**Shubhangi Agarwal**

**Self-Work Report**

**Week 1**

**Status**

Implemented work

* Basic database model for taking user input (Through Django, sqlite3)
* Handling http\_requests and responses (Through Django)
* Addition of user input into training data (csv file) (Through Django)
* A REST API for connecting with frontend (Through Django)
* Creation of serialiser to serialise and de-serialise the data (Through Django)
* API mapping to create a request from React to Django (Through React.js)

In Progress

* Connection of frontend with backend for getting an end-to-end workflow

**Backend Details**

Final Database Schema (As discussed with all)

* Partner : {

id : AutoField, primary key

firstName : CharField

lastName : CharField

email : EmailField, unique

contact : CharField

address : CharField

city : CharField

state : CharField

typestore : CharField

size : CharField

workingemployees : PositiveIntegerField

customers : PositiveIntegerField

service : CharField

password : CharField

confirmPassword : CharField

}

Initial Database Schema (self-thought)

* Partner : {

id

FirstName

LastName

Email

Contact

Age

Password

Location of the store (city,state,pincode)

Number of employees working in the store

Type of store

Size of store

Google maps rating

Annual sales of the store

Additional info.

Website link

Storage size willing to give to the locker

}

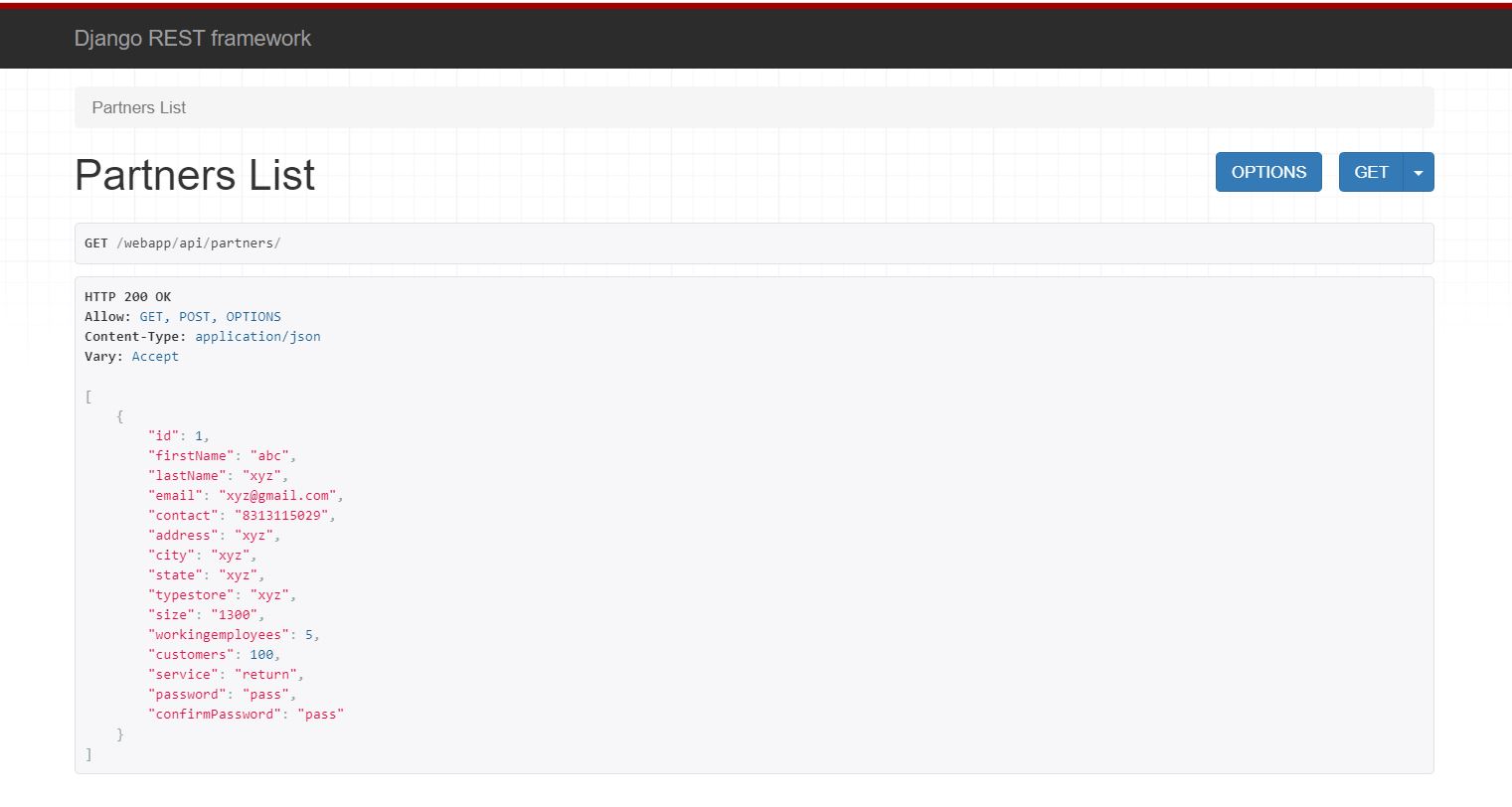
Routes

* webapp/api/partners/list/ : { method : ‘post’ , ‘get’ }
* webapp/api/partners/list/details/ : { method : ‘put’ , ‘delete’ }

Requests and responses

**{ ‘request\_method’ ? response\_if\_true : response\_if\_false }**

* ‘get’ ? displays data stored in ‘Partner’ database : HTTP\_400\_BAD\_REQUEST
* ‘post’ ? HTTP\_201\_CREATED : HTTP\_400\_BAD\_REQUEST
* ‘put’ ? HTTP\_204\_NO\_CONTENT : HTTP\_400\_BAD\_REQUEST
* ‘delete’ ? HTTP\_204\_NO\_CONTENT : HTTP\_400\_BAD\_REQUEST



**Fig. 1:** Response for ‘get’ request

**Technical Challenges Faced**

* Understanding MVC and MVT architecture : Solved
* Making migrations to the model through Django : Solved
* Sending requests from React app to Django : Solved

**Week 2**

**Status**

Implemented work

* Connection of Frontend with Backend
* Made required changes in the attributes of the database model
* Read about various classification algorithms
* Read about google maps API
* Discussed upon and finalized some key parameters to be used for prediction
* Established and thought about the relation between the parameters for prediction of correct program and discussed about it with the group
* Read about deploying pre-trained ML models using Django

In Progress

* Identification of more key parameters and algorithms to be used for prediction of program

**References**

* <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052>
* <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/>
* <https://towardsdatascience.com/productionize-a-machine-learning-model-with-a-django-api-c774cb47698c>
* <https://developers.google.com/maps/documentation/api-picker>
* <https://www.geeksforgeeks.org/understanding-logistic-regression/>

**Technical Challenges Faced**

None

**Week 3**

**Status**

Implemented work

* Integration of login, sign-up, add store, dashboard flow for the user and created different REST APIs for each respectively at backend
* Refactoring of the database model so that a user can onboard multiple stores
* Established a many-to-one relationship amongst the database model
* Alteration of the serialiser according to the database model

In Progress

* Authentication of the user from the backend

**Refactored Database Schema**

Customer Store

id 1 id

firstName n ownerId

lastName storeName

email address

contact city

confirmPassword state

pincode

typeStore

size

employees

customers

service

**Updated Routes**

* webapp/api/login/ : { method : ‘post’ }
* webapp/api/signup/ : { method : ‘post’ }
* webapp/api/dashboard/store-details/ : { method : ‘get’ }
* webapp/api/dashboard/add-store/ : { method : ‘post’ }
* webapp/api/dashboard/delete-store/ : { method : ‘put’ , ‘delete’ }

**Technical Challenges Faced**

None