

Chapter 5

Implementation

5.1. Data Collection

To find out what are the problems that the students really faced. An online survey via google form was done and also personal interaction with students was done, especially freshmen. It was found that the one of the major problem that they faced was the dilemma of selecting the most suitable engineering stream for them. Apart from that many students complained that some of the processes like hostel allocation took a long time and often led to unwanted scenarios. Many students also stated that they are unable to keep track of all the events and seminars around the campus and therefore sometimes would miss those that they were interested in. Apart from freshmen and juniors, some final year students were also interacted with to know about their problems, if any. It was noticed that because of the varied domains such as ML, IOT, Web Development present in the IT sector, they were confused which to pursue as a professional career.

5.2. Data Synthesise

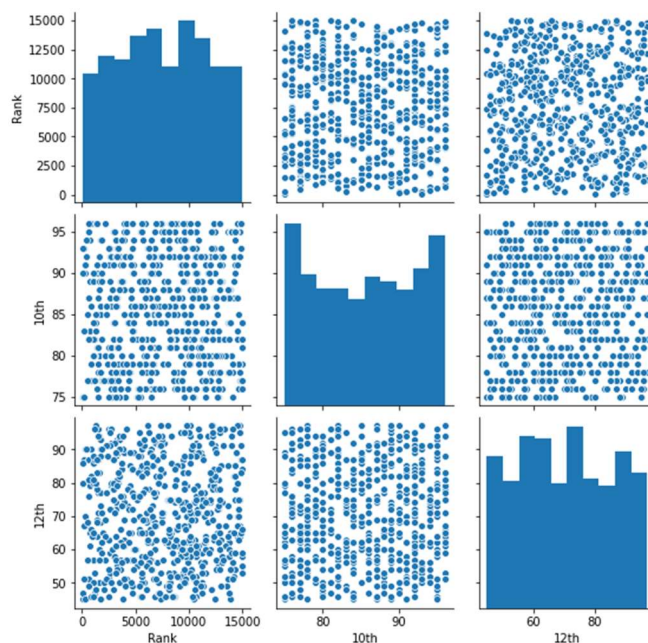


Fig 9. Data Visualization and Outliers Determination

To utilize the most useful information, from the complete dataset, data was plotted in several ways and tried to extract the best portions from the dataset. Along with that, some portion of data were synthesized based on the architecture of another dataset.

5.3. Feature Engineering

In both datasets, the columns with high covariance were removed. Also to build a better classifier, Gaussian transforms were applied to those data columns without normal distributions. Moreover using Forward Feature Selection procedure, best columns were searched for Output labels.

5.4. Ensemble Training and Model Building

Ensemble methods involve a group of predictive models to achieve a better accuracy and achieve a better accuracy and model stability. Some of the commonly used ensemble methods are bagging and boosting

In this Project, Bagging classifier Random Forest is being applied on the model. Bagging stands for Bootstrap Aggregation. The essence is to select bootstraps that fit a classifier on each of the created samples, train the models in parallel and then consider the average of the results.

5.5. Deep Learning and Model Building

To prepare a model on synthesised data, the MLP, RBM and CNN were used. Among which CNN provided the best result. The CNN was prepared of a architecture containing 3 one dimensional convolutional layer, treated with both 1D average and max pool layers among

Model: "sequential_8"

Layer (type)	Output Shape	Param #
conv1d_17 (Conv1D)	(None, 5, 128)	512
conv1d_18 (Conv1D)	(None, 5, 128)	49280
max_pooling1d_7 (MaxPooling1D)	(None, 1, 128)	0
dropout_4 (Dropout)	(None, 1, 128)	0
conv1d_19 (Conv1D)	(None, 1, 32)	12320
flatten_5 (Flatten)	(None, 32)	0
dense_9 (Dense)	(None, 32)	1056
dense_10 (Dense)	(None, 7)	231
Total params: 63,399		
Trainable params: 63,399		
Non-trainable params: 0		

which, maxpooling1D led to better result. For a relative faster convergence, Rectified Linear Unit (ReLU) had been used. Having all values in positive scale, there were less chance of leakage. Being multiclass classification problem, output layer activation us kept as "Softmax". An Adam optimizer with categorical_crossentropy type loss measurement function has been used for training phase. The ultimate weight files are stored to pickles.

Fig 10. ConvNet Architecture

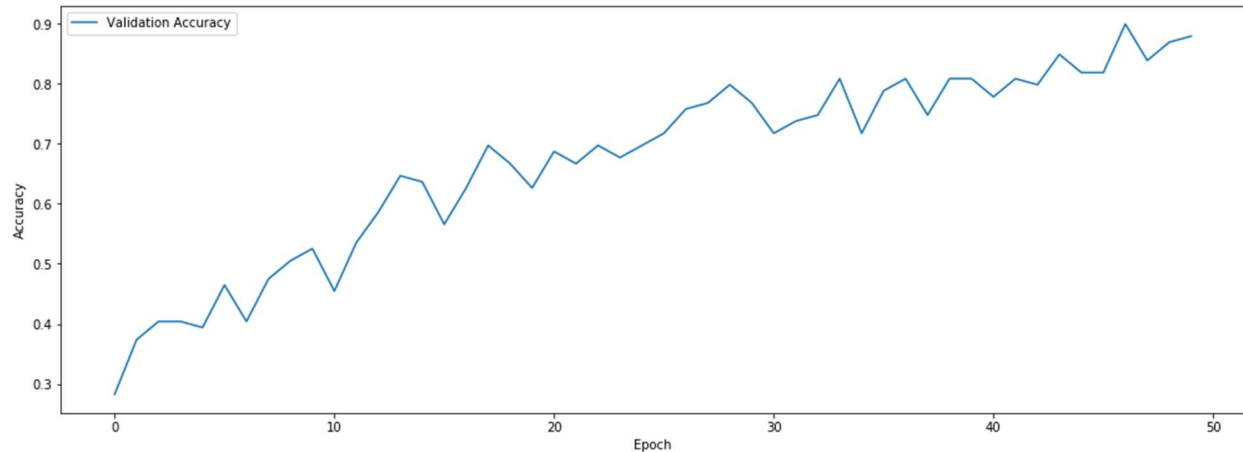


Fig 11. Validation Accuracy Curve

5.6. Database Preparation

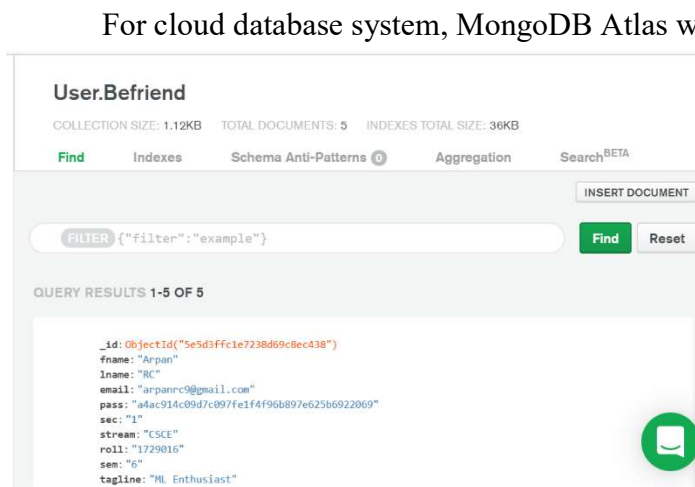


Fig 12. Mongo Atlas JSON View

system. MongoDB cloud is easy to integrate with most of the programming languages. For the complete development procedure, Befriend collection had been created containing different Databases for different purpose. MongoDB supports JSON type data, so all transferred data was JSON encoded.

5.7. API preparation and cloud host

WSGI server was establish in the local system and REST APIs were prepared using Python Flask. Necessary methods like “GET” and “POST” were used considering the size of transferred data chunks and status codes were used accordingly. Using the pickle, weight files were connected to the Flask Files. Pre-prepared database were connected where necessary. And after sufficient test, APIs were deployed in cloud and performed one more test.

5.8. Establishment of Storage

To store files in cloud storage, a S3 bucket with an ARN “arn:aws:s3:::befriendminor” was established.

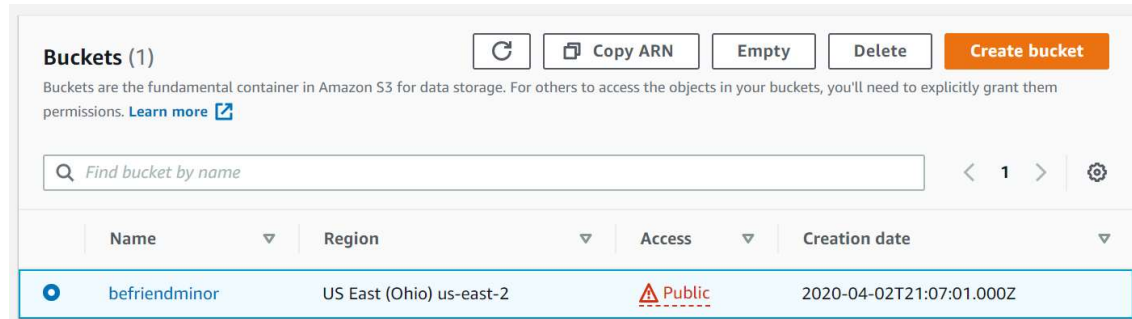


Fig 13. S3 Management Console

5.9. Front UI Preparation

After being established with quality standard tools and APIs, the UI has been prepared. The UI comes with specific session, starts when a user logs into the system. The session is destroyed when user logs out. User may access the dashboard from the member index page. Other necessary workflows start from that page. The attached backend is prepared with multiple PHP layers.

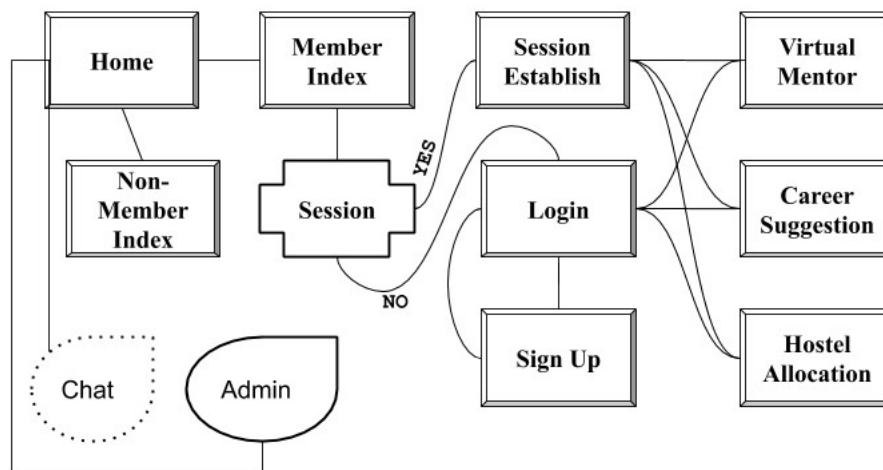
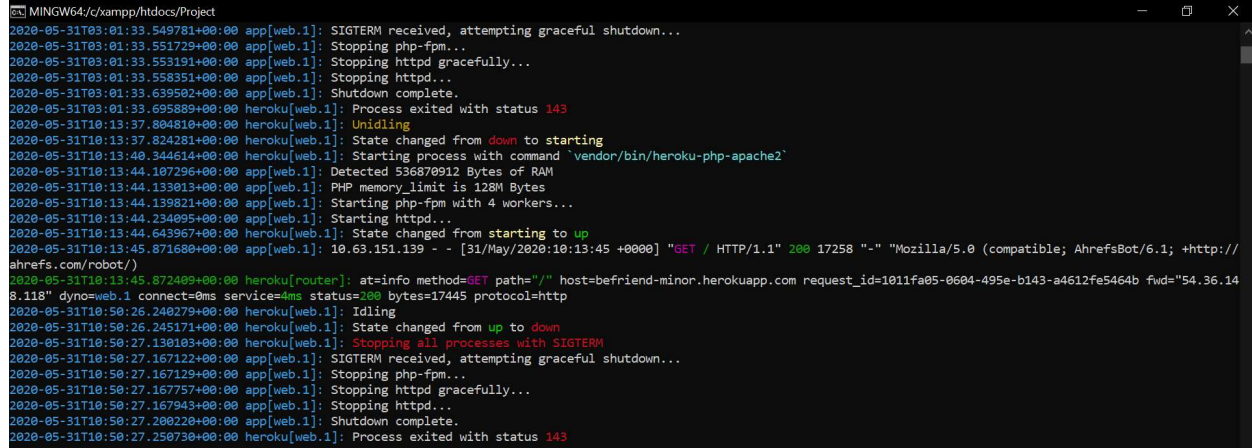


Fig 14. UI Frames Data Flow

5.10. System Deployment

The system is deployed in Heroku Cloud using version control git. A Procfile was set up with **heroku-php-apache 2** server and **Nginx reverse proxy**.



```

MINGW64/c/xampp/htdocs/Project
2020-05-31T03:01:33.549781+00:00 app[web.1]: SIGTERM received, attempting graceful shutdown...
2020-05-31T03:01:33.551729+00:00 app[web.1]: Stopping php-fpm...
2020-05-31T03:01:33.553191+00:00 app[web.1]: Stopping httpd gracefully...
2020-05-31T03:01:33.558351+00:00 app[web.1]: Stopping httpd...
2020-05-31T03:01:33.630502+00:00 app[web.1]: Shutdown complete.
2020-05-31T03:01:33.695889+00:00 heroku[web.1]: Process exited with status 143
2020-05-31T10:13:37.804810+00:00 heroku[web.1]: Unidling
2020-05-31T10:13:37.824281+00:00 heroku[web.1]: State changed from down to starting
2020-05-31T10:13:40.344614+00:00 heroku[web.1]: Starting process with command `vendor/bin/heroku-php-apache2`
2020-05-31T10:13:44.107296+00:00 app[web.1]: Detected 536870912 Bytes of RAM
2020-05-31T10:13:44.133013+00:00 app[web.1]: PHP memory limit is 128M Bytes
2020-05-31T10:13:44.139021+00:00 app[web.1]: Starting php-fpm with 4 workers...
2020-05-31T10:13:44.234095+00:00 app[web.1]: Starting httpd...
2020-05-31T10:13:44.643967+00:00 heroku[web.1]: State changed from starting to up
2020-05-31T10:13:45.871680+00:00 app[web.1]: 10.63.151.139 - - [31/May/2020:10:13:45 +0000] "GET / HTTP/1.1" 200 17258 "-" "Mozilla/5.0 (compatible; AhrefsBot/6.1; +http://ahrefs.com/robot/)"
2020-05-31T10:13:45.872409+00:00 heroku[router]: at=info method=GET path="/" host=befriend-minor.herokuapp.com request_id=1011fa05-0604-495e-b143-a4612fe5464b fwd="54.36.148.118" dyno=web.1 connect=0ms service=4ms status=200 bytes=17445 protocol=http
2020-05-31T10:50:26.240279+00:00 heroku[web.1]: Idling
2020-05-31T10:50:26.245171+00:00 heroku[web.1]: State changed from up to down
2020-05-31T10:50:27.130103+00:00 heroku[web.1]: Stopping all processes with SIGTERM
2020-05-31T10:50:27.167122+00:00 app[web.1]: SIGTERM received, attempting graceful shutdown...
2020-05-31T10:50:27.167129+00:00 app[web.1]: Stopping php-fpm...
2020-05-31T10:50:27.167757+00:00 app[web.1]: Stopping httpd gracefully...
2020-05-31T10:50:27.167943+00:00 app[web.1]: Stopping httpd...
2020-05-31T10:50:27.200220+00:00 app[web.1]: Shutdown complete.
2020-05-31T10:50:27.250730+00:00 heroku[web.1]: Process exited with status 143
  
```

Fig 15. Heroku Application Log