**Ayushi Gupta**

**Self-Work Report 1**

**Status**

Implemented work

* Discussed upon the attributes to be used for user input and further processing.
* Created a Database Model according to the need

**In Progress**

* Generating dataset for program prediction
* Identifying algorithms for multi class classification

**Technical Challenges Faced**

* None, so far.

**Self-Work Report 2**

**Status**

Implemented work

* Reading about various multilabel classification algorithms and their implementation.
* Narrowed down to:
* Discussing the criteria for program labelling, to be used in raw data generation and labelling.

**In Progress**

* Training data generation wrt parameters discussed and confirmed.
* Applying the algorithm and checking for accuracy.

**Technical Challenges Faced**

* Creating a good training dataset with parameters required to be used in prediction labelled correctly.

**Research done so far:**

Have been working on reading up and strudying about the algorithms which can help in multilabel and/multiclass classification of the programs, easily and effieciently.

Methods discovered so far are:

* Logistic regression

Adv: is most useful for understanding the influence of several independent variables on a single outcome variable.

Disadv: Works only when the predicted variable is binary, assumes all predictors are independent of each other, and assumes data is free of missing values.

* Random forests

Adv: Reduction in over-fitting and is more accurate than decision trees in most cases.

Disadv: Slow real time prediction; difficult to implement; complex algorithm

* Naïve Bayes

Adv:

1. Real time predictions: It is very fast and can be used in real time.
2. Scalable with Large datasets
3. Insensitive to irrelevant features.
4. Multi class prediction is effectively done in Naive Bayes
5. Good performance with high dimensional data(no. of features is large)

Disadv:

* 1. Independence of features does not hold: The fundamental Naive Bayes assumption is that each feature makes an independent and equal contribution to the outcome. However this condition is not met most of the times.
  2. Bad estimator: Probability outputs from predict\_proba are not to be taken too seriously.
  3. Training data should represent population well: If you have no occurrences of a class label and a certain attribute value together (e.g. class=”No”, shape=”Overcast “) then the posterior probability will be zero. So if the training data is not representative of the population, Naive bayes does not work well.(This problem is removed by smoothening techniques).
* Decision Tree

Adv:  simple to understand and visualise, requires little data preparation, and can handle both numerical and categorical data.

Disadv: can create complex trees that do not generalise well, and decision trees can be unstable because small variations in the data might result in a completely different tree being generated.

* XGBoost

Adv:

1. Less feature engg required
2. Feature importance can be found out
3. Fast to interpret
4. Minimal impact of outliers
5. **Handles large sized datasets** well.
6. Good execution speed
7. Good model performance
8. Less prone to overfitting
9. Good for any multiclass classification problem (Tried and tested at multiple Kaggle competitons)

Disadv:

1. Difficult interpretation, visualization
2. Overfitting possible if parameters aren’t tuned
3. Harder to tune

Also, parameters to be used for prediction in the model were discussed with some of the parameters being:

* No of customers visiting the shop
* No of shop employees
* Area/ size of shop
* Google rating

Further, studied and looked about google map api features offered, according to the need. It was found that we can capture the **no of stores in a given area**, and/or extracting a **store’s google rating**.

**Self-Work Report 3**

**Status**

Implemented work

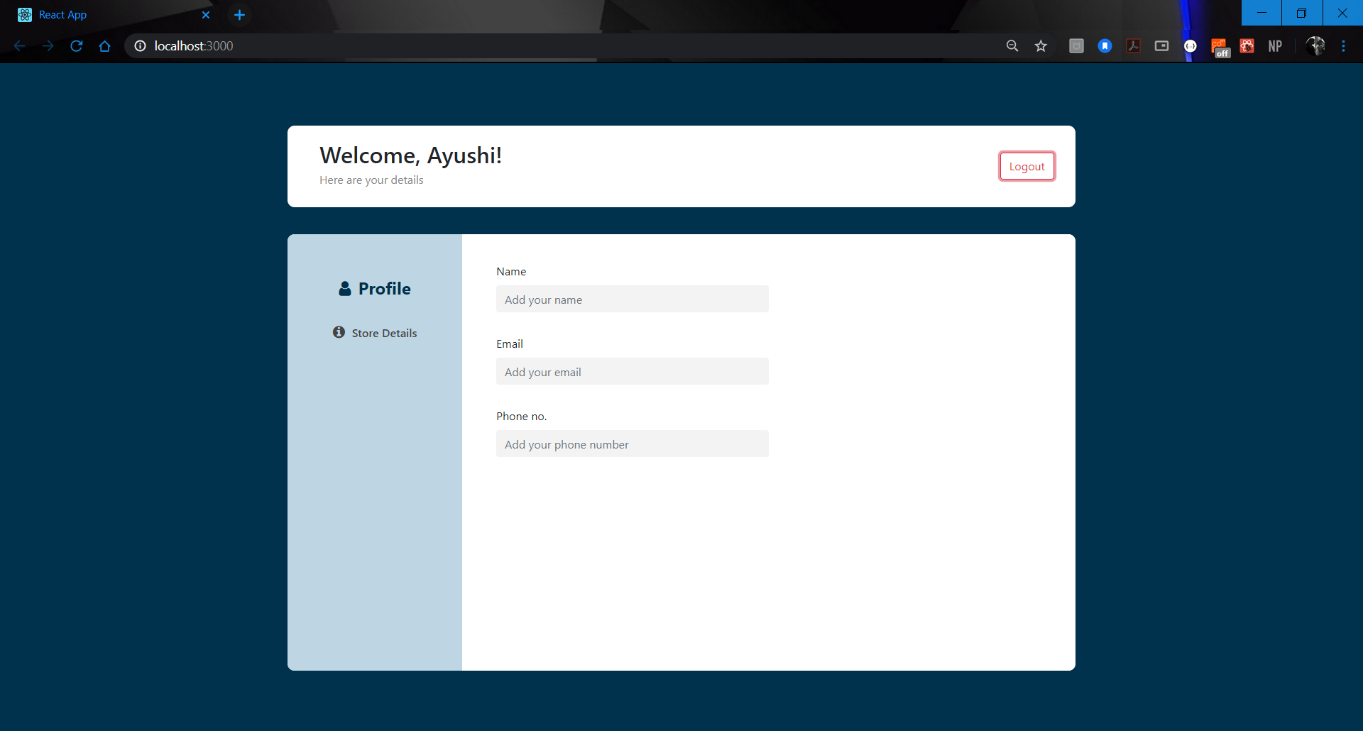
* Designed the dashboard component wrt to current design template
* Implemented dashboard component
* Added internal linking to the dashboard
* Corrected the UX of dashboard component
* Studied about Google Maps API
* Read about xgboost method
* Implemented xgboost on sample automatically generated data

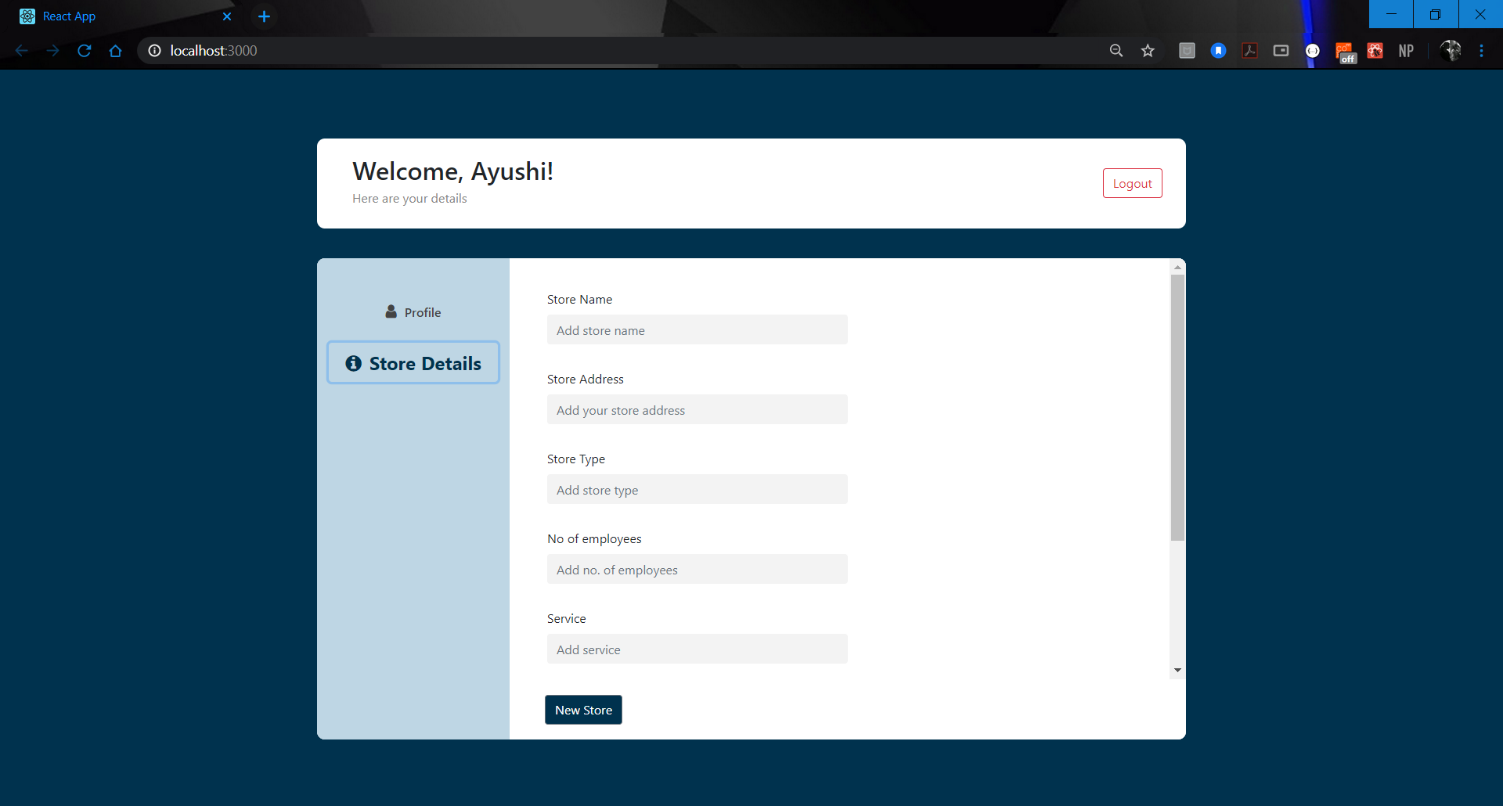
**In Progress**

* Adding user auth
* Connecting it to the login page
* UI/UX changes, if any

**Technical Challenges Faced**

* Understanding the used libraries: resolved





**Self-Work Report 4**

**Status**

Implemented work

* Studied about fetch apis and ReactJS implementation
* Added user authentication to Login Page(at folder Shubhangi Agarwal/Frontend/login.js)
* Cleaned Frontend login code(Removed warnings at login page)
* Added fetch API to dashboard Component

**In Progress**

* Merging dashboard with login component to complete the flow
* Responsiveness of dashboard component

**Technical Challenges Faced**

* Understanding fetch implementation (Post/Get) with ReactJS: resolved

**References:**

* <https://levelup.gitconnected.com/how-to-make-http-requests-with-fetch-api-be018730811f>
* <https://jasonwatmore.com/post/2020/02/01/react-fetch-http-post-request-examples>
* <https://stackblitz.com/edit/react-http-post-request-examples-fetch?file=App/PostRequestErrorHandling.jsx>
* <https://www.youtube.com/watch?v=oRL-pttfNSc>
* <https://www.youtube.com/watch?v=T3Px88x_PsA>