**K.J. Somaiya College of Science & Commerce Vidyavihar (E),**

**Mumbai – 400077.**



**Autonomous**

**Affiliated to University of Mumbai**

**PROJECT REPORT ON**

**Question Generation System using NLP and Django**

**Questioniz**

**SUBMITTED BY**

**Jagrut Gala**

**T.Y.BSC (COMPUTER SCIENCE)**

**UNIVERSITY OF MUMBAI  
2021 – 2022**

**PROJECT GUIDE**

**Mr. Mohammed Bilal Shaikh**

**K.J.Somaiya College of Science and**Somaiya Trust **Commerce**

Vidyavihar, Mumbai-400077

**Autonomous- Affiliated to University of Mumbai**

# **Department of Computer Science**

**CERTIFICATE**

*This is to certify that Jagrut Gala of T.Y.B.Sc. [Computer Science] Semester - V, Seat no. 2109805 has successfully completed the project entitled as Question Generation System using NLP and Django - Questioniz during the academic year 2021-2022.*

***Internal Guide***

***Date:***

***Course Coordinator of CS Department College seal Sign of Examiner***

***Date: Date***

**Acknowledgement**

First and foremost, I would like to thank our project guide Mr. Mohammed Bilal Shaikh who guided us in doing these projects. He provided us with invaluable advice and helped us in difficult periods. His motivation and help contributed tremendously to the successful completion of the project.

Besides, we would like to thank all the teachers who helped us by giving us advice and providing the equipment which we needed.

Finally, we would like to thank my friends and classmates who helped and motivated us to work on this project.

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Sr No** | **Praticular** | **Pg No** |
| 1 | Overview | 5 |
| 2 | Description of Existing System | 6 |
| 3 | Limitations of present system | 7 |
| 4 | Proposed System and its advantages | 8 |
| 5 | Stakeholders | 9 |
| 6 | Gantt chart | 10 |
| 7 | Technologies Used | 11 |
| 8 | Event Table | 12 |
| 9 | Use case Diagram | 13 |
| 10 | Entity-Relationship Diagram | 14 |
| 11 | Class Diagram | 15 |
| 12 | Data Flow Diagram | 16 |
| 13 | Sequence Diagram | 17 |
| 14 | Menu Tree | 18 |
| 15 | State Chart Diagram | 19 |
| 16 | System Coding | 20-44 |
| 17 | Screen Layouts and Report Layouts | 45-49 |
| 18 | Future Enhancements | 50 |
| 19 | Bibliography | 51 |

**Overview**

In recent times as the number of students appearing for competitive exams are increasing yet the number of admissions available in institutions remain the same. This is resulting in exams getting more intense and competitive.

So students need to practice more to master the subject, but teachers can give them a certain number of questions to students to practice. Even if students uses reference books questions to practice the book provides only a limited number of questions.

Question Generation systems are very useful in these types of situations especially trying times when education is online.

**Description of Existing System**

Existing Systems of Question Generation systems involve [Questgen](https://questgen.ai/), [Quillionz](https://www.quillionz.com/), [QuizEasy by KG-1510](https://github.com/KG-1510/QuizEasy) and [Question-Generation by KristiyanVachev](https://github.com/KristiyanVachev/Question-Generation) which are used by Teachers and Schools to generate practice worksheets or quiz’s for their students. Questgen is AI-based question generation web-application made using questgen api.

Some systems specialize in generating a specific type of questions like multiple-choice-question(MCQ), open questions, True/ False questions, etc. Many systems also provide functionality to export the generated quiz in a PDF format and download it.

Questioniz is a web-application that uses Natural language Processing(NLP) to generate questions.

**Limitations of Existing Systems**

Existing systems like Questgen and Quillionz are good and generate high quality questions but are locked behind a monthly subscription fees. They are also proprietary and the user has no way to control over what new feature will be added.

Other question generation systems which are free are self hosted and user needs technical skills to install and run the application. eg. QuizEasy by KG-1510 and Question-Generation by KristiyanVachev.

**Proposed System and Its Advantages**

This proposed work is made using Django Framework with sqlite3 database. For Natural Laguage Processing the system uses NLTK library.

The objective of this question generation system is to generate fill in the blank questions from the text provided by the user. It aims to generate the questions by following the given procedure:

1. Cleaning the given text
2. Finding keywords using TF-IDF algorithm
3. Swapping keywords with blanks
4. Presenting the list of questions

Some advantages of the system are:

* Highlights derived sentence
* New questions every time
* Easy to use
* Open Source
* Clean and modern user interface

**Stakeholders**

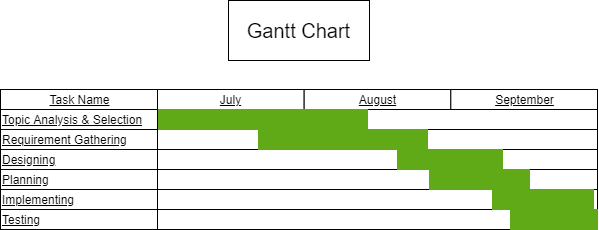
Individuals and organizations who are actively involved in the project, or whose interests may be positively or negatively affected as a result of project execution or successful project completion.

The project's stakeholders are as follows:

* Developer: The developer is the main person in charge of the making the system. All features of the system are well-understood by the developer. The developer is in responsible of keeping track of the information provided by the users.
* End User: The end user is the person who will use this application to benefit from themselves.

**Gantt Chart**

Gantt charts are a type of bar chart that depicts a project's progress. The tasks to be accomplished are shown on the vertical axis, while the time intervals are listed on the horizontal axis. The width of the horizontal bars in the graph show the time of each action.



**Technologies Used**

* Programming Environment-
  + **Operating System**: Windows
  + **Language**: Python
  + **Code Editor**: Visual Studio Code
  + **Browser**: Chrome
* Libraries and Frameworks used-
  + **Django**: Django is a high-level Python web framework that enables rapid development of secure and maintainable websites.
  + **SQLite3**: SQLite is a self-contained, file-based SQL database. SQLite comes bundled with Python and can be used in any of your Python applications without having to install any additional software.
  + **NLTK**: The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).

**Event Table**

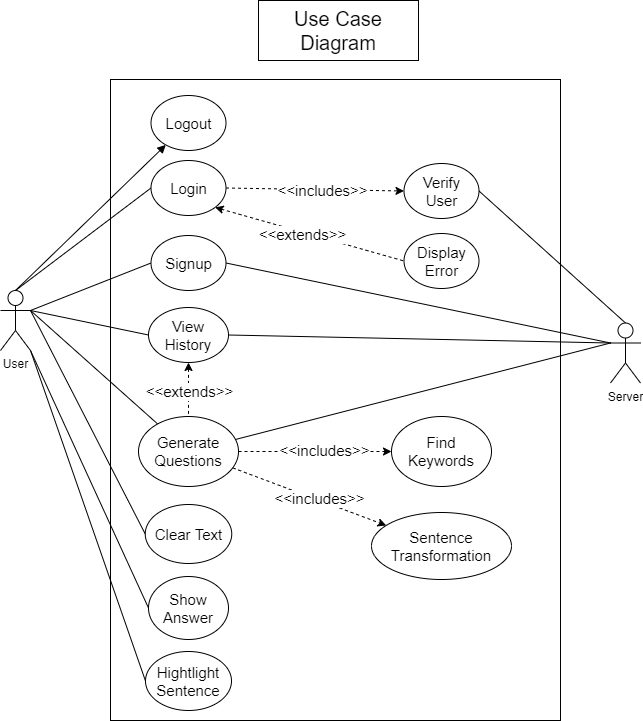
Event Table is a catalogue of use cases listed by event. Contains detailed information

* Trigger: Signal that indicates an event has occurred.
* Source: External agent that initiates event and supplies data for the event.
* Response: Output produced by the system.
* Destination: External agent that receives the response.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Event | Trigger | Source | Use Case | Response | Destination |
| user wants to log in | username and password | user | Login | session | End User |
| user wants to signup | username and password | user | Signup | session | End User Database |
| user wants to see answer | answer visibility | user | Show Answer | answer | End User |
| user wants to highlight orginal sentence | sentence index | user | Hightlight Sentence | original sentence | End User |
| user wants to see history | username | user | View History | previous 10 submissions | End User |
| user wants to clear text | - | user | Clear Text | empty text box | End User |
| user wants to generate questions | text | user | Generate Questions | questions | End User Database |
| user want to see a reult | text | user | Generate Questions | questions | End User Database |
| user want to logout | session | user | Logout | - | End User |

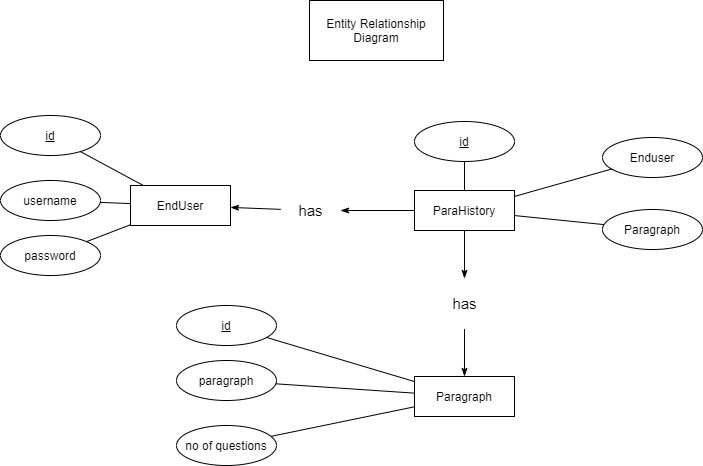
**Use Case Diagram**

Use Case Diagrams are used to summarize the details of any system's users (also known as actors) and their interactions with the system.



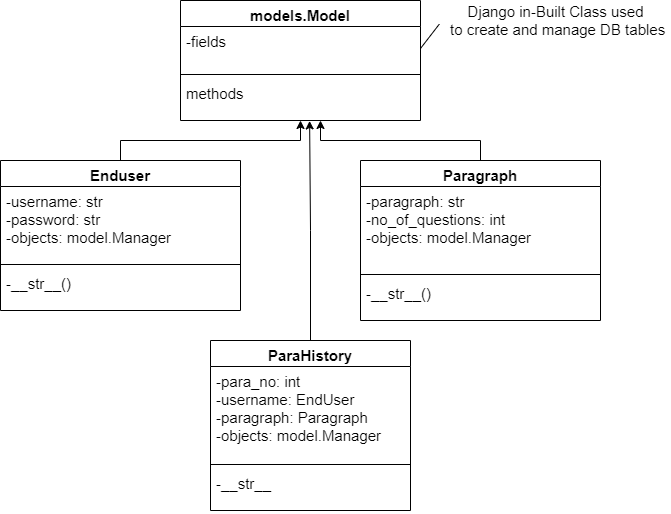
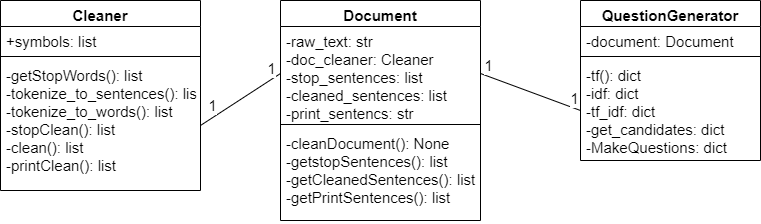
**Entity-Relationship Diagram**

Entity Relationship Diagram(ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties.



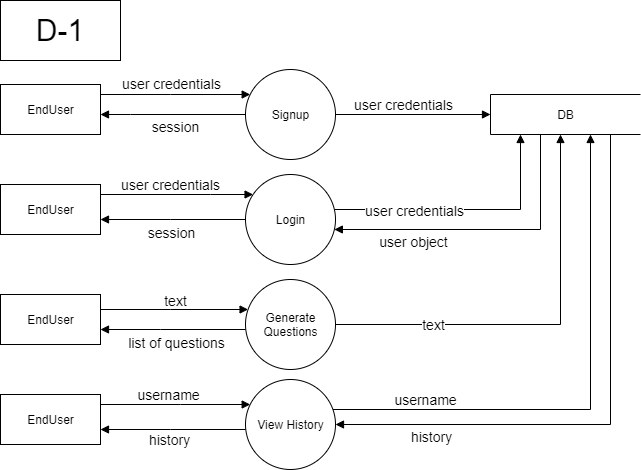
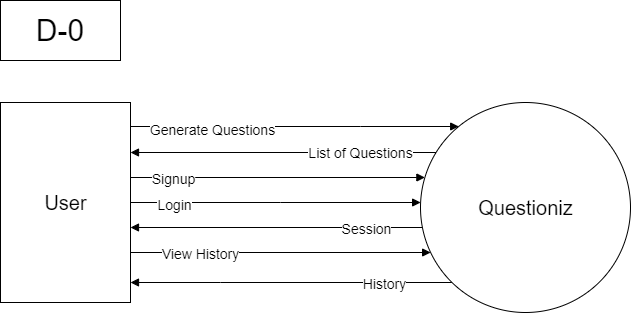
**Class Diagram**

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for understanding and documenting different aspects of a system but also for constructing executable code of the software application. Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.



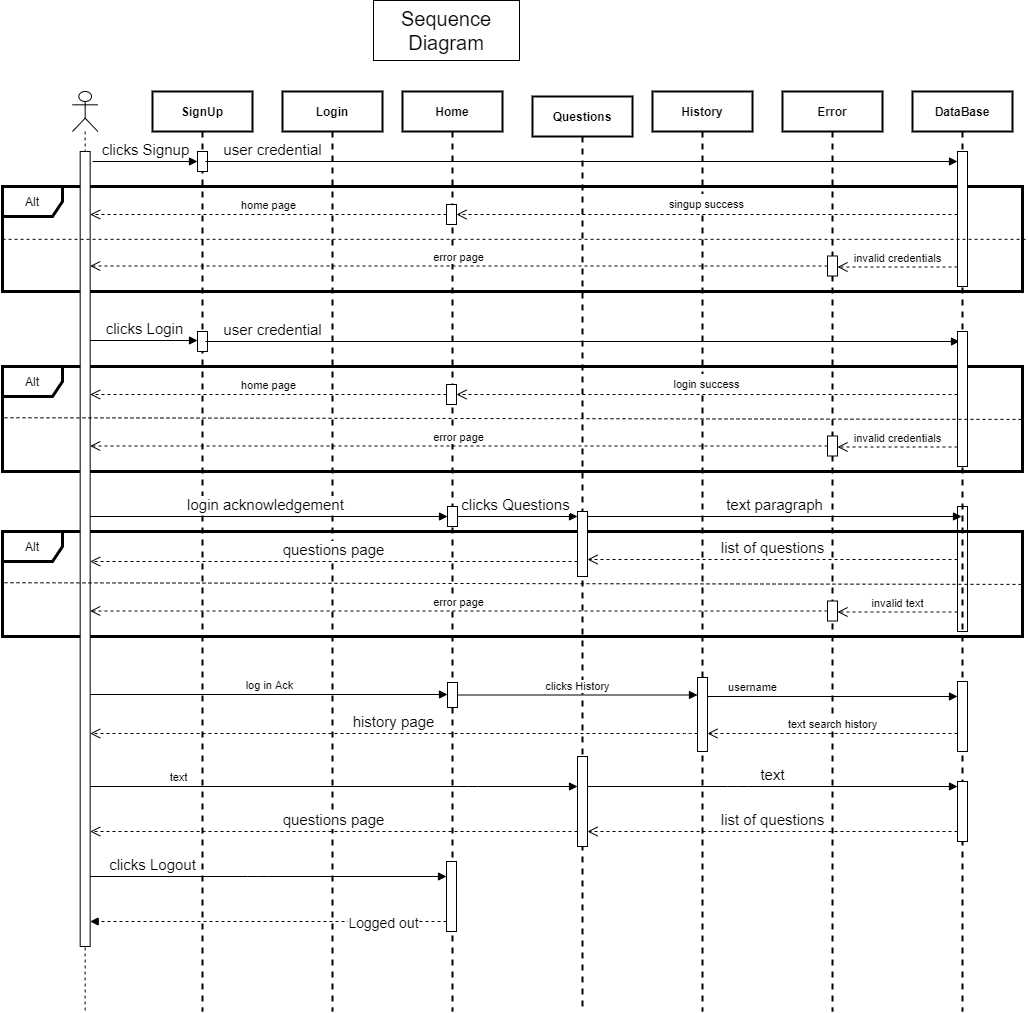
**Data Flow Diagram**

Data Flow Diagram (DFD) describes the in and out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.



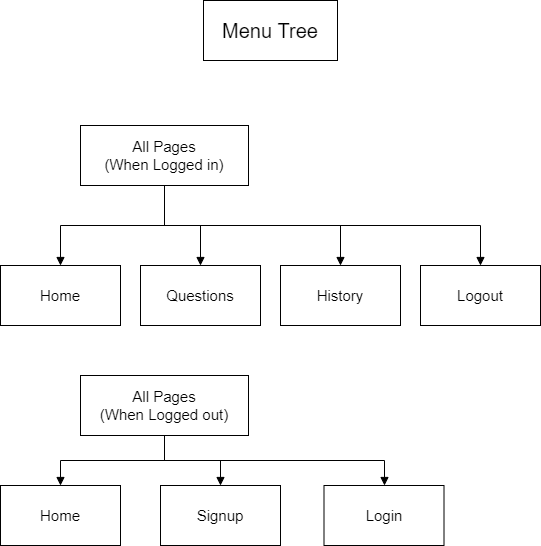
**Sequence Diagram**

Sequence Diagram is a diagram that explains the order of interactions that the any actor is going to have with the system. The diagram shows how—and in what order—a group of objects works together.



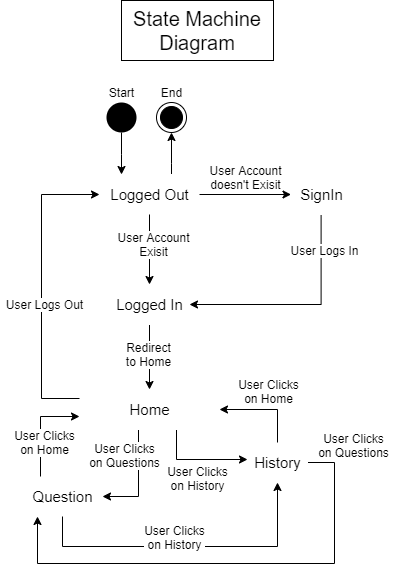
**Menu Tree**

Menu Tree shows all the pages and activities that are present in a system.



**State Machine Diagram**

A state machine is any device that stores the status of an object at a given time and can change status or cause other actions based on the input it receives. States refer to the different combinations of information that an object can hold, not how the object behaves.



**System Coding**

base.html

{% load static %}

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>{% block title%}Questioniz{% endblock %}</title>

    <link rel="stylesheet" href="{% static 'css/global.css' %}">

    {% block style %}{% endblock %}

    {% block script %}{% endblock %}

</head>

<body>

    <div class="root">

        <header>

            <section class="navigation">

                <a class="app" href="/">

                    <svg class="svg-logo svg" viewBox="0 0 16 16"><use xlink:href="#svg-logo" href="#svg-logo"></use></svg>

                    <h1>Questioniz</h1>

                </a>

                <ul class="navigation-list">

                    <li><a class="link" href="/">Home</a></li>

                    {% if username %}

                    <li><a class="link" href="/questions/">Questions</a></li>

                    <li><a class="link" href="/history/">History</a></li>

                    <form action="/logout/">

                        <button class="link logout-btn" type="submit" value="logout">Logout</button>

                    </form>

                    {% else %}

                        <li><a class="link" href="/signup/">Signup</a></li>

                        <li><a class="link" href="/login/">Login</a></li>

                    {% endif %}

                </ul>

            </section>

            {% block header %}{% endblock %}

        </header>

        <div class="content">{% block content %}{% endblock %}</div>

        <footer>{% block footer %}{% endblock %}</footer>

    </div>

    <svg style="display: none;">

        <symbol id="svg-logo" class="svg-symbol" width="16" height="16" xmlns="http://www.w3.org/2000/svg">

            <g id="Layer\_1">

                <rect stroke="#000" rx="2" fill-opacity="1" id="svg\_1" height="13.44136" width="13.87499" y="1.27932" x="1.0625" fill="#fff"/>

                <text transform="matrix(1 0 0 1 0 0)" xml:space="preserve" text-anchor="start" font-family="sans-serif" font-size="12" stroke-width="0" id="svg\_2" y="12.15313" x="4.65626" stroke="#000" fill="#000000">?</text>

            </g>

        </symbol>

        <symbol id="svg-highlight" class="svg-symbol" width="16" height="16" viewBox="0 0 16 16" fill="none" xmlns="http://www.w3.org/2000/svg">

            <path d="M2.5 1C2.77614 1 3 1.22386 3 1.5V4.5C3 4.77614 3.22386 5 3.5 5H12.502C12.7782 5 13.002 4.77614 13.002 4.5V1.5C13.002 1.22386 13.2259 1 13.502 1C13.7782 1 14.002 1.22386 14.002 1.5V4.5C14.002 5.1535 13.5841 5.70939 13.001 5.91502V7C13.001 8.10457 12.1055 9 11.001 9H10.9997L11.0029 10.7399C11.0038 11.2515 10.7439 11.7283 10.3134 12.0048L5.77287 14.9207C5.619 15.0196 5.42347 15.0265 5.26299 14.9388C5.10251 14.8512 5.00269 14.6829 5.00269 14.5V9H5.00098C3.89641 9 3.00098 8.10457 3.00098 7V5.91499C2.41787 5.70935 2 5.15347 2 4.5V1.5C2 1.22386 2.223861 2.5 1ZM6.00269 9V13.5847L9.77306 11.1634C9.91656 11.0712 10.0032 10.9123 10.0029 10.7417L9.99968 9H6.00269ZM4.00098 7C4.00098 7.55228 4.44869 8 5.00098 8H11.001C11.5533 8 12.001 7.55228 12.001 7V6H4.00098V7Z" fill="#212121"/>

        </symbol>

        <symbol id="svg-copy" class="svg-symbol" width="16" height="16" viewBox="0 016 16" fill="none" xmlns="http://www.w3.org/2000/svg">

            <path d="M4.00029 4.08525L4 10.5C4 11.8255 5.03154 12.91 6.33562 12.9947L6.5 13L10.9144 13.0007C10.7083 13.5829 10.1528 14 9.5 14H6C4.34315 14 3 12.6569 3 11V5.5C3 4.84678 3.41754 4.29109 4.00029 4.08525ZM11.5 2C12.3284 2 13 2.67157 13 3.5V10.5C13 11.3284 12.3284 12 11.5 12H6.5C5.67157 12 5 11.3284 5 10.5V3.5C5 2.67157 5.67157 2 6.5 2H11.5ZM11.5 3H6.5C6.22386 3 6 3.22386 6 3.5V10.5C6 10.7761 6.22386 11 6.5 11H11.5C11.7761 11 12 10.7761 12 10.5V3.5C12 3.22386 11.7761 3 11.5 3Z" fill="#212121"/>

        </symbol>

    </svg>

</body>

</html>

index.html

{% extends 'base.html' %}

{% load static %}

{% block title %}Home Page{% endblock %}

{% block style %}

    <link rel="stylesheet" href="{% static 'css/home.css' %}">

{% endblock %}

{% block script %}

    <script src="{% static 'js/loader.js' %}" defer></script>

    <script src="{% static 'js/home.js' %}" defer></script>

{% endblock %}

{% block content %}

    <section class="">

        Index Page

        about and how it works

        product description

    </section>

{% endblock %}

{% block footer %}

    <div class="no-display overlay" id="loading">

        <div class="loader"></div>

    </div>

    {% if c %}

        {{c}}

    {% endif %}

{% endblock %}

signup.html

{% extends 'base.html' %}

{% load static %}

{% block title %}Home Page{% endblock %}

{% block style %}

    <link rel="stylesheet" href="{% static 'css/home.css' %}">

{% endblock %}

{% block content %}

    <section class="form-container">

        <h1 class="pg-title">Sign up</h1>

        <form class="singup-form form" method="POST" action="/home/">

            {% csrf\_token %}

                <input class="form-input text" type="text" name="username" id="username" placeholder="Username">

                <input class="form-input password" type="password" name="password" id="password" placeholder="Confirm">

                <input class="form-input password" type="password" name="cpassword" id="cpassword" placeholder="Confirm Confirm">

            <button class="signup-btn btn round-border" type="submit" name="signup-btn" value="sign-up">Signup</button>

            <button class="reset-btn btn round-border" type="reset" name="reset-btn" value="reset">Reset</button>

        </form>

    </section>

{% endblock %}

login.html

{% extends 'base.html' %}

{% load static %}

{% block title %}Home Page{% endblock %}

{% block style %}

    <link rel="stylesheet" href="{% static 'css/home.css' %}">

{% endblock %}

{% block content %}

    <section class="form-container">

        <h1 class="pg-title">Log in</h1>

        <form class="login-form form" method="POST" action="/home/">

            {% csrf\_token %}

                <input class="form-input text" type="text" name="username" id="username" placeholder="Username">

                <input class="form-input password" type="password" name="password" id="password" placeholder="Confirm">

            <button class="login-btn btn round-border" type="submit" name="login-btn" value="let-me-in">Login</button>

            <button class="reset-btn btn round-border" type="reset" name="reset-btn" value="reset">Reset</button>

        </form>

    </section>

{% endblock %}

no\_questions.html

{% extends 'base.html' %}

{% load static %}

{% block style %}

    <link rel="stylesheet" href="{% static 'css/questions.css' %}">

    <link rel="stylesheet" href="{% static 'css/loader.css' %}">

{% endblock %}

{% block script %}

    <script>

        var textarea\_text= "{{text}}";

    </script>

    <script src="{% static 'js/home.js' %}" defer></script>

    <script src="{% static 'js/questions.js' %}"></script>

{% endblock %}

{% block content %}

    <section class="text-area--section" id="span-text">

        <form class="text-area--form" name="text-area-form" id="text-area-form" method="POST" action="/questions/">

            {% csrf\_token %}

            <label class="text-area--label" for="text">Text Here <span>🠗</span></label>

            <div class="text-area--container" words="0">

                {% if text %}

                    <textarea class="text-area--text" name="text-area" id="text-area" placeholder="Text here..." words="0">{{text}}</textarea>

                {% else %}

                    <textarea class="text-area--text" name="text-area" id="text-area" placeholder="Text here..." words="0"></textarea>

                {% endif %}

                <button type="reset" class="reset-btn btn " id="text-area-btn-reset" name="text-area-btn-reset" value="reset">X</button>

            </div>

                <button type="submit" class="submit-btn btn round-border-thick" id="text-area-btn-submit" name="text-area-btn-submit" value="generate-questions">Generate</button>

        </form>

    </section>

{% endblock %}

{% block footer %}

    <div class="no-display overlay" id="loading">

        <div class="loader"></div>

    </div>

{% endblock %}

questions.html

{% extends 'base.html' %}

{% load static %}

{% block style %}

    <link rel="stylesheet" href="{% static 'css/questions.css' %}">

{% endblock %}

{% block script %}

    <script>

        var textarea\_text= "{{text}}";

    </script>

    <script src="{% static 'js/questions.js' %}"></script>

{% endblock %}

{% block content %}

    <section class="text-area--section">

        <form class="text-area--form" name="text-area-form" id="text-area-form" method="POST" action="/questions/">

            {% csrf\_token %}

            <textarea class="text-area--text no-display" name="text-area" id="text-area" name="text-area" id="text-area">{{text}}</textarea>

            <button class="go-back btn round-border-thin" type="submit" name="goback-btn" value="go-back">🠔 Go Back</button>

        </form>

    </section>

    <section class="display-text round-border" id="span-sentences">

        {% for s in sentences %}

            <span indx="{{forloop.counter0}}">

                {{s}}

            </span>

        {% endfor %}

    </section>

    <section class="questions--section">

        <button class="clear-selection-btn btn round-border-thin no-display" id="clear-selection-btn" name="clear-selection-btn" onclick="clearSelection()">X</button>

        <ul class="questions--ul custom-marker" name="questions--ul" id="questions--ul">

            {% for i, s in questions\_with\_ans %}

                <li class="question--item">

                    <div class="question-contanier">

                        <span class="question" onclick="toggleAnswer('{{forloop.counter0}}')">{{s}}?</span>

                        <p class="answer no-display">Answer: {{i.1}}</p>

                    </div>

                    <div class="action-container">

                        <button class="highlight btn round-border-thin" hover="Highlight" onclick="highlightPrase('{{i.0}}')">

                            <svg class="svg-highlight svg" viewBox="0 0 16 16"><use xlink:href="#svg-highlight" href="#svg-highlight"></use></svg>

                        </button>

                        <button class="copy btn round-border-thin" hover="Copy" onclick="copyPhrase('{{forloop.counter0}}')">

                            <svg class="svg-copy svg" viewBox="0 0 16 16"><use xlink:href="#svg-copy" href="#svg-copy"></use></svg>

                        </button>

                    </div>

                </li>

            {% endfor %}

        </ul>

    </section>

{% endblock %}

history.html

{% extends 'base.html' %}

{% load static %}

{% block title %}Home Page{% endblock %}

{% block style %}

    <link rel="stylesheet" href="{% static 'css/history.css' %}">

{% endblock %}

{% block script %}

    <script src="{% static 'js/loader.js' %}" defer></script>

{% endblock %}

{% block content %}

    <section>

        <ul class="history--ul custom-marker">

            {% for i in paragraphs %}

            <li class="question--item">

                <form class="text-area--form" name="text-area-form" id="text-area-form" method="POST" action="/questions/">

                    {% csrf\_token %}

                    <textarea class="text-area--text no-display" name="text-area" id="text-area" placeholder="Text here..." words="0" type="hidden">{{i}}</textarea>

                    <p class="question--text">{{i}}</p>

                    <button type="submit" class="submit-btn btn round-border-thin" id="text-area-btn-submit" name="text-area-btn-submit" value="generate-questions">See Questions</button>

                </form>

            </li>

            {% endfor %}

        </ul>

    </section>

{% endblock %}

error.html

{% extends 'base.html' %}

{% load static %}

{% block title %}Home Page{% endblock %}

{% block style %}

    <link rel="stylesheet" href="{% static 'css/error.css' %}">

{% endblock %}

{% block script %}{% endblock %}

{% block header %}

    <p class="error big">Error</p>

{% endblock %}

{% block content %}

    <div>"{{error\_msg}}"</div>

{% endblock %}

{% block footer %}{% endblock %}

home.js

var edit\_area= document.querySelector("#text-area");

edit\_area.addEventListener("input", function (event) {

    let words= edit\_area.value.replace(/\n/g, " ").split(" ").filter((entry)=> {

        return entry.trim()!= '';

    });

    let no\_of\_words= words.length;

    edit\_area.parentElement.setAttribute("words", no\_of\_words);

});

loader.js

var q\_btn= document.querySelector("#send\_text-btn");

q\_btn.addEventListener("click", function() {

    console.log("show\_loading");

    document.querySelector("#loading").classList.remove("no-display");

});

questions.js

function highlightPrase(index) {

    // console.log(index);

    let sentences= document.querySelector("#span-sentences").children;

    let clear\_btn= document.querySelector("#clear-selection-btn");

    // console.log(clear\_btn);

    for(let i=0; i<sentences.length; i++) {

        sentences[i].classList.remove("selected");

        if(index == i) {

            sentences[i].classList.add("selected");

        }

    }

    document.querySelector("#span-sentences").scrollIntoView({ block: "center", behavior: "smooth"});

    clear\_btn.classList.remove("no-display");

}

function copyPhrase(index) {

    /\* Get the text field \*/

    let question\_li= document.querySelector("#questions--ul").children[index];

    // console.log(question\_li);

    let copyText= question\_li.querySelector(".question").innerHTML;

    console.log(copyText);

    navigator.clipboard.writeText(copyText);

}

function clearSelection() {

    let sentences= document.querySelector("#span-sentences").children;

    let clear\_btn= document.querySelector("#clear-selection-btn");

    // console.log(sentences);

    for(let i=0; i<sentences.length; i++) {

        sentences[i].classList.remove("selected");

    }

    clear\_btn.classList.add("no-display");

}

function toggleAnswer(indx) {

    console.log(indx);

    let clicked\_li= document.querySelector("#questions--ul").children[indx];

    let ans\_p= clicked\_li.querySelector(".answer");

    console.log(clicked\_li);

    if(ans\_p.classList.contains("no-display")) {

        ans\_p.classList.remove("no-display");

    } else {

        ans\_p.classList.add("no-display");

    }

}

Questioniz\_Web/urls.py

"""Questioniz\_web URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:

    https://docs.djangoproject.com/en/3.2/topics/http/urls/

Examples:

Function views

    1. Add an import:  from my\_app import views

    2. Add a URL to urlpatterns:  path('', views.home, name='home')

Class-based views

    1. Add an import:  from other\_app.views import Home

    2. Add a URL to urlpatterns:  path('', Home.as\_view(), name='home')

Including another URLconf

    1. Import the include() function: from django.urls import include, path

    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))

"""

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

    path('admin/', admin.site.urls),

    path('', include("qGen.urls")),

]

qGen/urls.py

from django.urls import path

from . import views

urlpatterns= [

    path("", views.index, name="index"),

    path("login/", views.login, name="login"),

    path("signup/", views.signup, name="signup"),

    path("logout/", views.logout, name="logout"),

    path("home/", views.home, name="home"),

    path("questions/", views.questions, name="questions"),

    path("history/", views.history, name="history"),

    path("error/<str:err>", views.error, name="error"),

]

qGen/views.py

from django.shortcuts import render, redirect

from django.http import HttpResponse

import nltk # test: test123

from .models import EndUser, Paragraph, ParaHistory

from .qg import Document, QuestionGenerator

from django.contrib.auth.models import User

# Create your views here.

def cleanText(string, v\_type):

    if(v\_type== "username"):

        v= True

        if(string.isalnum()):

            v= True

    elif(v\_type== "password"):

        v= True

    if(not v): return None

    return string

def validate(username, password): # validation of user

    cd= EndUser.objects.filter(username= username)

    if(cd.count()< 1 or cd.count()> 1): return None

    c= EndUser.objects.get(username= username)

    if(c.password== password): return c

    return None

def index(request):

    options= {

        "username": None,

    }

    try:

        options["username"]= request.session["username"]

    except Exception:

        pass

    return render(request, "index.html", options)

def home(request):

    # print(request)

    options= {

        "username": None,

    }

    try:

        options["username"]= request.session["username"]

    except Exception:

        pass

    if request.method== "POST":

        if(request.POST.get("signup-btn", False)== "sign-up"): #code to register signup

            username= cleanText(request.POST.get("username"), "username")

            password= cleanText(request.POST.get("password"), "password")

            cpassword= cleanText(request.POST.get("cpassword"), "password")

            if(not username or not password): return redirect("/error/invalid\_username\_or\_password")

            if(password!= cpassword): return redirect("/error/passwords\_dont\_match")

            if(EndUser.objects.filter(username= username)): return redirect("/error/user\_already\_exists")

            user= EndUser(username=username, password=password)

            user.save()

            request.session["username"]= user.username

        elif(request.POST.get("login-btn", False)== "let-me-in"): #code to authenticate login

            username= cleanText(request.POST.get("username"), "username")

            password= cleanText(request.POST.get("password"), "password")

            user= validate(username, password)

            if not user: return redirect("/error/invalid\_username\_or\_password")

            request.session["username"]= user.username

        try:

            options["username"]= request.session["username"]

        except Exception:

            pass

    return render(request, "index.html", options)

def signup(request):

    options= {

        "username": None,

    }

    try:

        options["username"]= request.session["username"]

    except Exception:

        pass

    if request.method == "POST":

        pass

    return render(request, "signup.html", options)

def login(request):

    options= {

        "username": None,

    }

    try:

        options["username"]= request.session["username"]

    except Exception:

        pass

    return render(request, "login.html", options)

def logout(request):

    options= {

        "username": None,

    }

    try:

        options["username"]= request.session["username"]

    except Exception:

        pass

    request.session.flush()

    return redirect("/")

def error(request, err):

    options= {

        "username": None,

        "error\_msg": err,

    }

    try:

        options["username"]= request.session["username"]

    except Exception:

        pass

    print(err)

    return render(request, "error.html", options)

    # b = Blog(name='Beatles Blog', tagline='All the latest Beatles news.')

    # b.save()

def questions(request):

    options= {

        "username": None,

        "text": None,

    }

    try:

        options["username"]= request.session["username"]

    except Exception:

        return redirect("/error/not\_logged\_in")

    text:str= None

    sentences= None

    questions\_with\_ans= None

    # print(request.POST)

    if(request.POST.get("text-area-btn-submit")== "generate-questions"):

        text= request.POST.get("text-area", None)

        # change to redirect

        if(not text): return redirect("/error/Empty Textarea")

        # All Validations Complete now processing Data

        doc= Document(text)

        sentences= doc.getPrintSentences()

        qg\_maker= QuestionGenerator(doc)

        questions:dict= qg\_maker.MakeQuestions()

        questions\_with\_ans= questions.items()

        options["text"]= text

        options["sentences"]= sentences

        options["questions\_with\_ans"]= questions\_with\_ans

        options["no\_of\_questions"]= len(questions\_with\_ans)

        if(options["username"]): # save paragraph if logged in

            user= EndUser.objects.get(username= options["username"])

            para= Paragraph(paragraph= text, no\_of\_questions= options["no\_of\_questions"])

            para.save()

            ph\_query= ParaHistory.objects.filter(username= user)

            if(ph\_query.count()> 0):

                ph= ParaHistory(para\_no= ph\_query.count()% 10, username= user, paragraph= para)

                ph.save()

            else:

                ph= ParaHistory(para\_no= 0, username= user, paragraph= para)

                ph.save()

        return render(request, "questions.html", options)

    elif(request.POST.get("goback-btn")== "go-back"):

        text= request.POST.get("text-area", None)

        options["text"]= text

        return render(request, "no\_questions.html", options)

    return render(request, "no\_questions.html", options)

def history(request):

    options= {

        "username": None,

        "text": None,

        "paragraph": None,

    }

    try:

        options["username"]= request.session["username"]

    except Exception:

        return redirect("/error/not\_logged\_in")

    if(options["username"]): # get paragraph if logged in

        user= EndUser.objects.get(username= options["username"])

        ph\_query= ParaHistory.objects.filter(username= user).order\_by("-para\_no")

        ph\_set= []

        for p in ph\_query:

            if(p.paragraph.paragraph in ph\_set): continue

            ph\_set.append(p.paragraph.paragraph)

        options["paragraphs"]= ph\_set[:10]

    return render(request, "history.html", options)

qGen/models.py

from django.db import models

# Create your models here.

class EndUser(models.Model):

    username = models.CharField(max\_length=50,default='')

    password = models.CharField(max\_length=50,default='')

    # is\_active = models.BooleanField(null=True)

    def \_\_str\_\_(self):

        return self.username

class Paragraph(models.Model):

    paragraph= models.CharField(max\_length=2500)

    no\_of\_questions= models.IntegerField()

    def \_\_str\_\_(self):

        return self.paragraph

class ParaHistory(models.Model):

    para\_no= models.IntegerField()

    username= models.ForeignKey("Enduser", on\_delete= models.CASCADE)

    paragraph= models.ForeignKey("Paragraph", on\_delete= models.CASCADE)

    def \_\_str\_\_(self):

        return f"{self.para\_no} - {self.paragraph}"

qGen/admin.py

from django.contrib import admin

from . import models

# Register your models here.

admin.site.register(models.EndUser)

admin.site.register(models.Paragraph)

qGen/qg.py

import nltk, math, string, random

from nltk.corpus import stopwords as StopWords

class Cleaner:

    symbols= ['\n', '~', ':', "'", "''", '+', '[', '\\', '@', '^', '{', '%', '(', '-', '"', '\*', '|', '&', '<', '`', '``', '}', '\_', '=', ']', '!', '>', ';', '?', '#', '$', ')', '/']

    def \_\_init\_\_(self):

        self.better\_stop\_words= self.getStopWords()

    def getStopWords(self, lang= None, symbols=True)->list:

        if(lang== None): lang= "english"

        sw= StopWords.words(lang)

        sw+= [w.capitalize() for w in sw]

        if(symbols):

            sw+= ['\n', '~', ':', "'", "''", '+', '[', '\\', '@', '^', '{', '%', '(', '-', '"', '\*', '|', ',', '&', '<', '`', '``', '}', '.', '\_', '=', ']', '!', '>', ';', '?', '#', '$', ')', '/']

        return sw

    def tokenize\_to\_sentences(self, data)->list:

        """tokenizes given data in sentences"""

        return nltk.sent\_tokenize(data)

    def tokenize\_to\_words(self, data:str)->list:

        """tokenizes given data in words"""

        return nltk.word\_tokenize(data)

    def stopClean(self, data):

        """sentences:list= list of sentences, sentence:list= list of words i.e. it is an element of sentences"""

        data= "".join([x.lower() for x in data if x not in Cleaner.symbols])

        data= "".join([x for x in data if not x.isdigit()])

        sentences= self.tokenize\_to\_sentences(data)

        for i, s in enumerate(sentences):

            word\_s= self.tokenize\_to\_words(s)

            word\_s= [w.lower() for w in word\_s if w not in self.better\_stop\_words]

            sentences[i]= word\_s

        for i, s in enumerate(sentences):

            tag\_word\_s= nltk.pos\_tag(s)

            sentences[i]= [tw[0] for tw in tag\_word\_s if tw[1] in ["NN", "NNS", "NNP", "NNPS"]]

        return sentences

    def clean(self, data):

        data= "".join([x for x in data if x not in Cleaner.symbols])

        sentences= self.tokenize\_to\_sentences(data)

        for i, s in enumerate(sentences):

            word\_s= self.tokenize\_to\_words(s)

            word\_s= [w for w in word\_s if w not in [".", ","]]

            sentences[i]= word\_s

        return sentences

    def printClean(self, data):

        cleaned\_data= self.clean(data)

        for i, s in enumerate(cleaned\_data):

            cleaned\_data[i]= " ".join(s)

            cleaned\_data[i]+= "."

        return cleaned\_data

class Document:

    def \_\_init\_\_(self, text):

        self.raw\_text= text

        self.doc\_cleaner= Cleaner()

        self.cleanDocument()

    def cleanDocument(self):

        self.stop\_sentences= self.doc\_cleaner.stopClean(self.raw\_text)

        self.cleaned\_sentences = self.doc\_cleaner.clean(self.raw\_text)

        self.print\_sentences = self.doc\_cleaner.printClean(self.raw\_text)

    def getstopSentences(self):

        return self.stop\_sentences

    def getCleanedSentences(self):

        return self.cleaned\_sentences

    def getPrintSentences(self):

        return self.print\_sentences

class QuestionGenerator:

    def \_\_init\_\_(self, doc):

        self.document= doc

# TF-IDF

    def tf(self, words, len\_words)->dict:

        tf\_score= {}

        for each\_word in words:

            tf\_score[each\_word]= tf\_score.get(each\_word, 0)+ 1

        # Dividing each element of dict by total\_no\_of\_words

        tf\_score.update((x, y/len\_words) for x, y in tf\_score.items())

        return tf\_score

    def idf(self, words, sentences, len\_sentences)->dict:

        idf\_score = {}

        def check\_in\_sentence(word, sentences): # check occurence of a word in all sentences

            final= [word in s for s in sentences]

            return final.count(True)

        for each\_word in words:

            if each\_word in idf\_score:

                idf\_score[each\_word] += check\_in\_sentence(each\_word, sentences)

            else:

                idf\_score[each\_word] = 1

        # Performing a log and divide

        idf\_score.update((x, math.log(len\_sentences)/y) for x, y in idf\_score.items())

        return idf\_score

    def tf\_idf(self, stop\_sentences)->dict:

        # Gather list of all sentences and all words

        no\_of\_sentences= len(stop\_sentences)

        words= [] # list of all words

        for s in stop\_sentences:

            words.extend(s)

        no\_of\_words= len(words)

        # Calculate TF

        tf\_score= self.tf(words, no\_of\_words) # Term Frequency

        # Calculate IDF

        idf\_score= self.idf(words, stop\_sentences, no\_of\_sentences) # Inverse Document Frequency

        # Calculate TF-IDF

        tf\_idf\_score = {key: tf\_score[key] \* idf\_score.get(key, 0) for key in tf\_score.keys()}

        # Sort tf\_idf\_scores

        tf\_idf\_score= dict(sorted(tf\_idf\_score.items(), key=lambda elem: elem[1], reverse = True))

        return tf\_idf\_score

    def get\_candidates(self, sentences, extracted\_words)->dict:

        q\_can= {}

        for s\_id, s in enumerate(sentences):

            ans\_words= [w for w in s if w.lower() in extracted\_words] # list of words from the sentence that can be answer

            for aw in ans\_words:

                for indx in [i for i,v in enumerate(s) if v==aw]: # find and replace ans\_word with '\_\_\_\_\_'

                    sent\_copy= [w.lower() for w in s]

                    sent\_copy[indx]= "\_\_\_\_\_"

                    q\_can[(s\_id, aw)]= " ". join(sent\_copy)

        return q\_can

    def MakeQuestions(self)->dict:

        stop\_sentences= self.document.getstopSentences()

        cleaned\_sentences= self.document.getCleanedSentences()

        tf\_idf\_score= self.tf\_idf(stop\_sentences)

        question\_candidates= self.get\_candidates(cleaned\_sentences, tf\_idf\_score)

        questions\_marked= {}

        random\_sent\_order= [n for n in range(len(cleaned\_sentences))]

        random.shuffle(random\_sent\_order)

        for i in random\_sent\_order:

            selected= False

            ith\_questions= []

            for k in question\_candidates:

                if(selected): break

                if(k[0]== i):

                    ith\_questions.append(k)

            rand\_indx= random.choice(ith\_questions)

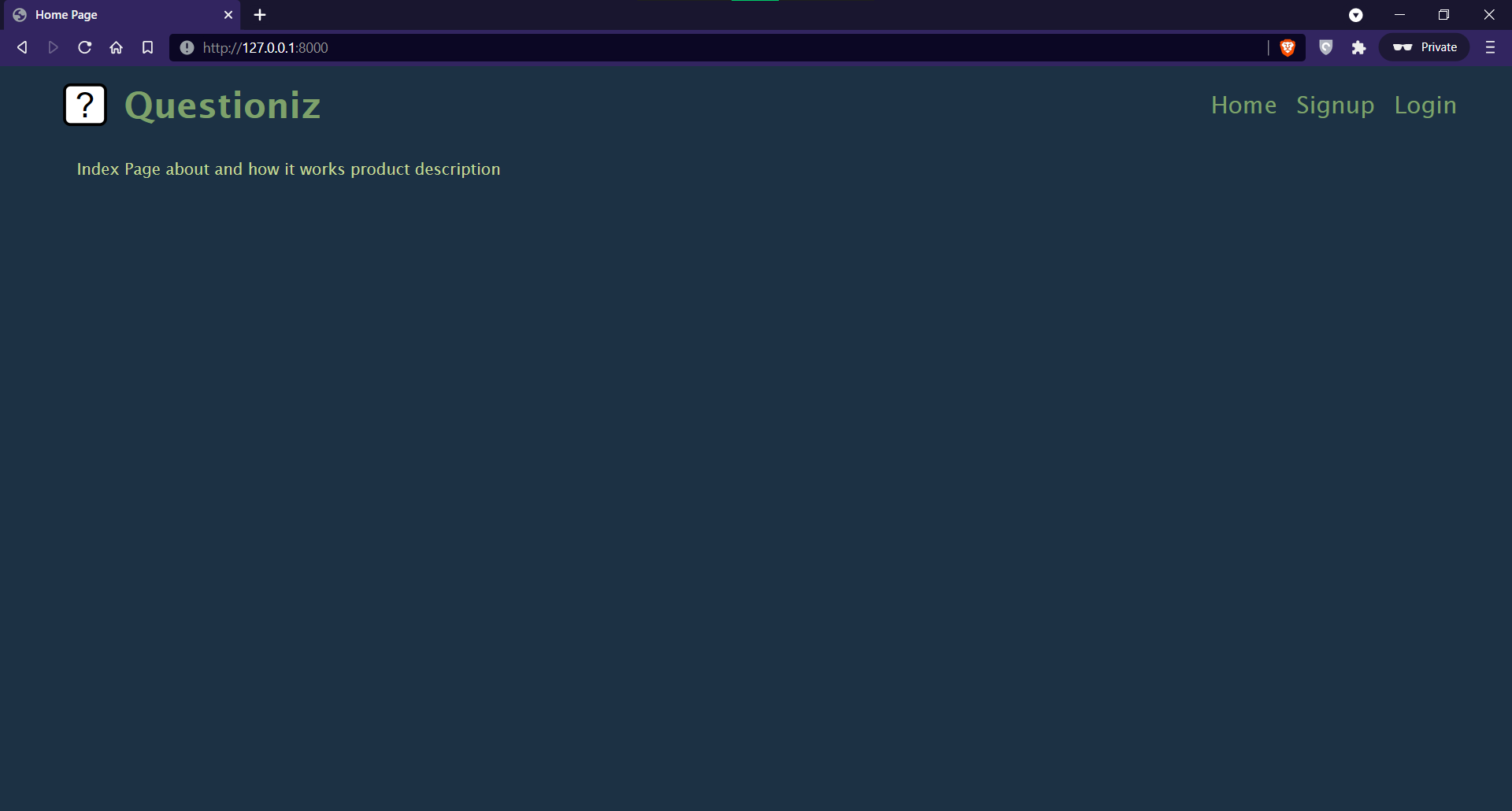
            questions\_marked[rand\_indx]= question\_candidates[rand\_indx]

        # question\_candidates= {tuple(sentence\_id, answer): dash\_question}

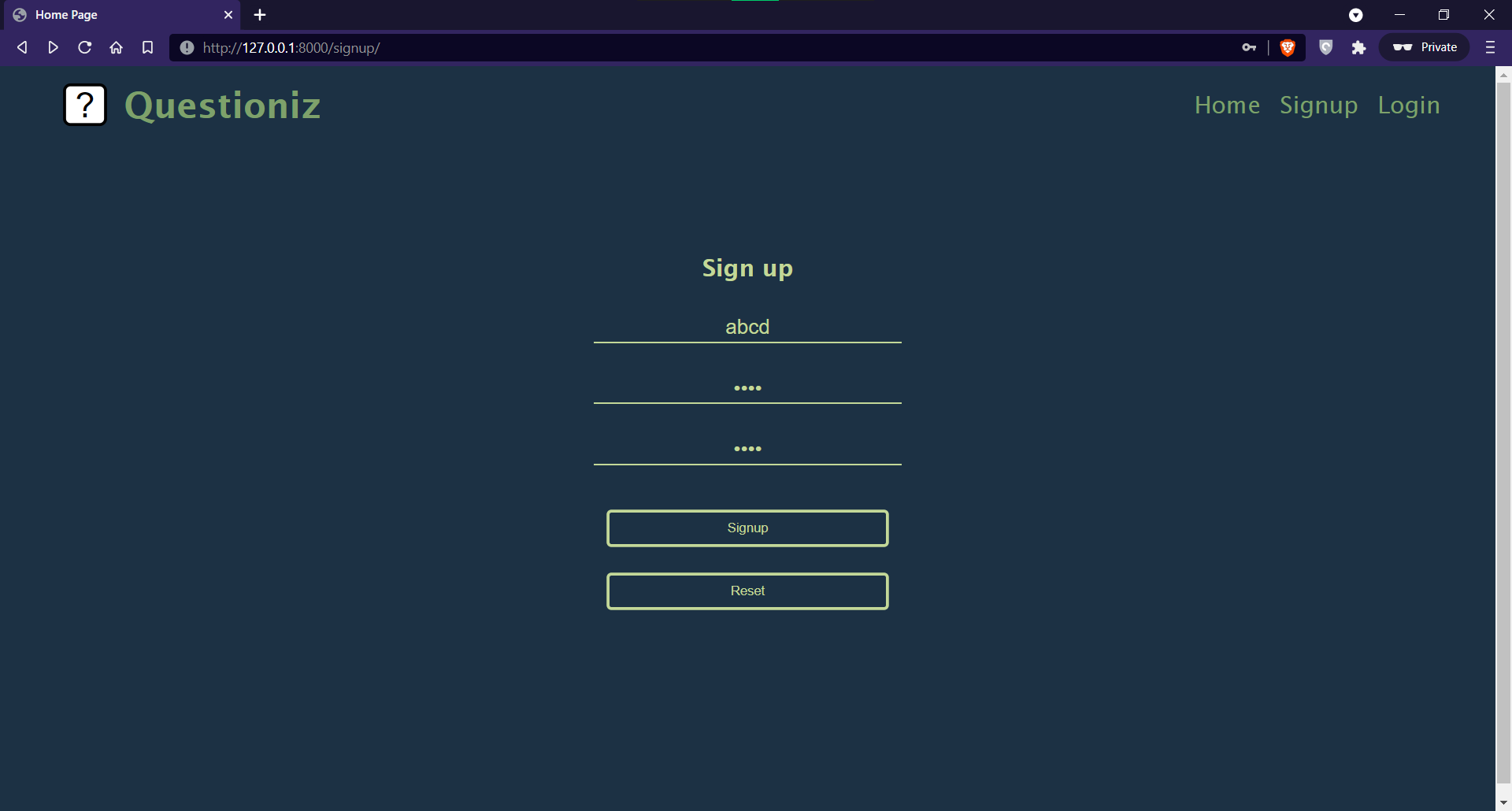
        return questions\_marked

**Screen Layout**

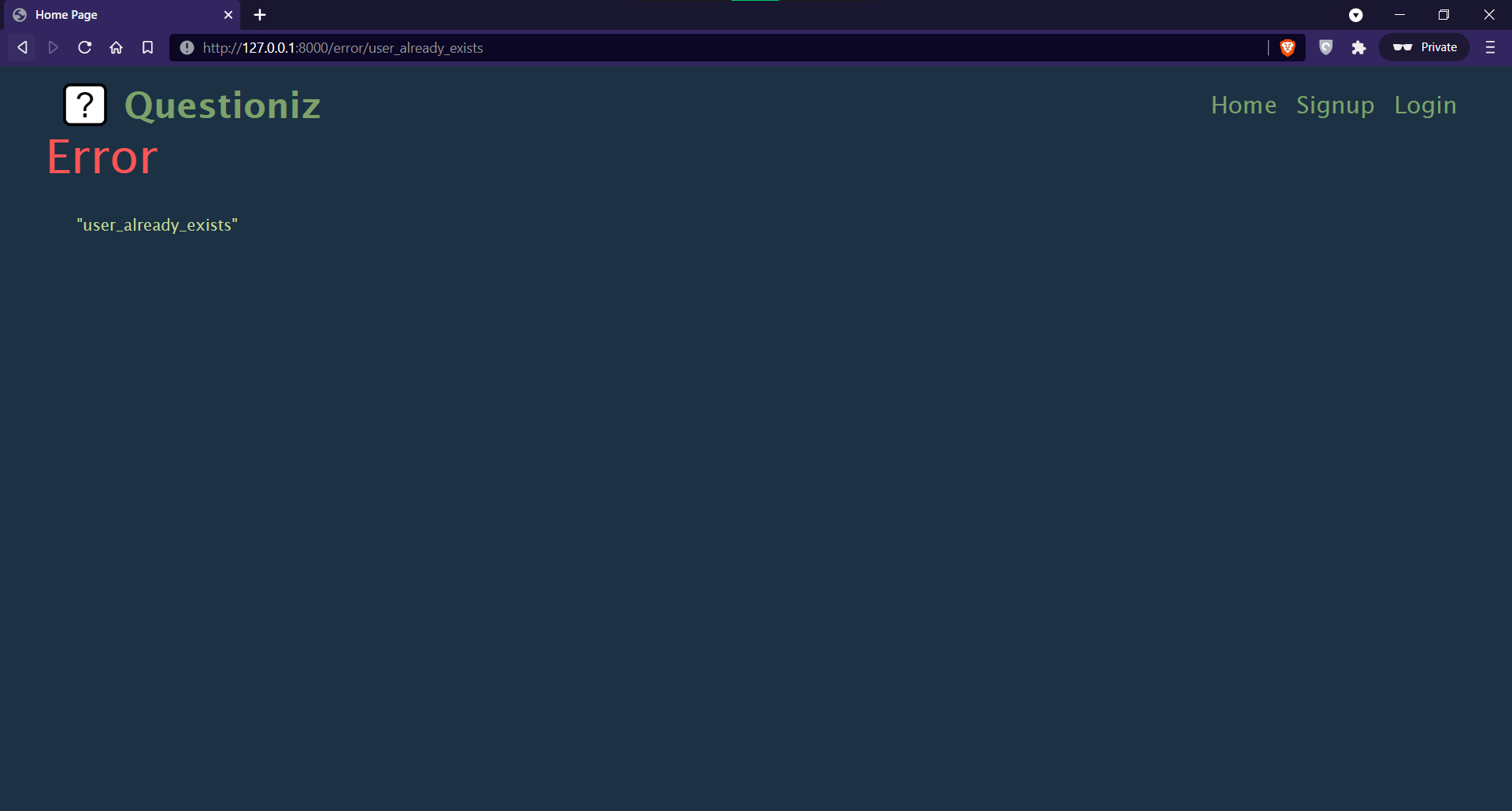
index page without login



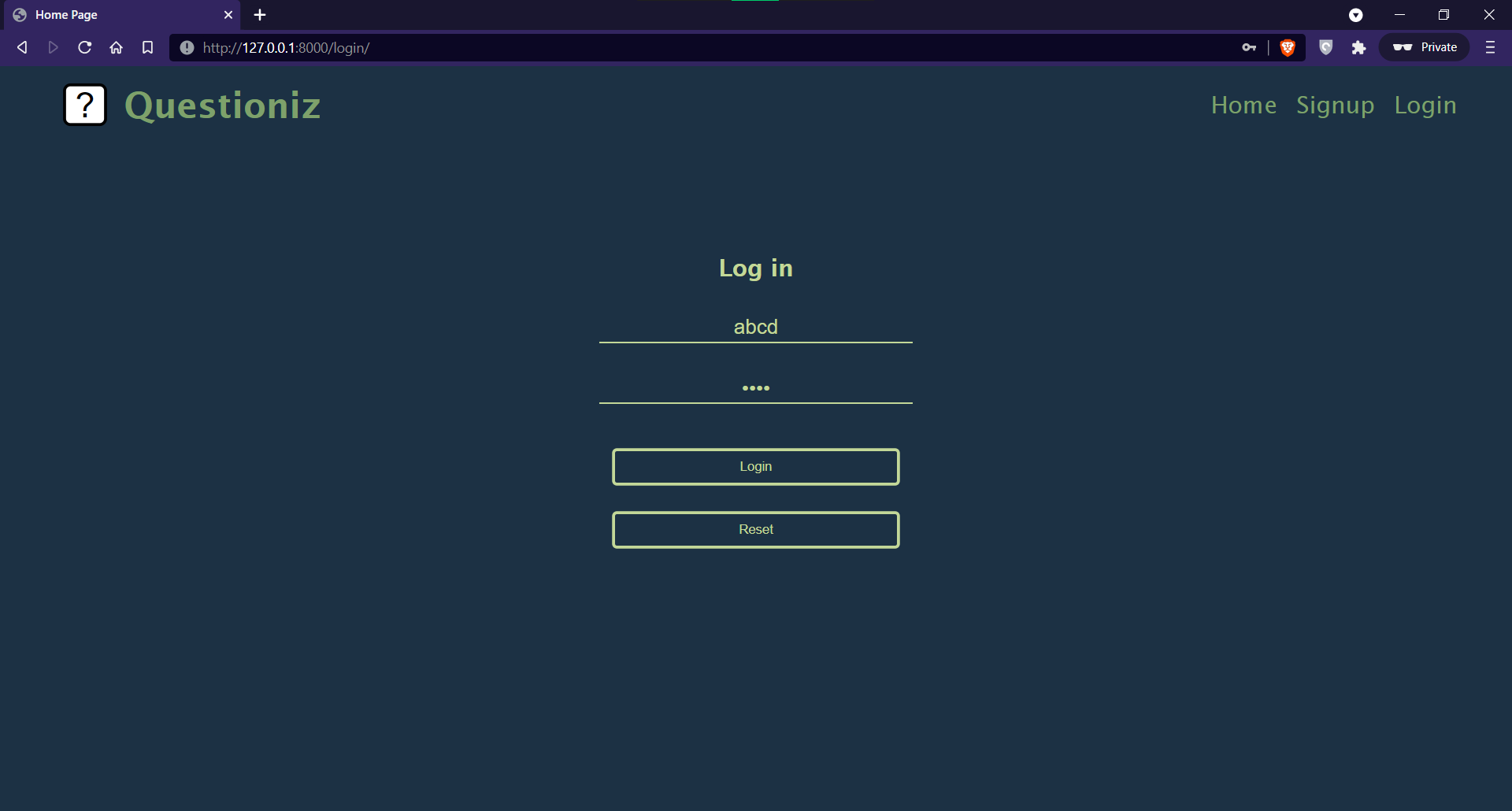
signup page



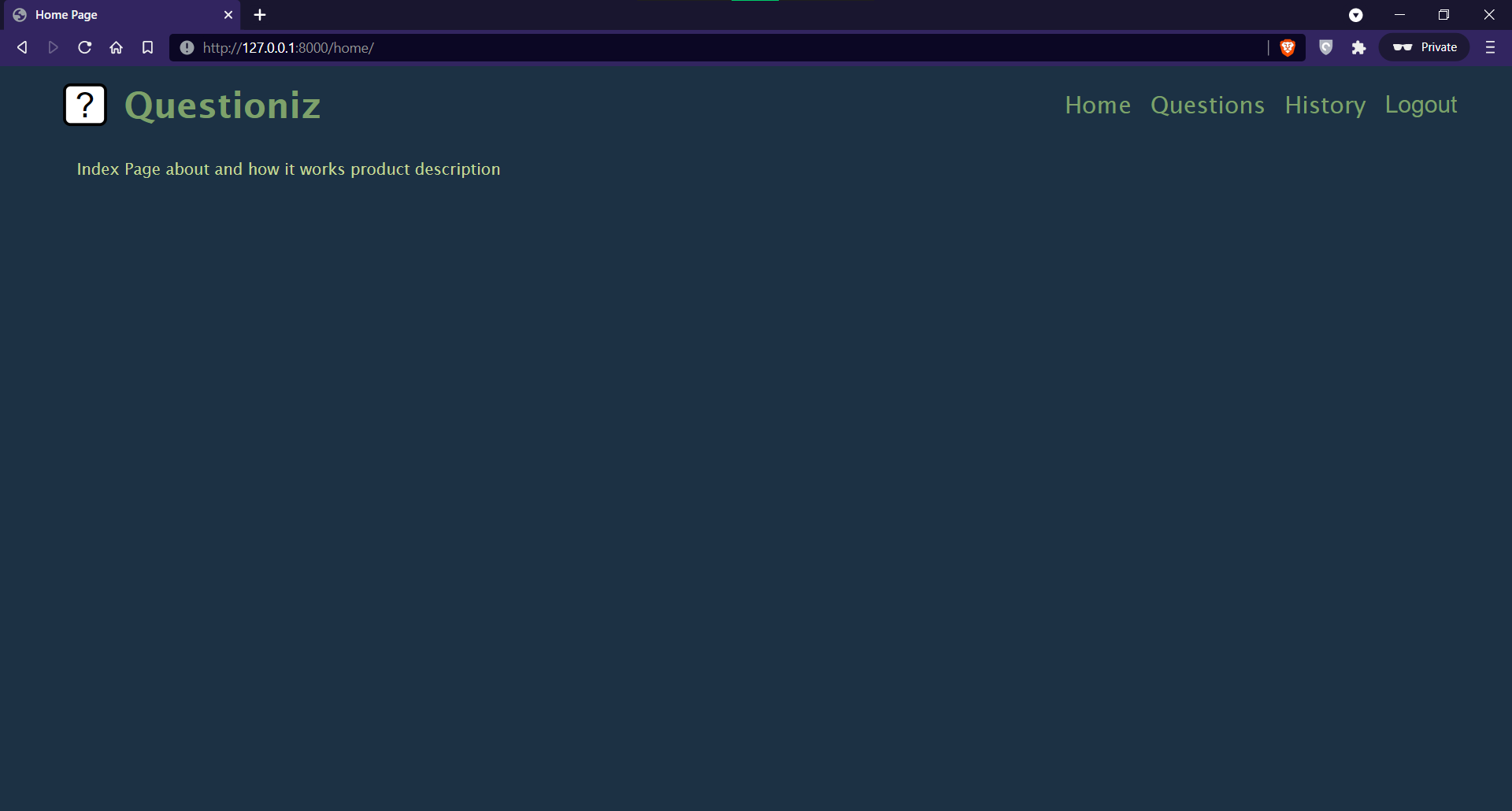
signup error page



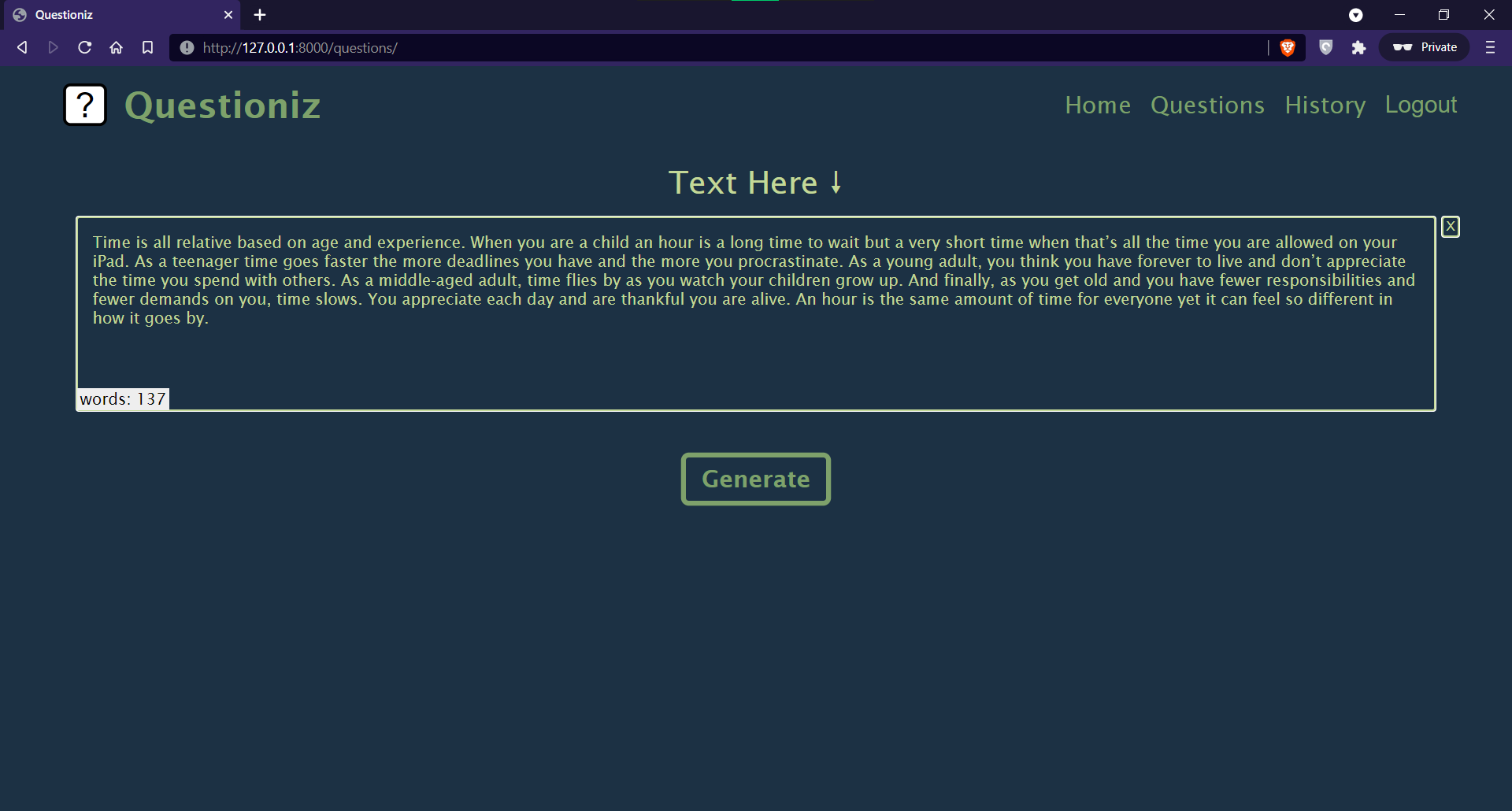
log in page



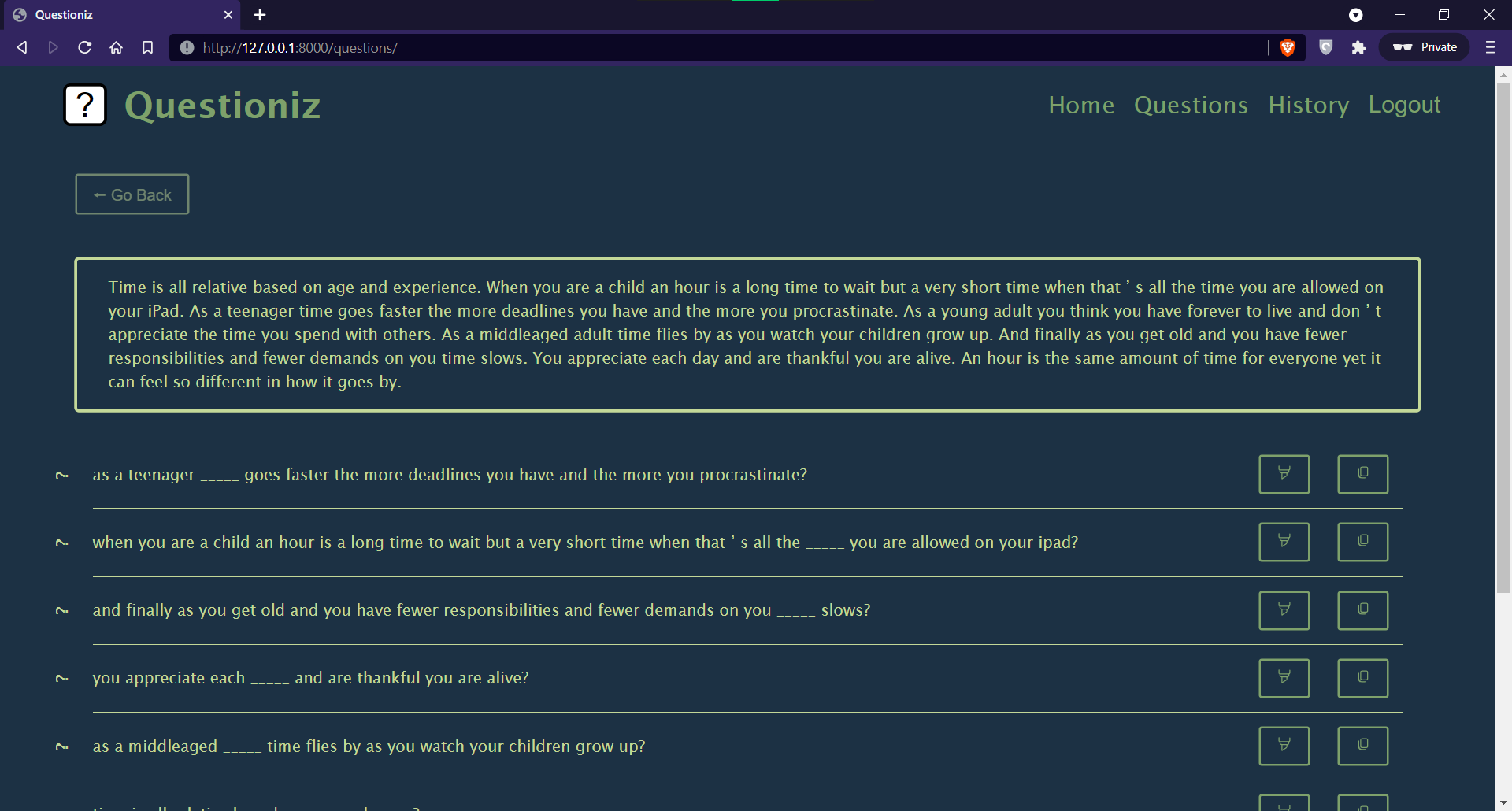
index page when logged in



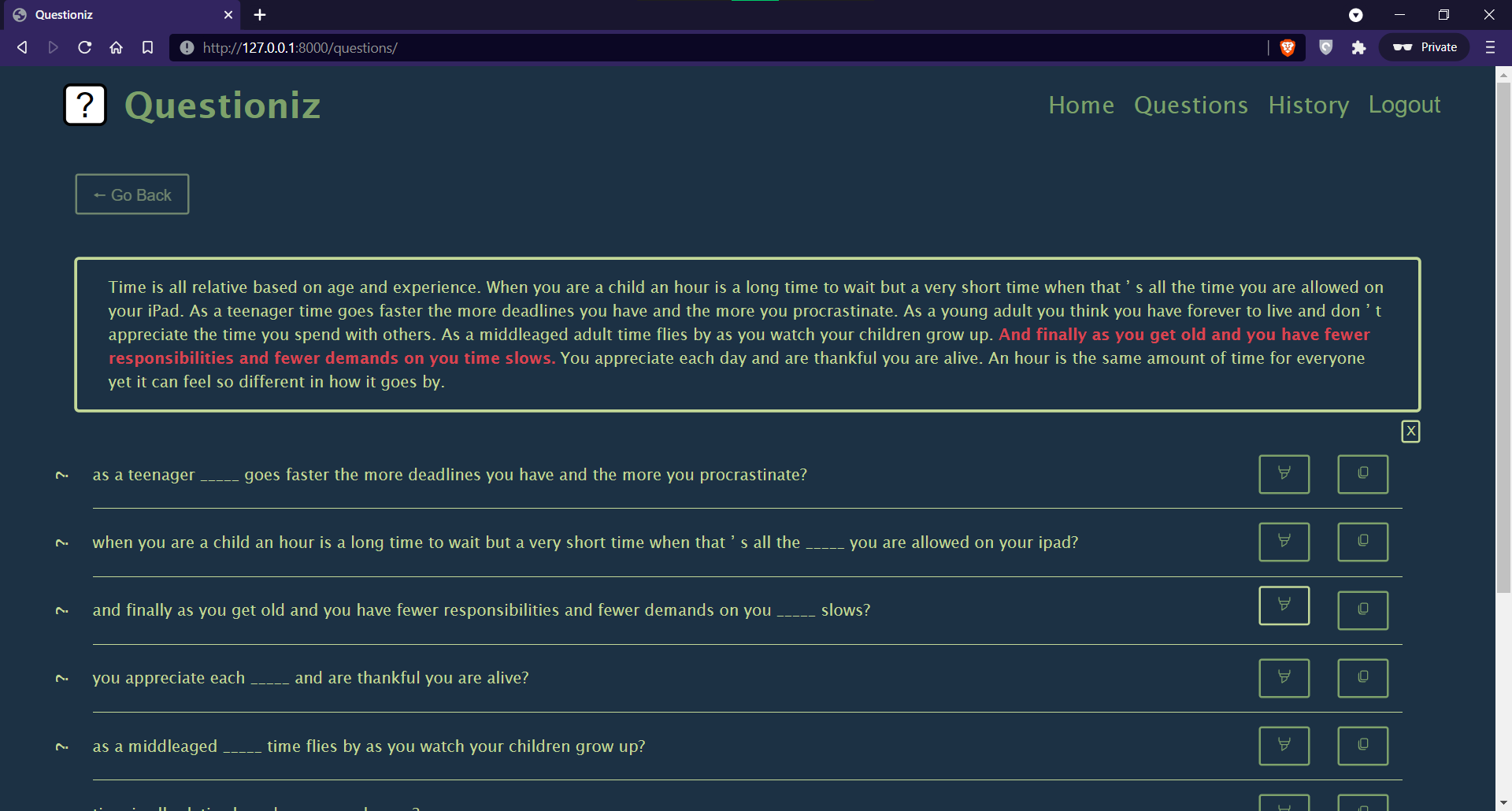
questions page



questions page with answer



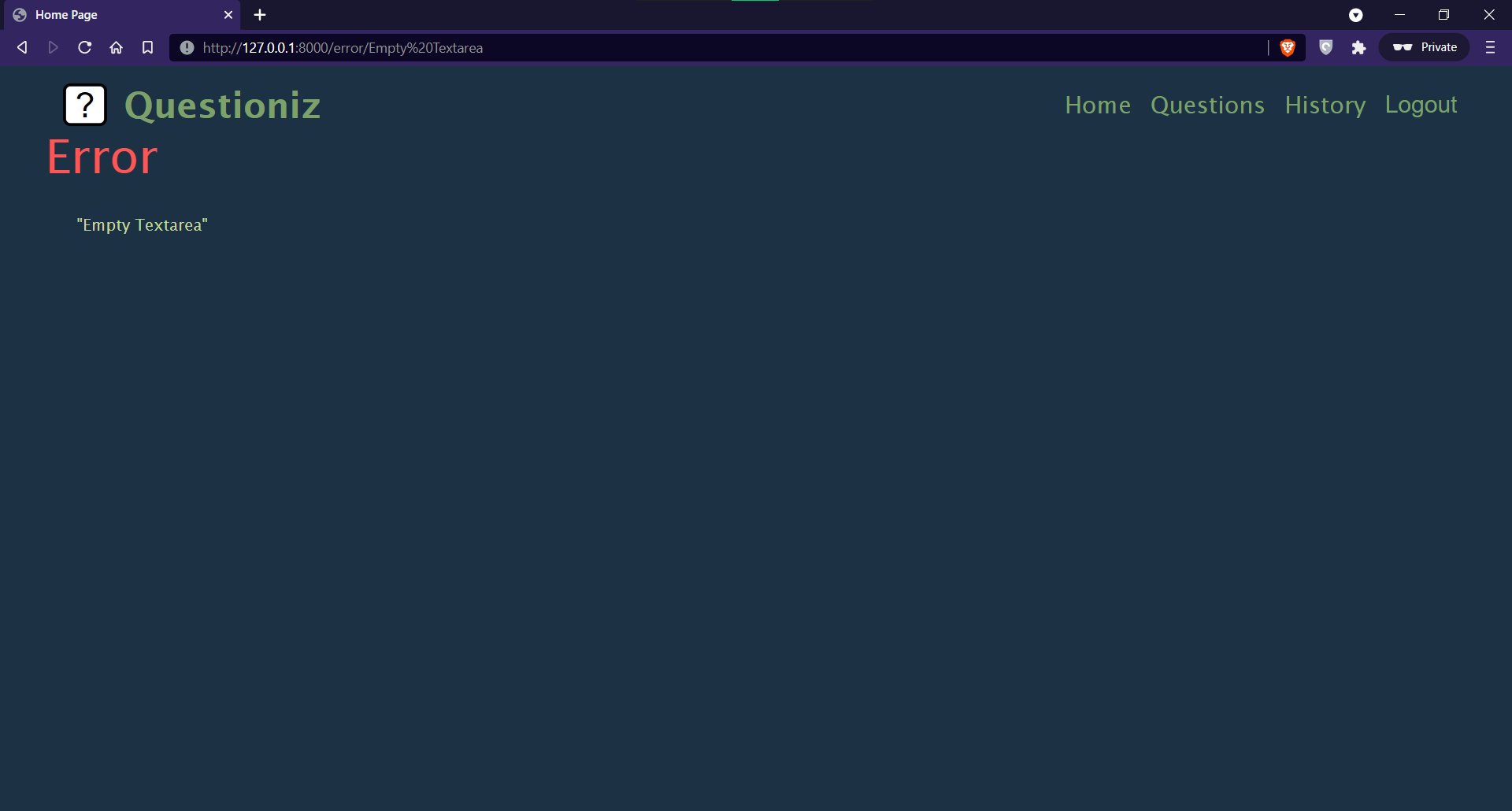
questions page highlight



history page



error page empty field



**Future Enhancements**

* Generate Wh- questions
* Generate MCQ Question
* Generate Mathematical Question
* Generate Short Answer Question
* Export and Download Generated Questions in PDF format
* Save specific question

**Bibliography**

[Python -](https://questgen.ai/) <https://www.python.org/>

Questgen - <https://questgen.ai/>

Quillionz - <https://www.quillionz.com/>

QuizEasy by KG-1510 - <https://github.com/KG-1510/QuizEasy>

Question-Generation by KristiyanVachev - <https://github.com/KristiyanVachev/Question-Generation>

NLTK - <https://www.nltk.org/>

Django - <https://docs.djangoproject.com/en/3.2/>