Milestone Report 2 (12 May 2019)

The description for the task can be viewed from here.

Our meeting reports and all the content here is also available on our Wiki Page.

1. Executive Summary

> Created by Irem Uguz
>
> Reviewed and edited by Ahmet Gedemenli
>
> Available here

We are designing a product, which will be used to teach, learn and make exercises on languages. Mainly, it is a language learning platform. This project is assigned to us by the instructor of the course CmpE 352 in Spring 2019. Our project mainly consists of two parts: designing the project on an abstract level and implementation.

Our team has ten members: Irem, Ibrahim, Arda, Burhan, Egemen, Emirhan, Gamze, Halit, Ahmet and Ali. We are all bunch of friends taking that course simultaneously.

The work done before our first Milestone can be found in the Milestone 1 report . After our Milestone 1, we produced two more deliverables. Project plan and API implementation for our project plan.

The project plan is the diagram that we plan what we have done and what we will be doing in the next steps of our project. The project plan is a dynamic deliverable as the project evolves, the project plan gets updated too.

Our API implementation has currently 9 mini API functionalities in it: Object detection, movie recommender, language detection from text, language detection from the location, word of the day, translation, text to speech, tag detection and inappropriate word detection.

For the future, we are planning to add a front-end to all the API functionalities. Also, as a group, we decided to learn about Web and Android basics to be able to implement the whole project. After learning the basics, we will be forming sub-teams for Web, Android and backend. Then we will be implementing the project. Our far future plans are not exactly determined yet but those are our main expectation from the future.

Of course, during our project, we met many challenges as a group.
The first challenge was the communication. While doing group work, communication between members is the most important thing. Especially between the first milestone and now, we had many difficulties with communication. We were not able to meet as a whole group for a long time during our meetings and while doing our personal share of the teamwork, when some group members didn't do their part without informing the other members, we were not able to complete the whole assignment.
The second challenge was about the division of the teamwork. When giving the members some weekly work, we weren't able to divide the whole job properly. This caused some jobs to be left without an assignee and left the job to the initiative of other group members.
* The third challenge was a more technical challenge. During the implementation of our API, many group members had hard time as we haven't implemented any API before and we didn't know about it. That's why some of the group members were not able to complete their parts. We tried to learn about a whole new concept so it was quite hard and challenging.

After facing those difficulties and trying to solve them as good as we can, we are more experienced as a group. We hope to face fewer problems in the next stages of our project.

2. Deliverables

> Created by Arda Baris Budak
>
> Reviewed and edited by Halit Ozsoy & Ahmet Gedemenli
>
> Available here

```
List and status of deliverables
```

Deliverable	Status
API	Nearly done. Documentation is fine; front-ends aren't complete.
API-URL	Done, check out: ass7.bounswe2019group9.tk
Project Plan	Nearly done. Some enhancements can be made.

More info on Evaluation of deliverables by Ibrahim & Evaluation of deliverables by Emirhan

3.1. Evalutation of Deliverables I

> Created by Ibrahim Can Kaplan
>
> Project Plan part reviewed and edited by Gamze Gulbahar
>
> API part reviewed and edited by Ahmet Gedemenli
>
> Available here

Project Plan

- The project plan has been helpful for observing the overall view and progress of our project.
- The start and finish dates of the tasks were not stated properly. We tried and fixed this problem as much as possible. Due to the date problem of Project Libre, those dates on it may not be the actual dates.
- In overall, most of the tasks stated in correct order but a few tasks are missing. For example, after getting feedback from customers, we nearly recreated the sequence diagrams.
- Milestones are included in the plan.

- Adding additional predecessors while defining new tasks will be better. For example, in Implementation/Android/Testing part should have a predecessor like Implementation/Android/Functional Implementation .

API Implementation

- The overall structure of the API seems fine. It works end-to-end, from the opening the issue to merging the pull request. But also it has some deficiencies.
- The functionalities of the API has no front-end part (except translation part), they only return json file. There should be a simple front-end part which can show the functionality of that part.
- There should be 10 functionalities but it seems only 9 functionality exist. The last one should be added.
- The number of test cases can be increased.

3.2. Evalutation of Deliverables II

> Created by Gamze Gulbahar
>
> Available on here

Project Plan

- During the process, we're substantially assisted by the project plan. To better see the general going of our work.
- The work flow seems good.
- After feedback, missing tasks are added.

API Implementation

- Functionalities of the API are very good. We can use these functionalities in the future during the implementation phase of the project.

- The good think about the APIs is that they enable us to make use of such functionalities that normally would be very difficult for us to create so perfectly in quite a short time.

4. Evaluation of Tools

> Created by Burhan Can Akkus
>
> Reviewed and edited by Ahmet Gedemenli
>
> Available here

Communication

Slack

- We mainly used Slack for in-group communication. At first we had a lackluster experience but once we got used to it, it worked fine.
- Slack's functionalities made us easy to communicate and share amongst ourselves.

Whatsapp

- We mostly used whatsapp for emergencies. When someone had an issue that needed immediate attention,(s)he used Whatsapp group.
- We didn't want to use Whatsapp as our main communication app since most people used Whatsapp for personal affairs.

Hangouts

- We never used Hangouts during this semester as it was not really needed.

Process Management

Github

- Github was our main tool for everything related to project management. We uploaded project documents and meeting notes to wiki page. Everything we did can be found in our repo's wiki page.
- We have some problems related to Issue followups. We still have 15+ issues open, some of them are almost a month old. In the future we

need a better system to make sure issues are done within their assigned timeframes.

Evaluation of Processes

Weekly Meetings

- We held a group meeting every week in BM Lounge on Tuesdays. After the first weeks we rarely had all 10 members of our group present in the meetings. In the future we may need a better meeting schedule.
- During the meetings we wasted a lot of time understanding what the assignment of week was about. We need to prepare before the meetings in the future.
- Discussions of our meetings were not really efficient. Instead of just distributing the workload we tried to tackle the issues during the meeting which slowed us down significantly. Most of our meetings lasted much longer than the intended 1 hour. But in time, we had more and more efficient meetings, in terms of time.

5. Work Done by Team Members

> Available here

Member	Work	Deliverable
Ahmet	Taken the notes of Meeting 11.	Meeting notes
Ahmet	Contributed to the evaluation of project repo part in the 1st Milestone Report.	Milestone Report
Ahmet	Added week-1 activities to the project plan sheet.	Project Plan
Ahmet	Made some research about APIs, sqlite, AWS, REST, Django and their different kinds of implementations.	API Implementation
Ahmet	Implemented a Django project 'The Word Of The Day' in my own repository.	API Implementation

Member	Work	Deliverable
Ahmet	Pushed 'The Word Of The Day' app to our repository and wrote tests for it.	API Implementation
Ahmet	Fixed and edited my app by suggestions and reviews of my teammates	API Implementation
Ahmet	Added the documentation of 'The Word Of The Day' API.	API Implementation
Ahmet	Wrote reviews to other team members' pull requests.	API Implementation
Ahmet	Edited the contributions of other members and compiled the whole milestone report with API documentation.	Milestone Report
İrem	Added suggestions for future activities	Project Plan
İrem	Added week two activities to the project plan sheet	Project Plan
İrem	Added the project plan to the ProjectLibre tool with Arda and İbrahim	Project Plan
İrem	Implemented the api about object detection	API Implementation
İrem	Write a test for the object detection API	API Implementation
İrem	Added the documentation of the object detection API	API Implementation
İrem	Taken the notes of Meeting 9	Meeting notes
İrem	Wrote the executive summary for the Milestone report 2	Milestone Report
İrem	Attended the lectures	-

Member	Work	Deliverable
İbrahim	Prepared the Week 4’s accomplishments to be a part of Project Plan	Project Plan
İbrahim	Contributed to document which we give opinions for the future part	Project Plan
İbrahim	Wrote review to other members' notes about past weeks	Project Plan
İbrahim	Added the project plan to the ProjectLibre tool with Arda and İrem	Project Plan
İbrahim	Made some research about APIs and its different kinds of implementations	API Implementation
İbrahim	Completed the part of the api which is ‘languagedetection’.	API Implementation
İbrahim	Fixed some bugs related to ‘languagedetection’ api. Added documentation.	API Implementation
İbrahim	Wrote reviews to other team members' pull requests.	API Implementation
Ali	Taken the notes of meeting 8	Meeting Notes
Ali	Gave the idea of documenting the plan's entries before actually making it in the meeting.	Project Plan
Ali	Helped fixing and rearranging the project plan according to the given feedback.	Project Plan
Ali	Learned about RESTful API's and how to make use of them	API Implementation
Ali	Implemented a text to speech converting project using an api	API Implementation
Ali	Wrote a documentation for texttospeech	API Implementation

Member	Work	Deliverable
Ali	Prepared the personal milestone report for the milestone 2	API Implementation
Ali	Wrote reviews to other team members' pull requests.	API Implementation
Arda	Wrote two items in the evaluation of project repo part in the 1st Milestone Report	Milestone Report
Arda	Added what we did in week 4 to the project plan document in ProjectLibre	Project Plan
Arda	Added the future part of the project plan based on the ideas written in project plan discussion document and assigned dates to them.	Project Plan
Arda	Implemented movie recommendation functionality for the API using OMDB API	API Implementation
Arda	Implemented tests for the movie recommendation functionality	API Implementation
Arda	Wrote documentation for the movie recommendation functionality	API Implementation
Arda	Prepared List and Status of deliverables part of the report	Milestone Report II
Halit Özsoy	Fixed durations & dates of the existing tasks with Burhan	Project Plan
Halit Özsoy	Made lots of research about primarily on AWS, Django, CICD (Travis-CodeDeploy)	API

Member	Work	Deliverable
Halit Özsoy	Created initial skeleton Django project Implemented a method for adding secret keys Separated settings into production and local (used inheritance) Explained how to add new apps and secrets to the skeleton on a Readme file Created a sample app called Polls to the skeleton	API Implementation
Halit Özsoy	Created an EC2 instance prepared it for deployment Created a TravisCI Configuration, connected it to CodeDeploy Created start/stop/restart bash scripts Took bounswe2019group9.tk free domain and configured its DNS via Cloudflare Configured nginx as reverse proxy, serving static files separately in production mode Configured uwsgi to deploy Django App in production mode Installed MySQL(MariaDB) to be used in production mode	API-URL / CICD / DevOps

Member	Work	Deliverable
Halit Özsoy	Researched translation APIs, (Google/Yandex/Azure) Implemented Django App, `translate` that supports both JSON input-output and standard FORM data Created a simple frontend for `translate app` Created a simple documentation for translation feature Implemented useful tests to `translate` app for both frontend and model validation	API Implementation
Halit Özsoy	Created a `home` app that lists demo links for all the apps inside Reviewed others' pull requests Bug fixes to others' apps whenever necessary	API Implementation
Burhan	Added Week 8 to project plan	Project Plan
Burhan	Implemented the api about language prediction	API Implementation
Burhan	Write a test for the language prediction API	API Implementation
Burhan	Added the documentation of the language predicion API	API Implementation
Burhan	Wrote evaluation of processes and tools used on project until this point	Milestone II
Emirhan	Added notes to project plan	Project Plan

Member	Work	Deliverable
Emirhan	Wrote executive summary in the 1st Milestone Report.	Milestone Report
Emirhan	Wrote the second part of the evaluation of deliverables in Milestone Report II	Milestone Report
Emirhan	Made some research about APIs, sqlite, AWS, REST, Django and their different kinds of implementations.	API Implementation
Emirhan	Implemented a Django project 'Music Suggester' in my own repository.	API Implementation
Emirhan	Pushed 'Music Suggester' app to our repository and wrote tests for it.	API Implementation
Emirhan	Fixed and edited my app by suggestions and reviews of my teammates	API Implementation
Emirhan	Added the documentation of 'Music Suggester' API.	API Implementation
Emirhan	Made research and implementations about Amazon AWS	API Implementation
Emirhan	Summarized my personal efforts	Milestone Report
Gamze	Taken the notes of Meeting 10.	Meeting notes
Gamze	Wrote the evaluation of class diagram in the 1st Milestone Report.	Milestone Report
Gamze	Updated project plan according to feedback.	Project Plan
Gamze	Added works of the milestone reports and project plan to project plan.	Project Plan
Gamze	Contributed to Ibrahim's evaluation of the project plan for second Milestone report.	Milestone Report

Member	Work	Deliverable
Gamze	Wrote the evaluation of the project plan and API implementation for second Milestone report.	Milestone Report
Gamze	Research on the RESTful API's.	API Implementation

6. API with Documentation and URL

> Created with the contributions of all team.
>
> Special thanks to Halit
>
> Edited and compiled by Ahmet Gedemenli
>
> Source code available on here
>
> Will be live at ass7.bounswe2019group9.tk

You can find the explanations and endpoints for our API below.

Project Folder Structure

```
practice-app
├── LICENSE                 # License information
├── README.md               # This readme file
├── appspec.yml             # Details for Amazon Server EC2 Depl
├── uwsgi-conf.ini          # Details for uWSGI configuration fo
├── scripts                 # A folder consisting of scripts to
├── docs                    # Documentation of the project (docs
├── manage.py               # Django CLI file
├── requirements.txt        # Project Dependencies (pip install
├── prod_requirements.txt   # Extra Project Dependencies for Pro
├── practice_app            # Main Project Settings & Details
│   ├── __init__.py             # empty init file to make practi
│   ├── base_settings.py        # Most Django Environemnt Settin
│   ├── settings.py             # Development Environemnt Settin
│   ├── prod_settings.py        # Production Environemnt Setting
│   ├── urls.py                 # url mappings to custom subapps
│   └── wsgi.py                 # not sure if this is used (auto
└── subapp                  # Sample Django App
    ├── __init__.py             # empty init file to make subapp
    ├── admin.py                # Admin permissions, definitions
    ├── apps.py                # subapp App settings / configur
    ├── docs                   # Documentation of the subapp (d
    ├── migrations             # db migrations & seeds
    │   ├── 0001_initial.py        # generated from model (pyth
    │   ├── __init__.py            # empty init file to make mi
    ├── models.py              # model definitions for db
    ├── static                 # static files (css/js/img) for
    │   └── subapp                 # namespacing subapp to prev
    │       ├── images                 # images folder to hold
    │       │   ├── sample.gif         # sample image
    │       │   └── background.png     # another sample image
    │       └── style.css          # sample stylesheet for suba
    ├── templates              # template files (django html) f
    │   └── subapp                 # namespacing subapp to prev
    │       ├── etc.html               # sample django template
    │       ├── index.html             # another sample django
    │       └── somethingelse.html     # yet another sample dja
    ├── tests.py               # tests of subapp
    ├── urls.py                # url mappings to views of subap
    └── views.py               # define WHAT HAPPENS ON EACH RE
```

Database

In production environment MySQL (MariaDB) is used, in development environment SQLite is used.

Running on your local machine

You should prepare a `secret.json` file that contains below fields:

```
{
  "SECRET_KEY1": "series of random characters",
  "SECRET_KEY2": "series of random characters",
  "SECRET_KEY3": "series of random characters",
  "DB_NAME": "mysql db name",
  "DB_USER": "mysql db user",
  "DB_PASSWORD": "mysql user password",
  "STATIC_ROOT": "root folder for collectstatic command"
}
```

Usage

6.1. Movie Recommendation

For your questions, please contact the developer, Arda Baris Budak

Returns name and year of a random movie.

Url extension: `/rec/<language_code>`

Example

`rec/en`

```
{
"movie_name": "Honey, I Shrunk the Kids",
"movie_year": "1989"
}
```

language_code should be one of `en, de, tr` for English, German, and Turkish respectively.

6.2. The Word of the Day

For your questions, please contact the developer, Ahmet Gedemenli

Returns the word of the day with its meaning, in any of the supported languages.

The respond you get will, of course, differ every day.

Url extension: `/wordoftheday/<language_code>`

Examples

`wordoftheday/en`

```
{
    "word": "Efficient",
    "definition": "Achieving maximum productivity with minimum w
}
```

`wordoftheday/tr`

```
{
    "word": "Niyet",
    "definition": "Bir şeyi yapmayı önceden isteyip düşünme, mak
}
```

`wordoftheday/aq`

```
NO SUCH LANGUAGE AVAILABLE
```

6.3. Object Recognition

For your questions, please contact the developer, Irem Uguz

Takes a picture and recognizes the objects in that picture with the correction percentage and returns an Json object with that data. This can be used in our project to help the users learn the vocabulary of the object in that picture.

Url extension: `/obj/<picture_url>`

Return Content:

```
{
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "default",
          "name": "default",
          "classes": [
            {
              "class": "<Object Name>",
              "score": <Score of the object>,
              "type_hierarchy": "<Type of the object>"
            }
          ]
        }
      ],
      "source_url": "<Url of the image>",
      "resolved_url": "<Url of the image>"
    }
  ],
  "images_processed": 1,
  "custom_classes": 0
}
```

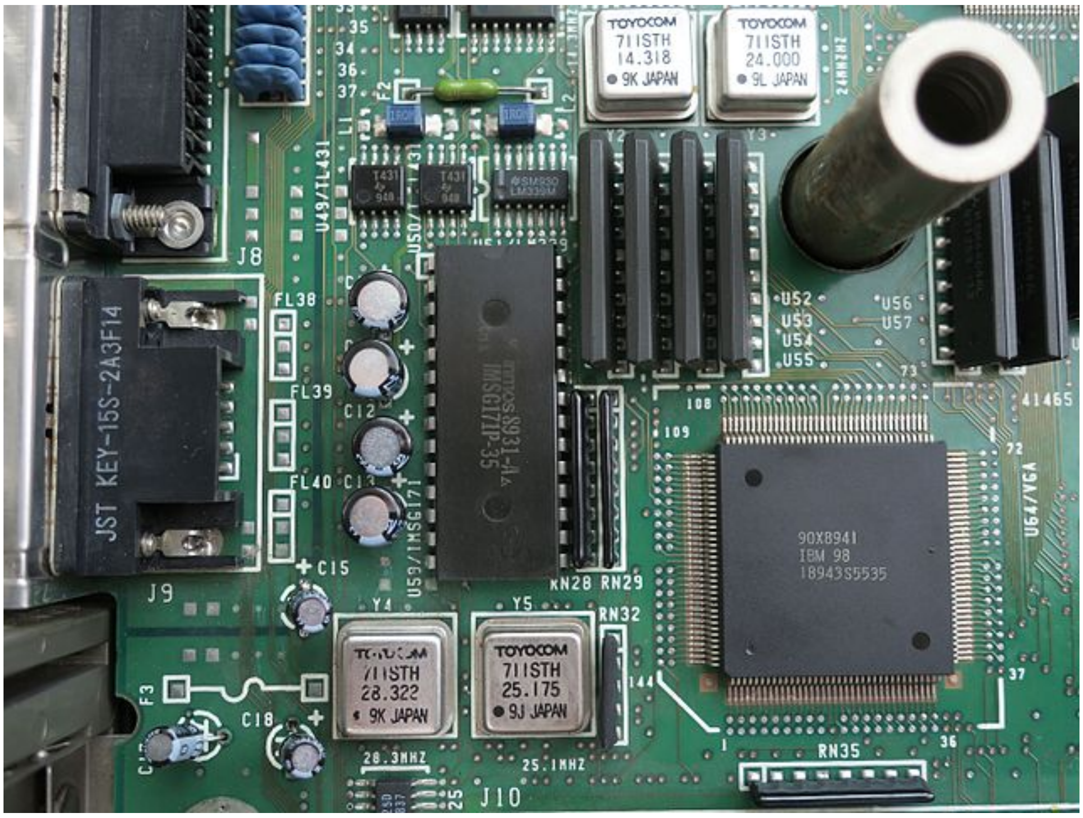
Class means basically the name of the object in the picture, score is how confident the object detection is about the classification. Also the api doesn't return any classification that has a score that is less than 0.5. Type hierarych is the bigger classes that the class of the object belongs to. For example a dog is an animal too. Custom classes papameter will always be 0 for our api as we use the classes that are built-in in the watson-object-detection api which is the api that I am using for this case. Images processed will be one as we are giving single image to the api. Source url and resolved will be the same as well.

Example

`/obj/https://watson-developer-cloud.github.io/doc-tutorial-downloads/visual-recognition/640px-IBM_VGA_90X8941_on_PS55.jpg`

```
{
  "images": [
    {
      "classifiers": [
        {
          "classifier_id": "default",
          "name": "default",
          "classes": [
            {
              "class": "circuit board",
              "score": 0.578,
              "type_hierarchy": "/electrical device/computer cir
            },
            {
              "class": "computer circuit",
              "score": 0.755
            },
            {
              "class": "electrical device",
              "score": 0.757
            },
            {
              "class": "disk controller",
              "score": 0.553,
              "type_hierarchy": "/controller/disk controller"
            },
            {
              "class": "controller",
              "score": 0.558
            },
            {
                "class": "central processing unit",
                "score": 0.535
              },
              {
                "class": "PC board",
                "score": 0.501,
                "type_hierarchy": "/electrical device/computer cir
              },
              {
                "class": "CPU board",
                "score": 0.5,
                "type_hierarchy": "/electrical device/computer cir
              },
              {
                "class": "electronic equipment",
                "score": 0.6
              },
              {
                "class": "memory device",
                "score": 0.599
              },
              {
                "class": "microchip",
                "score": 0.592
              },
              {
                "class": "jade green color",
                "score": 0.838
              },
              {
                "class": "emerald color",
                "score": 0.787
              }
            ]
          }
        ],
        "source_url": "https://watson-developer-cloud.github.io/do
        "resolved_url": "https://watson-developer-cloud.github.io/
      }
    ],
    "images_processed": 1,
  "custom_classes": 0
}
```

Picture in the example url:



6.4. Language Detection by Text

For your questions, please contact the developer, Ibrahim Can Kaplan

Returns the language code, language name, probability of the response being correct and the confidence(true or false).

Url extension: `/detect/<text>`

Example

`detect/I%20like%20eating%20apples%20while%20I%20am%20reading%20a%20book.`

```
{
    "language_code": "en",
    "language_name": "English",
    "percentage": 100,
    "reliable_result": true
}
```

6.5. Language Detection from IP

For your questions, please contact the developer, Burhan Can Akkus

Returns the user's language based on their IP address.

Url extension: `/burhan/check` or `/burhan/<IP>`

Example

`burhan/check`

```
{
    "ip": "193.140.194.16",
    "Language": "Turkish"
}
```

`burhan/193.140.194.16`

```
{
    "ip": "193.140.194.16",
    "Language": "Turkish"
}
```

`burhan/333.1.3.2`

```
{
    "ip": "null",
    "Language": "null"
}
```

6.6. Text to Speech Converter

For your questions, please contact the developer, Ali Ramazan Mert

Returns an audio file and downloads it when the text to be converted is given.

Url extension: `/texttospeech/<language_code>/<text>`

Examples

`texttospeech/en/Hello`

```
speech.wav(voice of a person saying 'Hello') appears on the brow
```

`texttospeech/fr/Bonjour`

```
speech.wav(voice of a person saying 'Bonjour') appears on the br
```

`texttospeech/es/Hola`

```
speech.wav(voice of a person saying 'Hola') appears on the brows
```

`texttospeech/invalidlanguage/text`

```
UNDEFINED LANGUAGE
```

6.7. Translation

For your questions, please contact the developer, Halit Ozsoy

Provides translaing a text from one language to another using Azure Translator Text Services.
It has a simple interface you can use.

Endpoints

`GET /translate/`

```
Returns an HTML page with a form to make translations.
```

`GET /translate/<from_language_code>/`

```
Returns the same HTML page with the given language preselected a
```

`GET /translate/-/<to_language_code>`

```
Returns the same HTML page with the given language preselected a
```

`GET /translate/<from_language_code>/<to_language_code>`

```
Returns the same HTML page with the given languages preselected
```

`POST /translate/`

If `Content-Type` Header is set to `json/application` such request expected

```
{
  "from": "from-language-code",
  "to": "to-language-code",
  "source": "text-to-be-translated"
}
```

When successful returns a response as such: (StatusCode: 200)

```
{
  "errors": [],
  "from": "from-language-code",
  "to": "to-language-code",
  "source": "original-text",
  "translation": "translated-text"
}
```

When a parameter is missing or wrong it returns a response as such: (StatusCode: 400)

```
{
  "errors": [
    "Whatever error happened.",
    "More than one error can have been happened"
  ],
  "from": "from-language-code-or-null-if-invalid-from-input",
  "to": "to-language-code-or-null-if-invalid-to-input",
  "source": "original-text-or-null-if-invalid-source-text",
  "translation": null
}
```

When the request is correct but an unknown error occured it returns a response as such: (StatusCode: 500)

```
{
  "errors": [
    "Failed to translate the text for some unknown reason. Maybe
  ],
  "from": "from-language-code",
  "to": "to-language-code",
  "source": "original-text",
  "translation": null
}
```

- *in such case you should contact the developer.*

If the `Content-Type` header is not set or something else, such request is expected (Form Encoded)

```
POST http://ass7.bounswe2019group9.tk/translate/
from=from-language-code
to=to-language-code
source=original-text
```

Whether the translation is successful or not, it will return the same html form as `GET /translate/` ,however,
displaying any errors occured as well as the translated text if it did succeed.
Also, it preselects the given
from & to languages in the form as well as filling the source text with the given one.

6.8. Inappropriate Word Detection

For your questions, please contact the developer, Egemen Gol

Swear detection in english language.

Url extension: `/noswear/plain/<text>` or `/noswear/json/<text>`

Example

`noswear/json/This%20is%20my%20fucking%20sentence.`

```
{
    "wasSwearing": true,
    "censored": "This is my ******* sentence."
}
```

`noswear/plain/This%20is%20my%20sentence.`

```
This is my ******* sentence.
```

`noswear/json/Pretty%20standard`

```
{
    "wasSwearing": false,
    "censored": "Pretty standard"
}
```

`noswear/plain/Pretty%20standard`

```
Pretty standard
```

6.9. Music Suggester

For your questions, please contact the developer, Emirhan Yasin Cetin

A user can GET, PUT and DELETE songs and the endpoint returns a `JSON` object.

Usage

Url extension: `/music/v1/songs`

`GET`

```
{
"title": "Dark Side of The Moon",
"artist": "Pink Floyd"
}
```

`PUT`

If successful:

```
{
"title": "Symphony No. 40",
"artist": "Amadeus Mozart"
}
```

If not successful:

```
{
"message": "Song with id: {} does not exist",
}
```

`DELETE`

If successful:

```
{
"title": "They Don't Care About Us",
"artist": "Michael Jackson"
}
```

If not successful:

```
{
"message": "Song with id: {} does not exist",
}
```

7. Project Plan

> Created with the contributions of all team.
>
> Edited and compiled by Gamze Gulbahar & Irem Uguz

You can find our Project Plan here.