Proiect Inginerie Software

-Remote Debugging App-

Autori:

Podaru Bogdan

Pentek Tamas

Rentea Robert

A. Rezumat

Aplicatia este un tool de depanare ce poate fi descarcat de pe un website. In urma descarcarii pe calculatorul clientului si asignarea unui depanator, acesta din urma poate investiga cu ajutorul comenzilor in terminal anumite probleme raportate de client. Descarcarea executabilului se realizeaza printr-un protocol de transfer (TCP) de pe web server. In final, clientul poate oferi feedback aplicatiei printr-un review.

Autor sectiune : Podaru Bogdan

Review-urile clientilor sunt procesate cu ajutorul Machine Learning (Text classification – Sentiment Analysis) pentru a stabili daca un review este negativ sau pozitiv (clientul este sau nu multumit de prestatia moderatorului sau de functionalitatea tool-ului). Modelul utilizat (reteaua) este antrenat pe setul de date IMDB, avand o acuratete de aproximativ 90%. Datele obtinute sunt valori cuprinse intre 0 si 1. O valoare mai mica de 0.5 semnifica un client nemultumit, iar alta mai mare de 0.5 semnifica unul multumit. Se realizeaza un grafic in 2 coloane ce prezinta numarul de clienti multumiti si numarul de clienti nemultumiti.

Autor sectiune: Rentea Robert

Specialistii vor lucra cu un fisier executabil(bot) care va rula pe calculatorului clientului asteptand o conexiune. Odata ce un specialist se conecteaza la bot acesta va putea trimite comenzi de terminal pe care botul le va putea executa direct pe sistemul clientului.

Botul este construit cu ajutorul framework-ului RPyC prin care pe calculatorul clientului se construieste un server care asculta pe un anumit port dupa conexiuni de la un specialist.

Autor sectiune: Pentek Tamas

Fisierul executabil (bot) este descarcat pe calculatorul clientului prin FTP (File Transfer Protocol). FTP-ul functioneaza pe principiul client-server, in cazul nostru serverul este Django ftp-server la care se conecteaza clientul si automat se descarca fisierul executabil. Partea de client este implementata cu ajutorul ftplib (FTP library) .

Clientul descarca un fisier executabil care ruleaza un script python, acest script descarca bot-ul pe calculatorul clientului si dupa ce clientul deschide acest fisier, un specialist se conecteaza la bot si se executa comenzi de terminal prin care rezolva anumite probleme raportate de client.

B. Referinte + Bibliografie

- Podaru Bogdan :
  1. https://www.tensorflow.org/tutorials/keras/text_classification_with_hub
  2. https://keras.io/
  3. https://www.learnopencv.com/neural-networks-a-30000-feet-view-for-beginners/
- Rentea Robert :
  1. https://rpyc.readthedocs.io/en/latest/tutorial.html
- Pentek Tamas :
  1. https://docs.python.org/3/library/ftplib.html
  2. https://www.techinfected.net/2017/07/create-simple-ftp-server-client-in-python.html

C. MiniProiect (comun)

Pentru realizarea acestuia s-a folosit Django + React js.

C.1 Cod sursa

a) models.py

```
from django.db import models

class Todo(models.Model):
title = models.CharField(max_length=120)
description = models.TextField()
completed = models.BooleanField(default=False)

def _str_(self):
return self.title
```

b) views.py

```
from django.shortcuts import render
from rest_framework import viewsets
from .serializers import TodoSerializer
from .models import Todo

class TodoView(viewsets.ModelViewSet):
serializer_class = TodoSerializer
```

```
queryset = Todo.objects.all()
```

c) serializers.py

```
from rest_framework import serializers
from .models import Todo
class TodoSerializer(serializers.ModelSerializer):
class Meta:
model = Todo
fields = ('id', 'title', 'description', 'completed')
```

d) admin.py

```
from django.contrib import admin
from .models import Todo
class TodoAdmin(admin.ModelAdmin):
     list_display = ('title', 'description', 'completed')
admin.site.register(Todo, TodoAdmin)
```

e) apps.py

```
from django.apps import AppConfig
class TodoConfig(AppConfig):
name = 'todo'
```

f) urls.py

```
from django.contrib import admin
from django.urls import path, include
from rest_framework import routers
from todo import views

router = routers.DefaultRouter()
router.register(r'todos', views.TodoView, 'todo')

urlpatterns = [
path('admin/', admin.site.urls),
path('api/', include(router.urls))
]
```

g) settings.py

```
import os

os.path.join(BASE_DIR, ...)
BASE_DIR =
os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
SECRET_KEY =
'%h5jh243&4_&0fi*z#i)^@iq0w#xs!!owc@e3=p8dsbi-_sp01'
DEBUG = True
ALLOWED_HOSTS = []
INSTALLED_APPS = [
'django.contrib.admin',
'django.contrib.auth',
```

```
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'corsheaders',
'rest_framework',
'todo',
]
MIDDLEWARE = [
'corsheaders.middleware.CorsMiddleware',
'django.middleware.security.SecurityMiddleware',
'django.contrib.sessions.middleware.SessionMiddleware',
'django.middleware.common.CommonMiddleware',
'django.middleware.csrf.CsrfViewMiddleware',
'django.contrib.auth.middleware.AuthenticationMiddleware',
'django.contrib.messages.middleware.MessageMiddleware',
'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'backend.urls'

TEMPLATES = [
{
'BACKEND': 'django.template.backends.django.DjangoTemplates',
'DIRS': [],
'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',
],
},
},
]

WSGI_APPLICATION = 'backend.wsgi.application'
DATABASES = {
'default': {
'ENGINE': 'django.db.backends.sqlite3',
'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
}
}

AUTH_PASSWORD_VALIDATORS = [
{
'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
},
{
```

```
'NAME':
'django.contrib.auth.password_validation.MinimumLengthValidato
r',

},

{

'NAME':
'django.contrib.auth.password_validation.CommonPasswordValidat
or',

},

{

'NAME':
'django.contrib.auth.password_validation.NumericPasswordValida
tor',

},

]

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


STATIC_URL = '/static/'

CORS_ORIGIN_WHITELIST = (

'http://localhost:3000',

)
```

h) App.css

```
.App {

text-align: center;
```

```
}

.App-logo {
height: 40vmin;
}

.App-header {
background-color: #282c34;
min-height: 100vh;
display: flex;
flex-direction: column;
align-items: center;
justify-content: center;
font-size: calc(10px + 2vmin);
color: white;
}

.App-link {
color: #09d3ac;
}
```

i) App.js

```
import React, { Component } from "react";
import Modal from "./components/Modal";
import axios from "axios";
```

```
class App extends Component {
constructor(props) {
super(props);
this.state = {
viewCompleted: false,
activeItem: {
title: "",
description: "",
completed: false
},
todoList: []
};
}
componentDidMount() {
this.refreshList();
}
refreshList = () => {
axios
.get("http://localhost:8000/api/todos/")
.then(res => this.setState({ todoList: res.data }))
.catch(err => console.log(err));
};
displayCompleted = status => {
if (status) {
return this.setState({ viewCompleted: true });
```

```
}
return this.setState({ viewCompleted: false });
};
renderTabList = () => {
return (
<div className="my-5 tab-list">
<span
onClick={() => this.displayCompleted(true)}
className={this.state.viewCompleted ? "active" : ""}
>
complete
</span>
<span
onClick={() => this.displayCompleted(false)}
className={this.state.viewCompleted ? "" : "active"}
>
Incomplete
</span>
</div>
);
};
renderItems = () => {
const { viewCompleted } = this.state;
const newItems = this.state.todoList.filter(
item => item.completed === viewCompleted
);
```

```
return newItems.map(item => (

<li

key={item.id}

className="list-group-item d-flex justify-content-between
align-items-center"

>

<span

className={`todo-title mr-2 ${

this.state.viewCompleted ? "completed-todo" : ""

}`}

title={item.description}

>

{item.title}

</span>

<span>

<button

onClick={() => this.editItem(item)}

className="btn btn-secondary mr-2"

>

{" "}

Edit{" "}

</button>

<button

onClick={() => this.handleDelete(item)}

className="btn btn-danger"

>

Delete{" "}
```

```
</button>
</span>
</li>
));
};
toggle = () => {
this.setState({ modal: !this.state.modal });
};
handleSubmit = item => {
this.toggle();
if (item.id) {
axios
.put(`http://localhost:8000/api/todos/${item.id}/`, item)
.then(res => this.refreshList());
return;
}
axios
.post("http://localhost:8000/api/todos/", item)
.then(res => this.refreshList());
};
handleDelete = item => {
axios
.delete(`http://localhost:8000/api/todos/${item.id}`)
.then(res => this.refreshList());
};
createItem = () => {
```

```
const item = { title: "", description: "", completed: false };
this.setState({ activeItem: item, modal: !this.state.modal });
};
editItem = item => {
this.setState({ activeItem: item, modal: !this.state.modal });
};
render() {
return (
<main className="content">
<h1 className="text-white text-uppercase text-center
my-4">Todo app</h1>
<div className="row ">
<div className="col-md-6 col-sm-10 mx-auto p-0">
<div className="card p-3">
<div className="">
<button onClick={this.createItem} className="btn btn-primary">
Add task
</button>
</div>
{this.renderTabList()}
<ul className="list-group list-group-flush">
{this.renderItems()}
</ul>
</div>
</div>
</div>
{this.state.modal ? (
```

```
<Modal
activeItem={this.state.activeItem}
toggle={this.toggle}
onSave={this.handleSubmit}
/>
) : null}
</main>
);
}
}
export default App;
```

j) index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import 'bootstrap/dist/css/bootstrap.min.css';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root'));

serviceWorker.unregister();
```

k) index.css

```
/__ frontend/src/index.css __/


body {
margin: 0;
padding: 0;
font-family: -apple-system, BlinkMacSystemFont, "Segoe UI",
"Roboto", "Oxygen",
"Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica
Neue",
sans-serif;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
background-color: #282c34;
}
.todo-title {
cursor: pointer;
}
.completed-todo {
text-decoration: line-through;
}
.tab-list > span {
padding: 5px 8px;
border: 1px solid #282c34;
border-radius: 10px;
margin-right: 5px;
cursor: pointer;
}
```

```
.tab-list > span.active {
background-color: #282c34;
color: #ffffff;
}
```

l) Modal.js

```
// frontend/src/components/Modal.js

import React, { Component } from "react";
import {
Button,
Modal,
ModalHeader,
ModalBody,
ModalFooter,
Form,
FormGroup,
Input,
Label
} from "reactstrap";

export default class CustomModal extends Component {
constructor(props) {
super(props);
this.state = {
```

```
activeItem: this.props.activeItem

};

}

handleChange = e => {

let { name, value } = e.target;

if (e.target.type === "checkbox") {

value = e.target.checked;

}

const activeItem = { ...this.state.activeItem, [name]: value
};

this.setState({ activeItem });

};

render() {

const { toggle, onSave } = this.props;

return (

<Modal isOpen={true} toggle={toggle}>

<ModalHeader toggle={toggle}> Todo Item </ModalHeader>

<ModalBody>

<Form>

<FormGroup>

<Label for="title">Title</Label>

<Input

type="text"

name="title"

value={this.state.activeItem.title}

onChange={this.handleChange}

placeholder="Enter Todo Title"
```

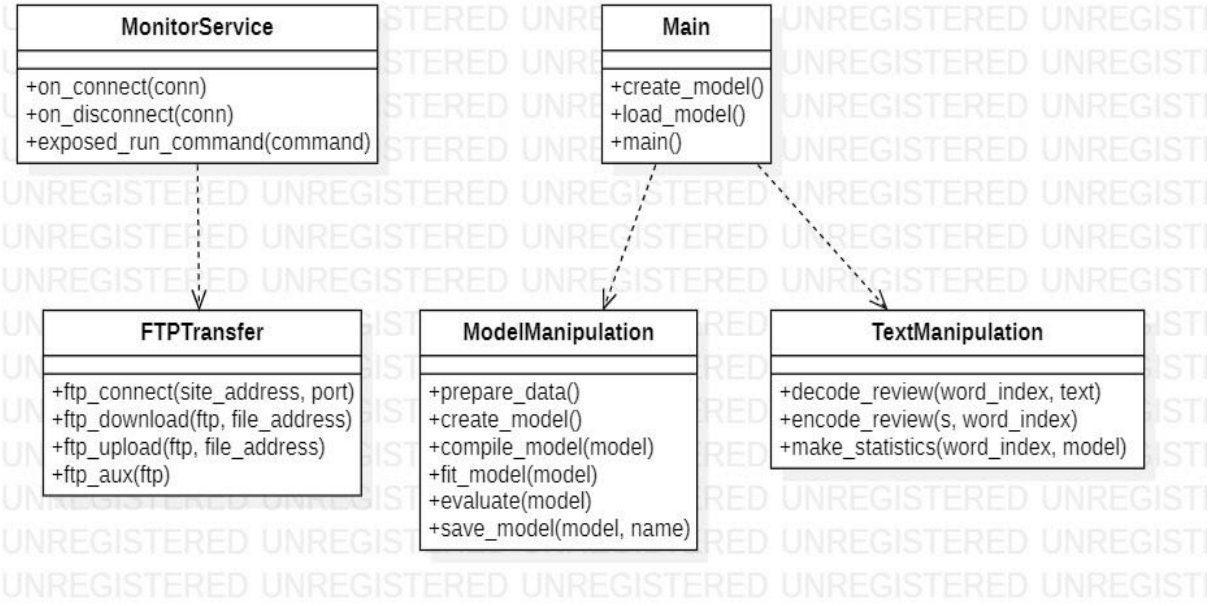
```
/>
</FormGroup>
<FormGroup>
<Label for="description">Description</Label>
<Input
type="text"
name="description"
value={this.state.activeItem.description}
onChange={this.handleChange}
placeholder="Enter Todo description"
/>
</FormGroup>
<FormGroup check>
<Label for="completed">
<Input
type="checkbox"
name="completed"
checked={this.state.activeItem.completed}
onChange={this.handleChange}
/>
Completed
</Label>
</FormGroup>
</Form>
</ModalBody>
<ModalFooter>
```

```
<Button color="success" onClick={() =>
onSave(this.state.activeItem)}>

Save

</Button>

</ModalFooter>

</Modal>

);

}

}
```

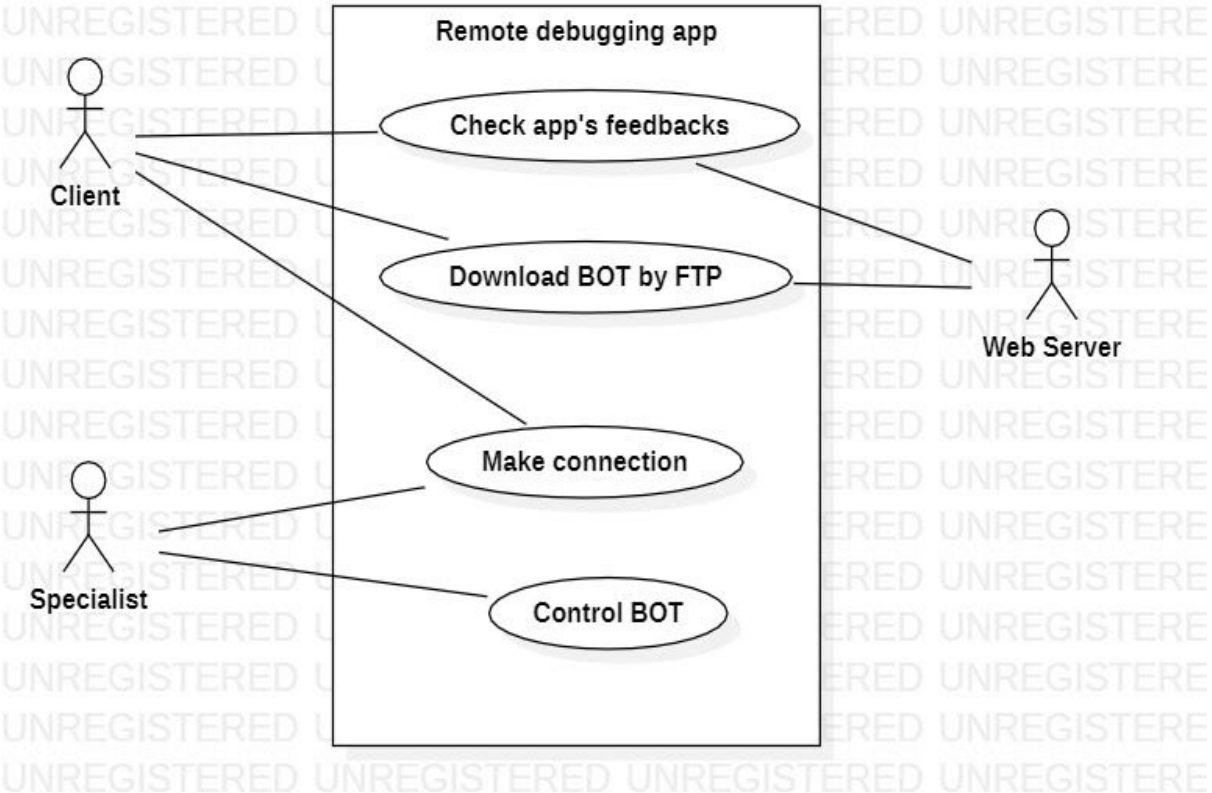
D. Descrierea proiectului final

D.1 Diagrame

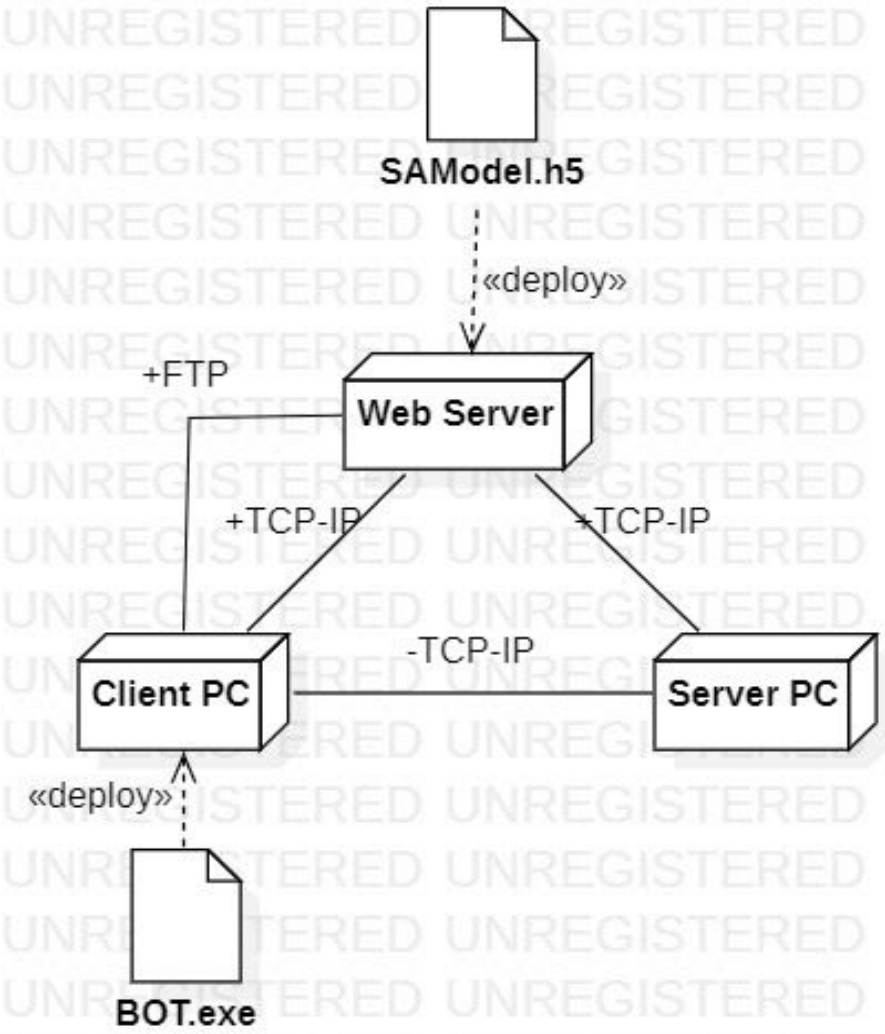
D.1.1 Diagrama de clase



D.1.2 Diagrama de cazuri de utilizare



D.1.3 Diagrama de deployment


SAModel.h5
«deploy»
+FTP
Web Server
+TCP-IP
+TCP-IP
-TCP-IP
Client PC
Server PC
«deploy»
BOT.exe

D.2 Cod sursa

---Podaru Bogdan---

a) ModelManipulation.py

```
import numpy as np

from keras.datasets import imdb

from keras.preprocessing.sequence import pad_sequences

from keras import Sequential, layers


# resolve allow_pickle error within load_data method

# problem in numpy module

np_load_old = np.load

np.load = lambda *a, **k: np_load_old(*a, allow_pickle=True,
**k)



class ModelManipulation:

     def __init__(self):

          # load and split data into train and test

          (train_data, train_labels), (test_data, test_labels)
          = imdb.load_data(num_words=10000)

          self.test_data = test_data

          self.test_labels = test_labels

          self.train_data = train_data

          self.train_labels = train_labels


     def prepare_data(self):
```

```
        # get the dictionary where entries are words mapped
        to integers

        word_index = imdb.get_word_index()

        word_index = {k:(v+3) for k, v in
        word_index.items()} # reserve first 4 indices for
        unknown words like spaces or words not from imdb
        data set

        word_index["<PAD>"] = 0

        word_index["<START>"] = 1

        word_index["<UNK>"] = 2

        word_index["<UNUSED>"] = 3

        # allow maximum of 250 words in a review

        # if the number of words < 250 then add padding tags

        self.train_data = pad_sequences(self.train_data,
        value=word_index["<PAD>"], padding="post",
        maxlen=250)

        self.test_data = pad_sequences(self.test_data,
        value=word_index["<PAD>"], padding="post",
        maxlen=250)

        return word_index


def create_model(self):

        # create the model

        model = Sequential()

        # 16 dimensions for each word vector from those
        10000 words

        # group similar words

        model.add(layers.Embedding(10000, 16))

        # scale down the entry dimension of the previous
        layer by averaging words vectors

        model.add(layers.GlobalAveragePooling1D())
```

```
        # add complexity by activation function rectified
        liniar unit

        model.add(layers.Dense(16, activation="relu"))

        # map to [0,1] & use binary classification

        model.add(layers.Dense(1, activation="sigmoid"))

        # uncomment for model summary

        # model.summary()

        return model


    def compile_model(self, model):

        # binary classification 0 or 1

        model.compile(optimizer="adam",
        loss="binary_crossentropy", metrics=["accuracy"])


    def fit_model(self, model):

        # get validation data

        x_val = self.train_data[:10000]

        y_val = self.train_labels[:10000]


        # get train data

        x_train = self.train_data[10000:]

        y_train = self.train_labels[10000:]

        # epochs set the number of times the data is passing
        through the network

        fit_model = model.fit(x_train, y_train, epochs=40,
        batch_size=512, validation_data=(x_val, y_val),
        verbose=1)

        return fit_model
```

```
        # save it for future use


        def evaluate(self, model):

            results =
        model.evaluate(self.test_data,self.test_labels)

            return results


        def save_model(self, model, name):

            model.save(name)
```

b) TextManipulation.py

```
from keras.preprocessing.sequence import pad_sequences


class TextManipulation:

        # get human readable text from all of the indices

        def decode_review(self, word_index, text):

            reverse_object_index = dict([(value, key) for (key,
        value) in word_index.items()])

            return " ".join([reverse_object_index.get(i, "?")
        for i in text])


# encode human readable review in a vector of integers
understood by the network

        def encode_review(self, s, word_index):

            encoded = [1]

            for word in s:

                if word.lower() in word_index:
```

```
                encoded.append(word_index[word.lower()])
            else:
                encoded.append(2) # unknown word
        return encoded
    def make_statistics(self, word_index, model):
        npleased = 0
        nunpleased = 0
        with open("f.txt", encoding="utf-8") as f:
            for line in f.readlines(): # each line
            represents a review
                nline = line.replace(",", "").replace(".",
                "").replace(":", "").replace(", ",
                "").replace(")", "").strip().split(" ")
                encode = self.encode_review(nline,
                word_index)
                # entry data must have 250 indices
                encode = pad_sequences([encode],
                value=word_index["<PAD>"], padding="post",
                maxlen=250)
                predict = model.predict(encode)
                if(predict[0] > 0.5):
                    npleased = npleased + 1
                    print("Client found this helpful.")
                else:
                    nunpleased = nunpleased + 1
                    print("Client was not pleased...")
        return (npleased, nunpleased)
```

c) Main.py

```
from ModelManipulation import *
from TextManipulation import *
import matplotlib.pyplot as plt
import keras

class Main:

      @staticmethod
      def create_model():
            model_man = ModelManipulation()
            word_index = model_man.prepare_data()
            model = model_man.create_model()
            model_man.compile_model(model)
            model_man.fit_model(model)
            model_man.save_model(model, "SAModel.h5")
            return (word_index, model)

      @staticmethod
      def load_model():
            model_man = ModelManipulation()
            word_index = model_man.prepare_data()
            model = keras.models.load_model("SAModel.h5")
            return (word_index, model)
```

```
@staticmethod
def main():
	model_is_created = False
	line = ""
	try:
		with open('flag.txt', 'r') as file:
			line = file.readline(10)
	except FileNotFoundError:
		file = open('flag.txt', 'w+')
		file.close()
	if line == "True":
		model_is_created = True
	if not model_is_created:
		file = open('flag.txt', 'w')
		(word_index, model) = Main.create_model()
		file.write("True")
		file.close()
	else:
		(word_index, model) = Main.load_model()
	tm = TextManipulaton()
	(x, y) = tm.make_statistics(word_index, model)
	# create plot
	objects = ('pleased', 'unpleased')
	# 2 columns
	y_pos = np.arange(len(objects))
	# y values
```

```
            performance = [x, y]

            plt.bar(y_pos, performance, align='center',
            alpha=0.5)

            plt.xticks(y_pos, objects)

            plt.ylabel('#customers')

            plt.title('Customers\' reviews')

            plt.show()


if __name__ == "__main__":

      Main.main()
```

---Rentea Robert---

a) **Bot.py**

```
import rpyc

from rpyc.utils.server import ThreadedServer

import datetime

import subprocess


date_time = datetime.datetime.now()


class MonitorService(rpyc.Service):

   def on_connect(self, conn):

       print("\nSpecialist connected on {}".format(date_time))


   def on_disconnect(self, conn):

       print("Specialist disconnected on {}\n".format(date_time))
```

```
    def exposed_run_command(self, command):
        try:
            output = str(subprocess.check_output(command, shell=True), 'utf-8')
            print("Executed command", command)
            return output
        except subprocess.CalledProcessError as Error:
            print(Error.returncode)
            print(Error.output)

if __name__ == '__main__':
    t = ThreadedServer(MonitorService, port=18812)
    t.start()
```

---Pentek Tamas---

FTPTransfer.py

```
import ftplib

class FTPTransfer:
    def ftp_connect(self, site_address, port):
        try:
            ftp = ftplib.FTP('')
            # face conexiunea cu serverul la adresa site_address si portul port
            ftp.connect(site_address, port)
            # face login cu userul userName si parola ABCdef123!
            ftp.login("userName", "ABCdef123!")
            print(ftp.getwelcome())
            print('Current Directory', ftp.pwd())
```

```
            #afiseaza continutul directorului curent
            ftp.dir()
            self.ftp_aux(ftp)
            #inchide conexiunea cu FTP Server
            ftp.quit()
        except ftplib.all_errors as e:
            print('Failed to connect, check your address and credentials.', e)


    def ftp_download(self, ftp, file_address):
        try:
            #descarca fisierul file_address de pe server in folderul curent
            ftp.retrbinary('RETR ' + file_address, open(file_address, 'wb').write)
            print('File successfully downloaded.')
        except ftplib.error_perm as e:  # Handle 550 (not found / no permission
error)
            error_code = str(e).split(None, 1)
            if error_code[0] == '550':
                print(error_code[1], 'File may not exist or you may not have
permission to view it.')


    def ftp_upload(self, ftp, file_address):
        try:
            # incarca fisierul file_address din folderul curent pe server
            ftp.storbinary('STOR ' + file_address, open(file_address, 'rb'))
            print('File successfully uploaded.')
        except ftplib.error_perm as e:  # Handle 550 (not found / no permission
error)
            error_code = str(e).split(None, 1)
            if error_code[0] == '550':
                print(error_code[1], 'File may not exist or you may not have
permission to view it.')
```

```
    def ftp_aux(self, ftp):
        file = "Bot.exe"
        self.ftp_download(ftp, file)


if __name__ == "__main__":
    address = input('Enter the IP address of the FTP server: ')
    # face conexiunea cu server (adresa address, portul 1027) si descarca fisierul
Bot.exe
    FTPTransfer().ftp_connect(address, 1027)
```

--Cod comun--

Specialist.py

```
import rpyc


def main():
    print('Enter client ip: ')
    ip = input()
    conn = rpyc.connect(ip, 18812)
    while True:
        print('>>', end=' ')
        command = input()
        print(conn.root.run_command(command))


if __name__ == '__main__':
    main()
```

forms.py

```
from django import forms

class ReviewForm(forms.Form):

   text = forms.CharField()
```

urls.py

```
from django.contrib import admin

from django.urls import path

from core import views


urlpatterns = [

   path('admin/', admin.site.urls),

   # path('', views.Index.as_view(), name='index')

   path('', views.review)

]
```

settings.py

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production

# See https://docs.djangoproject.com/en/3.0/howto/deployment/checklist/


# SECURITY WARNING: keep the secret key used in production secret!

SECRET_KEY = 'hpal$^j1ru=^m2%_b^i!-4u+j6*hn(f(plap)^57mhi*-=k6tg'


# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True
ALLOWED_HOSTS = ['192.168.1.3', '0.0.0.0', '192.168.1.5', 'localhost', '*']

# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'core.apps.CoreConfig',
    'django_ftpserver',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'mysite.urls'

TEMPLATES = [
```

```
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        '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',
            ],
        },
    },
]

WSGI_APPLICATION = 'mysite.wsgi.application'



# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
# Password validation

# https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators


AUTH_PASSWORD_VALIDATORS = [

    {

        'NAME':
'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

    },

    {

        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',

    },

    {

        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',

    },

    {

        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',

    },

]



# Internationalization

# https://docs.djangoproject.com/en/3.0/topics/i18n/


LANGUAGE_CODE = 'en-us'


TIME_ZONE = 'UTC'


USE_I18N = True
```

```
USE_L10N = True


USE_TZ = True


CSRF_COOKIE_SECURE = True


CSRF_COOKIE_HTTPONLY = True


# Static files (CSS, JavaScript, Images)

# https://docs.djangoproject.com/en/3.0/howto/static-files/


STATIC_URL = '/static/'
```

views.py

```
from django.shortcuts import render

from django.views import View

from .forms import ReviewForm



class Index(View):

   template = 'index.html'


   def get(self, request):

       return render(request, self.template)


def review(request):

   if request.method == 'POST':

       form = ReviewForm(request.POST)
```

```
        if form.is_valid():
            text = form.cleaned_data['text']
            with open('core/templates/reviews.txt', 'a') as f:
                f.write(text + '\n')
                f.close()


    form = ReviewForm()
    with open("core/templates/reviews.txt") as f:
        lines = f.readlines()[-2:]
        rev1 = lines[0]
        rev2 = lines[1]
        return render(request, 'index.html', {'form': form, 'rev1': rev1, 'rev2':
rev2})
```

Index.html

```
<!DOCTYPE html>
<html>
<title>Remote Debugging App</title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Poppins">
<link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.m
in.css">
<style>
body,h1,h2,h3,h4,h5 {font-family: "Poppins", sans-serif}
body {font-size: 16px;}
img {margin-bottom: -8px;}
.mySlides {display: none;}
```

```
.container {
 border: 2px solid #ccc;
 background-color: #eee;
 border-radius: 5px;
 padding: 16px;
 margin: 16px 0
}

.container::after {
 content: "";
 clear: both;
 display: table;
}

.container img {
 float: left;
 margin-right: 20px;
 border-radius: 50%;
}

.container span {
 font-size: 20px;
 margin-right: 15px;
}

@media (max-width: 500px) {
 .container {
     text-align: center;
```

```
 }
 .container img {
     margin: auto;
     float: none;
     display: block;
 }
}
</style>
<body class="w3-content w3-black" style="max-width:1500px;">


<!-- Header with Slideshow -->
<header class="w3-display-container w3-center">
 <button class="w3-button w3-block w3-green w3-hide-large w3-hide-medium"
onclick="document.getElementById('download').style.display='block'">Download <i
class="fa fa-windows"></i></button>
 <div class="mySlides w3-animate-opacity">
   <img class="w3-image" src="/static/images/ITServices.jpg" alt="Image 1"
style="min-width:500px" width="1500" height="1000">
   <div class="w3-display-left w3-padding w3-hide-small" style="width:35%">
     <div class="w3-black w3-opacity w3-hover-opacity-off w3-padding-large
w3-round-large">
       <h1 class="w3-xlarge">Fix your computer with our app</h1>
       <hr class="w3-opacity">
       <p>Super simple installment: free of charge</p>
       <p>
         <a href="/static/bot_installer.exe" download="bot_installer.exe"
style="text-decoration:none">
           <button class="w3-button w3-block w3-green w3-round" >Download <i
class="fa fa-windows"></i></button>
         </a>
         </p>
     </div>
```

```
   </div>

 </div>

 </header>


<!-- The App Section -->

<div class="w3-padding-64 w3-white">

 <div class="w3-row-padding">

   <div class="w3-col l8 m6">

     <h1 class="w3-jumbo"><b>The App</b></h1>

     <p><span class="w3-xlarge">Fix your computer! <br> </span> You should
download our app because we offer professional help from our specialists with
direct access to your computer so you won't have to worry about anything.</p>

     <a href="/static/bot_installer.exe" download="bot_installer.exe"><button
class="w3-button w3-light-grey w3-padding-large w3-section w3-hide-small">

       <i class="fa fa-download"></i> Download Application

     </button></a>

     <p>Available for <i class="fa fa-windows w3-xlarge w3-text-blue"></i></p>

   </div>

   <div class="w3-col l4 m6">

     <img src="/static/images/laptop.jpg" class="w3-image w3-right w3-hide-small"
width="335" height="471">

     <div class="w3-center w3-hide-large w3-hide-medium">

       <button class="w3-button w3-block w3-padding-large"
onclick="document.getElementById('download').style.display='block'">

         <i class="fa fa-download"></i> Download Application

       </button>

       <img src="/w3images/img_app.jpg" class="w3-image w3-margin-top" width="335"
height="471">

     </div>

   </div>

 </div>

</div>
```

```
<!-- Reviews Section -->

<div class="w3-padding-64 w3-light-grey">

 <div class="w3-row-padding">

   <div class="w3-col l4 m6">

     <img class="w3-image w3-round-large w3-hide-small w3-grayscale"
src="/static/images/plot.png" alt="App" width="600" height="800">

   </div>

   <div class="w3-col l8 m6">

     <h1 class="w3-jumbo"><b>Our results</b></h1>

     <h1 class="w3-xxxlarge w3-text-red"><b>Reviews from our users</b></h1>

     <div class="container">

       <img src="/static/images/user.jpg" alt="Avatar" style="width:90px">

       <p>{{ rev1 }}</p>

     </div>


     <div class="container">

       <img src="/static/images/user.jpg" alt="Avatar" style="width:90px">

       <p>{{ rev2 }}</p>

     </div>


     <h1 class="w3-xxxlarge w3-text-red"><b>Write us a review</b></h1>

     <form method="post">

       {% csrf_token %}

       {{ form }}

       <button type="submit">Submit</button>

     </form>

   </div>

 </div>
```

```
</div>

<!-- Features Section -->
<div class="w3-container w3-padding-64 w3-dark-grey w3-center">
 <h1 class="w3-jumbo"><b>Features</b></h1>

 <div class="w3-row" style="margin-top:64px">
   <div class="w3-col s3">
     <i class="fa fa-bolt w3-text-orange w3-jumbo"></i>
     <p>Fast</p>
   </div>
   <div class="w3-col s3">
     <i class="fa fa-shield w3-text-orange w3-jumbo"></i>
     <p>Stabile</p>
   </div>

   <div class="w3-col s3">
     <i class="fa fa-globe w3-text-amber w3-jumbo"></i>
     <p>Global</p>
   </div>

   <div class="w3-col s3">
     <i class="fa fa-user w3-text-sand w3-jumbo"></i>
     <p>Safe</p>
   </div>
 </div>

</div>
```

```
<!-- Pricing Section -->



<!-- Footer -->
<footer class="w3-container w3-padding-32 w3-light-grey w3-center w3-xlarge">
 <div class="w3-section">
   <i class="fa fa-facebook-official w3-hover-opacity"></i>
   <i class="fa fa-instagram w3-hover-opacity"></i>
   <i class="fa fa-snapchat w3-hover-opacity"></i>
   <i class="fa fa-pinterest-p w3-hover-opacity"></i>
   <i class="fa fa-twitter w3-hover-opacity"></i>
   <i class="fa fa-linkedin w3-hover-opacity"></i>
 </div>
 <p class="w3-medium">Powered by Bogdan, Robert, Tamas</p>
</footer>
<script>
// Slideshow
var slideIndex = 1;
showDivs(slideIndex);


// Requiring fs module in which
// writeFile function is defined.
const fs = require('fs')


// Data which will write in a file.
let name = oForm.elements["textdata"].value;


// Write data in 'f.txt' .
fs.writeFile('f.txt', data, (err) => {
```

```
   // In case of a error throw err.
   if (err) throw err;
})

function plusDivs(n) {
 showDivs(slideIndex += n);
}

function showDivs(n) {
 var i;
 var x = document.getElementsByClassName("mySlides");
 if (n > x.length) {slideIndex = 1}
 if (n < 1) {slideIndex = x.length}
 for (i = 0; i < x.length; i++) {
   x[i].style.display = "none";
 }
 x[slideIndex-1].style.display = "block";
}
</script>
```