

Developers Manual:

What problem our project solves?

In a company that operates globally has different working hours for the same team in different countries. When working hour in one country ends they have to handover all the information to their counterpart working in another country. This entire process is manual and many man hours are wasted to accomplish this. This project reduces man hours wasted significantly.

How it does?

1. **Speech to Text:** Our project uses inbuilt speech recognition feature of web browsers like chrome. Person involved speaks into the web app after clicking on **'Start'** button whenever anything happens like **'billing cycle 5 started'**.
2. **Classification:** Once the text has been recognized it is passed to a python backend made using flask that deserialize a machine learning model and predicts the class of received text into any one of **'Exceptions', 'On progress', 'Completed Task', 'Follow up'**.
3. **Spreadsheet Updation & displaying to front end:** After classification backend puts the text received into its category along with time stamp in the spreadsheet. Results is also displayed on the website by rendering template using flask.
4. **Add Another:** Result is displayed an an option **'Add Another'** is given to the user to add more info if occurs during their working hours.
5. **Show Sheet:** Users are given an option to show the current updates. A sheet is displayed using the data from spreadsheets.

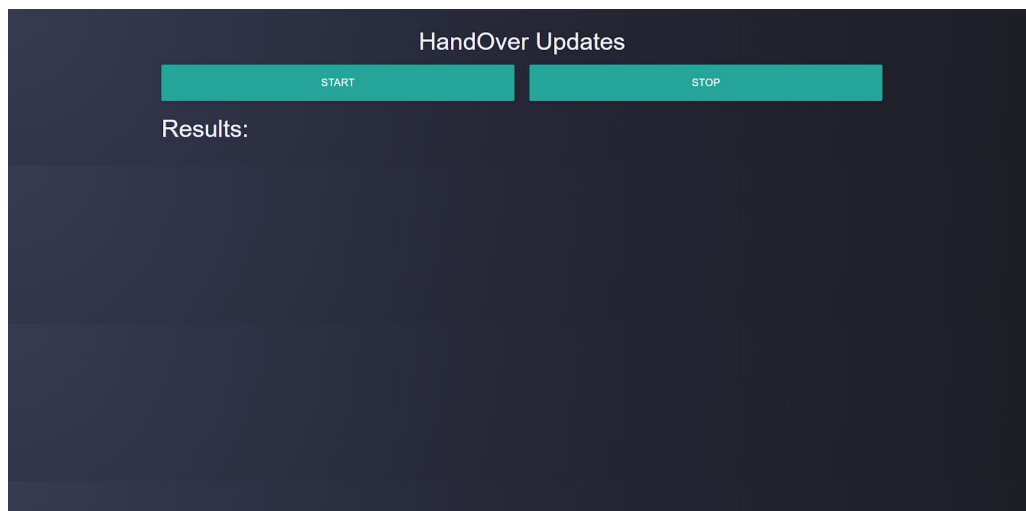
6. **Send Email:** Users can send entire updates to their counterpart using '**Send Email**' button. Entire data is collected from spreadsheet and sent through flask to the recipient entered by the user.

Files:

1. First Page(Home):

Basic page built on html,javascript and some css for styling it. '**Firstpage.js**' contains the javascript for this page. Once text has been received it is sent to flask backend to classify it. "<http://127.0.0.1:8000/>" is where our flask server is running. Received text is concatenated with above link and page is redirected to this concatenated address.

This is how the first page looks:



Example: if user has spoken '**billing cycle 5 started**' then following address will be called : "<http://127.0.0.1:8000/billing cycle 5 started>".

```

recognition.onresult = function(event) {
  var text = event.results[0][0].transcript;
  result_container.innerText = text;
  var str = "http://127.0.0.1:8000/" ;
  str = str + text;
  console.log(str);
  window.location.href = str;
  start_view.style.backgroundColor = "";
}

```

2. Classify.py (Present in **js** folder contains the main flask server)

Contains end points for various things like to classify text received, for taking recipients email, for sending email . Like for above text said by user it will match with following endpoint in **classify.py**.

```
@app.route('/<string:text>')
```

Then it enters this function and does following things:

Loads Machine Learning model (saved using pickle python library) :

```

filename = 'model.pk'
with open(filename, 'rb') as file:
    cl = pickle.load(file)
    print('Model loaded')
    str = cl.classify(text)

```

Str contains the classified result like if text is '**billing cycle 5 started**' it contains '**On progress**'.

After classification current time is calculated using time library present python.

```
#Adding time
timeprefix = '[' + strftime("%H:%M:%S",localtime()) + ']' : '
text = timeprefix + text
```

Example: if text is '**billing cycle 5 started**' and the time is '11:46:23' then text will now contain [11:46:23] : billing cycle 5 started.

Adding to Google Spreadsheet:

First authorization is done so that data can be inserted into google spreadsheet.
Following code does this.

```
SCOPES = 'https://www.googleapis.com/auth/drive'
store = file.Storage('credentials.json')
creds = store.get()
if not creds or creds.invalid:
    flow = client.flow_from_clientsecrets('client_secret.json', SCOPES)
    creds = tools.run_flow(flow, store)
service = build('sheets', 'v4', http=creds.authorize(Http()))

spreadsheet_id = '1TQM0ys75pq2GmQFIPM1_hiucmR2dhoPhYz6CuAN_lE'
```

If you want to use a different spreadsheet enter the id of that spreadsheet and you're done!

Then text is updated in its column by the following code:

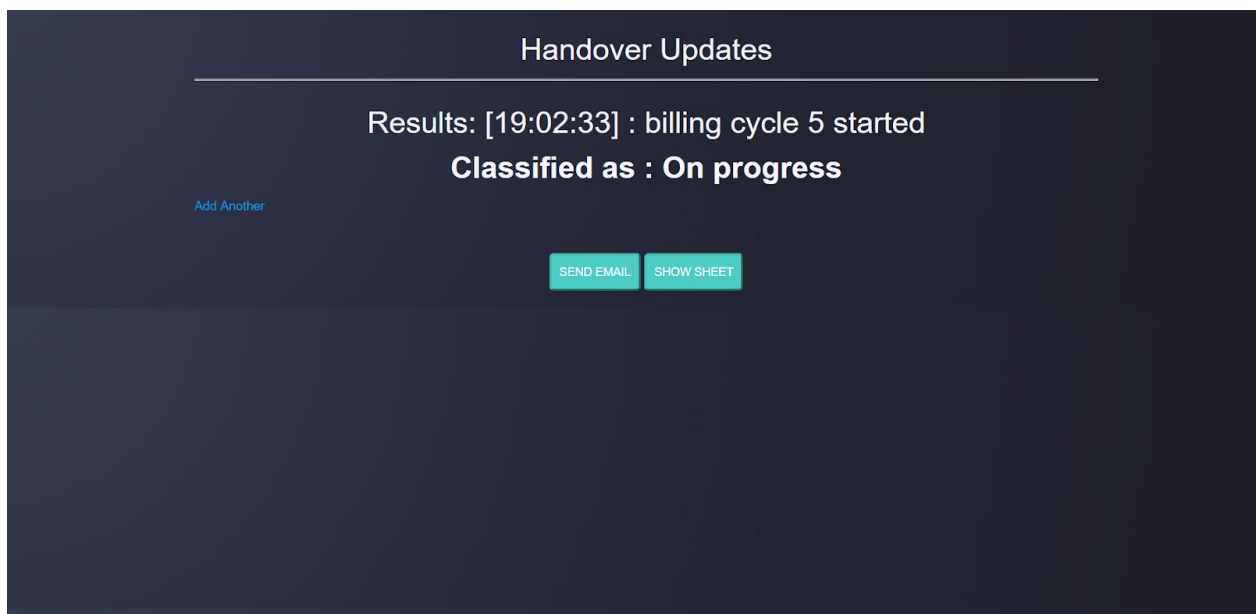
```
request = service.spreadsheets().values().append(spreadsheetId=sheet_id, range=range_, valueInputOption=value_input_option, insertDataOption=insert_data_option,
response = request.execute()
pprint(response)

return render_template('stt.html',values=str,text=text)
```

Once spreadsheet is updated then flask server renders a html page with the results displayed. **Str** contains the classification result (like '**On progress**') and **text** contents are explained above.

```
return render_template('stt.html',values=str,text=text)
```

Results look like this:



3. Sending email:

Classify.py contains end point for sending email. When user clicks on Send Email button then a page is rendered by flask to take recipients email. Following endpoint does this:

```
@app.route('/takeemail',methods=['GET','POST'])
def takeemail():
    form = EmailForm(request.form)

    if request.method == 'POST':
        email=request.form['email']
        print (email)

        if form.validate():
            flash('Recieved :' + email)
            return redirect(url_for('sendemail',email=email))
        else:
            flash('Email Not correct |!')
    return render_template('takeemail.html',form=form)
```

Once correct email is received a new page is rendered by flask by following endpoint which does the task of sending email:

```
@app.route('/sendemail/<email>')
```

If email entered is xyz@gmail.com then localhost:5000/sendemail/xyz@gmail.com will match with this endpoint in flask.

Once the mail has been received it a Daily report is send to it using flask_mail. Before using flask_mail some configuration namely '**MAIL_SERVER**', '**MAIL_PORT**', '**MAIL_USERNAME**', '**MAIL_PASSWORD**' need to be done.

4. Showing Current Updates:

For showing Current Updates there is another endpoint as shown below in **classify.py**.

```
@app.route('/showsheets')
```

After this endpoint is reached(when user clicks **show sheet**) '**sheet.html**' is rendered by flask.

To show sheets from google spreadsheet to website Google Sheets to Html was used. File **google-sheets-html.js** does this task.

```
google.load('visualization', '1', {
  packages: ['table']
});
var visualization;

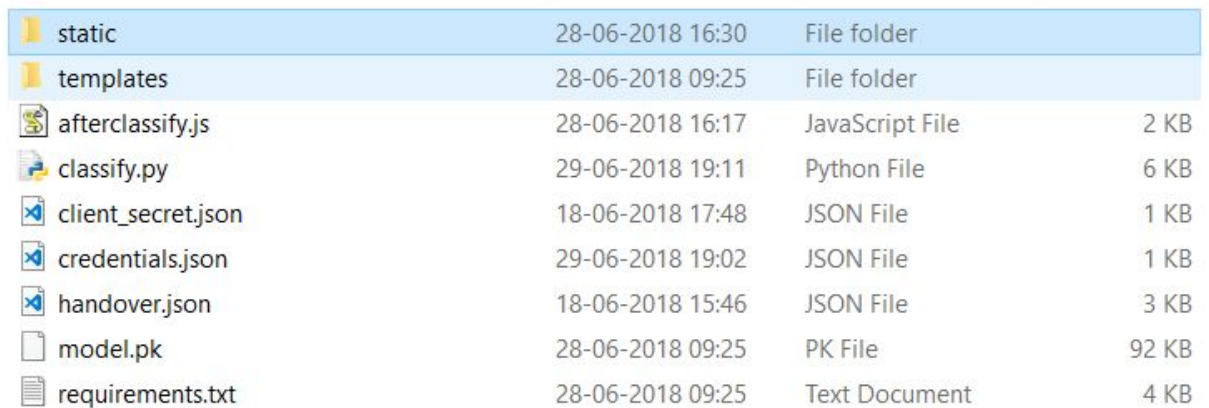
function drawVisualization() {
  var query = new google.visualization.Query('https://spreadsheets.google.com/tq?key=1TQM0ys75pq2GmQFPipM1_hiucmR2dhoPhYz6CuAW_lE&output=html&usp=sharing');
  query.setQuery('SELECT A, B, C, D label A "Exception", B "On Progress", C "Completed Task", D "Follow Up");
  query.send(handleQueryResponse);
}
```

And this is how it looks after this:

CURRENT UPDATES

| Exception | On Progress | Completed Task | Follow Up |
|---|---|--|---|
| Exception | On progress | Completed Task | Follow Up |
| [11:46:27] : billing cycle 5 started | | | |
| [11:46:42] : billing cycle 5 crashed | | | |
| | | | [11:46:52] : look after billing cycle 5 |
| | | [11:47:10] : billing cycle 7 completed | |
| [11:47:19] : memory out of bound | | | |
| [11:47:32] : memory out of bound when running billing cycle 5 | | | |
| [13:14:52] : server crashed maintaining balance cycle 4 | | | |
| [13:15:51] : server crashed when running billing cycle 5 | | | |
| | [13:16:04] : billing cycle 7 is currently running | | |
| | [14:25:46] : billing cycle 1234 is an hour | | |
| | | [14:26:03] : billing cycle 1, 2, 3 | |
| | [15:10:53] : billing cycle 7 started | | |
| [15:11:17] : memory out of bound when running | | | |
| [15:15:28] : memory out of bound when running billing cycle 5 | | | |
| | [19:02:33] : billing cycle 5 started | | |

Note : Entire backend is made using flask. In flask a directory structure needs to be followed. All the css files comes under **static/styles** folder, all the html files rendered by the flask server comes under **templates/** folder.



| | | | |
|--------------------|------------------|-----------------|-------|
| static | 28-06-2018 16:30 | File folder | |
| templates | 28-06-2018 09:25 | File folder | |
| afterclassify.js | 28-06-2018 16:17 | JavaScript File | 2 KB |
| classify.py | 29-06-2018 19:11 | Python File | 6 KB |
| client_secret.json | 18-06-2018 17:48 | JSON File | 1 KB |
| credentials.json | 29-06-2018 19:02 | JSON File | 1 KB |
| handover.json | 18-06-2018 15:46 | JSON File | 3 KB |
| model.pk | 28-06-2018 09:25 | PK File | 92 KB |
| requirements.txt | 28-06-2018 09:25 | Text Document | 4 KB |

Above screenshot explains the directory structure required for flask app.

Requirements:

XAMPP, Python v3.6, Web browser with speech recognition feature. Along with this one needs to import all the libraries used in making flask web server. 'requirements.txt' contains all the required libraries. It can be done by doing following in terminal:

```
pip install requirements.txt
```

Code Links:

<https://github.com/itsabhijeet/Handover>

Contributors:

Abhijeet Kumar(<https://www.linkedin.com/in/itsabhijeetkr>)

Pratyush Kesarwani