Plan of Action

T64PF1 & T64NE1
Aashish G, Athreya Chandramouli, Devesh Verma, Sachin Dev

1 Brief Summary

The IBM Watson *Natural Language Understanding* API, can be utilized to analyze features of natural langauge text. The goal of submission 1 is to build a bare-bones application which integrates the *POST /analyze* functionality of the API, and utilize it to categorize some given input text. The application should be suitable for use as a boilerplate starter-project for /hub.

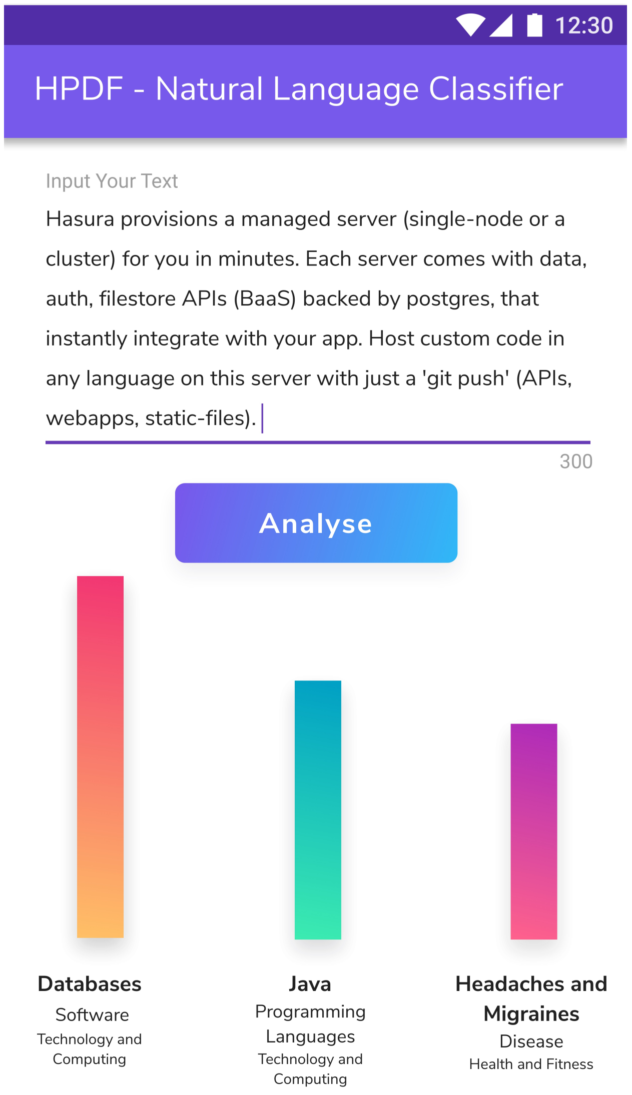
As of now, we've successfully managed to integrate the API's basic functionality, using the provided wrappers and have tested a few hardcoded inputs. We've drawn up wireframes and mockups for the frontend and are in the process of implementing them. Once this is completed, we can integrate it with the backend using Jinja templates. More details can be found in the next section.

2 Sub-Task Checklist

- **Backend - Python-Flask/NodeJS-Express** **Status**
  - **Installation of Requirements** **Complete**
    To integrate the API, *watson-developer-cloud* must be installed. This is done using pip (*pip install –upgrade watson-developer-cloud*) for Flask and npm (*npm install watson-developer-cloud*) for Express.
  - **Authentication using BlueMix Account** **Complete**
    In order to use the API, authentication is required. To generate the service credentials, the user has to sign up for the service using their IBM BlueMix account.

  - **Basic integration with Flask/Express** **Complete**
    Once the authentication is successful, the basic integration is done by importing the appropriate modules(*NaturalLanguageUnderstandingV1*), instantiating the required class as an object(*understandingObj*), and calling the appropriate methods (*understandingObj.analyze()*).
  - **Testing with Hardcoded inputs** **Complete**
    Given a body of text, the API returns a JSON object consisting of the category and the corresponding confidence. The integration was tested using some hardcoded bodies of text to check its functioning. An example of the input and output can be found in the next section.
  - **Integration with Frontend** **Pending**
    Using Jinja templates to combine the Frontend with Flask. Similarly, integration is done with Express as well.

- **Frontend - React-Native**
  - **Form UI** **Wireframe**
    TextInput tag is used for inputting the text to be passed to the API. The UI is minimalist with a simple Gradient style for the button.(Wireframe Uses: Gradient #7956EC - #2FB9F8)
  - **Result UI** **Wireframe**
    The result is shown as 3 TextViews for the 3 sets of variable. It is graphically represented using 3 ImageViews with the background colour as a Gradient (Wireframe Uses: #FFC066 - #F33573, #3CECB0 - #009FC5, #FF618C - #AD2AB9) distributed by the tag <*JustifyContent="space-between"* >. Currently this is a static image, but it can be modified to dynamically resize depending on the confidence value.
  - **Error Handling** **Pending**
    Any obvious errors, such as invalid inputs, insufficient characters, excessive characters etc... are handled at the Frontend directly.
  - **Integration with Backend** **Pending**
    Using Jinja templates to combine the Frontend with Flask. Similarly, integration is done with Express as well.



Hasura provisions a managed server (single-node or a cluster) for you in minutes. Each server comes with data, auth, filestore APIs (BaaS) backed by postgres, that instantly integrate with your app. Host custom code in any language on this server with just a 'git push' (APIs, webapps, static-files).

Figure 1: Frontend Wireframe

3 Sample I/O

Input Text:

> ”Hasura provisions a managed server (single-node or a cluster) for you in minutes. Each server comes with data, auth, filestore APIs (BaaS) backed by postgres, that instantly integrate with your app. Host custom code in any language on this server with just a ’git push’ (APIs, webapps, static-files).”

Output JSON:[1]

```json
{
  "categories": [
    {
      "label": "/technology and computing
          /software/databases",
      "score": 0.436991
    },
    {
      "label": "/technology and computing
          /programming languages/java",
      "score": 0.311278
    },
    {
      "label": "/health and fitness/disease
          /headaches and migraines",
      "score": 0.259054
    }
  ],
  "language": "en",
  "usage": {
    "features": 1,
    "text_characters": 300,
    "text_units": 1
  }
}
```

[1]**Note**: The output is posted verbatim. This does not reflect the thoughts and views of any of the team members. We don’t think Hasura is about headaches and migraines, but for some reason Watson seems to.

4 Resources/Docs

The Documentation for the API can be found here. It serves as a comprehensive guide for understanding and using the API functionalities, including the *POST /analyze*. The docs provide syntax (for the Node and Python wrappers), input and output format, and details about parameters and parameter types.

5 Responsibilities

Name	**Email**	**Allotted Framework**
Aashish G	aashishgrao1@gmail.com	React-Native
Athreya Chandramouli	cathreya98@gmail.com	Python-Flask
Devesh Verma	deveshverma619@gmail.com	NodeJS-Express
Sachin Dev	hellosachindev@gmail.com	React-Native