not registered themselves previously then they need to create an account with their details.

After logging in they will be taken to the Dashboard, where they can see available quizzes. They can appear the quiz by clicking the Appear Button. After appearing the quiz they can view their results.

## III. Background

Offline exam is paper-pen based. Students need to come to the institute to appear the exam. The offline exam requires question paper, pen, sitting arrangement for each and every student and also some teachers (i.e., invigilators) are required. Also in this very pandemic situation (COVID-19), we can't afford offline examination system due to maintaining social distance and avoid the risk factor which could be caused by offline examination system.

Online Examination System is a technology-driven way to simplify examination tasks like defining an exam question banks, question and answer storage. Online Examination System is a cost-effective, scalable way to convert traditional pen and paper-based exams to online and paperless mode. Students can appear for the exam using any desktop, laptop, or mobile device with any browser. Exam results can be generated instantly for the appeared questions. It can simplify overall examination management and result in generation activity.

This Online Examination System provides students, a quick and easier way to take the examination with accuracy. Students can enter to take the test only with the valid credentials and they are offered multiple choices. The user can check their results just after taking the examination.

## IV. Objectives

- To conduct exam from anywhere, at anytime
- Reduces the use of pen and paper which is good for environment.
- Also, there is advantage of appearing as many times the user wishes to give any exam for his/her practice and to increase his/her skills in that very criteria and have habit of giving exams online.
- Students can view their improvement after giving each and every exam of same criteria.

## V. Methodology

- We have used Web-based interface for our project.
- We used Django to create the project. Django is a framework written in python which helps us to create websites faster and with high efficiency. This framework uses majorly three types of files:-

    1. **models.py**:- This file contains all the database models.
    2. **urls.py**:- This file is used to map the urls.
    3. **views.py**:- This file is used to render the html pages with required data.

- Django uses **MVC (**Model, View, Controller**)** pattern.

## VI. Implementation

- We have created four different Database Models that represent the tables in the database. Those are:-

1. Class Quiz defines the quiz table and attributes of the class defines the attributes of the database table.

```
class Quiz(models.Model):
    name = models.CharField(max_length=100, unique=True)
    pass_marks = models.IntegerField()

    def __str__(self):
        return self.name
```

2. Class Question defines the question table which consist of a foreign key **quiz**, which refers to the class quiz.

```
class Questions(models.Model):
    quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE)

    question_text = models.CharField(max_length=500)
    option_A = models.CharField(max_length=100)
    option_B = models.CharField(max_length=100)
    option_C = models.CharField(max_length=100)
    option_D = models.CharField(max_length=100)

    answer_A = models.BooleanField()
    answer_B = models.BooleanField()
    answer_C = models.BooleanField()
    answer_D = models.BooleanField()


    def __str__(self):
        return self.question_text
```

3. AppearedQuizes class has two foreign keys which are **user** and **quiz**. This represents which user has appeared which quiz.

```
class AppearedQuizzes(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    quiz = models.ForeignKey(Quiz, on_delete=models.CASCADE)

    date = models.DateField(null=True)
```

```
    time_taken = models.IntegerField(default=0)

    marks = models.IntegerField()

    def __str__(self):
        return self.quiz.name
```

4. Answers class represents the answers provided by every student or user for each and every quiz they have appeared.

```
class Answers(models.Model):
    appeared_quiz = models.ForeignKey(AppearedQuizzes, on_delete=models.CASCADE)

    question_text = models.ForeignKey(Questions, on_delete=models.CASCADE)

    answer_A = models.BooleanField()
    answer_B = models.BooleanField()
    answer_C = models.BooleanField()
    answer_D = models.BooleanField()

    correct = models.BooleanField()
```

- To represent the data we have used **HTML** pages and in the backend we have written some python scripts to render the data through **get & post requests**.

```
def signin(request):
    return render(request, 'loginRegister.html', {'method': 'login'})

def register(request):
    users = User.objects.all()

    return render(request, 'loginRegister.html', {'method': 'register'})

@csrf_protect
def logIn(request):
    if request.method == 'POST':
        username = request.POST['username']
        password = request.POST['password']

        user = authenticate(username=username, password=password)

        if user is not None:
            login(request, user)
            messages.success(request, "Logged In Successfully")
            return redirect('/')

        else:
            messages.error(request, "Invalid credentials")
            return redirect('/signin-form')
```

```
    else:
        messages.error(request, "Something Went Wrong!!")
        return redirect('/')

def logOut(request):
    logout(request)
    messages.success(request, 'Successfully Logged Out')
    return redirect('/')

def addUser(request):
    if request.method == 'POST':
        email = request.POST['email']
        username = request.POST['username']
        fname = request.POST['fname']
        lname = request.POST['lname']
        password = request.POST['password']

        users = User.objects.all()
        user_list = []
        for user in users :
            user_list.append(user.username)
        if username in user_list :
            messages.success(request, "Username Exists")
            return redirect('/register-form/')

        myUser = User.objects.create_user(username, email, password)
        myUser.first_name = fname
        myUser.last_name = lname

        myUser.school_admin = True


        myUser.save()
        messages.success(request, "User Created")
        return redirect('/signin-form/')
        # messages.success(request, "Account has been created successfully")
    else:
        pass

def dashboard(request):
    user = ''
    appeared_quiz = {}
    if request.user.is_authenticated:
        user = request.user
        appeared_quiz = AppearedQuizzes.objects.filter(user=user)


    all_quiz = Quiz.objects.all()
    available = len(all_quiz)

    all_not_appeared_quiz = []
    all_appeared_quiz = []
    quiz_details = {}
    if len(appeared_quiz) == 0:
        for quiz in all_quiz:
            all_not_appeared_quiz.append(quiz)
```

```
    else:
        for quiz in all_quiz:
            query = appeared_quiz.filter(quiz=quiz)
            if query:
                all_appeared_quiz.append(quiz)
                quiz_details[quiz.name] : {
                    'name': quiz.name,
                    'marks':query.Marks,
                }

            else:
                all_not_appeared_quiz.append(quiz)


    constraints = {
        'available': available,
        'user': user,
        'all_appeared_quiz' : all_appeared_quiz,
        'all_not_appeared_quiz' : all_not_appeared_quiz,
        'no_appeared_quiz': len(all_appeared_quiz),
        'no_not_appeared_quiz' : len(all_not_appeared_quiz),
        'appeared_quizzes' : appeared_quiz,
        }



    return render(request, 'home.html', constraints)
```

```
# Student Only Section ( Appear and View Quiz )
def appearQuestion(request, qzid, qnid):
    quiz = Quiz.objects.get(id = qzid)
    questions = Questions.objects.filter(quiz=quiz)

    allquestions = []
    for q in questions:
        allquestions.append(q)

    if qnid > len(allquestions):
        return render(request, '404.html')


    appeared = AppearedQuizzes.objects.filter(user=request.user)

    if not appeared:
        return render(request, '404.html')

    if qnid == len(allquestions):
        last = True
    else:
        last = False


    constraints = {
        'name' : quiz.name,
        'question' : allquestions[qnid - 1],
        'number' : qnid,
```

```
        'mode': 'appear',
        'last': last,
    }


    return render(request, 'addAppearQuiz.html', constraints)

def viewQuiz(request, id):
    quiz = Quiz.objects.get(id=id)
    query = AppearedQuizzes.objects.filter(user=request.user).filter(quiz=quiz)
    for iterator in query:
        appearQuiz = iterator

    if appearQuiz:
        appeared = "YES"
    else:
        appeared = "NO"

    qns = Questions.objects.filter(quiz=quiz)
    questions = []
    for question in qns:
        questions.append(question)

    all_answers = Answers.objects.filter(appeared_quiz=appearQuiz)

    result = []
    result_unit = {}

    rng = len(questions)
    for i in range(rng):
        result_unit = {}
        result_unit['id'] = i+1
        result_unit['question'] = questions[i].question_text
        for ans in Answers.objects.filter(question_text=questions[i]).filter(appeared_quiz=appearQui
z):
            answer = ans

        if answer.correct == True:
            result_unit['answer'] = "Correct"
        elif answer.correct == False:
            result_unit['answer'] = "Wrong"

        else:
            return HttpResponse("Something went wrong")

        result.insert(i, result_unit)


    constraints = {
        'mode' : 'view',
        'name' : quiz.name,
        'appeared': appeared,
        'passmarks' : quiz.pass_marks,
        'marks' : appearQuiz.marks,

        'results' : result,
    }
```

```
    return render(request, 'quizDetails.html', constraints)


def checkQuestion(request, qzid, qnid):
    if request.method == 'POST':
        quiz = Quiz.objects.get(id = qzid)
        for iterator in AppearedQuizzes.objects.filter(user=request.user).filter(quiz=quiz):
            appearQuiz = iterator

        questions = Questions.objects.filter(quiz=quiz)

        last = len(questions)

        allquestions = []
        for question in questions:
            allquestions.append(question)
        question = allquestions[qnid - 1]

        a = request.POST.get("op1")
        b = request.POST.get("op2")
        c = request.POST.get("op3")
        d = request.POST.get("op4")

        time = request.POST.get("time")


        answer = Answers()
        answer.appeared_quiz = appearQuiz
        answer.question_text = question

        yourAnwerString = ''
        if str(a) == 'on':
            yourAnwerString = yourAnwerString + 'a'
            answer.answer_A = True
        else:
            answer.answer_A = False
        if str(b) == 'on':
            yourAnwerString = yourAnwerString + 'b'
            answer.answer_B = True
        else:
            answer.answer_B = False
        if str(c) == 'on':
            yourAnwerString = yourAnwerString + 'c'
            answer.answer_C = True
        else:
            answer.answer_C = False
        if str(d) == 'on':
            yourAnwerString = yourAnwerString + 'd'
            answer.answer_D = True
        else:
            answer.answer_D = False

        originalAnswerString = ''
        ansList = []
        if question.answer_A == True:
            originalAnswerString = originalAnswerString + 'a'
            ansList.append(question.option_A)
```

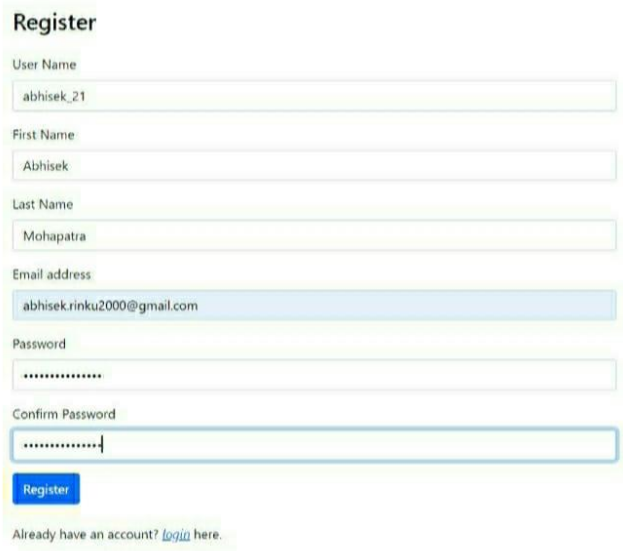
```
        if question.answer_B == True:
            originalAnswerString = originalAnswerString + 'b'
            ansList.append(question.option_B)

        if question.answer_C == True:
            originalAnswerString = originalAnswerString + 'c'
            ansList.append(question.option_C)

        if question.answer_D == True:
            originalAnswerString = originalAnswerString + 'd'
            ansList.append(question.option_D)


        if yourAnwerString == originalAnswerString:
            message = 'Correct Answer'
            appearQuiz.marks = appearQuiz.marks + 10
            appearQuiz.save()
            answer.correct = True

        else:
            message = 'Wrong Answer'
            answer.correct = False


        answer.save()
        appearQuiz.time_taken += int(time)
        appearQuiz.save()




        constraints = {
            'message' : message,
            'question' : question.question_text,
            'ansList' : ansList,
            'qzid' : qzid,
            'qnid' : qnid+1,
            'last': last,
        }


        return render(request, 'answer.html', constraints)

    return render(request,'404.html')


def result(request, qzid):
    quiz = Quiz.objects.get(id=qzid)
    query = AppearedQuizzes.objects.filter(user=request.user).filter(quiz=quiz)
    for q in query:
        appearedQuiz = q

    constraints = {
        'quiz' : quiz,
        'passmarks' : quiz.pass_marks,
        'yourmarks' : appearedQuiz.marks,
    }
```

```
    return render(request, 'result.html', constraints)


# APIs
#----------------------------------------#


def appearQuiz(request, id):
    quiz = Quiz.objects.get(id=id)


    appeared_quiz = AppearedQuizzes(quiz=quiz, user=request.user, marks=0)

    appeared_quiz.save()


    return redirect(f'/appearquiz/{quiz.id}/{1}/')
```

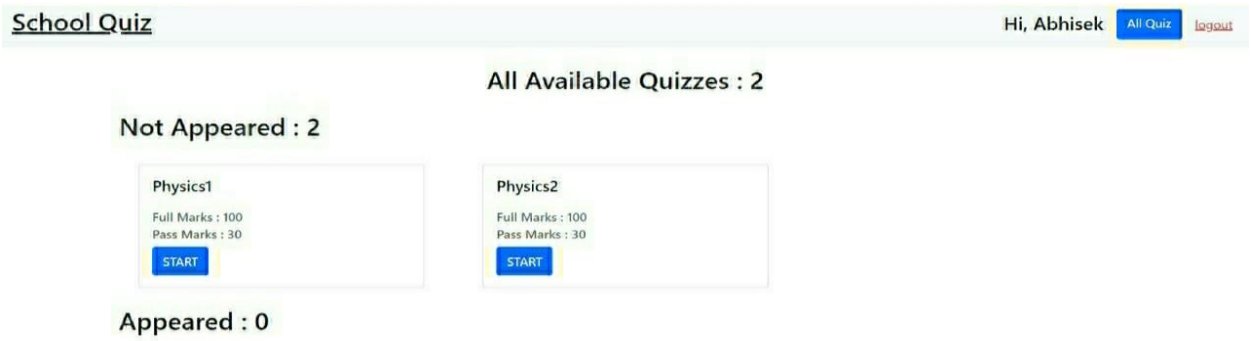
# VII. Results & Disscussion

## 1. LOGIN PAGE

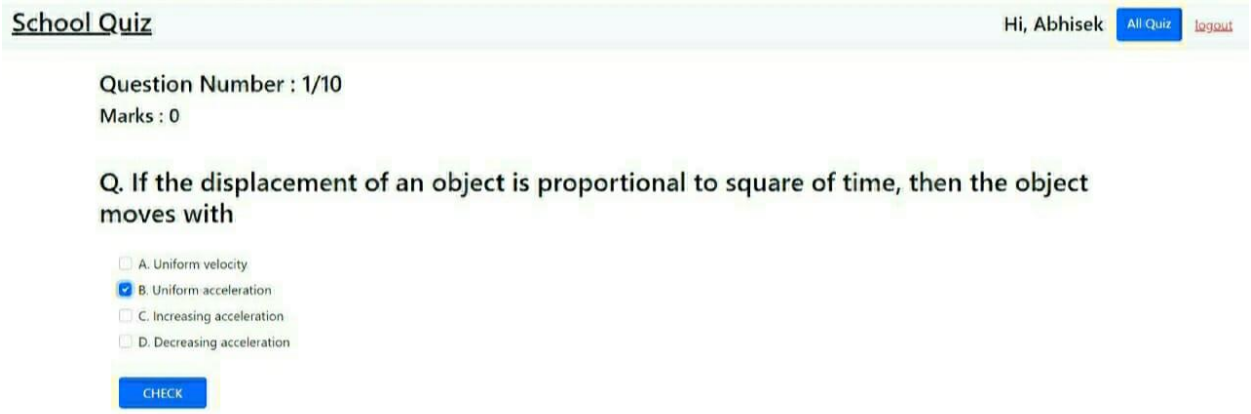

## 2. REGISTER PAGE



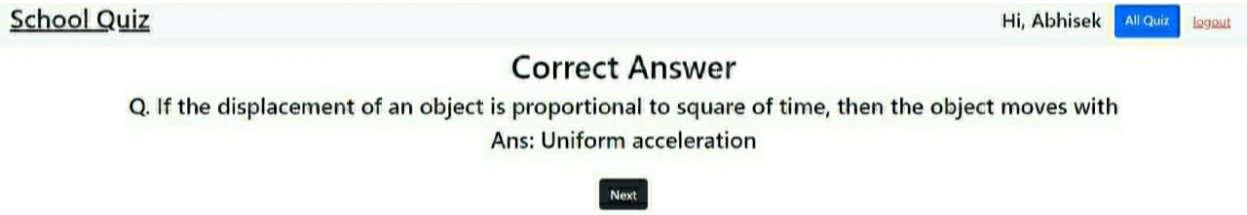
## 3. DASHBOARD



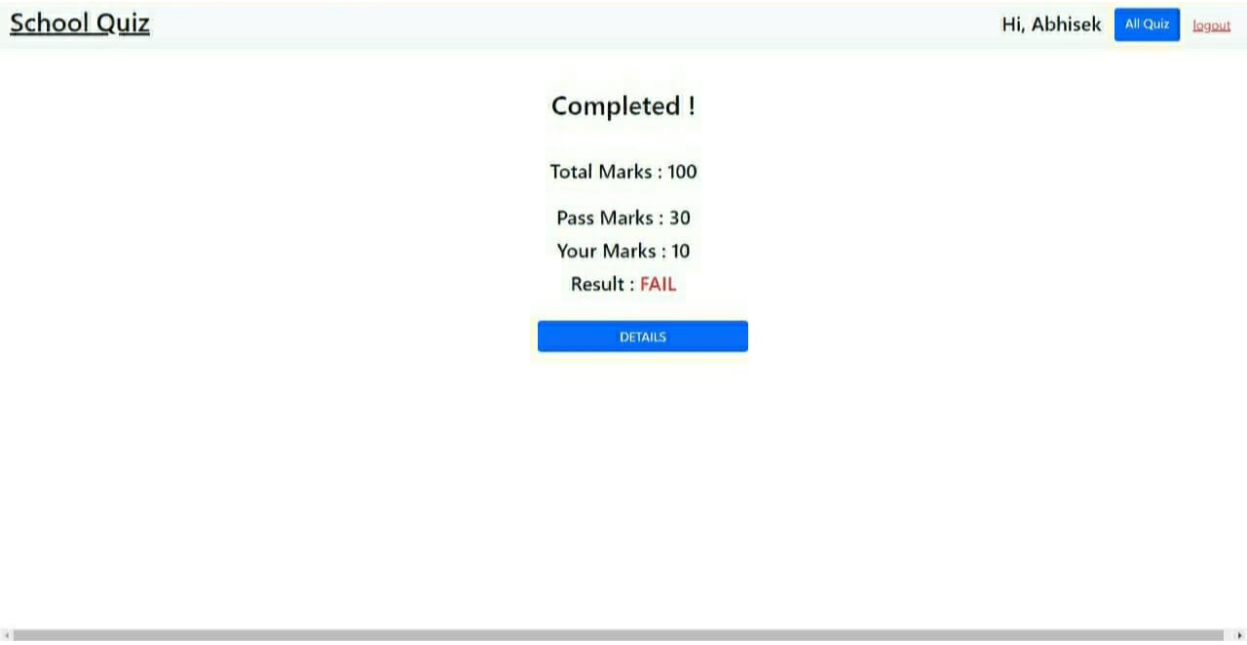

## 4. QUIZ PAGE



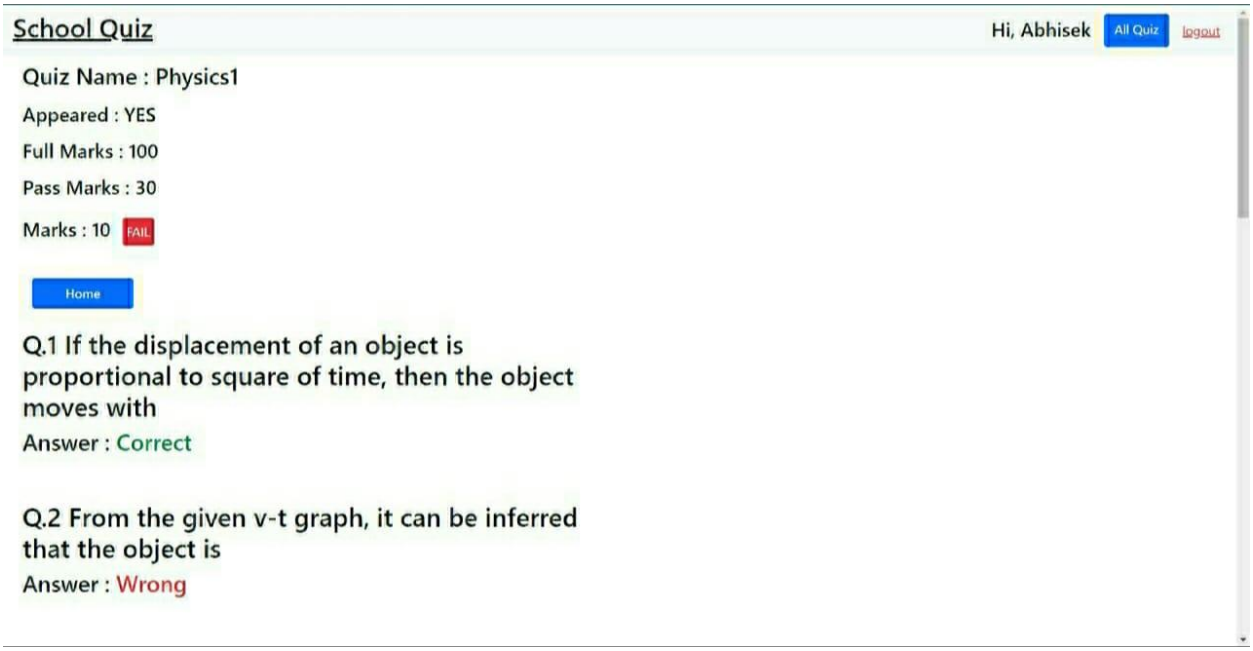
## 5. CHECK ANSWER PAGE



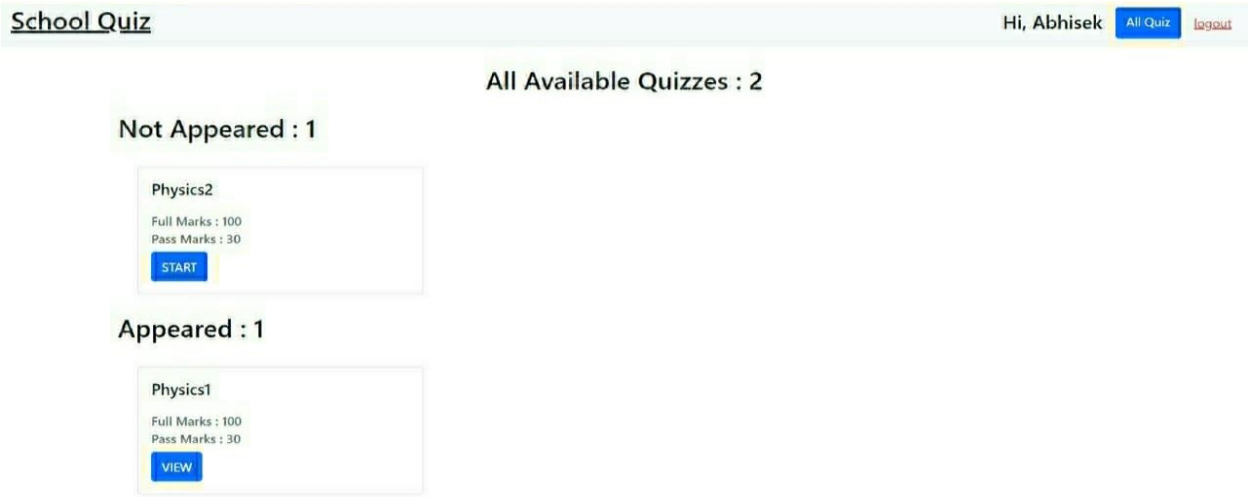

## 6. RESULT PAGE



## 7. QUIZ DETAIL PAGE




## 8. APPEARED QUIZ PAGE



## VIII. Conclusion and Future Works

It is a user friendly system, which is very easy and convenient to use.
The system is complete in the sense that it is operational and it is tested by entering data and getting the reports in proper order. But there is always a scope for improvement and enhancement.

We have prepared new system after identify issue in existing manual examination system. Scope of this online examination system is very broad in terms of other manually taking exams. However, we can improve our system in next version like.

- A smoother interface
- Improvement of Admin module
- Improved accuracy and security
- When a large set of data would be obtained, it can be used analyse student's performance and provide a student with appropriate guide on finding a right approach to study.
- Fair conduction

## IX. Contributions

We have divided this project into four modules that are:-

- **Login/Register**: - This part of project is done by SAGAR MOHANTY.
- **Quiz**: - The quiz appearing page is done by ANAND KUMAR JHA.
- **Dashboard**: - This module is done by ABHISEK MOHAPATRA.
- **Result**: - The result and result info page is done by BANISHREE SWAIN.

(All the required codes for frontend, backend and database are done by each of us on the basis of module divided between us.)

The Report and PPT are prepared by all of the members of group and we have done all the things on ZOOM MEETING.

## X. References


1. https://stackoverflow.com
2. https://docs.python.org
3. https://wiki.python.org
4. https://www.w3schools.com
5. https://www.youtube.com
6. https://docs.djangoproject.com/en/3.2/