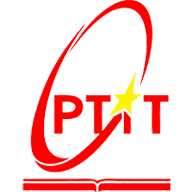
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

**CƠ SỞ THÀNH PHỐ HỒ CHÍ MINH**

**--🖎🕮✍--**



**BÁO CÁO**

**BÀI TẬP LỚN MÔN PYTHON**

**ĐỀ TÀI: WEB TRẮC NGHIỆM PYTHON VỚI DJANGO**

**Giảng viên: Nguyễn Thị Tuyết Hải**



Sinh viên thực hiện: Đặng Hoàng Hà – N19DCCN046

Phan Văn Hiểu – N19DCCN056

Lớp: D19CQCN03-N

TPHCM, 12/2021

**Mục Lục**

[CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI: 1](#_Toc16363)

[1. Giới thiệu chung: 1](#_Toc27881)

[2. Mô tả đề tài: 1](#_Toc2447)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT: 2](#_Toc21993)

[1. Sử dụng thư viện Django trong python: 2](#_Toc8811)

[1.1 Thư viện Django là gì: 2](#_Toc7915)

[1.2 Tại sao lại sử dụng thư viện Django: 2](#_Toc24391)

[1.3 Sơ lược về Django 2](#_Toc10443)

[1.3.1 Thiết kế model: 2](#_Toc26600)

[1.3.2 Đăng ký ứng dụng: 3](#_Toc25930)

[1.3.3 Đăng kí model 4](#_Toc21481)

[1.3.4 Thiết kế URLs: 4](#_Toc20204)

[1.3.6 Thiết kế templates: 6](#_Toc22447)

[1.4 Sử dụng các tác vụ của thư viện django: 9](#_Toc9389)

[1.4.1 Authenticating users 9](#_Toc18893)

[1.4.2 Permissions and Authorization 9](#_Toc12372)

[1.4.3 Default permissions 10](#_Toc19583)

[1.4.4 The LoginRequired mixin 10](#_Toc21616)

[1.4.5 Login, Logout 11](#_Toc1077)

[1.4.6 Django\_pbkdf2\_sha256 12](#_Toc18251)

[1.4.7 Render, Redirect 13](#_Toc19269)

[1.4.8 Re, Datetime, json 14](#_Toc19026)

[CHƯƠNG 3: THỰC NGHIỆM: 15](#_Toc5168)

[1. Cây phân cấp chức năng: 15](#_Toc27063)

[2. Biểu đồ cơ sở dữ liệu của Quiz App: 16](#_Toc30506)

[3. Mô tả trang web: 16](#_Toc28269)

[3.1. Trang chủ, đăng nhập, đăng ký và thêm, chỉnh sửa thông tin tài khoản: 16](#_Toc3263)

[3.1.1. Trang chủ 16](#_Toc15525)

[3.1.2. Đăng nhập tài khoản 20](#_Toc1743)

[3.1.3. Đăng kí tài khoản 20](#_Toc27500)

[3.1.4. Chỉnh sửa thông tin cá nhân 21](#_Toc16951)

[3.2. Tạo và xem trước bài trắc nghiệm: 23](#_Toc1574)

[3.3. Thực hiện bài trắc nghiệm: 30](#_Toc7895)

[CHƯƠNG 4: TÀI LIỆU THAM KHẢO: 33](#_Toc12024)

[1. Sách tham khảo: 33](#_Toc6374)

[2. Link tham khảo: 33](#_Toc20828)

# CHƯƠNG 1: GIỚI THIỆU ĐỀ TÀI:

## Giới thiệu chung:

* Như chúng ta đã thấy trong trình trạng COVID-19 diễn biến phức tạp hiện nay thì yêu cầu về các ứng dụng, trang web có hình thức thi, các trò chơi họp nhóm trực tuyến càng trở nên phổ biến và cần thiết.
* Để đáp ứng nhu cầu đó, bài toán đặt ra là cần làm thể nào để có 1 website vừa đáp ứng nhu cầu thi trực tuyến, vừa đáp ứng nhu cầu các trò chơi hội họp trực tuyến.
* Biết được nhu cầu đó Trang web “QUIZ APP” được chúng em xây dựng trên nền tảng backend của **Python** (bằng Module Django) sẽ đáp ứng nhu cầu cho mọi người về việc tạo một bài kiểm tra trắc nghiệm hay tổ chức các trò chơi trắc nghiệm tính điểm trực tuyến.

## Mô tả đề tài:

* Trang web “QUIZ APP” bao gồm các chức năng cơ bản đảm bảo việc thực hiện, thêm, xóa, chỉnh sửa các bài trắc nghiệm cụ thể như sau:
* Cho phép người dùng làm bài trắc nghiệm
* Xem trước bài trắc nghiệm
* Hiện số điểm đã làm được và lưu dữ liệu khi hoàn thành bài trắc nghiệm
* Đăng nhập và đăng ký tài khoản để có thể thêm, xóa, chỉnh sửa các bài trắc nghiệm của người dùng đã tạo

1. Link Github:

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT:

## Sử dụng thư viện Django trong python:

### Thư viện Django là gì:

**-** Django là công cụ giúp đưa các ứng dụng web từ khái niệm có thể khởi chạy trong vài giờ. Django giúp giải quyết các rắc rối trong quá trình phát triển web. Ngoài ra, Django là nguồn mở và hoàn toàn miễn phí.

### Tại sao lại sử dụng thư viện Django:

**-** Nhanh đáng kinh ngạc: Django được thiết kế để giúp các nhà phát triển đưa ra các ứng dụng từ ý tưởng đến khi hoàn thành một cách nhanh nhất.

- Hỗ trợ nhiều tính năng: Django bao gồm rất nhiều tính năng bổ sung giúp nhà phát triển có thể sử dụng để xử lý các tác vụ phát triển web thông thường như: xác thực người dùng, quản trị nội dung, sơ đồ trang web, nguồn cấp dữ liệu RSS và nhiều tác vụ khác.

- Đảm bảo an toàn: Django giúp các nhà phát triển tránh nhiều lỗi bảo mật phổ biến như: SQL injection, cross-site script, giả mạo yêu cầu cross-site và clickjacking. Hệ thống xác thực người dùng của Django cung cấp an toàn để quản lý tài khoản và mật khẩu người dùng.

- Có thể mở rộng: nhiều trang web sử dụng khả năng mở rộng quy mô nhanh chóng, linh hoạt của Django để đáp ứng nhu cầu lưu lượng truy cập lớn.

- Linh hoạt: nhiều công ty, tổ chức và chính phủ đã sử dụng Django để xây dựng mọi thứ từ hệ thống quản lý nội dung đến mạng xã hội, nền tảng máy tính khoa học.

### Sơ lược về Django

### Thiết kế model:

**-** Tạo model trong file ../models.py

- Thêm models:

*from* django.db *import* models

**-** Tạo class:

*class* Quiz(models.Model):  
 user = models.ForeignKey(User, on\_delete=models.CASCADE)  
 name = models.CharField(max\_length=120)  
 topic = models.CharField(max\_length=120)  
 number\_of\_questions = models.IntegerField()  
 time = models.IntegerField(help\_text="Duration of the quiz in minutes ")  
 required\_score\_to\_pass = models.IntegerField(help\_text="requires score is %")  
 difficulty = models.CharField(max\_length= 6,choices=DIFF\_CHOIES)

**+** Tạo các trường thông tin cho class. vd: user, name, topic, …

+ Các kiểu dữ liệu:

CharField: kiểu kí tự

IntegerField: kiểu số nguyên

+ Các ràng buộc:

ForeignKey: khóa ngoại

on\_delete=models.CASCADE: ràng buộc khi các đối tượng tham chiếu bị xóa thì các đối tượng các tham chiếu đến cũng bị xóa

max\_length: kích thước tối đa

help\_text: chữ hiển thị để giải thích yêu cầu cho người dùng

Choices: nội dung để chọn

- Sau khi tạo model thì thực hiện lệnh:

$ python manage.py makemigrations

$ python manage.py migrate

- Lệnh **makemigrations** xem xét tất cả các mô hình có sẵn của nhà phát triển và tạo di chuyển cho các bảng chưa tồn tại. migrate chạy quá trình di chuyển và tạo các bảng trong cơ sở dữ liệu , cũng như tùy chọn cung cấp khả năng kiểm soát giản đồ phong phú hơn nhiều.

### Đăng ký ứng dụng:

**-** Đăng kí ứng dụng trong file ../settings.py

# Application definition  
  
INSTALLED\_APPS = [  
 'django.contrib.admin',  
 'django.contrib.auth',  
 'django.contrib.contenttypes',  
 'django.contrib.sessions',  
 'django.contrib.messages',  
 'django.contrib.staticfiles',  
 'questions',  
 'quizes',  
 'results',  
 'user',  
]

**-** Thêm các tên ứng dụng cần đăng kí vào INSTALLED\_APPS, các tên được đặt trong nháy đơn hoặc nháy kép và ngăn các nhau bởi dấu phẩy

### Đăng kí model

**-**Khi các mô hình của nhà phát triển được xác định, Django có thể tự động tạo giao diện quản trị chuyên nghiệp, sẵn sàng sản xuất - một trang web cho phép người dùng đã xác thực thêm, thay đổi và xóa các đối tượng. Bước duy nhất cần thiết là đăng ký mô hình của nhà phát triển trong trang web quản trị:

- Đăng kí model trong ../admin.py

- Thêm models

- Đăng kí model bằng lệnh: admin.site.register()

*from* django.contrib *import* admin  
*from* .models *import* Quiz  
# Register your models here.  
  
admin.site.register(Quiz)

### Thiết kế URLs:

**-** Để thiết kế URL cho ứng dụng, cần phải tạo model python.

- Thêm path và các hàm cần dùng để thiết kế url

*from* django.urls *import* path  
*from* .views *import* quiz\_data\_view, viewHome, delete\_quiz, \  
 ViewSerch, viewDoTest, AddTest, ChangeQuiz, viewTest, tryTest, saveResult, exportFile

**-** Ghi tên ứng dụng để phân biệt với các ứng dụng khác: app=’Ten app’

app\_name = 'quizes'

**-** Thiết kế các url trong urlpatterns =[], theo cú pháp

path(‘đường dẫn’, tên hàm hoặc tên class,name= ‘tên đường dẫn’)

urlpatterns = [  
 path('', viewHome, name='home'),  
 path('delete\_quiz/<pk>/',delete\_quiz,name='delete\_quiz'),  
 path('quiz\_data\_view/<pk>/',quiz\_data\_view,name= 'quiz\_data\_view'),  
 path('add\_test',AddTest.as\_view(),name='add\_test'),  
 path('view\_search/',ViewSerch.as\_view(),name='view\_search'),  
 path('make\_test/', viewDoTest, name='make\_test'),  
 path('save\_result/', saveResult, name='save\_result'),  
 path('change\_quiz/<pk>/', ChangeQuiz.as\_view(), name='change\_quiz'),  
 path('view\_test/<pk>/', viewTest, name='view\_test'),  
 path('try\_test/', tryTest, name='try\_test'),  
 path('export\_file/<pk>/', exportFile, name='export\_file'),  
]

**-** Đoạn mã trên ánh xạ các đường dẫn URL đến các hàm gọi lại Python (“lượt xem”). Các chuỗi đường dẫn sử dụng các thẻ tham số để "nắm bắt" các giá trị từ các URL. Khi người dùng yêu cầu một trang, Django sẽ chạy qua từng đường dẫn theo thứ tự và dừng lại ở đường dẫn đầu tiên phù hợp với URL được yêu cầu. (Nếu không có cái nào phù hợp, Django gọi một dạng xem 404 trong trường hợp đặc biệt.) Điều này cực kỳ nhanh, vì các đường dẫn được biên dịch thành các biểu thức chính quy tại thời điểm tải.

- Khi một trong các mẫu URL khớp, Django gọi chế độ xem đã cho, đó là một hàm Python. Mỗi chế độ xem được chuyển một đối tượng yêu cầu - chứa siêu dữ liệu yêu cầu - và các giá trị được ghi lại trong mẫu.

- vd: nếu người dùng yêu cầu URL ‘/delete\_quiz/1/’, Django sẽ gọi hàm quizes.views.delete\_quiz(request,pk=1)

* + 1. Tạo views:

**-**Mỗi chế độ xem có trách nhiệm thực hiện một trong hai việc: Trả lại đối tượng HttpResponse có chứa nội dung cho trang được yêu cầu hoặc nêu ra một ngoại lệ chẳng hạn như Http404. Phần còn lại là tùy thuộc vào người thiết kế.

-Nói chung, một dạng xem truy xuất dữ liệu theo các tham số, tải một mẫu và hiển thị mẫu với dữ liệu đã truy xuất.

Vd: ../views.py

*from* django.shortcuts *import* render

*def* viewHome(*request*):  
 *global* option\_sort  
 option\_sort = {'Test Name-ASC': 'name', 'Test Name-DESC': '-name', 'Author-ASC': 'user', 'Author-DESC': '-user'}  
 quiz = Quiz.objects.all().order\_by(option\_sort['Test Name-ASC'])  
 *return* render(*request*, 'quizes/home.html',{'object\_list':quiz,'sort\_by': 'Test Name-ASC','request': *request* })

- Sử dụng rendender để tạo ra giao diện người xem với các tham số:

+ request: yêu cầu của người dùng

+ ‘quizes/home.html’: templates để hiển thị

+{{'object\_list':quiz,'sort\_by': 'Test Name-ASC','request': *request } : context*

### Thiết kế templates:

- Django có một đường dẫn tìm kiếm mẫu, cho phép nhà phát triển giảm thiểu sự dư thừa giữa các mẫu. Trong cài đặt Django, nhà phát triển chỉ định một danh sách các thư mục để kiểm tra các mẫu với DIRS. Nếu một mẫu không tồn tại trong thư mục đầu tiên, nó sẽ kiểm tra thư mục thứ hai, v.v.

- Cài đặt thư mục chứa templates trong ../settings.py

TEMPLATES = [  
 {  
 'BACKEND': 'django.template.backends.django.DjangoTemplates',  
 'DIRS': [  
 BASE\_DIR/'template',  
 BASE\_DIR/'user'/'template',  
 BASE\_DIR / 'questions' / 'template',  
 BASE\_DIR / 'results' / 'template',  
 ],  
 'APP\_DIRS': *True*,  
 'OPTIONS': {  
 'context\_processors': [  
 'django.template.context\_processors.debug',  
 'django.template.context\_processors.request',  
 'django.contrib.auth.context\_processors.auth',  
 'django.contrib.messages.context\_processors.messages',  
 ],  
 },  
 },  
]

**-** Thêm các thư mục chứa templates trong TEMPLATES[‘DIRS’], các thư mục các nhau bằng dấu phẩy

- Thiết kế template bằng html:

{% extends "quizes/base.html" %}  
{% load static %}  
  
{% block img %}  
 <link rel**="icon"** href**="{% static 'quizes/logo.png' %}"** type**="image/x-icon"**>  
 <link rel**="shortcut icon"** href**="{% static 'quizes/logo.png' %}"** type**="image/x-icon"**>  
{% endblock img %}  
  
{% block style %}  
<link rel**="stylesheet"** href**="{% static 'quizes/style.css' %}?{% now 'U' %}"** type**="text/css"** media**="screen"**>  
<link rel**="stylesheet"** href**="{% static 'quizes/admin.css' %}?{% now 'U' %}"** type**="text/css"** media**="screen"**>  
<script >  
 var questions = {{ data\_test|safe }};  
 var time = {{ quize.time }};  
</script>  
  
{% endblock style %}  
  
{% block title %}VIEW ANSWER {% endblock title %}  
{% block username %}{{ request.user.username }}{% endblock username %}  
  
  
{% block content %}  
 <div class**="quizContainerSmall container-fluid well well-lg"**>  
 <div id**="quiz1"** class**="text-center"**>  
 <h3> Test's name:</h3>  
 <h4 class**="number-question"** ><span >Total Question: </span>  
 <br/><span style**="font-size:25px;"**> {{ quize.number\_of\_questions }} q.</span></h4>  
 <h4 class**="count-time"**><span id**="iTimeShow"**>Time Remaining: </span>  
 <br/><span id**='timer'** style**="font-size:25px;"**>{{ quize.time }} min</span></h4>  
 <h2 > {{ quize.name }} </h2>  
 </div>  
 </div>  
 <div class**="quizContainer container-fluid well well-lg"**>  
 {% for q in quize.get\_questions %}  
 <div class**="question"**>{{ q.text }}</div>  
 {% for a in q.get\_answers %}  
 {% if a.correct == True %}  
 <ul class**="choiceList"**>  
 <li style**="border:2px solid green;margin-top:10px;"**>  
 <input disabled type**="radio"** class**="radio-inline"** checked**="checked"** name**="dynradio"** > {{ a.text }}</li>  
 </ul>  
 {% else %}  
 <ul class**="choiceList"**>  
 <li>  
 <input disabled type**="radio"** class**="radio-inline"** name**="dynradio"** />{{ a.text }}</li>  
 </ul>  
 {% endif %}  
 {% endfor %}  
 {% endfor %}  
 </div>  
 <a href**="{% url 'quizes:home' %}"**><button class**="submitBtn"**>Close</button></a>  
{% endblock content %}

- Các biến được bao quanh bởi dấu ngoặc nhọn kép. {{quize.number\_of\_questions}} có nghĩa là "Xuất ra giá trị của thuộc tính tên của bài kiểm tra." Những dấu chấm không chỉ được sử dụng để tra cứu thuộc tính. Ngoài ra, cũng có thể thực hiện tra cứu khóa từ điển, tra cứu chỉ mục và gọi hàm.

- Django sử dụng khái niệm "kế thừa khuôn mẫu". Đó là những gì {% extends "quizes/base.html"%} làm. Nó có nghĩa là “Đầu tiên hãy tải mẫu có tên là‘ base ’, đã xác định một loạt các khối và điền vào các khối bằng các khối sau”. Nói tóm lại, điều đó cho phép nhà phát triển cắt giảm đáng kể lượng dư thừa trong các mẫu: mỗi mẫu chỉ phải xác định những gì duy nhất cho mẫu đó.

### Sử dụng các tác vụ của thư viện django:

### Authenticating users

Sử dụng authenticate () để xác minh một tập hợp các thông tin xác thực. Nó lấy thông tin xác thực làm đối số từ khóa, tên người dùng và mật khẩu cho trường hợp mặc định, kiểm tra chúng với từng chương trình phụ trợ xác thực và trả về đối tượng Người dùng nếu thông tin đăng nhập hợp lệ cho chương trình phụ trợ. Nếu thông tin xác thực không hợp lệ cho bất kỳ chương trình phụ trợ nào hoặc nếu một chương trình phụ trợ làm tăng PermissionDenied, nó sẽ trả về None

Vd:

*class* LoginClass(View):  
 *def* get(self,*request*):  
 *return* render(*request*,'user/login.html')  
  
 *def* post(self,*request*):  
 username=*request*.POST.get('username')  
 password = *request*.POST.get('password')  
 my\_user = authenticate(username=username, password=password)  
 *if* my\_user *is None*:  
 *return* render(*request*,'user/login.html',{"error":'Login failed! Username does not exist or password is wrong'})  
 login(*request*,my\_user)  
 *return* redirect("/")

### **Permissions and Authorization**

**-** Quyền truy cập để xem các đối tượng được giới hạn cho người dùng có quyền “xem” hoặc “thay đổi” đối với loại đối tượng đó.

- Quyền truy cập để xem biểu mẫu “thêm” và thêm đối tượng bị giới hạn đối với người dùng có quyền “thêm” cho loại đối tượng đó.

- Quyền truy cập để xem danh sách thay đổi, xem biểu mẫu “thay đổi” và thay đổi đối tượng được giới hạn cho người dùng có quyền “thay đổi” đối với loại đối tượng đó.

- Quyền truy cập để xóa một đối tượng bị giới hạn ở những người dùng có quyền "xóa" đối với loại đối tượng đó

- Quyền có thể được đặt không chỉ cho mỗi loại đối tượng, mà còn cho mỗi cá thể đối tượng cụ thể. Bằng cách sử dụng các phương thức has\_view\_permission (), has\_add\_permission (), has\_change\_permission () và has\_delete\_permission () được cung cấp bởi lớp ModelAdmin, có thể tùy chỉnh quyền cho các trường hợp đối tượng khác nhau của cùng một loại.

- Đối tượng người dùng có hai trường nhiều-nhiều: nhóm và người dùng cấp quyền. Các đối tượng người dùng có thể truy cập các đối tượng liên quan của họ theo cách giống như bất kỳ mô hình Django nào khác:

myuser.groups.set([group\_list])

myuser.groups.add(group, group, ...)

myuser.groups.remove(group, group, ...)

myuser.groups.clear()

myuser.user\_permissions.set([permission\_list])

myuser.user\_permissions.add(permission, permission, ...)

myuser.user\_permissions.remove(permission, permission, ...)

myuser.user\_permissions.clear()

### **Default permissions**

- Khi django.contrib.auth được liệt kê trong cài đặt INSTALLED\_APPS của nhà phát triển, nó sẽ đảm bảo rằng bốn quyền mặc định - thêm, thay đổi, xóa và xem - được tạo cho mỗi kiểu Django được xác định trong một trong các ứng dụng đã cài đặt của nhà phát triển.

- Các quyền này sẽ được tạo khi nhà phát triển chạy management.py migrate; lần đầu tiên nhà phát triển chạy di chuyển sau khi thêm django.contrib.auth vào INSTALLED\_APPS, quyền mặc định sẽ được tạo cho tất cả các kiểu máy đã cài đặt trước đó, cũng như cho mọi kiểu máy mới đang được cài đặt tại thời điểm đó. Sau đó, nó sẽ tạo quyền mặc định cho các mô hình mới mỗi khi nhà phát triển chạy management.py migrate (chức năng tạo quyền được kết nối với tín hiệu post\_migrate).

### **The LoginRequired mixin**

- Nếu một chế độ xem đang sử dụng mixin này, tất cả các yêu cầu của người dùng không được xác thực sẽ được chuyển hướng đến trang đăng nhập hoặc hiển thị lỗi HTTP 403 Forbidden, tùy thuộc vào tham số raise\_exception.

Vd:

*class* UpdateProfile(LoginRequiredMixin, View):  
 login\_url = '/user/Login/'  
 *def* get(self,*request*):  
 *return* render(*request*, 'user/update\_profile.html')  
 *def* post(self, *request*):  
 user\_pk = *request*.user.pk  
 username = *request*.POST.get('username')  
 first\_name = *request*.POST.get('firstname')  
 last\_name = *request*.POST.get('lastname')  
 email = *request*.POST.get('Email')  
 *if* whiteSpace(username):  
 *return* render(*request*,'user/update\_profile.html',{"error":"Do not contain spaces in usernames"})  
 User.objects.filter(pk=user\_pk).update(username=username, first\_name=first\_name, last\_name= last\_name, email= email )  
 *return* redirect('/')

### **Login, Logout**

- Đăng nhập: sử dụng cú pháp:

login(request, user, backend=None)

- Khi người dùng đăng nhập, ID của người dùng và phần phụ trợ được sử dụng để xác thực sẽ được lưu trong phiên của người dùng.

Vd:

*class* LoginClass(View):  
 *def* get(self,*request*):  
 *return* render(*request*,'user/login.html')  
  
 *def* post(self,*request*):  
 username=*request*.POST.get('username')  
 password = *request*.POST.get('password')  
 my\_user = authenticate(username=username, password=password)  
 *if* my\_user *is None*:  
 *return* render(*request*,'user/login.html',{"error":'Login failed! Username does not exist or password is wrong'})  
 login(*request*,my\_user)  
 *return* redirect("/")

- Đăng xuất:

*def* logoutPage(*request*):  
 logout(*request*)  
 *return* redirect('/')

-Khi gọi logout (), dữ liệu phiên cho yêu cầu hiện tại sẽ bị xóa hoàn toàn. Tất cả dữ liệu hiện có sẽ bị xóa. Điều này nhằm ngăn người khác sử dụng cùng một trình duyệt Web để đăng nhập và có quyền truy cập vào dữ liệu phiên của người dùng trước đó.

### **Django\_pbkdf2\_sha256**

- Lớp này thực hiện băm PBKDF2-HMAC-SHA256 của Django và tuân theo mật khẩu-băm-api.

- Nó hỗ trợ một muối có độ dài thay đổi và số lượng vòng thay đổi.

- Các phương thức mã hóa và genconfig chấp nhận các từ khóa tùy chọn sau:

+ Loại muối: str

+ Muối param: Chuỗi muối tùy chọn. Nếu không được chỉ định, một 12 ký tự sẽ được tự động tạo (điều này được khuyến nghị). Nếu được chỉ định, có thể là bất kỳ chuỗi ký tự nào được vẽ từ dải ô regexp [0-9a-zA-Z].

gõ salt\_size: int

param salt\_size:

+ Số lượng ký tự tùy chọn để sử dụng khi tự động tạo muối mới. Mặc định là 12, nhưng có thể là bất kỳ giá trị dương nào.

+Loại vòng: int

+Vòng param: Số lượng vòng tùy chọn để sử dụng. Mặc định là 29000, nhưng phải nằm trong phạm vi (1,1 << 32).

gõ thư giãn: bool

+ Param relax: Theo mặc định, việc cung cấp giá trị không hợp lệ cho một trong các từ khóa khác sẽ dẫn đến lỗi ValueError. Nếu relax = True và lỗi có thể được sửa, PasslibHashWarning sẽ được cấp thay thế. Các lỗi có thể sửa được bao gồm vòng quá nhỏ hoặc quá lớn và chuỗi muối quá dài.

*class* ChangePassword(LoginRequiredMixin, View):  
 *def* get(self,*request*):  
 *return* render(*request*, 'user/change\_password.html')  
 *def* post(self, *request*):  
 user = *request*.user  
 password\_user = user.password  
 password\_old = *request*.POST.get('password\_old')  
 password\_new = *request*.POST.get('password\_new')  
 password\_cofirm = *request*.POST.get('password\_cofirm')  
 *if* django\_pbkdf2\_sha256.verify(password\_old, password\_user) == *False*:  
 *return* render(*request*, 'user/change\_password.html', {'error': "Old password entered is not correct"})  
 *if* len(password\_new)<8:  
 *return* render(*request*,'user/change\_password.html',{"error":"Password should Atleast have 8 character"})  
 *elif not* lower(password\_new):  
 *return* render(*request*, 'user/change\_password.html', {"error": "You didn't use LOWER case letter"})  
 *elif not* upper(password\_new):  
 *return* render(*request*, 'user/change\_password.html', {"error": "You didn't user UPPER case letter"})  
 *elif not* digit(password\_new):  
 *return* render(*request*, 'user/change\_password.html', {"error": "You didn't use DIGIT"})  
 *elif* password\_new!=password\_cofirm:  
 *return* render(*request*,'user/change\_password.html',{"error":"Password and confirmation do not match"})  
  
 rounds = password\_user.split('$')[1]  
 salt = password\_user.split('$')[2]  
 password\_new\_hash=django\_pbkdf2\_sha256.hash(password\_new, rounds=rounds, salt=salt)  
 User.objects.filter(pk=user.pk).update(password= password\_new\_hash)  
 my\_user = authenticate(username=user.username, password=password\_new)  
 login(*request*, my\_user)  
 *return* redirect("/")

### **Render, Redirect**

Render:

- Cú pháp: render(request, template\_name, context=None, content\_type=None, status=None, using=None)

- Kết hợp một mẫu nhất định với một từ điển ngữ cảnh nhất định và trả về một đối tượng HttpResponse với văn bản được hiển thị đó.

vd:

*def* viewDoTest(*request*):

*return* render(*request*,'quizes/makeTest.html')

Redirect:

- Cú pháp: redirect(to, \*args, permanent=False, \*\*kwargs)

**-** Trả về một HttpResponseRedirect đến URL thích hợp cho các đối số được truyền vào.

vd:

*def* logoutPage(*request*):

logout(*request*)

*return* redirect('/')

### **Re, Datetime, json**

Re:

* Mô-đun re cung cấp một tập hợp các chức năng cho phép chúng tôi tìm kiếm một chuỗi cho một kết quả phù hợp

+findall trả về một danh sách chứa tất cả các kết quả phù hợp

+find trả về một đối tượng Khớp nếu có một đối tượng khớp ở bất kỳ đâu trong chuỗi

+ split trả về danh sách trong đó chuỗi đã được tách ở mỗi trận đấu

+ sub thay thế một hoặc nhiều kết quả phù hợp bằng một chuỗi

**vd: re.sub(r"[^a-zA-Z0-9]","",quiz.name)**

Datetime:

* Mô-đun datetime cung cấp các lớp để thao tác ngày và giờ.

vd: **datetime.now().strftime('%YY%MM%dD\_%Hh%Mm%Ss')**

Json:

* Python has a built-in package called json, which can be used to work with JSON data

vd: **r = json.dumps(questions\_list)**

# CHƯƠNG 3: THỰC NGHIỆM:

## Cây phân cấp chức năng:

Quiz App

Guest

Xem danh sách bài trắc nghiệm

Làm bài trắc nghiệm

Xem trước bài trắc nghiệm

Tạo tài khoản user

User

Tương tự các chức năng cơ bản của Guest

Tạo bài trắc nghiệm

Xóa bài trắc nghiệm

Chỉnh sửa bài trắc nghiệm

Thêm thông tin người dùng

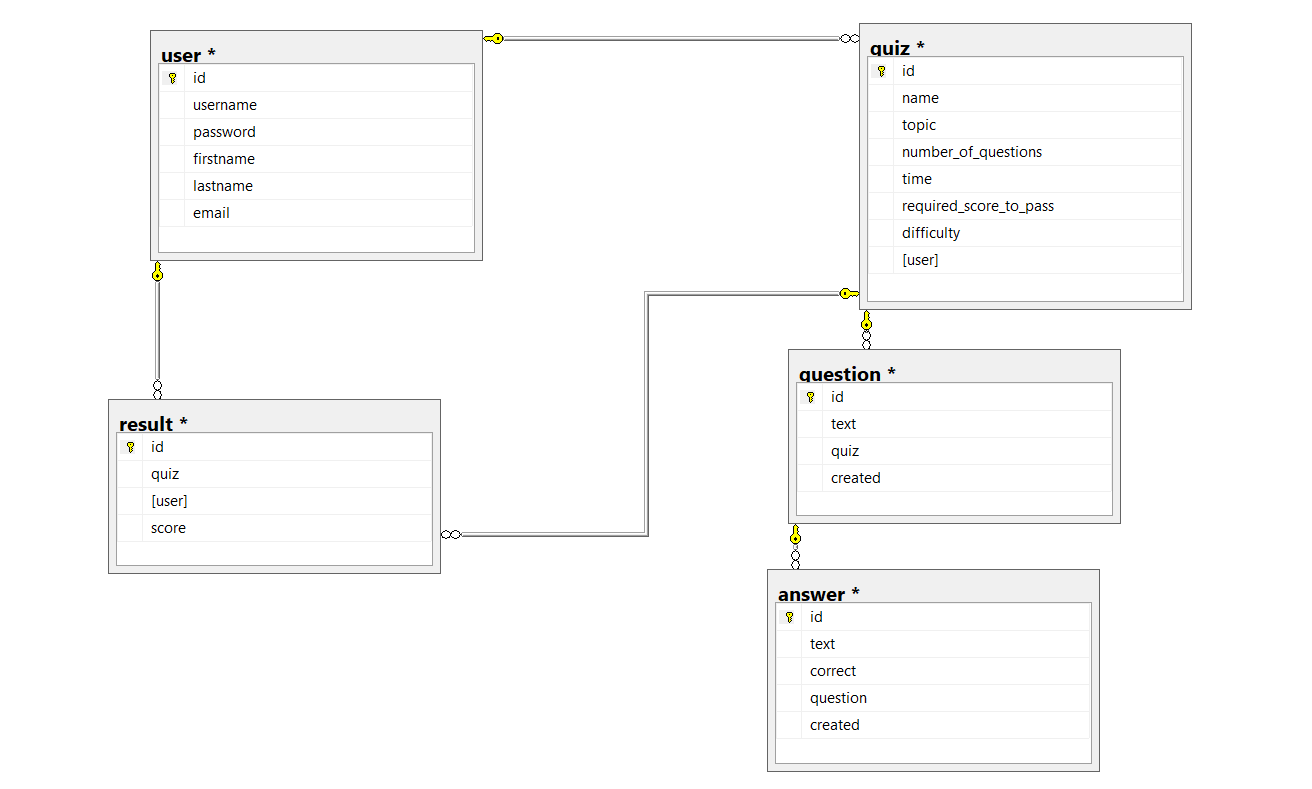
Admin

Quản lý CSDL user

Quản lý CSDL bài trắc nghiệm, câu hỏi,

Hình 1. Cây phân cấp chức năng của trang web

## Biểu đồ cơ sở dữ liệu của Quiz App:

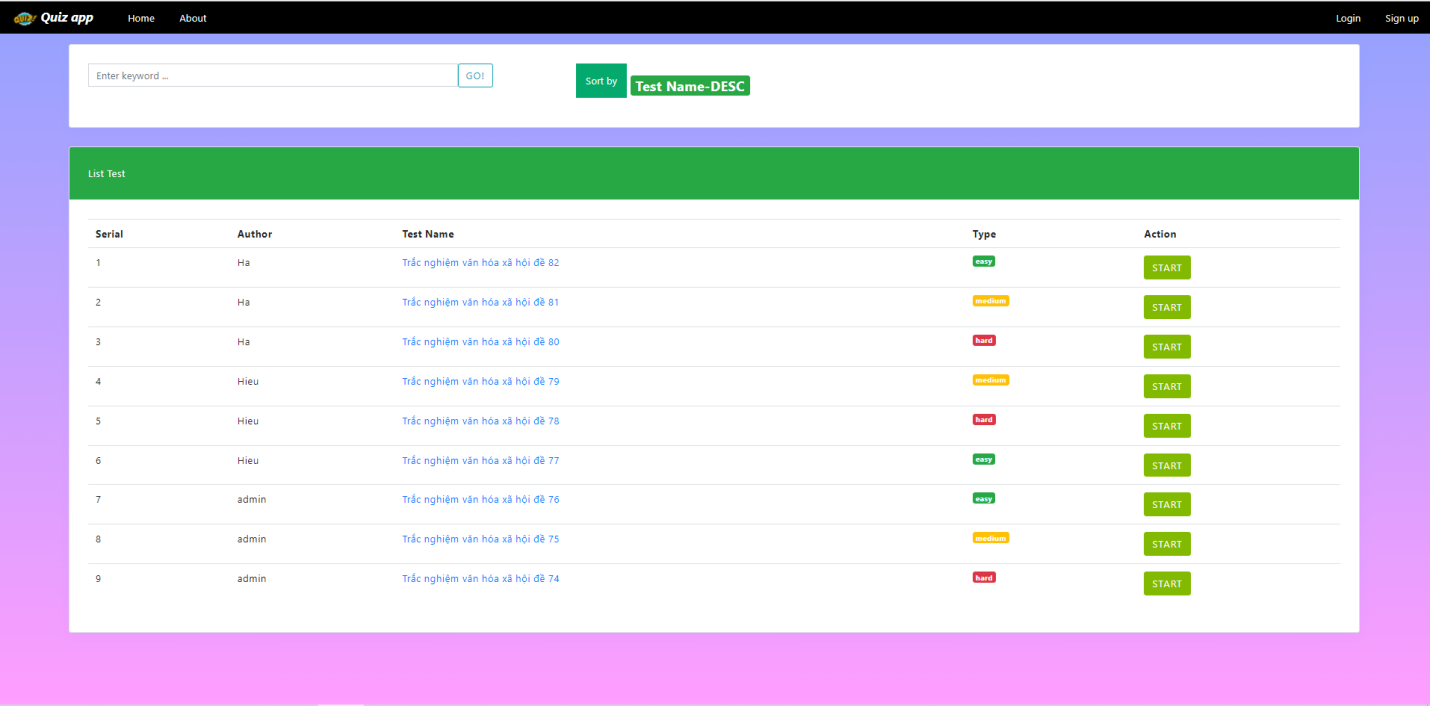
****

Hình 2. Mô hình diagram

## Mô tả trang web:

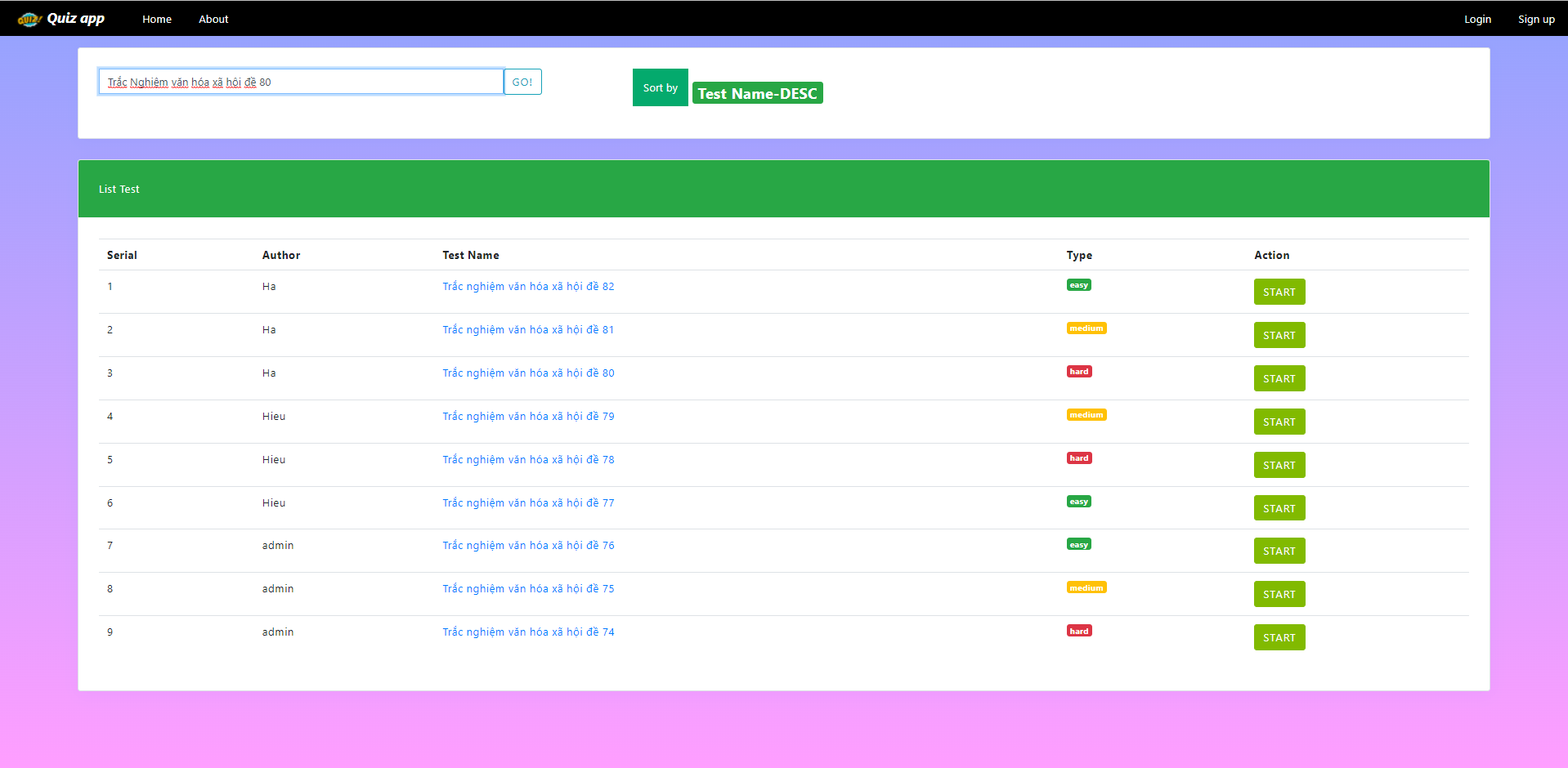
### Trang chủ, đăng nhập, đăng ký và thêm, chỉnh sửa thông tin tài khoản:

### Trang chủ

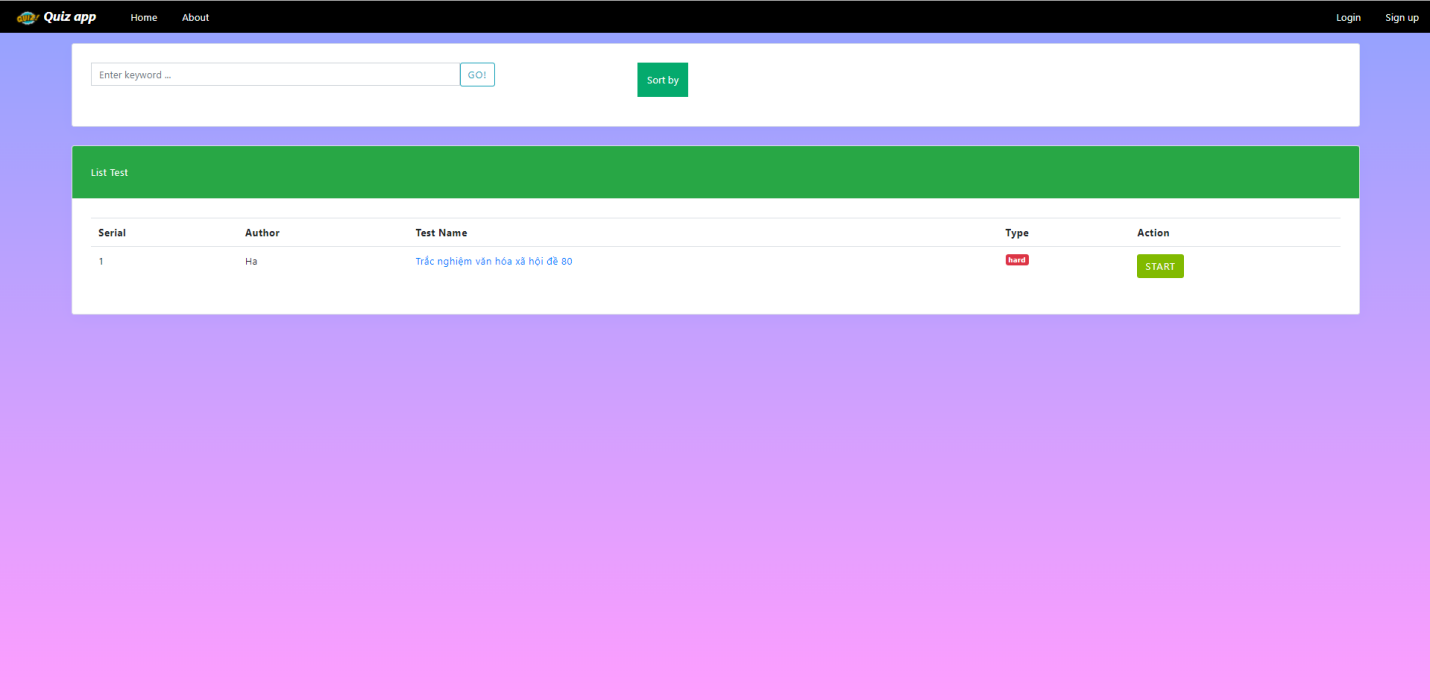
- Trang chủ Home sẽ truy xuất dữ liệu thông tin của các bài trắc nghiệm để xuất hiện trên trang web. Người dùng có thể xem trước các câu hỏi của bài trắc nghiệm bất kỳ bằng cách nhấp vào tên của bài trắc nghiệm, riêng các bài trắc nghiệm chưa có dữ liệu thì người dùng không thể xem trước. 

Hình 3. Giao diện home

- Người dùng có thể tìm kiếm bằng cách gõ tên bài trắc nghiệm cần tìm vào ô Enter keyword, rồi nhấn nút go để bắt đầu tìm kiếm theo từ khóa vừa nhập. Kết quả tìm kiếm trả về là tất cả các tên bài trắc nghiệm có chứa từ khóa đó hoặc giống hoàn toàn với từ khóa tìm kiếm vừa nhập. Nếu người dùng không nhập từ khóa thì không thể bắt đầu tìm kiếm, kích thước tối thiểu của từ khóa là một kí tự, có thể là chữ cái hoặc chữ số. Nếu không có bài kiểm tra nào có trùng từ khóa hoặc chứa từ khóa thì kết quả sẽ trả về danh sách rỗng.

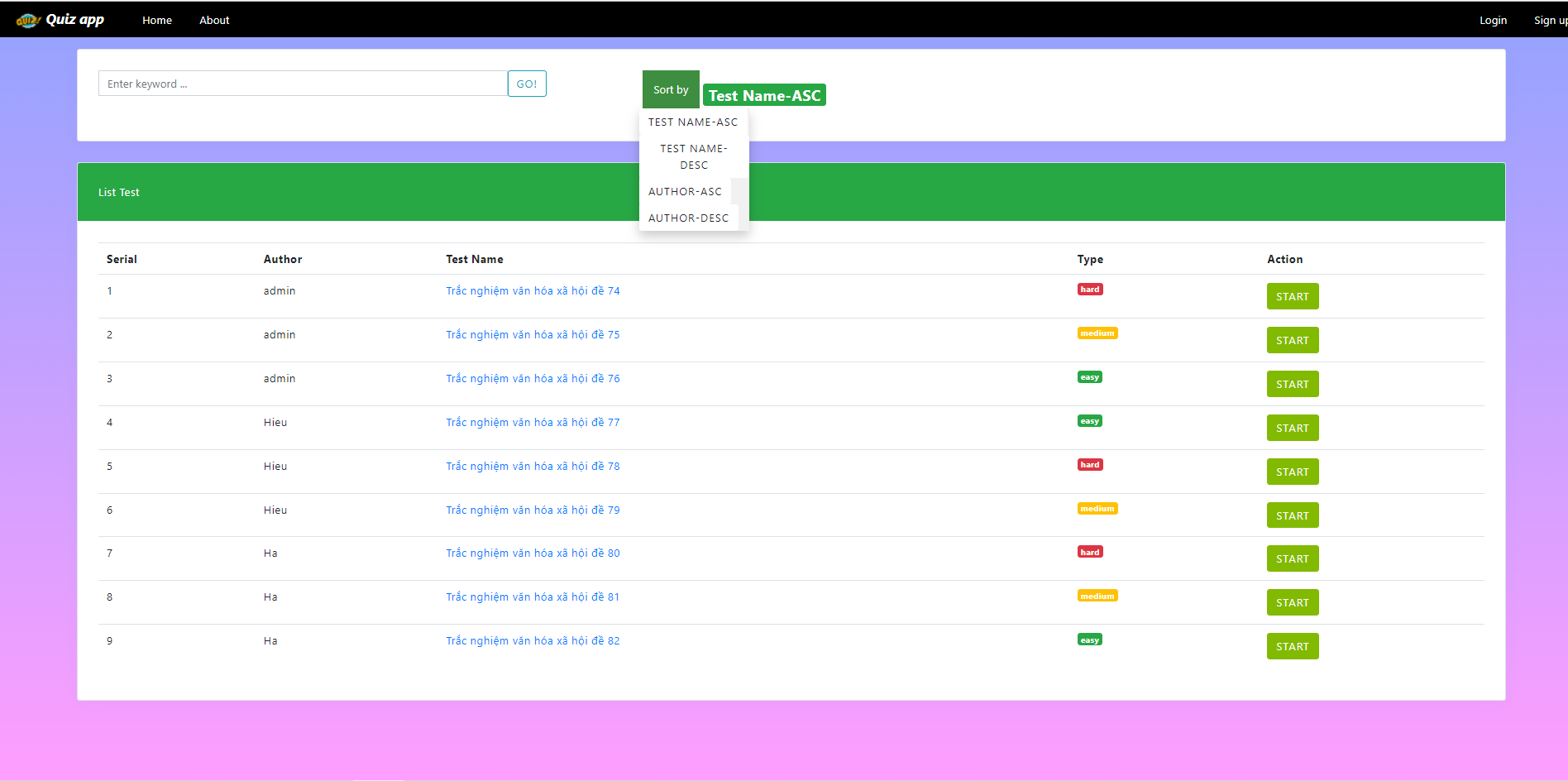


Hình 4. Giao diện khi nhập từ khóa tìm kiếm

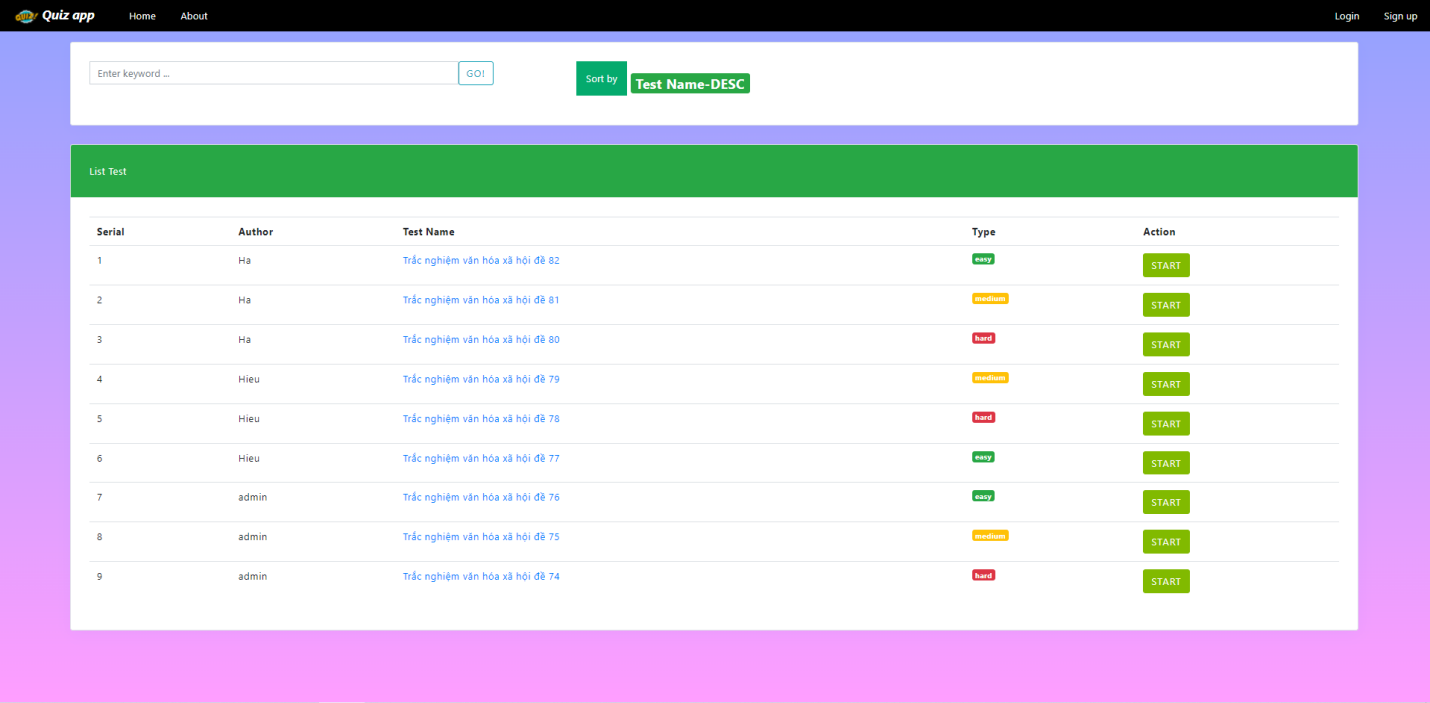


Hình 5. Giao diện sau khi tìm kiếm

- Rê chuột vào sort by để chọn kiểu sắp xếp phù hợp với người dùng. Có các kiểu sắp xếp như test name - asc, author -asc: sắp xếp theo tên tăng dần theo bảng chữ cái của tên bài kiểm tra hay tên tác giả hoặc giảm dần test name -desc, author-desc. Sau đó chọn kiểu sắp xếp thì danh sách bài kiểm tra sẽ tự động sắp xếp theo kiểu đã chọn và không cần phải bấm thêm nút nào. Nếu người dùng không chọn kiểu mới mà bấm sort by thì danh sách bài kiểm tra được sắp xếp mặc định theo kiểu bên phải của nút sort by.



Hình 6. Giao diện chọn kiểu sắp xếp



Hình 7. Giao diện sau khi sắp xếp

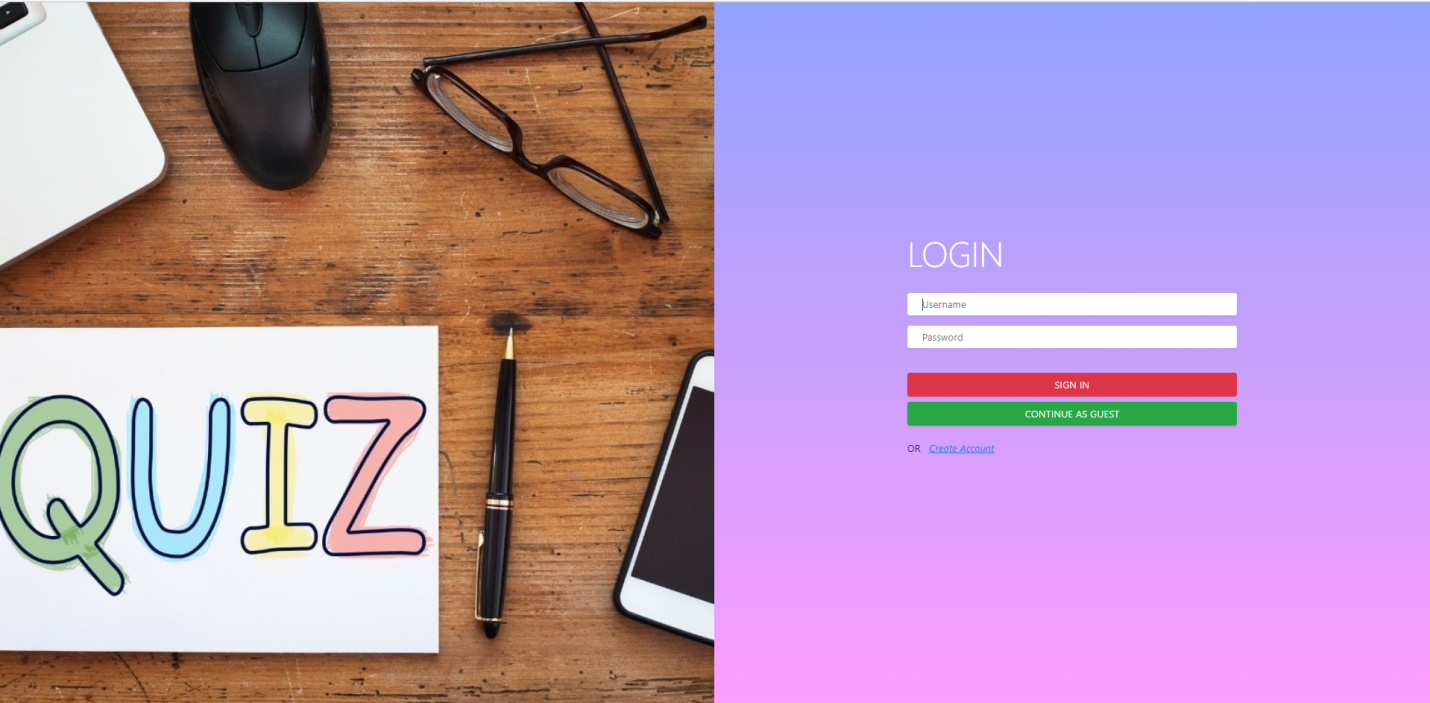
### Đăng nhập tài khoản

- Khi người dùng đăng nhập, tại giao diện đăng nhập sẽ có 3 sự lựa chọn.

+ Lựa chọn 1 là nhập tên tài khoản và mật khẩu và nhấp sign in để đăng nhập. Nếu người dùng nhập đúng tên tài khoản và mật khẩu thì sẽ được chuyển về trang home, nếu sai thì bị báo lỗi.

+ Lựa chọn 2 là không cần đăng nhập nhưng vẫn có thể làm được bài, thì người dùng chỉ cần nhấn nút continue as guest để vào mà không cần đăng nhập tài khoản.

+ Lựa chọn 3 là nếu người dùng muốn đăng nhập nhưng chưa có tài khoản thì nhấp vào create account để chuyển đến trang đăng kí để tạo tài khoản mới.



Hình 8. Giao diện đăng nhập tài khoản

### Đăng kí tài khoản

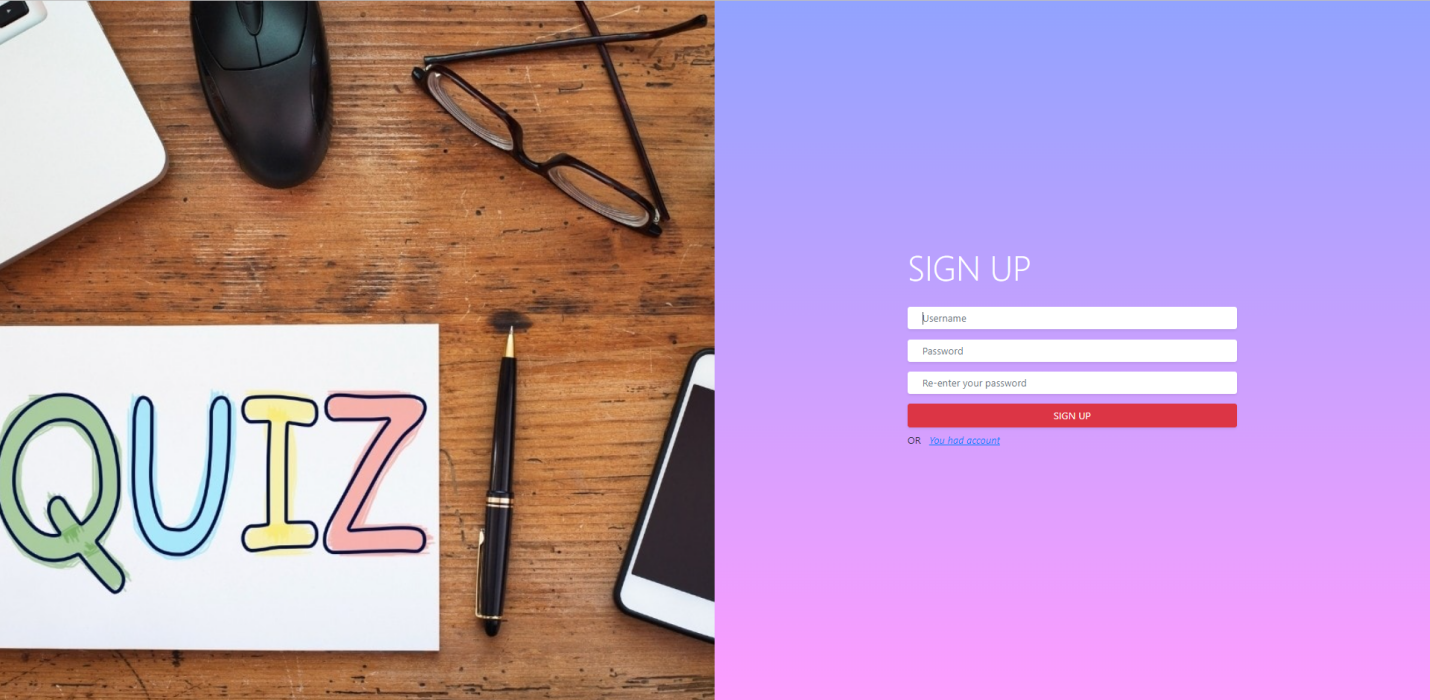
- Tại giao diện đăng kí tài khoản, người có 2 sự lựa chọn. Một là tạo tài khoản mới để sử dụng. Hai là, khi người dung đã có tài khoản và không muốn tạo thêm tài khoản mới thì chỉ cần nhấn vào you had account để chuyển đến trang đăng nhập. Khi đăng kí tài khoản mới sẽ có các ràng buộc :

+ Ràng buộc 1: Tên tài khoản không được chứa khoảng trắng

+ Ràng buộc 2: Mật khẩu có kích thước tối thiểu là 8 kí tự, trong mật khẩu có chứa ít nhất một kí tự chữ hoa, một kí tự chữ thường và một kí tự chữ số.

+ Ràng buộc 3: Mật khẩu và xác nhận mật khẩu phải giống nhau.

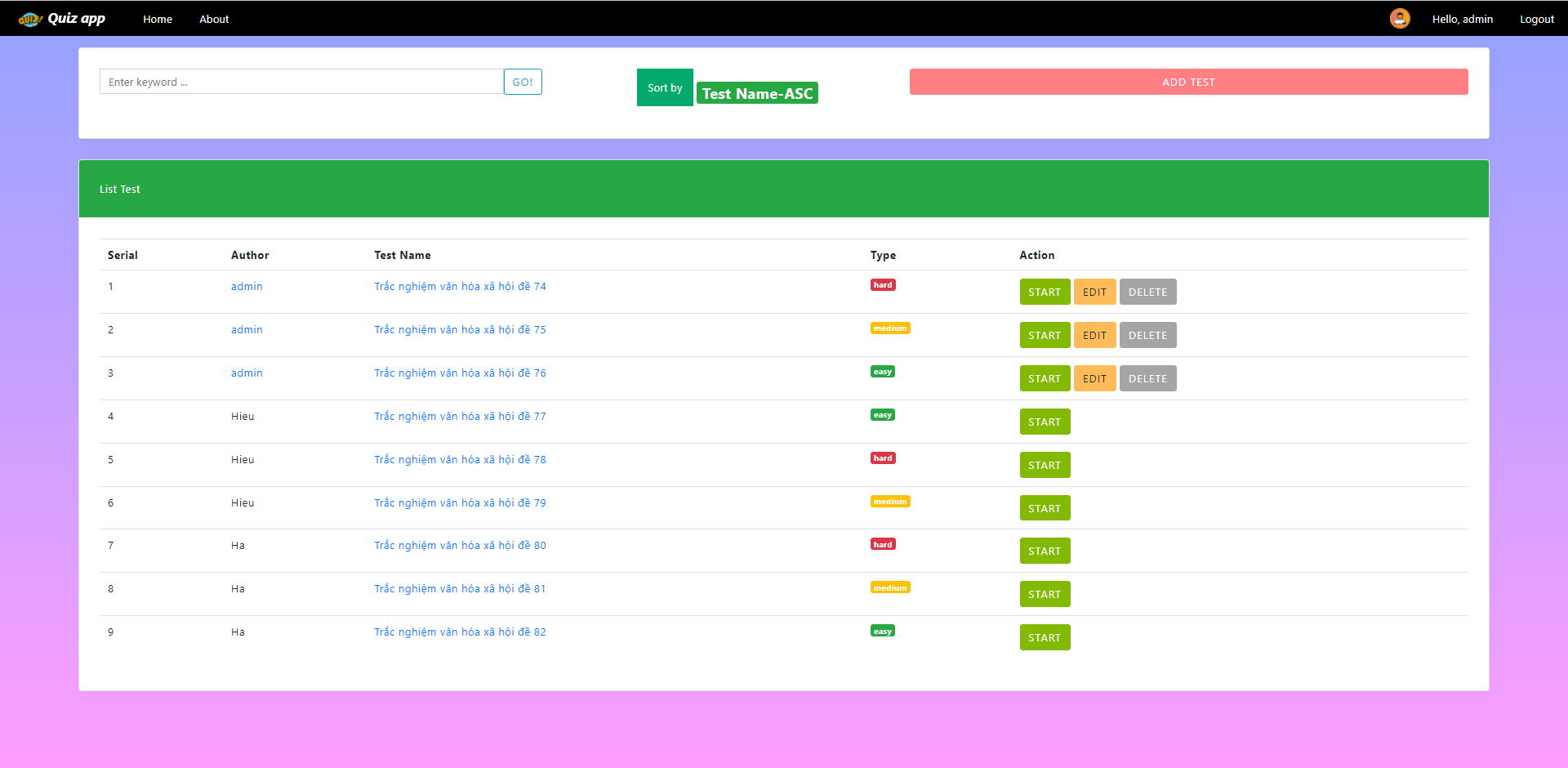
- Sau khi đăng kí tài khoản thành công sẽ được chuyển đến trang đăng nhập để đăng nhập.



Hình 9. Giao diện đăng kí tài khoản

### Chỉnh sửa thông tin cá nhân

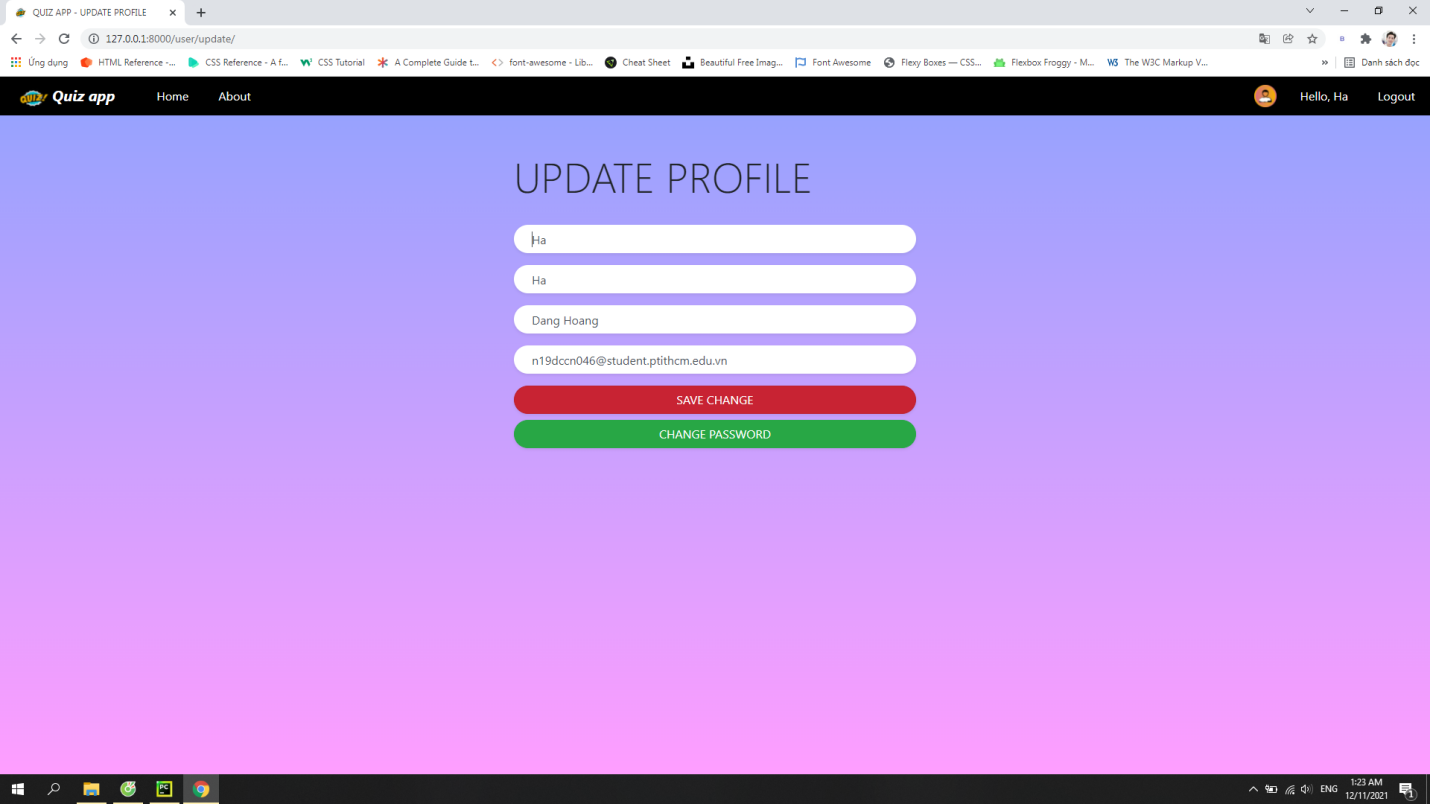
- Màn hình chính sau khi đăng nhập.



Hình 10. Giao diện sau khi đăng nhập thành công

- Các bài trắc nghiệm thuộc quyền sở hữu của user khác nên user hiện tại không có quyền chỉnh sửa hay xóa các bài trắc nghiệm đó.

- Khi người dùng nhấn vào biểu tượng người hoặc chỗ “Hello user” thì màn hình thông tin tài khoản của người dùng sẽ xuất hiện

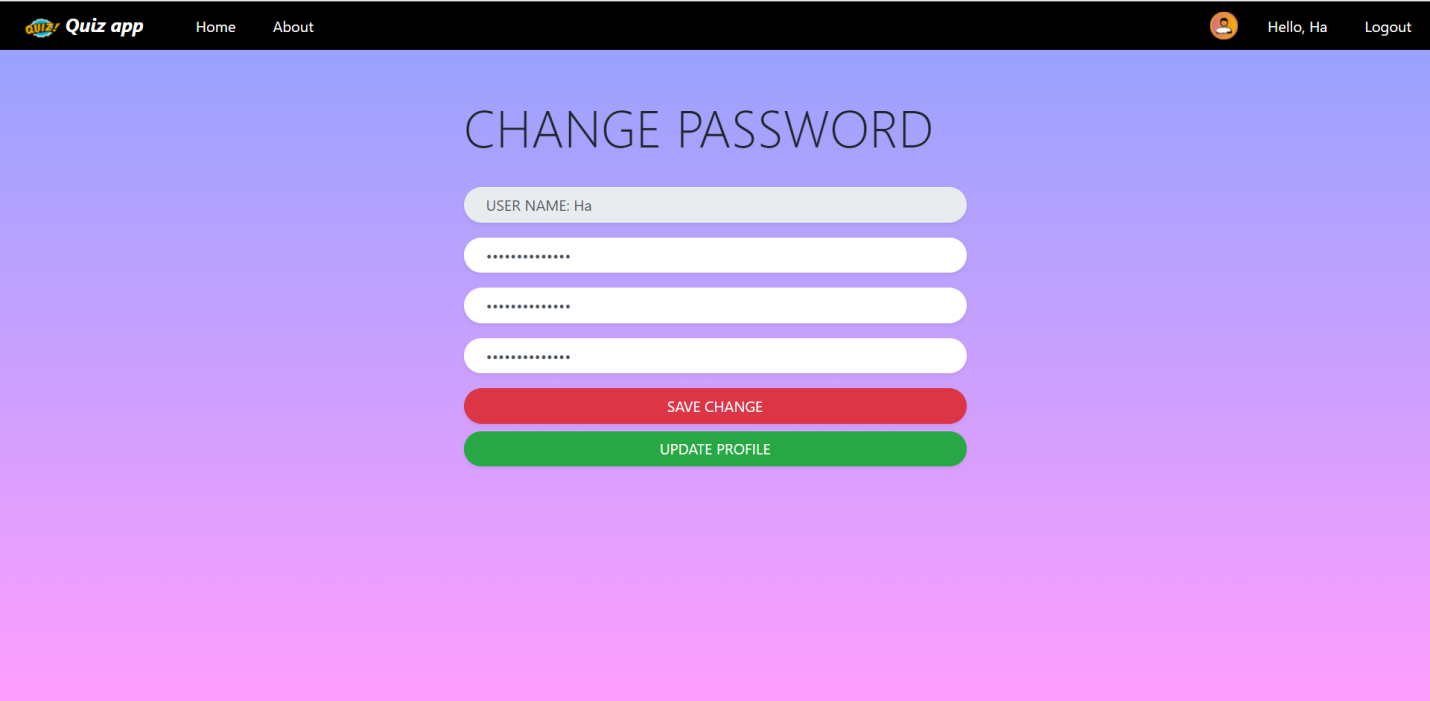


Hình 11a. Giao diện cập nhật thông tin cá nhân

tại đây người dùng sẽ điền các thông tin để thay đổi tài khoản như tên tài khoản; tên, họ; email; thay đổi password….Để thay đổi mật khẩu thì nhấn vào nút change password, sau khi nhấn vào sẽ được chuyển đến trang thay đổi mật khẩu để tiến hành thay đổi.

- Khi thay đổi mật khẩu, bắt buộc phải nhập mật khẩu cũ và phải khớp với mật khẩu cũ trong cơ sở dữ liệu. Sau đó sẽ nhập mật khẩu mới và xác nhận mật khẩu mới. Các ràng buộc mật khẩu mới giống như khi tạo tài khoản, mật khẩu mới có độ dài tối thiểu là 8 kí tự, có chứa ít nhất một kí tự chữ hoa, một kí tự chữ thường, một kí tự chữ số.

- Tạo giao diện này nếu không muốn thay đổi mật khẩu nữa thì nhấn vào nút update profile chuyển sang trang chỉnh sửa thông tin cá nhân.

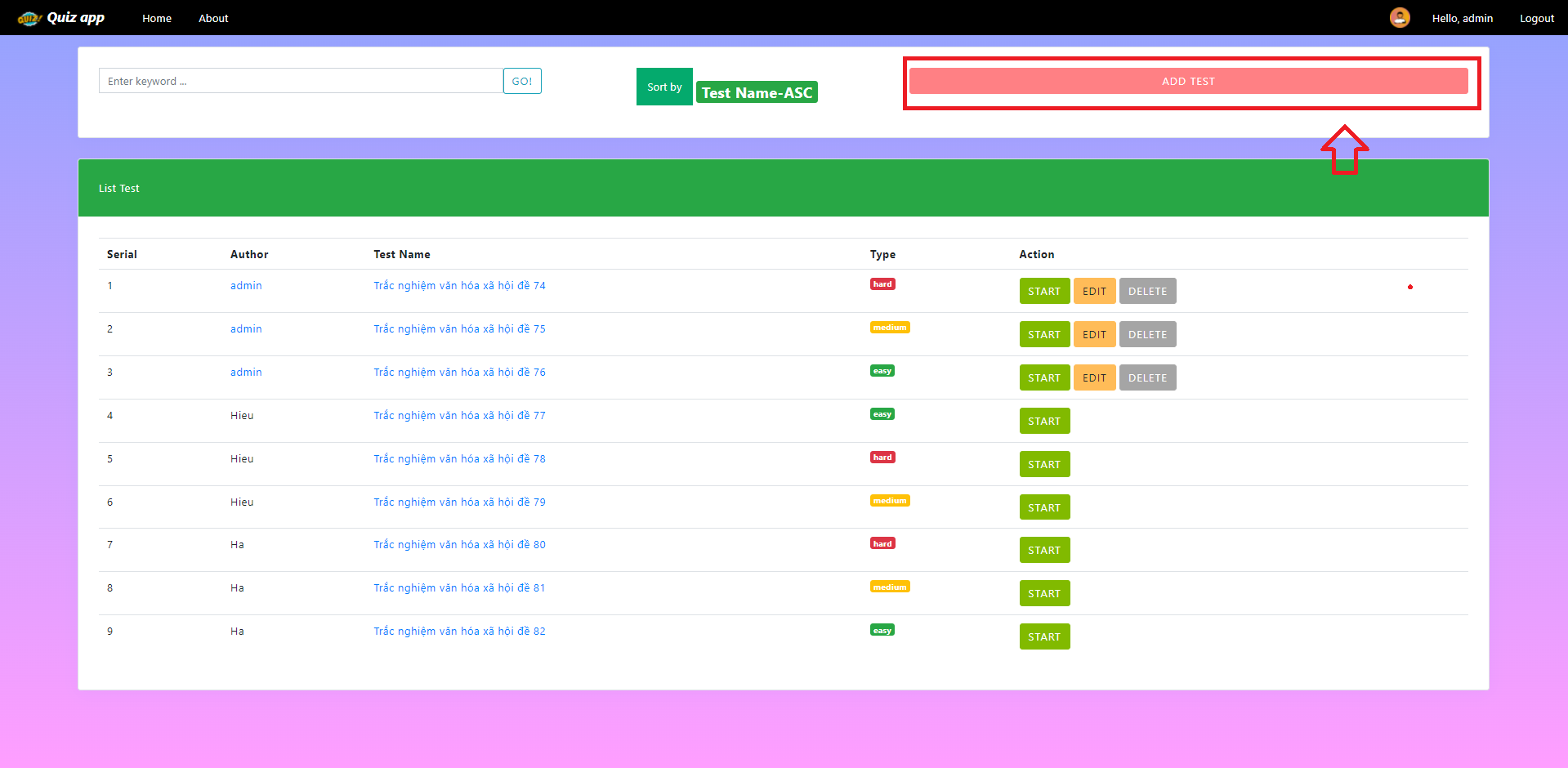


Hình 11b. Giao diện thay đổi mật khẩu

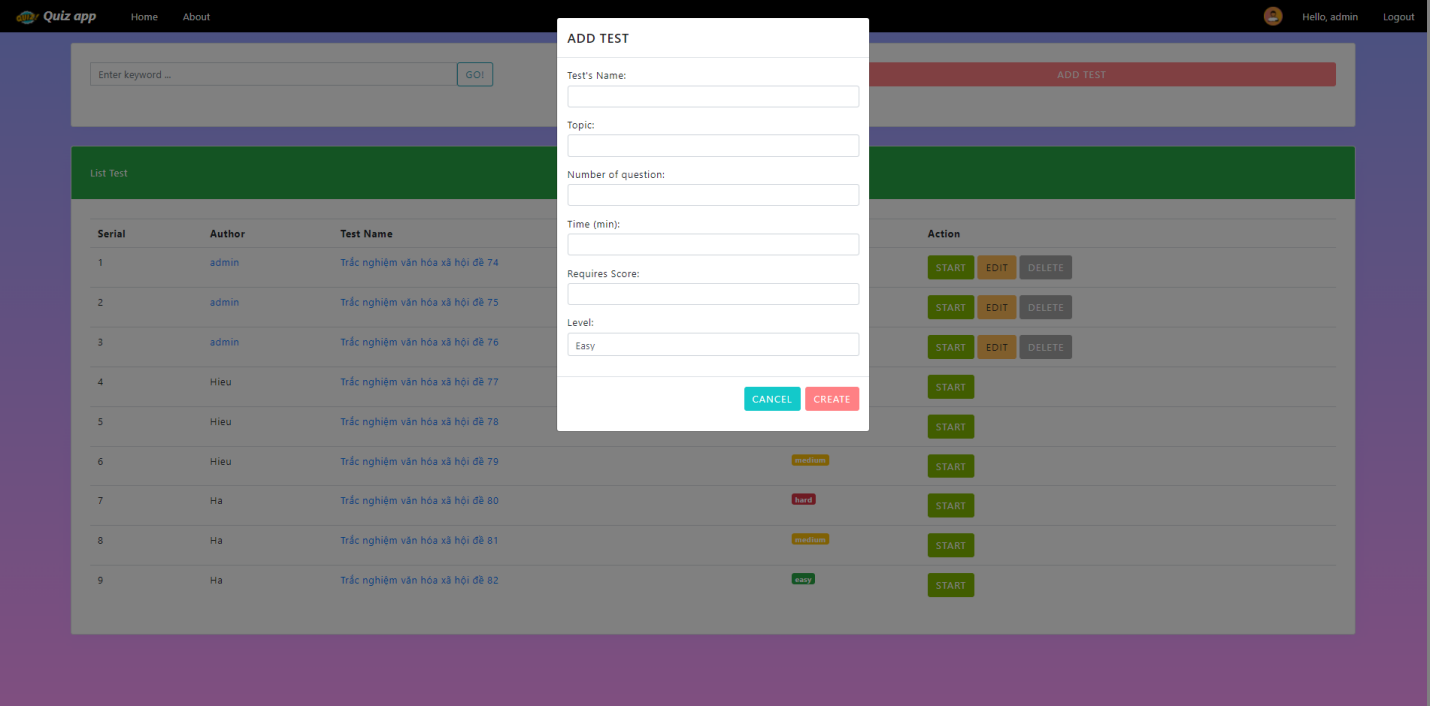
### Tạo và xem trước bài trắc nghiệm:

- Bây giờ chúng ta bắt đầu tạo một bài trắc nghiệm bằng Quiz App.

- Nhấn vào nút ADD TEST và điền thông tin của bài trắc nghiệm vào.

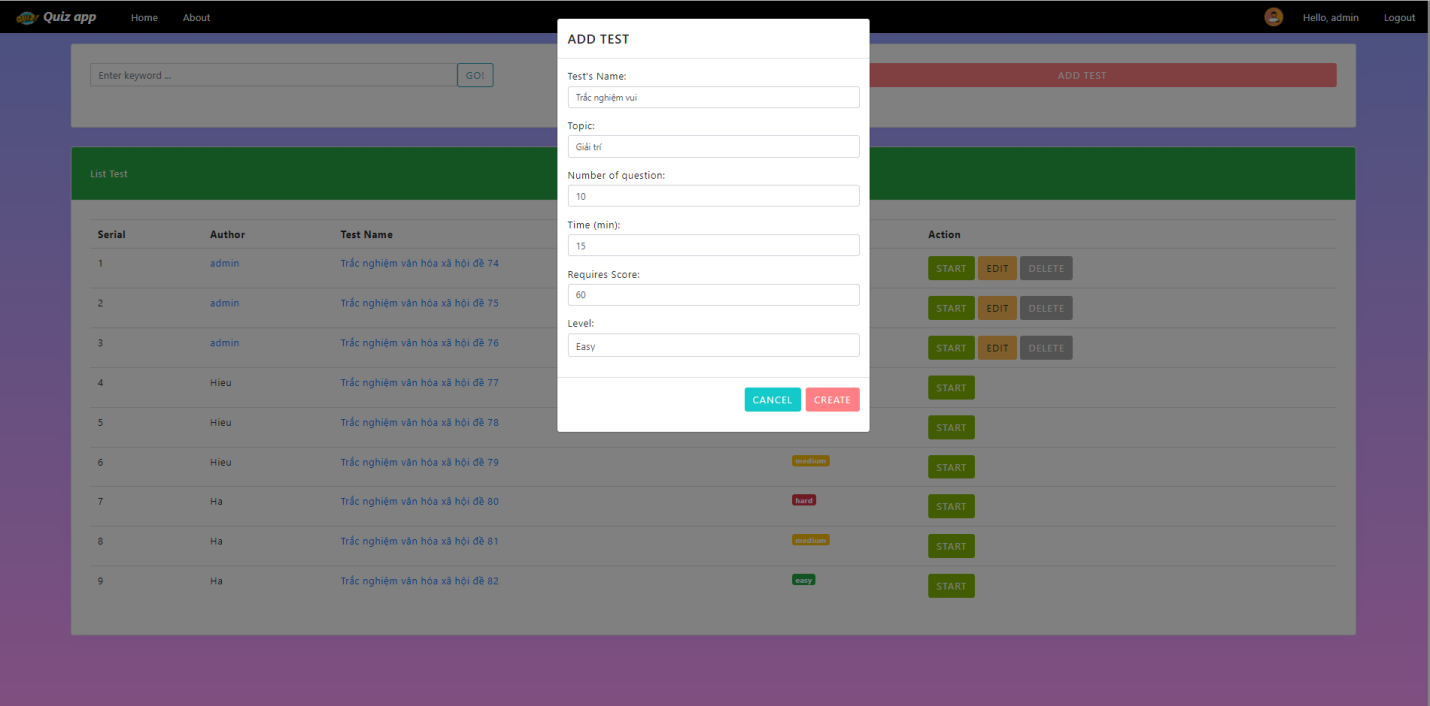


Hình 12a. Nút thêm bài kiểm tra



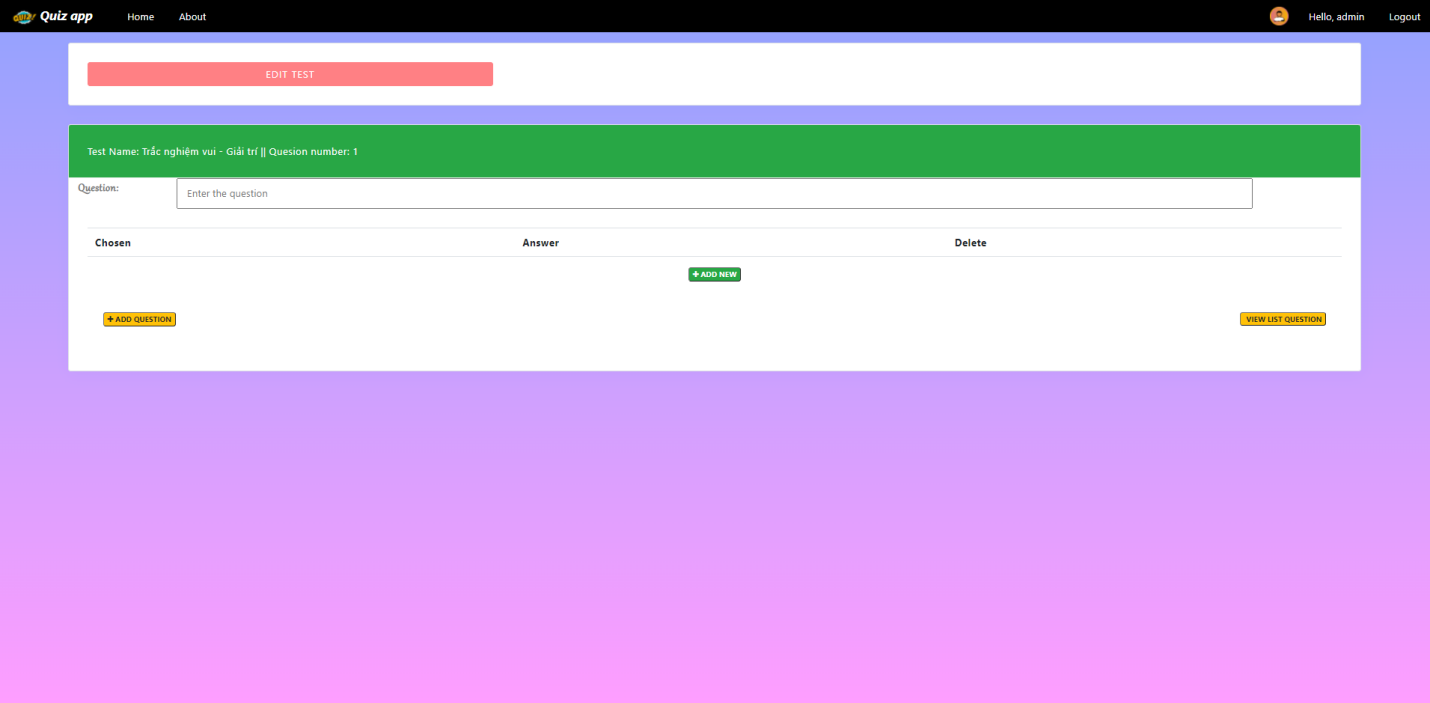
Hình 12b. Giao diện thêm bài kiểm tra

- Sau khi điền các thông tin bài trắc nghiệm xong và bấm vào nút create thì sẽ hiện màn hình như sau để điền thông tin câu hỏi vào. Khi nhập nội dung bài kiểm tra, thì không được bỏ trống bất cứ trường thông tin nào. Số câu hỏi, thời gian, ràng buộc điểm vượt qua không âm. Ràng buộc điểm vượt qua tối đa là 100. Thời gian làm bài kiểm tra tính bằng phút.



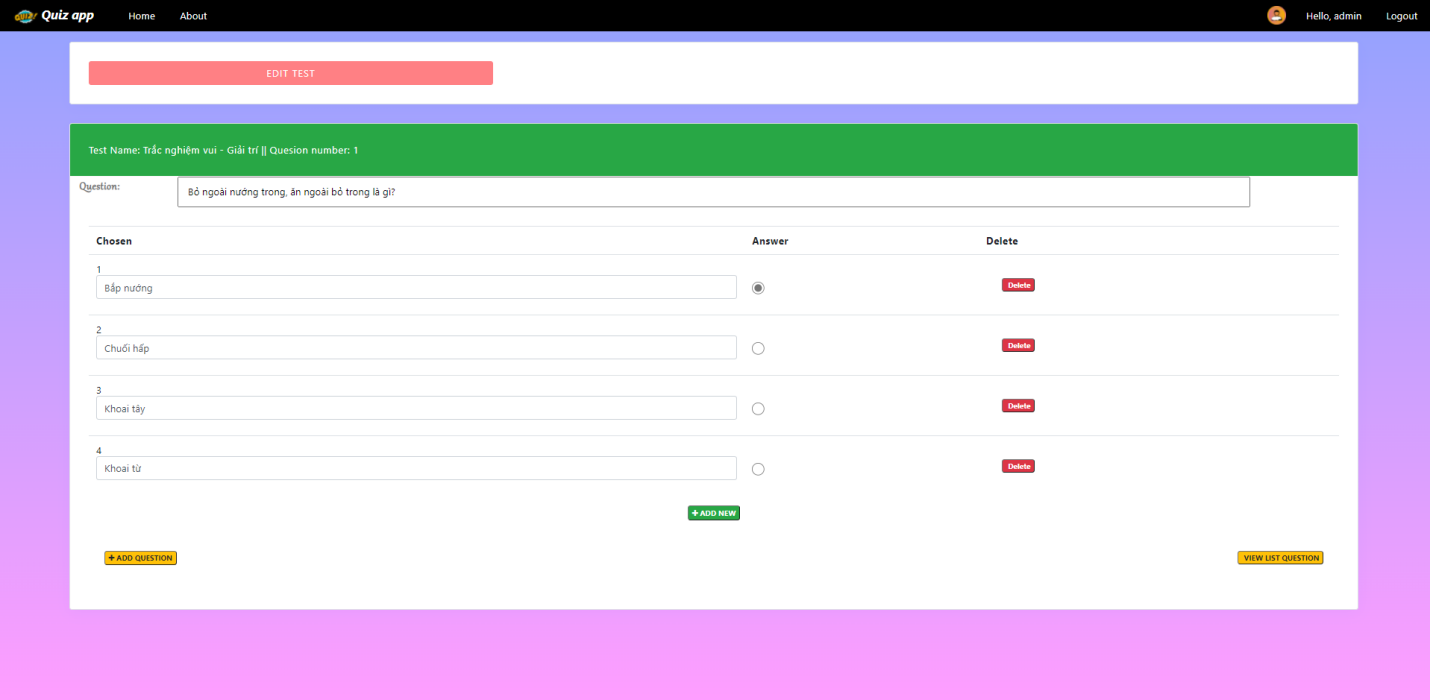
Hình 12c. Màn hình sau khi nhập thông tin bài kiểm tra

- Sau khi nhập thông tin bài kiểm tra, thì sẽ được chuyển đến trang thêm câu hỏi cho bài kiểm tra. Tại đây, người dùng vừa có thể thêm câu hỏi, vừa có thể chỉnh sửa thông tin bài kiểm tra thông qua nút edit test. Khi sửa sẽ có các ràng buộc giống lúc thêm và thêm một ràng buộc, số lượng câu hỏi không được ít hơn số câu hỏi hiện có trong cơ sở dữ liệu của bài kiểm tra đó.



Hình 13. Giao diện thêm câu hỏi

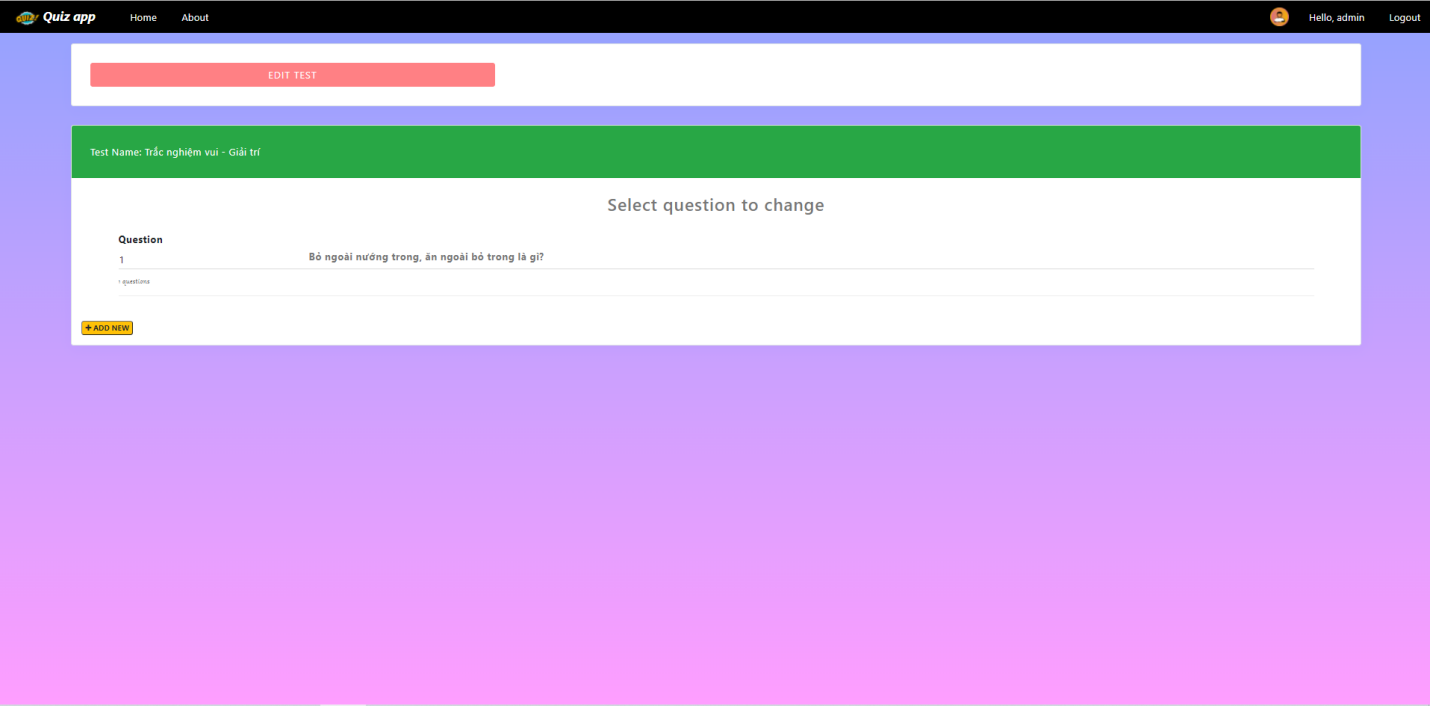
- Đầu tiên người dùng điền câu hỏi vào ô Question và bấm add new để thêm các sự lựa chọn vào.



Hình 14. Màn hình sau khi đã điền câu hỏi và đáp án

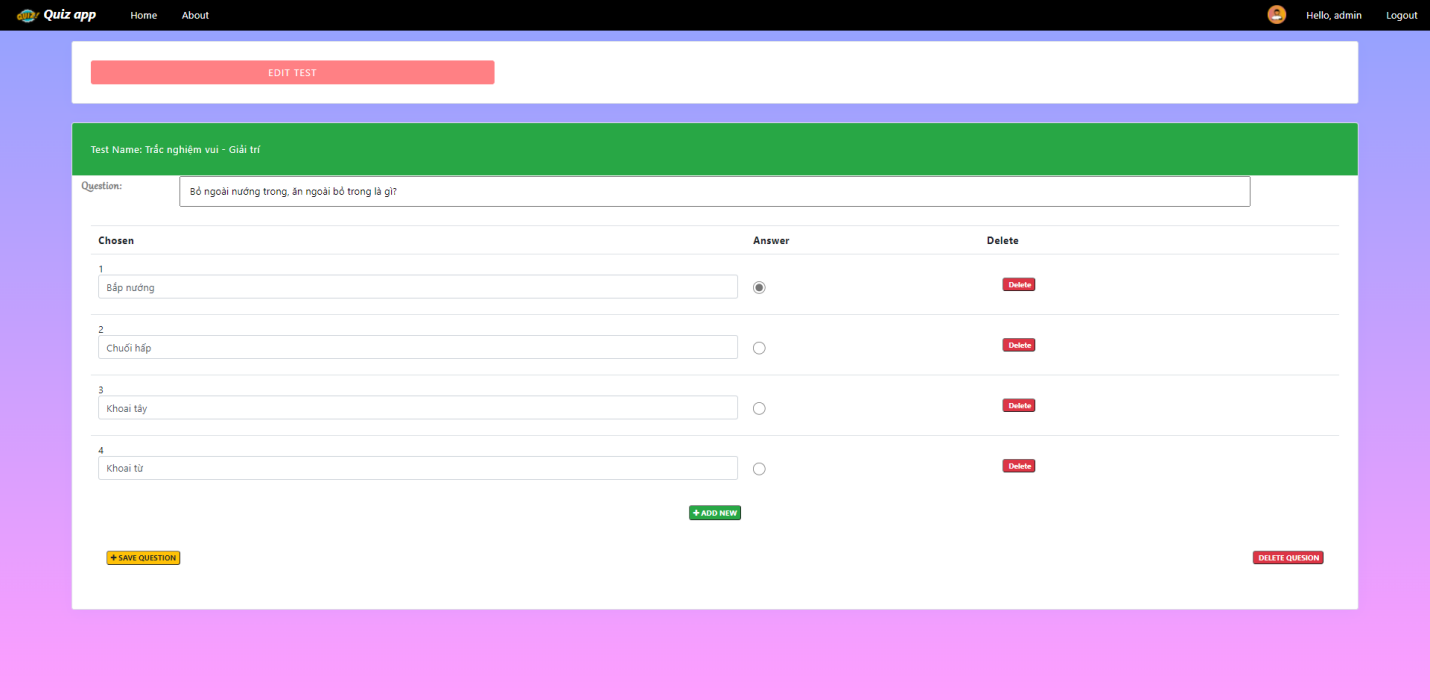
- Với sự lựa chọn được tick vào cột Answer sẽ chính là câu trả lời, người dùng có thể xóa các sự lựa chọn bằng nút delete bên cạnh nhưng nên xóa từ dưới lên vì nếu không sẽ gây lỗi cho sự lựa chọn đúng. Khi người dùng hoàn tất câu hỏi thì người dùng ấn add question để thêm câu hỏi vào cơ sở dữ liệu. Khi thêm, người dùng không được bỏ trống bất kì thông tin nào đăch biệt là chọn đáp án đúng. Khi thêm câu hỏi, thì không giới hạn số lượng câu trả lời của câu hỏi đó, nhưng tối thiểu phải có một câu trả lời, nếu không có câu trả lời nào thì không thể thêm được câu hỏi. Bên cạnh đó người dùng có thể xem các câu hỏi đã được thêm bằng nút view list question.

- Dưới đây là giao diện khi bấm vào View list question. Tại giao diện này người dùng có thể xem nội dung câu hỏi bằng cách nhấn vào tên câu hỏi trong danh sách câu hỏi.



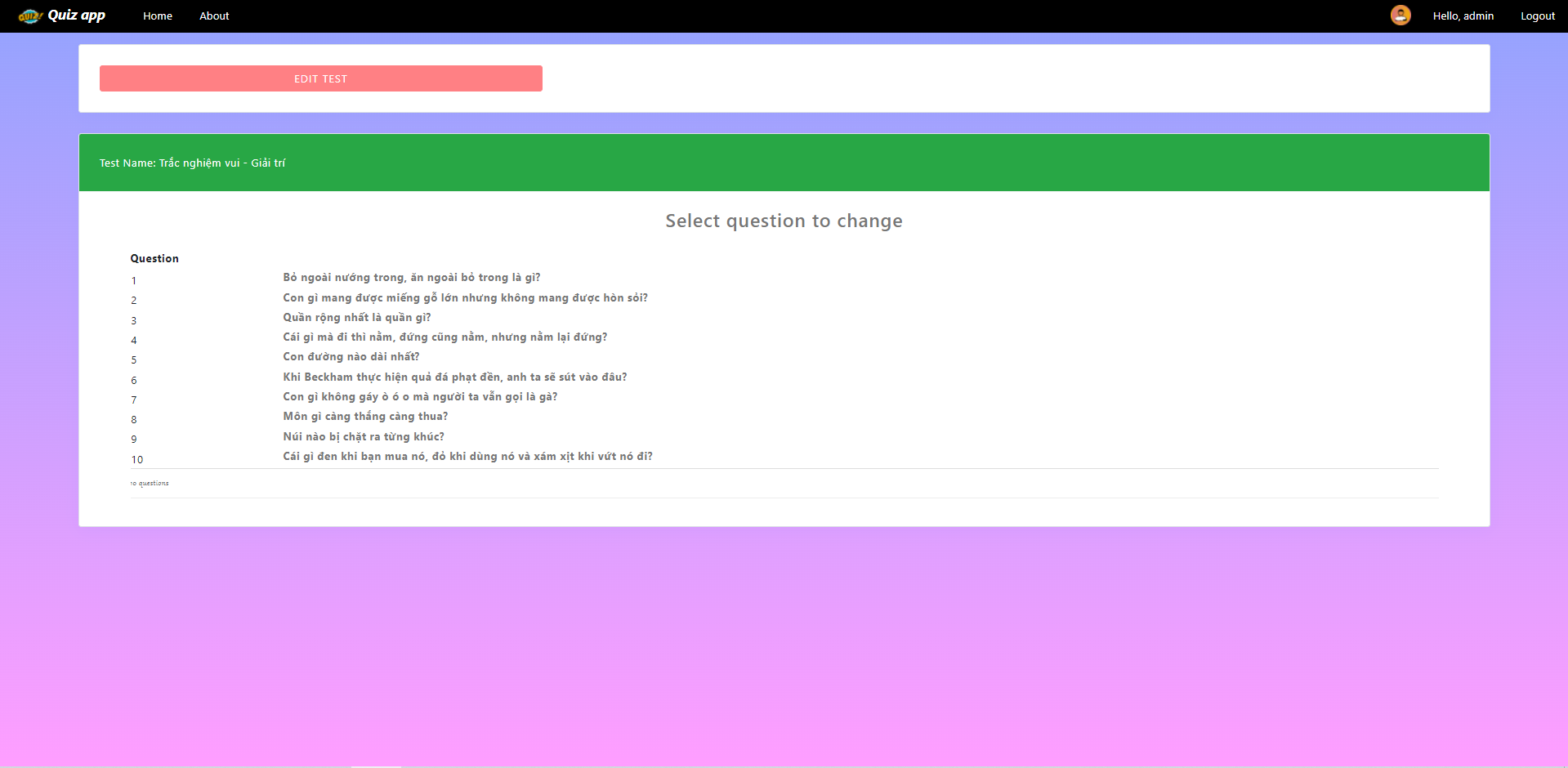
Hình 15. Giao diện xem danh sách câu hỏi

- Và khi người dùng xem lại các question đã thêm vào thì người dùng sẽ có thêm lựa chọn sẽ xóa câu hỏi đó ra khỏi danh sách câu hỏi của bài trắc nghiệm đó hay không.



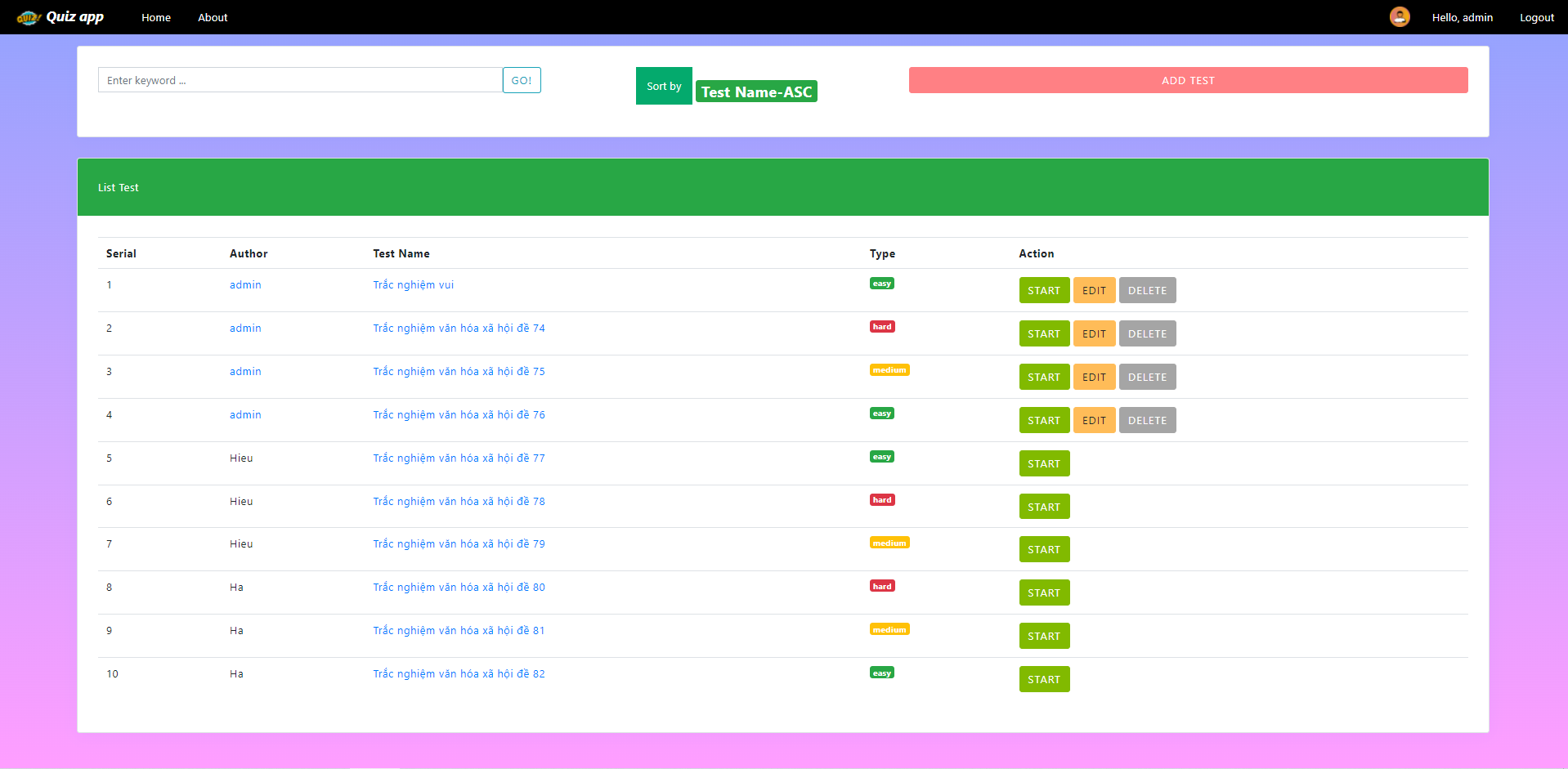
Hình 16. Giao diện xem nội dung câu hỏi

- Sau khi thêm đủ số lượng câu hỏi đã ghi trong thông tin bài trắc nghiệm thì chúng ta sẽ có giao diện sau.



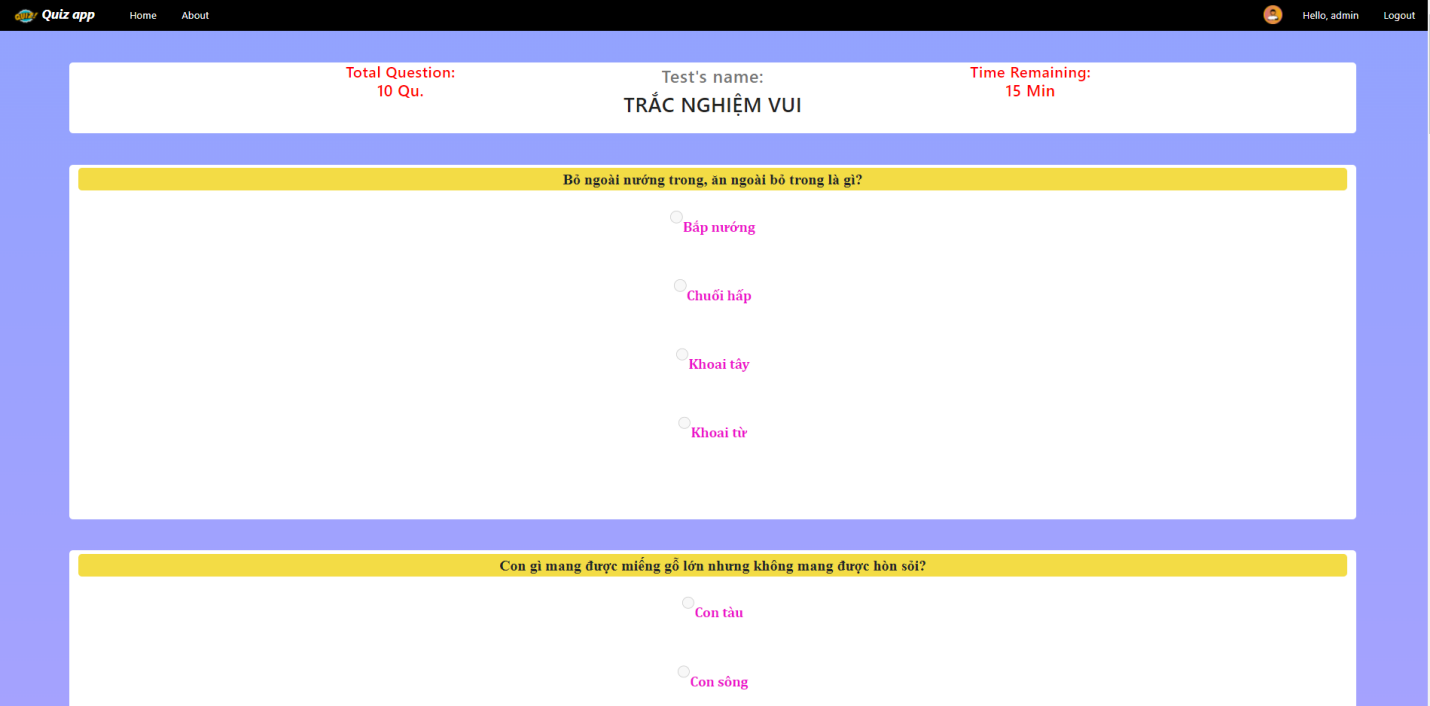
Hình 17. Giao diện xem danh sách câu hỏi đầy đủ

- Bấm vào home để trở về trang chủ và bài trắc nghiệm của người dùng đã được tạo



Hình 18. Màn hình chính sau khi đã thêm bài kiểm tra

- Người dùng có thể bấm vào tên của bài trắc nghiệm để xem câu hỏi và sự lựa chọn có giống với thông tin cũng như các câu hỏi người dùng đã làm hay không.



Hình 19a. Giao diện xem nội dung bài kiểm tra phần đầu

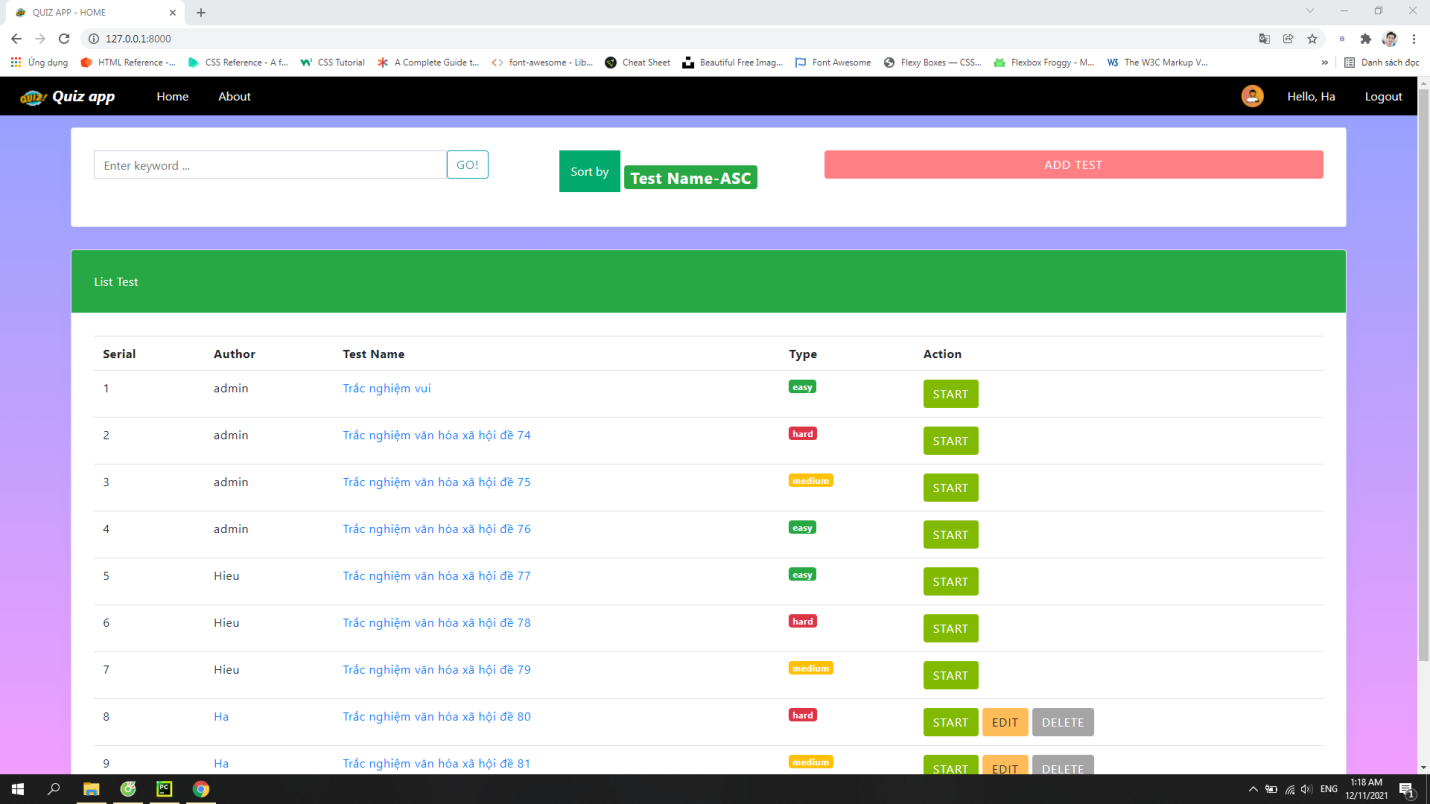
- Tại giao diện xem nội dung bài kiểm tra bạn có thể nhấn nút export file để xuất file nội dung bài kiểm tra.



Hình 19b. Giao diện xem nội dung bài kiểm tra phần cuối

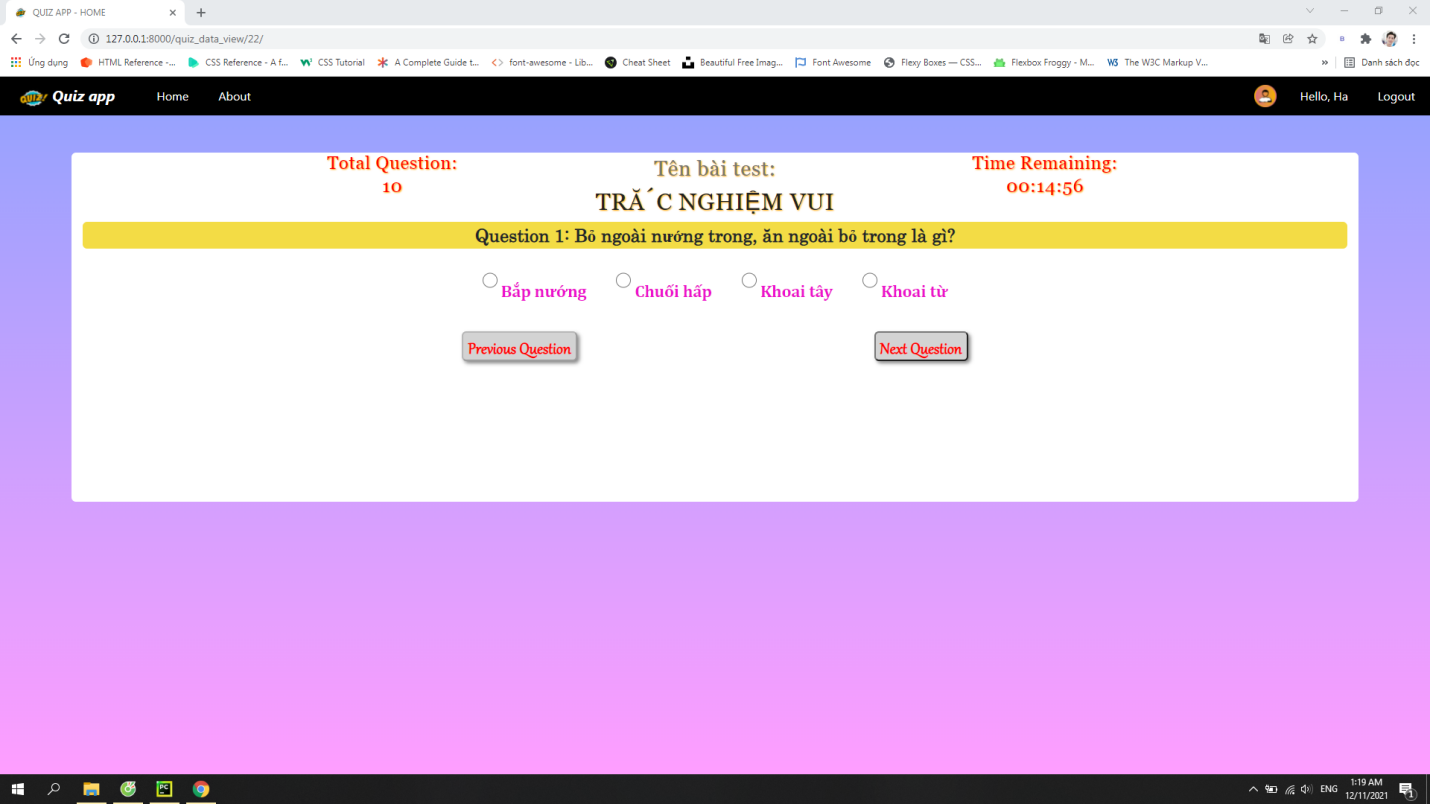
### Thực hiện bài trắc nghiệm:

- Để bắt đầu làm bài trắc nghiệm người dùng bấm vào nút start ở danh sách bài trắc nghiệm.



Hình 20a. Nút bắt đầu làm bài

- Người dùng sẽ có được giao diện làm bài như sau:



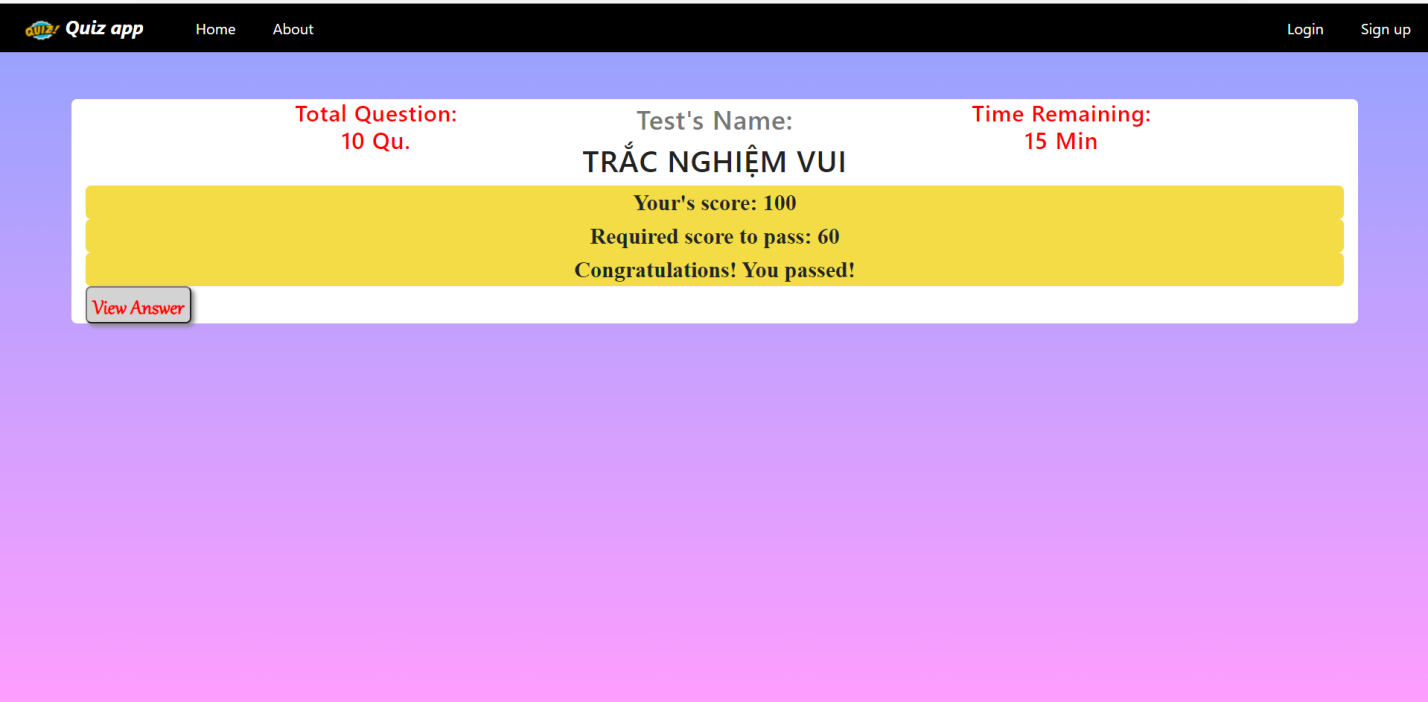
Hình 20b. Giao diện làm bài

với bên trái là tổng số câu hỏi, bên phải là đồng hồ đếm ngược thời gian làm bài trắc nghiệm. Người dùng phải hoàn thành xong câu trắc nghiệm hiện tại trước khi bấm câu kế tiếp ở nút Next Question. Khi đến câu cuối cùng hoặc hết thời gian làm bài kết quả sẽ hiện ra cùng một nút submit để xem kết quả bài làm.



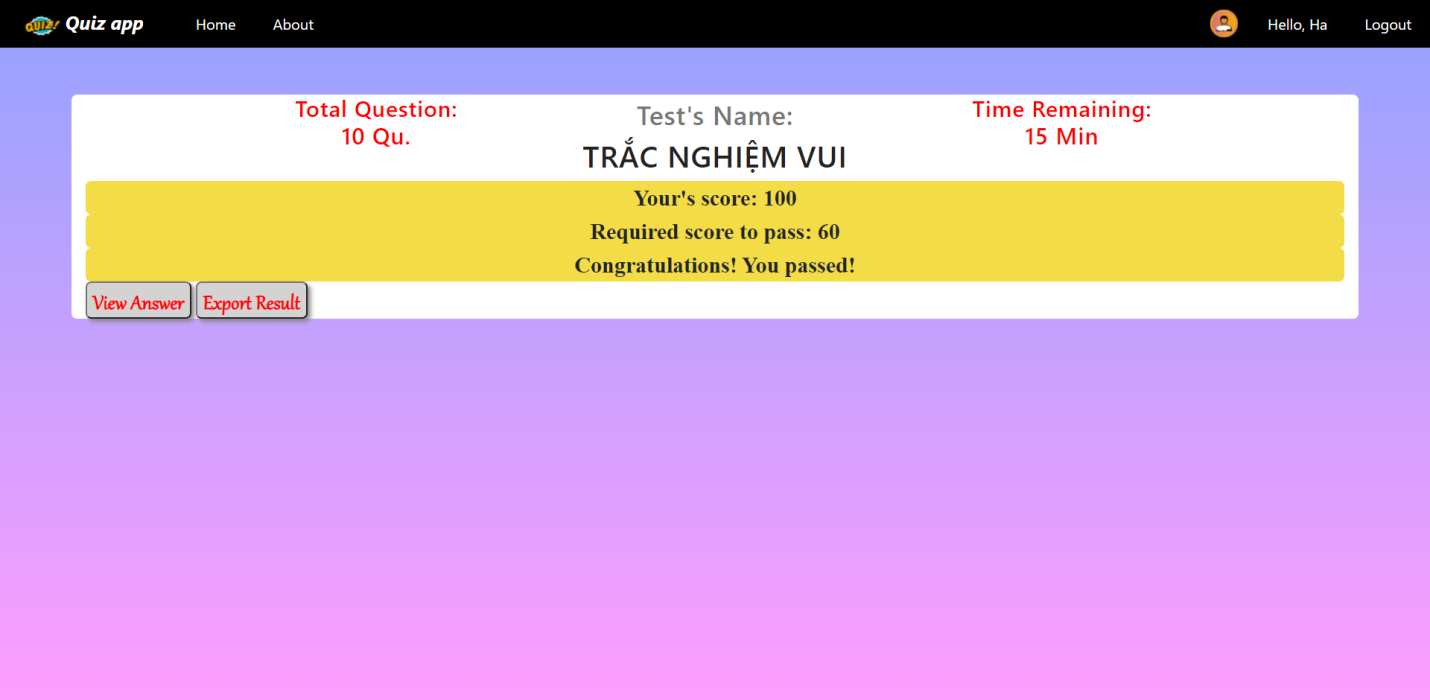
Hình 20c. Giao diện khi làm đến câu cuối cùng

**-** Nếu không đăng nhập thì chỉ xem được kết quả bài làm, và kết quả này sẽ không được lưu lại.



Hình 20d. Giao diện xem kết quả khi không đăng nhập

- Nếu có đăng nhập thì kết quả sẽ được lưu, và tại giao diện xem kết quả có thêm nút export result để xuất file điểm.



Hình 20e. Giao diện xem kết quả khi có đăng nhập

# CHƯƠNG 4: TÀI LIỆU THAM KHẢO:

## Sách tham khảo:

## Link tham khảo:

**-** [**https://www.djangoproject.com/**](https://www.djangoproject.com/)

**-** [**https://www.w3schools.com/**](https://www.w3schools.com/)

**-** [**https://docs.python.org/**](https://docs.python.org/)