UNIVERSITY OF MICHIGAN-DEARBORN

Department of Computer and Information Science

CIS 571 Web Services Project 1

BY

Siyu Yang

UMID: 76998080

content

1. Introduction

1.1 Background

Memorizing words has always been one of the most difficult but important things in learning English. Although the importance of memorizing words has been emphasized many times, still almost no one likes to memorize words with a huge and boring dictionary. People have never given up looking for both fun and effective ways to memorize words with a huge and boring dictionary. People have never given up looking for both fun and effective ways to memorize words.

1.2 Goal

In order to give an interesting way to memorize words, I build this A Simple Word Game web application. This web application consists of 5 major functionalities, they are:

- Show the definitions of words and check the answer.
- Give two hints -- letter and sample of use about the word.
- Allow new user register and oud user login.
- Give one more hint -- translation in the native language.
- Store the words list of answers in database after login.

1.3 Overall architecture

```
~/Project
    |-- run.py
    |-- config.py
    |-- storage.py
    |-- config.py
    |__ /app
        |-- __init__.py
        |-- register.py
        |-- login.py
        |-- play.py
        |-- /soap
            |-- soap_utils.py
            |-- server.py
            |-- client.py
            |__ /templates
                |-- index.html
                |-- login.html
                |-- layout.html
                |__ ..
            |__ /static
                |-- style.css
                |-- google_api.json
```

1.4 Technologies

- Back-end: SQLite Database,
- Front-end: HTML, CSS
- APIs: Oxford Dictionaries API, Google Cloud Translation API
- Python Flask Web Framework, Zeep

2. Existing APIs

2.1 Oxford Dictionaries API

2.1.1 Brief Overview

The Oxford Dictionaries API gives access to our world-renowned dictionary data

in an ever-growing list of languages. Based on intensive language research programme – one of the largest in the world – data is up-to-date, accurate, and reliable and, for the first time, can be quickly and easily incorporated into your apps via one single API. The Oxford Dictionaries API includes:

- Flexible endpoints including headwords, parts of speech, synonyms, audio, example sentences, and more.
- Lexical frequency and n-gram information from the integrated LexiStats API.
- Data expertly pre-processed by the in-house engineers to ensure accuracy and consistency of format across datasets – in many languages, for the first time.
- Continually updated content, giving a immediate access to the latest words and linguistic resources.

2.1.2 Code Fragments

Oxford Dictionaries API:

```
# oxford API configuration more info: https://developer.oxforddictionaries.com/
            app_id = '2b16a5ff'
            app_key = '4f6e8cb06f2eab504ab5aa72a0d8422c'
            language = 'en'

            try:
                word_id = question
                url = 'https://od-api.oxforddictionaries.com/api/v2/entries/' +
language + '/' + word_id.lower()
                urlFR = 'https://od-
api.oxforddictionaries.com/api/v2/stats/frequency/word/' + language +
'/?corpus=nmc&lemma=' + word_id.lower()
                r = requests.get(url, headers={'app_id': app_id, 'app_key': app_key})
            except:
                flag = "NOK (OXFORD API)"
                print(flag + " " + question)
                db.execute("UPDATE dict SET flag = :flag WHERE word = :word",
flag=flag, word=question)
                db.execute("INSERT INTO bug (code, question) VALUES
```

```
(:code, :question)", code=flag, question=question)
                return bug()

            try:
                definition = []
                api = r.json()

                # iterate over json object to get list of definitions
                for i in api["results"]:
                    for j in i["lexicalEntries"]:
                        for k in j["entries"]:
                            for v in k["senses"]:
                                definition.append(v["definitions"])
            except:
                flag = "NOK (OXFORD DEF)"
                print(flag + " " + question)
                db.execute("UPDATE dict SET flag = :flag WHERE word = :word",
flag=flag, word=question)
                buglog = db.execute("INSERT INTO bug (code, question) VALUES
(:code, :question)", code=flag, question=question)
                return bug()

            try:
                # try to get first exaple of use from json object
                samples =
r.json()["results"][0]["lexicalEntries"][0]["entries"][0]["senses"][0]["examples"][0][
"text"]
                censoredS = str(samples).replace(question, len(question) * ".")
            except:
                flag = "NOK (JSON -> SAMPLES)"
                print(flag + " " + question)
                buglog = db.execute("INSERT INTO bug (code, question) VALUES
(:code, :question)", code=flag, question=question)
                censoredS = "sorry, not this time..."

            # make "letters hint"
```

```
        letters = []
        for c in question:
            letters += c
        shuffle(letters)
```

Here, in my play() function, I use the Oxford Dictionary API to get words' information. Firstly, I select a random word from the database. Then take the word as the input of Oxford Dictionary API, all the definitions of this word are the output as well as the question in the game.

Besides, in order to make several hints, I separate the word into letters and give every word a sample of use. Users will be benefit to remember words in the sentences. Finally, I return all of my outputs to the play.html as:

```
# render play template
   return render_template("play.html", definition=definition, censoredS=censoredS, question=question, letters=letters)
```

2.2 Google Cloud Translation API

2.2.1 Brief Overview

The Google Cloud Translation API is a part of Google's larger Cloud Machine Learning API family. This API translates text between thousands of language pairs, with new features for this latest version that includes; Glossaries and Batch requests. It enables you with the ability to create a custom dictionary to correctly and consistently translate terms that are customer-specific and to make an asynchronous request to translate large amounts of text. The Cloud Translation API lets developers programmatically integrate the service with third party sites and applications.

2.2.2 Code Fragments

Google Cloud Translation API:

```
# google translator API // please download and copy to static/ folder credentials file.
        # More info: https://cloud.google.com/translate/
        if session.get("user_id") != None:
```

```
                # registered user
                try:
                    os.environ["GOOGLE_APPLICATION_CREDENTIALS"] =
"static/google_api.json"
                    translate_client = translate.Client()

                    target = session["nativelang"]
                    text = question

                    translation = translate_client.translate(
                        text,
                        source_language='en',
                        target_language=target)
                except:
                    flag = "NOK (GOOGLE TRANSLATOR API)"
                    print(flag + " " + question)
                    buglog = db.execute("INSERT INTO bug (code, question)
VALUES (:code, :question)", code=flag, question=question)
                    translation = {'translatedText': 'sorry, not this time...'}
            # unregisterd user
            else:
                translation = {'translatedText': 'sorry, not this time...'}

            # render play template
            return render_template("play.html",
translation=translation['translatedText'])
```

Here, in my play() function, I also use the google translator API to get words' translation. When the users register in the system, their native language information will be collected and stored into the database. The translation API will return the word's translation in the user's native language as another hint to the game. I apply for the key of Google Cloud Translation API and save the json file named google_api. Then I use the following command to get the translation and return it to the play.html also as following:

```
result = translate_client.translate( text, source_language, target_language=target)
```

```
# render play template
return render_template("play.html", translation=translation['translatedText'])
```

3. Own APIs

3.1 REST API

3.1.1 Design

I design two REST API for the register and login system.

In the register system, the register API has HTTP GET and HTTP POST two protocols. The massage format is JSON. As for POST, the request is the information of new user, and the response is apology or save data into dataset, as:

Request:	Response:
{ "username": "siyu" "hash": "password" "nativelang": "Chinese" }	{ "apology": "concept of apology" "user_id": "new_user" " nativelang": "Chinese" }

In the login system, the login API also has two protocols, JSON format massage like register API. As for POST, the request is the information of new user, and the response is apology or save data into dataset, as:

Request:	Response:
{ "username": "siyu" "hash": "password" }	{ "apology": "concept of apology" "user_id": "new_user" " nativelang": "Chinese" }

3.1.2 Code Fragments

Register API:
@app.route("/register", methods=["GET", "POST"]) def register(): """Register user""" # Forget any user_id

```
    session.clear()

    # User reached route via POST (as by submitting a form via POST)
    if request.method == "POST":

        # Ensure username was submitted
        if not request.form.get("username"):
            return apology("must provide username")

        # Ensure username is avilable
        new_user = db.execute("SELECT * FROM user WHERE username
= :username", username=request.form.get("username"))
        if len(new_user) > 0:
            return apology("username allready exist")

        # Ensure password was submitted
        elif not request.form.get("password"):
            return apology("must provide password")

        # Ensure password confirmation was submitted
        elif not request.form.get("confirmation"):
            return apology("must provide password confirmation")

        # Ensure password and password confirmation match
        elif request.form.get("password") != request.form.get("confirmation"):
            return apology("password and password confirmation must match")

        # Register new user
        new_user = db.execute("INSERT INTO user (username, hash, nativelang)
VALUES (:username, :hash, :nativelang)",
                              username=request.form.get("username"),
hash=generate_password_hash(request.form.get("password")),
                              nativelang=request.form.get("nativelang"))

        # Remember which user has logged in
        session["user_id"] = new_user
```

```
        session["nativelang"] = request.form.get("nativelang")

        # Redirect user to play page
        return redirect("/play")

    # User reached route via GET (as by clicking a link or via redirect)
    else request.method == "GET":
        return render_template("register.html")
```

Login API:

```
@app.route("/login", methods=["GET", "POST"])
def login():
    """Log user in"""

    # Forget any user_id
    session.clear()

    # User reached route via POST (as by submitting a form via POST)
    if request.method == "POST":

        # Ensure username was submitted
        if not request.form.get("username"):
            return apology("must provide username")

        # Ensure password was submitted
        elif not request.form.get("password"):
            return apology("must provide password")

        # Query database for username
        rows = db.execute("SELECT * FROM user WHERE username
= :username", username=request.form.get("username"))

        # Ensure username exists and password is correct
        if len(rows) != 1 or not check_password_hash(rows[0]["hash"],
request.form.get("password")):
            return apology("invalid username and/or password")
```

```
        # Remember which user has logged in
        session["user_id"] = rows[0]["id"]
        session["nativelang"] = rows[0]["nativelang"]

        # Redirect user to play page
        return redirect("/play")

    # User reached route via GET (as by clicking a link or via redirect)
    else request.method == "GET":
        return render_template("login.html")
```

3.2 SOAP API

3.2.1 Design

I use python's SOAP client - Zeep to build a SOAP API which can take the user information as input and return the wordlist of this user as output.

First, I import Client from zeep, get the WSDL file and create the Userkey in client.py. The response of client is return a get_mywords service. Then the client send the SOAP massages from WSDL file to the localhost URL by HTTP POST. After receive the POST request from client, the server translate the SOAP massage and decide to call method get_mywords() to finish the service and return the response as response(etree_to_string(to_return.content), mimetype='text/xml').

3.2.2 Code Fragments

SOAP API (server):

```
import faker
import soap_utils
from zeep import Client
from zeep.wsdl.utils import etree_to_string
from flask import Flask, request
from flask import make_response, Response
```

```
fake = faker.Faker()

APPLICATION = '{spyne.examples.hello.http}Application'
client = Client(wsdl='user.wsdl')
binding = client.wsdl.bindings[APPLICATION]

app = Flask('wordslist_api')

# Types from WSDL
# we still need to type this, but Zeep makes sure that types comply with the WSDL
mywords = client.get_type('ns0:mywords')

# Actual implementation of endpoints (return random values)
def get_mywords(key):
    return Words(
        userid = key.userid,
        word = fake.word(),
        timestamp = fake.timestamp()
    )

# Actual SOAP endpoint
@app.route('/', methods=['POST', 'GET'])
def soap_endpoint():

    if request.method == 'GET':
        with open('word.wsdl', 'r') as fp:
            return Response(fp.read(), mimetype='text/xml')

    document = soap_utils.parse_xml(request.data)
    operation_name = soap_utils.operation_name(document)
    operation = binding.get(operation_name)

    request_object = operation.input.deserialize(document)

    if operation_name == 'get_mywords':
```

```
            response_object = get_mywords(request_object)
        else:
            raise NotImplementedError()
        to_return = operation.output.serialize(response_object)
        return Response(etree_to_string(to_return.content), mimetype='text/xml')

@app.errorhandler(Exception)
def handle_server_error(error):
        # Needs to return proper errors
        response = make_response('<error>SOAP Error</error>')
        response.mimetype = 'text/xml'
        response.status_code = 500
        return response

if __name__ == '__main__':
        app.run()
```

SOAP API (client):

```
from zeep import Client

# client = Client(wsdl='http://localhost:7789/?wsdl')
client = Client(wsdl='word.wsdl')

UserKey = client.get_type('ns0:userKey')
siyu_key = UserKey(username='siyu', hash='123456')

print(client.service.get_mywords(siyu_key))
```

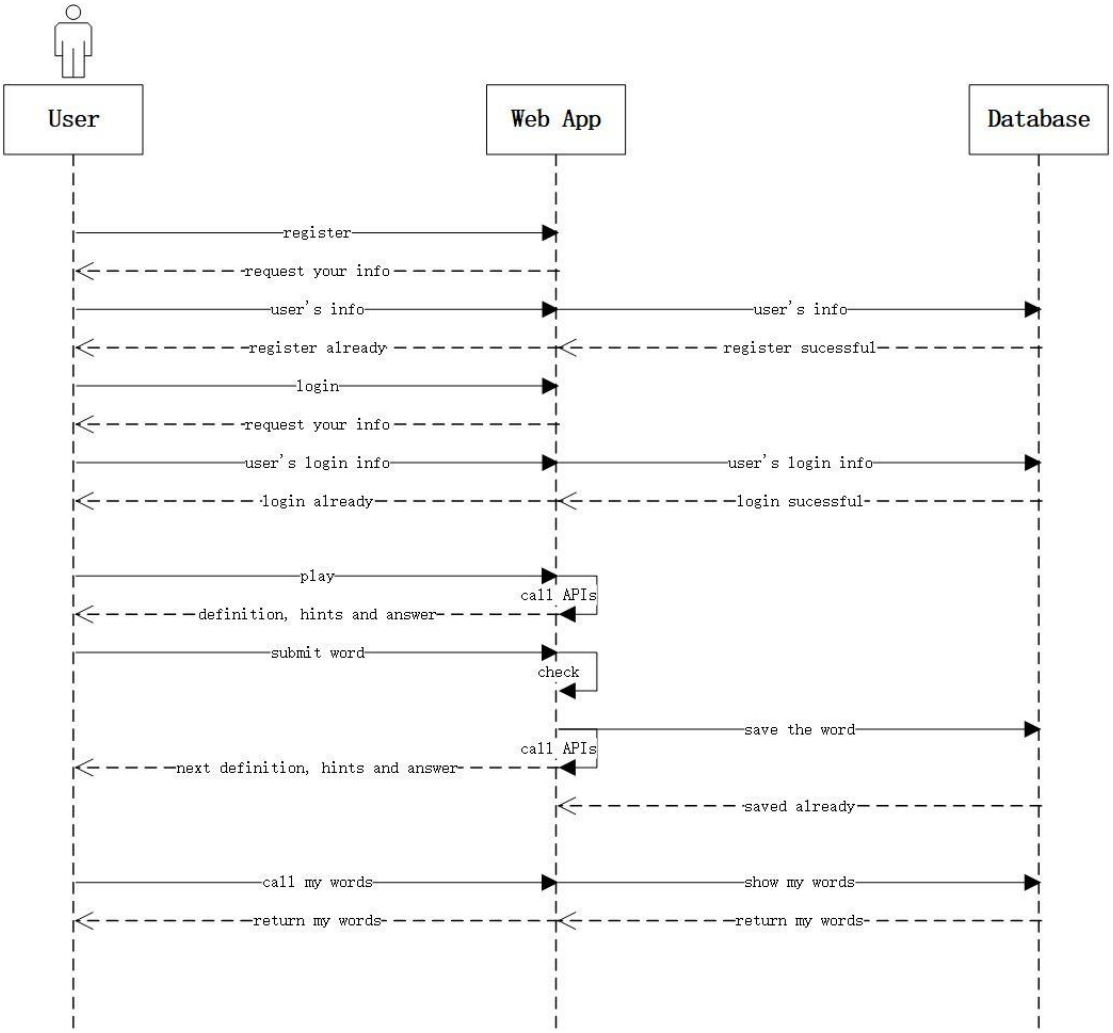
SOAP API (soap_utils):

```
import lxml
import zeep.loader
from zeep.settings import Settings

def operation_name(document):
        return lxml.etree.QName(document[0][0].tag).localname
```

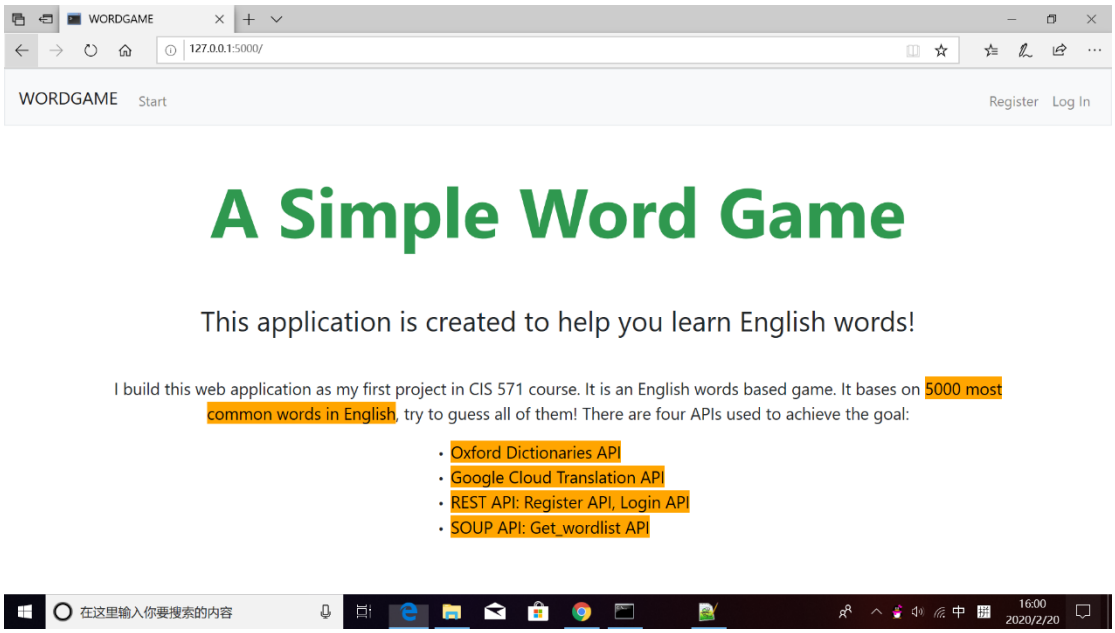
```
def parse_xml(data):
    return zeep.loader.parse_xml(data, None, settings=Settings())
```

4. UML Sequence Diagram

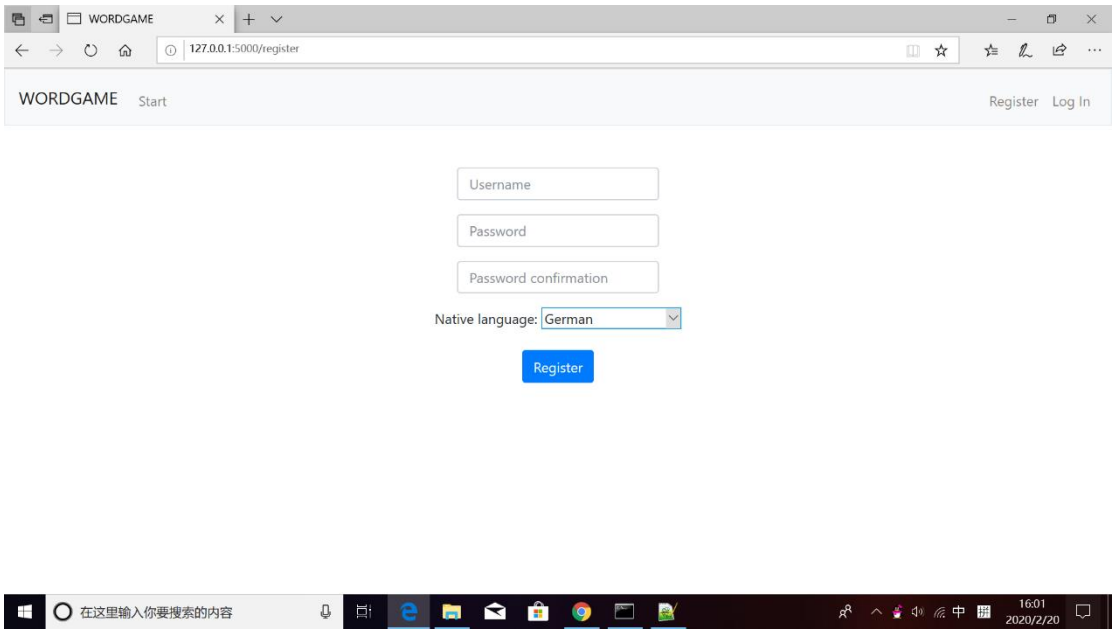


5. Screenshots

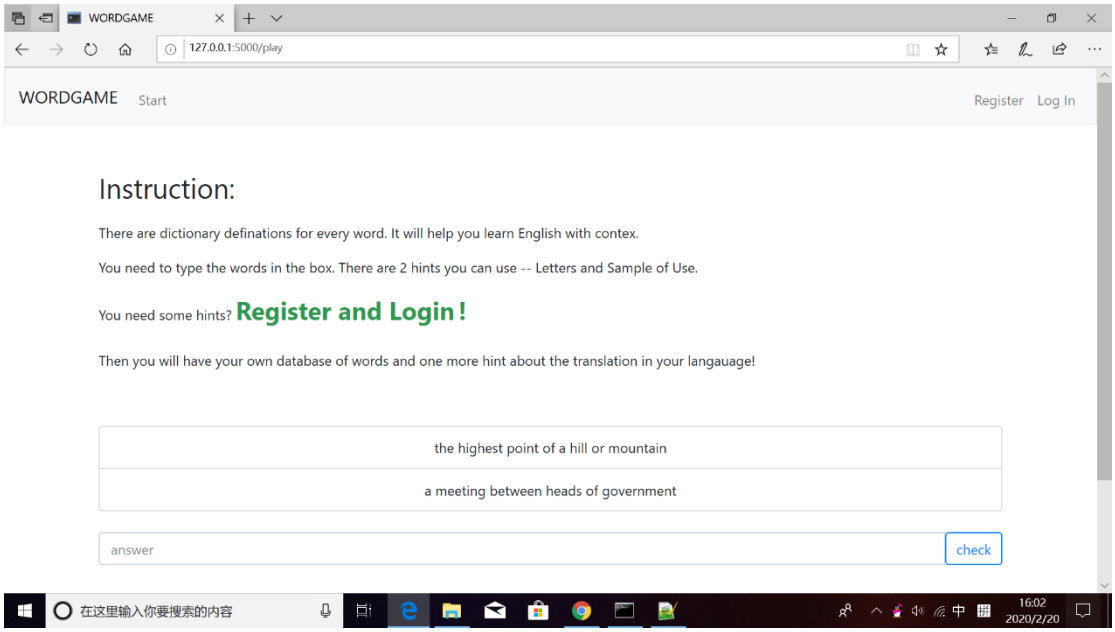
5.1 Index:



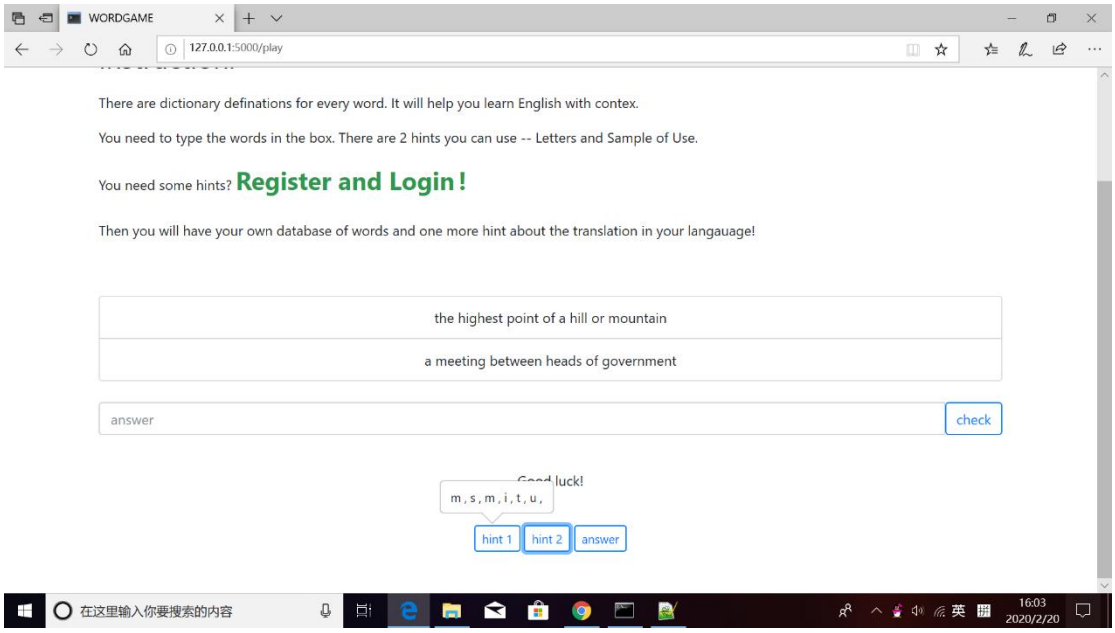
5.2 Register:

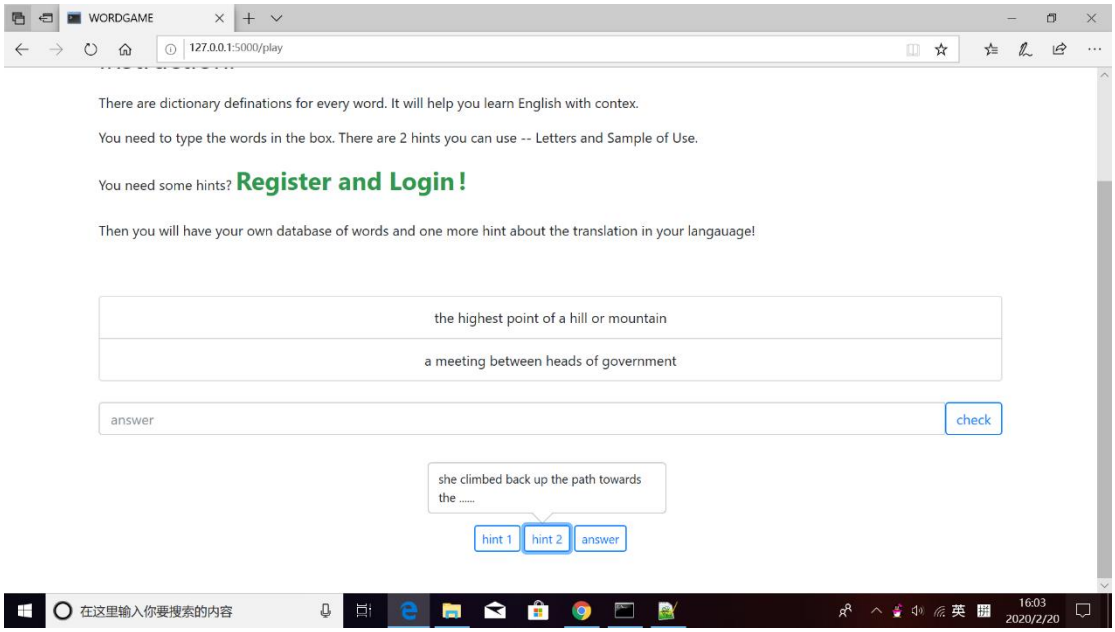


5.3 Play without login:

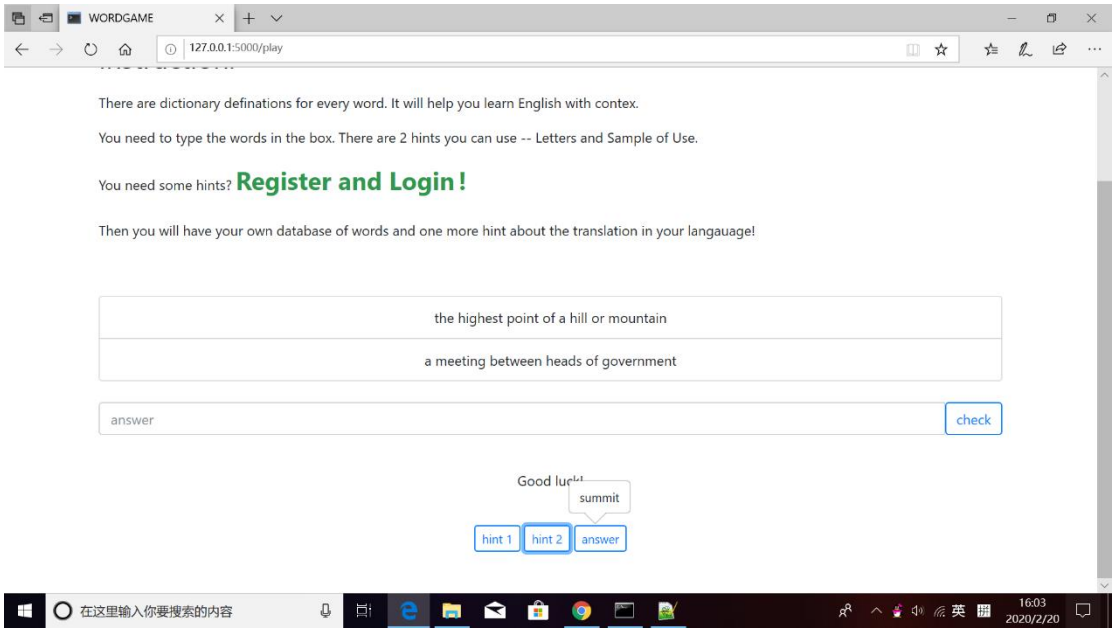

WORDGAME
127.0.0.1:5000/play
WORDGAME Start
Register Log In
Instruction:
There are dictionary defination for every word. It will help you learn English with contex.
You need to type the words in the box. There are 2 hints you can use -- Letters and Sample of Use.
You need some hints? Register and Login!
Then you will have your own database of words and one more hint about the translation in your langauage!
the highest point of a hill or mountain
a meeting between heads of government
answer
check
在这里输入你要搜索的内容
16:02
2020/2/20

5.4 Hint:

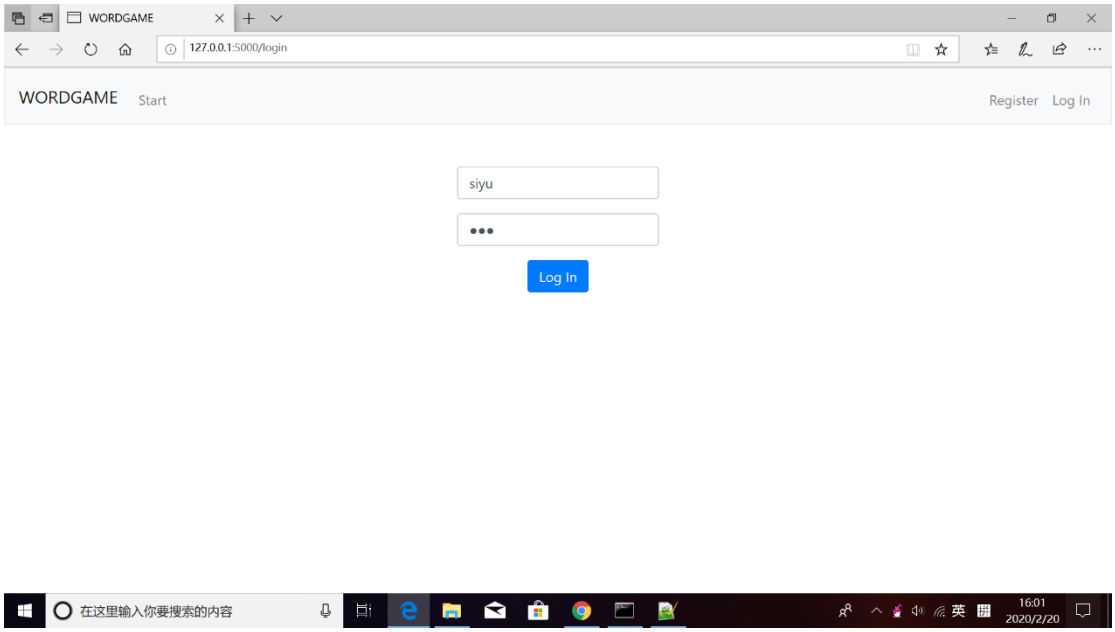

WORDGAME
127.0.0.1:5000/play
There are dictionary defination for every word. It will help you learn English with contex.
You need to type the words in the box. There are 2 hints you can use -- Letters and Sample of Use.
You need some hints? Register and Login!
Then you will have your own database of words and one more hint about the translation in your langauage!
the highest point of a hill or mountain
a meeting between heads of government
answer
check
Good luck!
m , s , m , i , t , u ,
hint 1
hint 2
answer
在这里输入你要搜索的内容
16:03
2020/2/20


WORDGAME
127.0.0.1:5000/play
There are dictionary defination for every word. It will help you learn English with contex.
You need to type the words in the box. There are 2 hints you can use -- Letters and Sample of Use.
You need some hints? Register and Login!
Then you will have your own database of words and one more hint about the translation in your langauage!
the highest point of a hill or mountain
a meeting between heads of government
answer
check
she climbed back up the path towards the .....
hint 1
hint 2
answer
在这里输入你要搜索的内容
16:03
2020/2/20

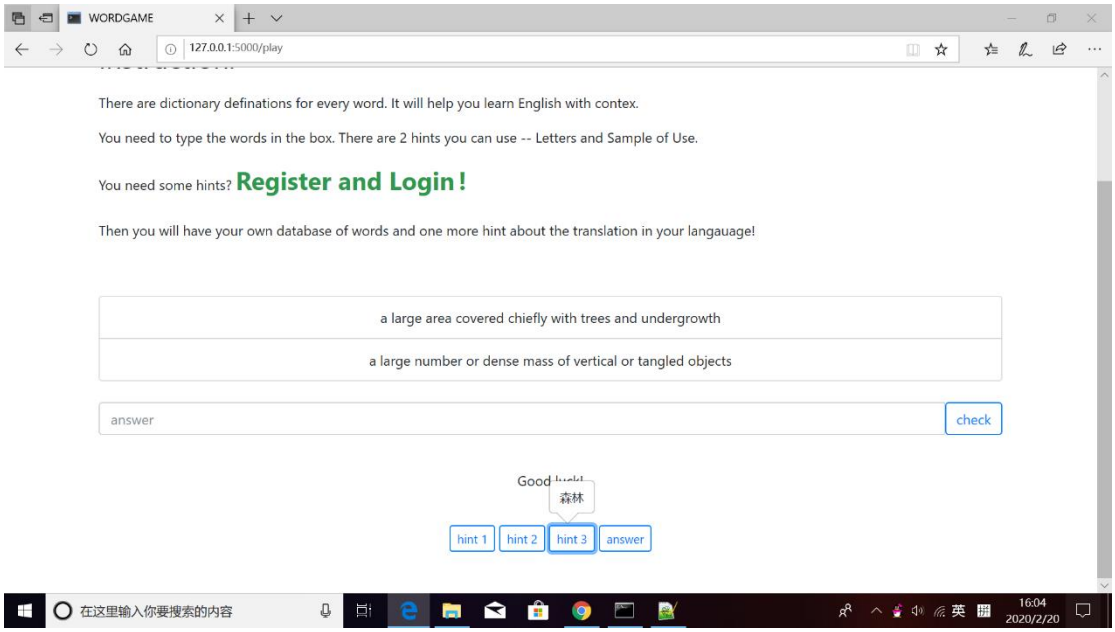
5.5 Answer:


WORDGAME
127.0.0.1:5000/play
There are dictionary defination for every word. It will help you learn English with contex.
You need to type the words in the box. There are 2 hints you can use -- Letters and Sample of Use.
You need some hints? Register and Login!
Then you will have your own database of words and one more hint about the translation in your langauage!
the highest point of a hill or mountain
a meeting between heads of government
answer
check
summit
hint 1
hint 2
answer
在这里输入你要搜索的内容
16:03
2020/2/20

5.6 Login:



5.7 Hint 3 (Play after login):



6. References

[1] https://programminghistorian.org/en/lessons/creating-apis-with-python-and-flask

[2]https://rapidapi.com/blog/20-tutorials-on-how-to-create-your-own-api-sorted-by-programming-language/

[3]https://github.com/rock-chock/color_emotions

[4]https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/

[5]https://www.w3school.com.cn/soap/soap_example.asp

[6]https://python-zeep.readthedocs.io/en/master/in_depth.html