**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:

* + 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.
  + Bad estimator: Probability outputs from predict\_proba are not to be taken too seriously.
  + 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

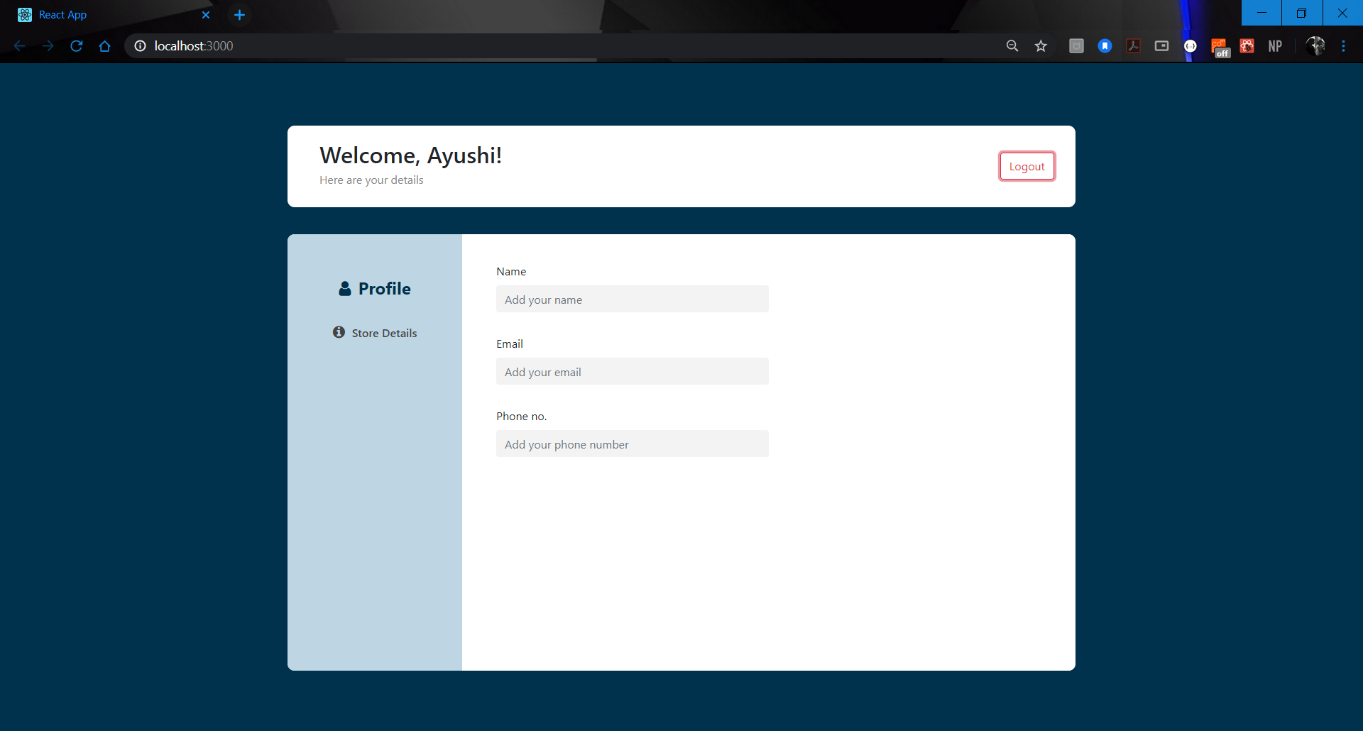
* Designed the dashboard component wrt to current design template.
* Implemented dashboard component.
* Added internal linking to the dashboard using tabs.
* Corrected the UX of dashboard component.
* Studied about Google Maps API.
* Read about xgboost method.
* Implemented xgboost on sample automatically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

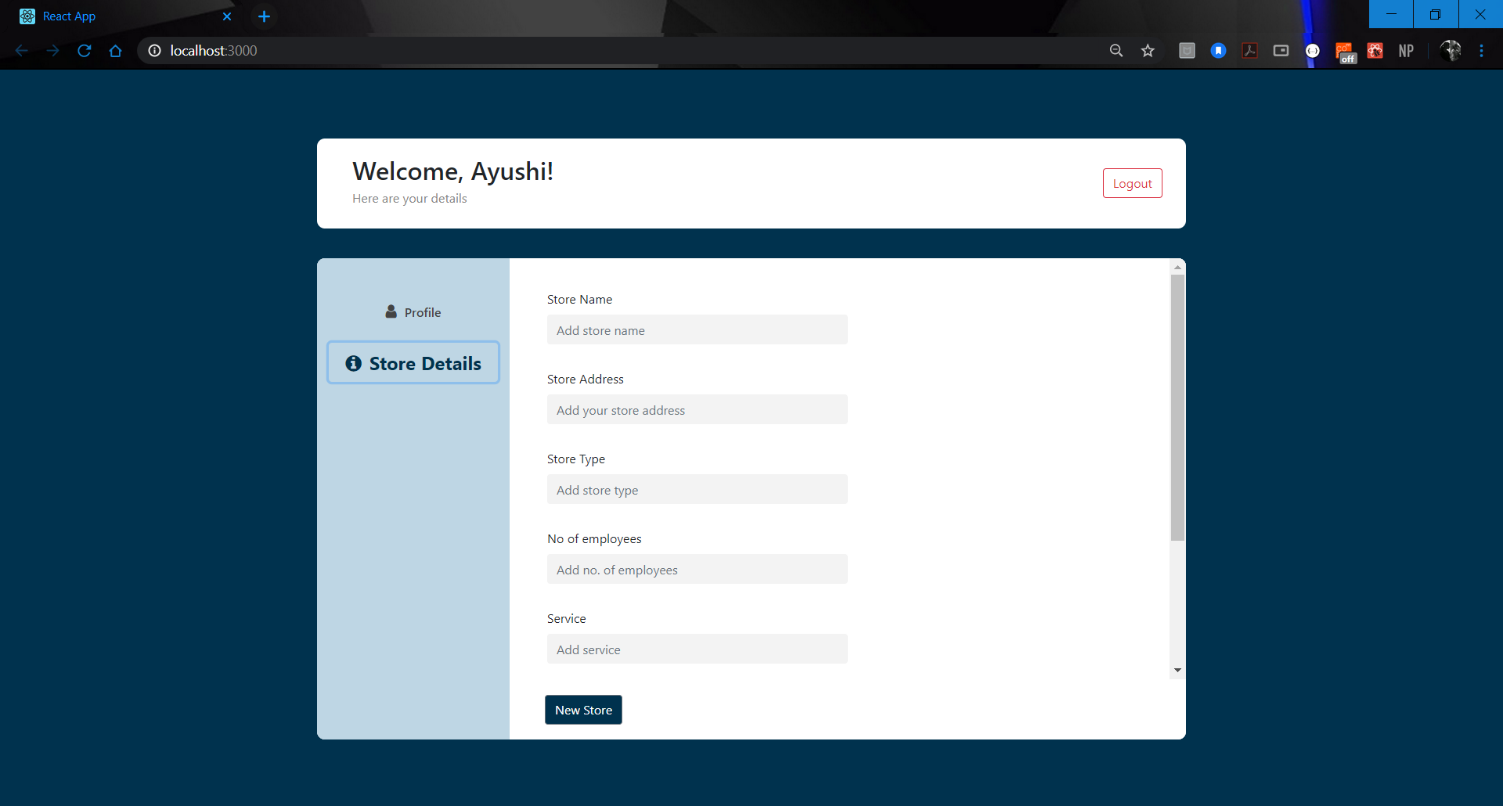
**In Progress**

* 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>

**Self-Work Report 4**

**Status**

Implemented work

* Studied further about fetch apis and ReactJS implementation.
* Merging dashboard with login component to complete the flow.
* Added fetch API to Dashboard Component.
* Moved all work/dashboard component to Project folder and completed the end to end connections, frontend wise.
* Testing of end-to-end flow of the application built till now.
* Tried deploying ML model at the server side.

Remaining work

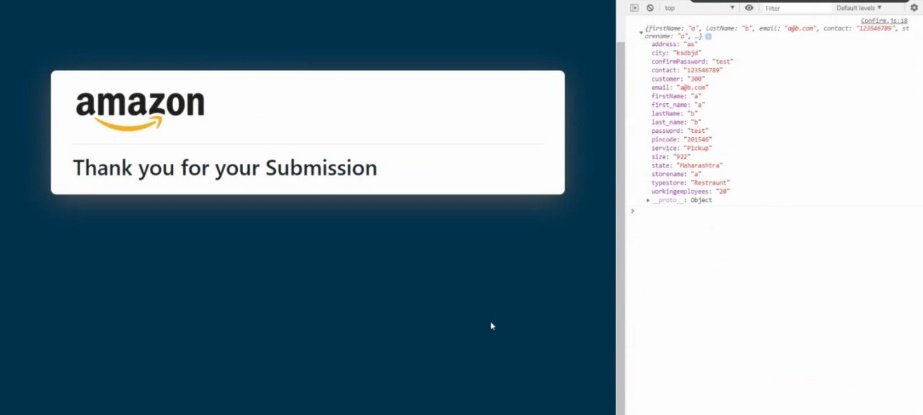
* Adding selected program variable from the backend, after successful ML-backend server integration.

Issue/Error reported and status, and the updates made:

1. ***Confirm.js was not hitting the signup api.***
   * This led to no information being stored from frontend to backend.
   * Status: **Fixed**; *there was an error in the api done as before, corrected it to the right one.*
2. ***Registration api was not being hit, bad code.***
   * Wrong code was written. There was a .then().catch().then() which led to the last then() always being executed: wrong and bad practice
   * Status: **Fixed;** *changed the code to correct, industry format. Changed .then().catch().then() to .then().catch() only.*
3. ***No .gitignore at both frontend and backend parts of Project folder.***
   * This led to problems while committing and pushing a file through the command line as all the node modules and py\_cache was being read too.
   * Status: **Fixed;** *added .gitignore to the both the folders.*
4. ***Changed Project title name to Pickup Point Classification.***
   * The project displayed name as React App in the title bar of the local host.
   * Status: **Fixed;** *changed the name to Pickup Point Classification for the title bar.*
5. ***No token being received at the login page from backend.***
   * Due to an issue in the Registration files, no token was received for login auth, for valid entries.
   * Status: **Fixed.**
6. ***Updated the login auth to successfully receive tokens and data.***
   * *Stored token at the localStorage() for ease of access during data fetch at the dashboard.*
7. ***User Firstname and Lastname not being stored in the database.***
   * The first name and last name are empty.

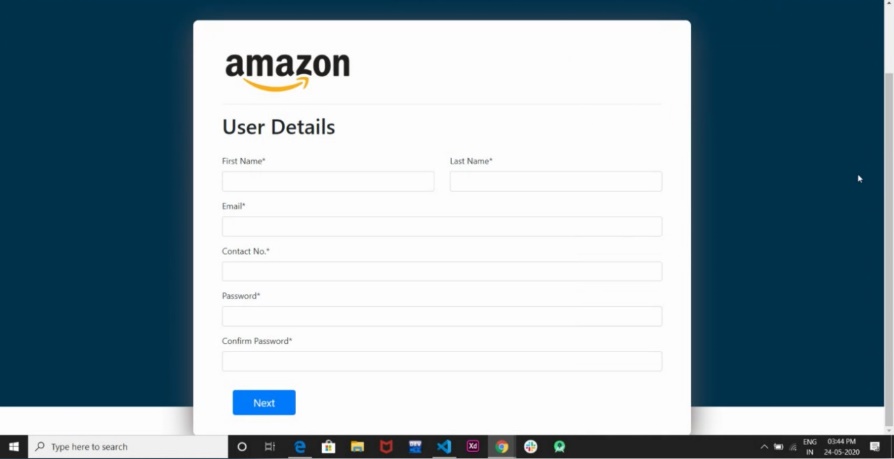


* + Status: **Fixed;** *mismatched variabls were being used to connect between the frontend and backend. Variable name in the backend was first\_name and last\_name, while at the frontend(confirm.js), firstName and lastName was being used. Due to this, the data was never being read at the backend. (As checked for, below).*

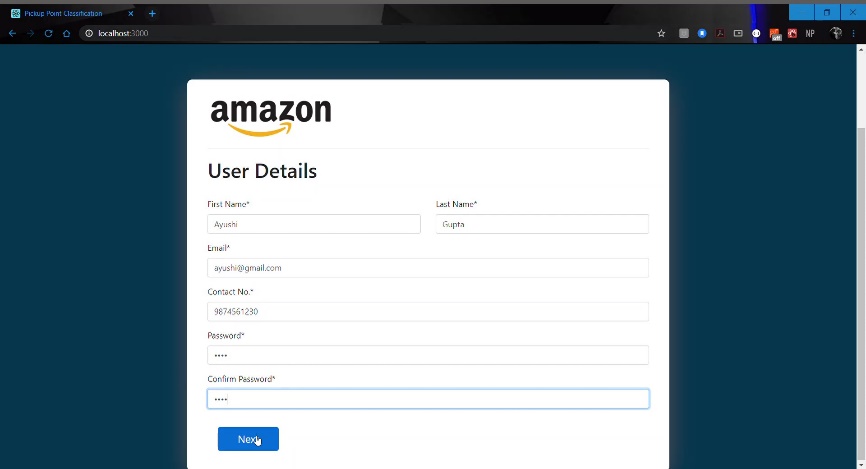


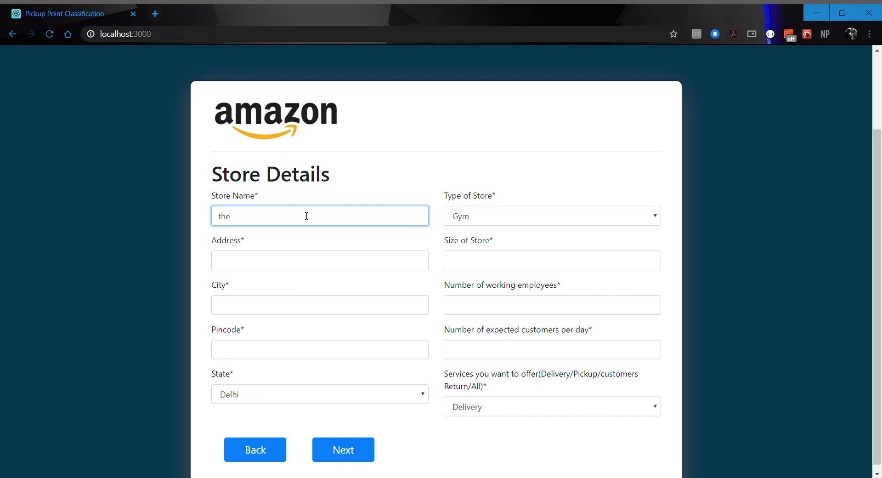
* + *Issue was resolved by refactoring “firstName” to “first\_name” and similar for lastName at confirm.js folder, so that it is recorded in the api.*

1. ***Signup/Registration form css overflow(all screens).***

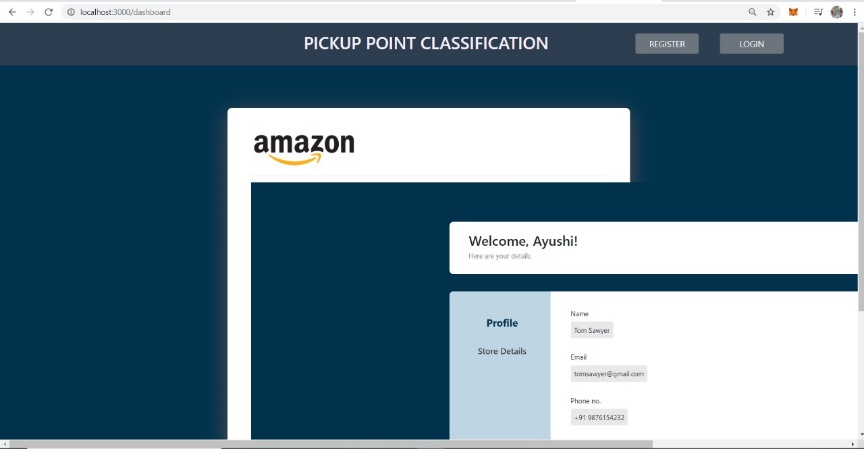


* + Page CSS overflow due to previous code being written for only one particular screen.
  + Status: **Fixed;** *replaced fixed height with dynamic min-height.*

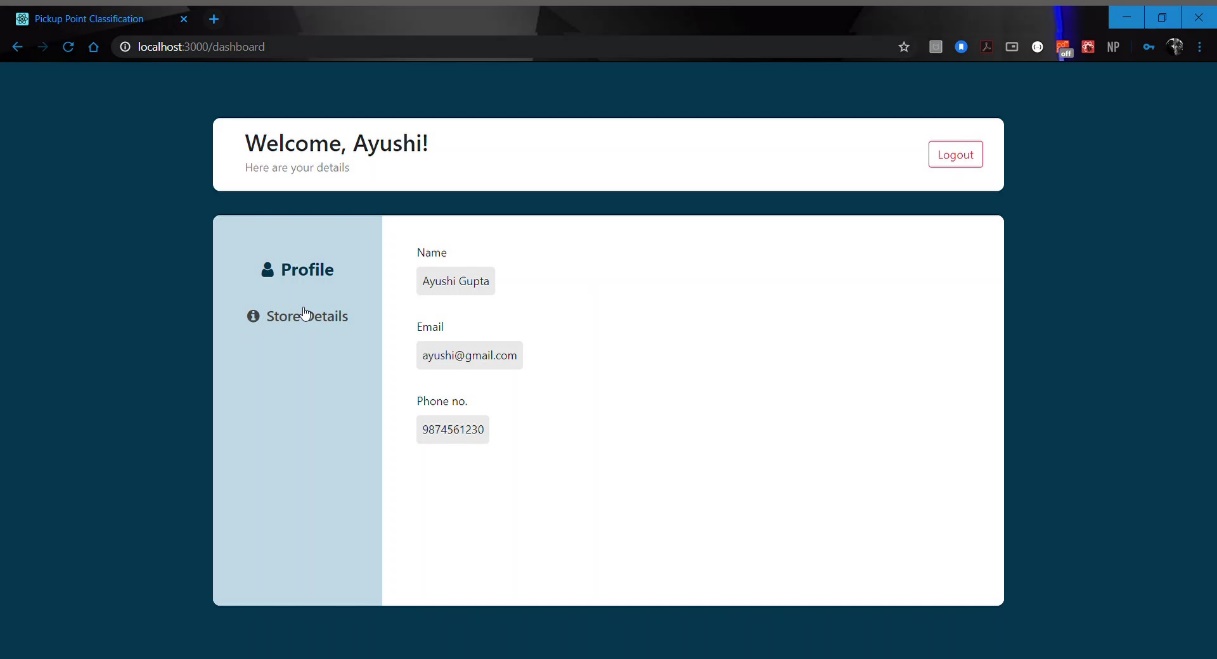




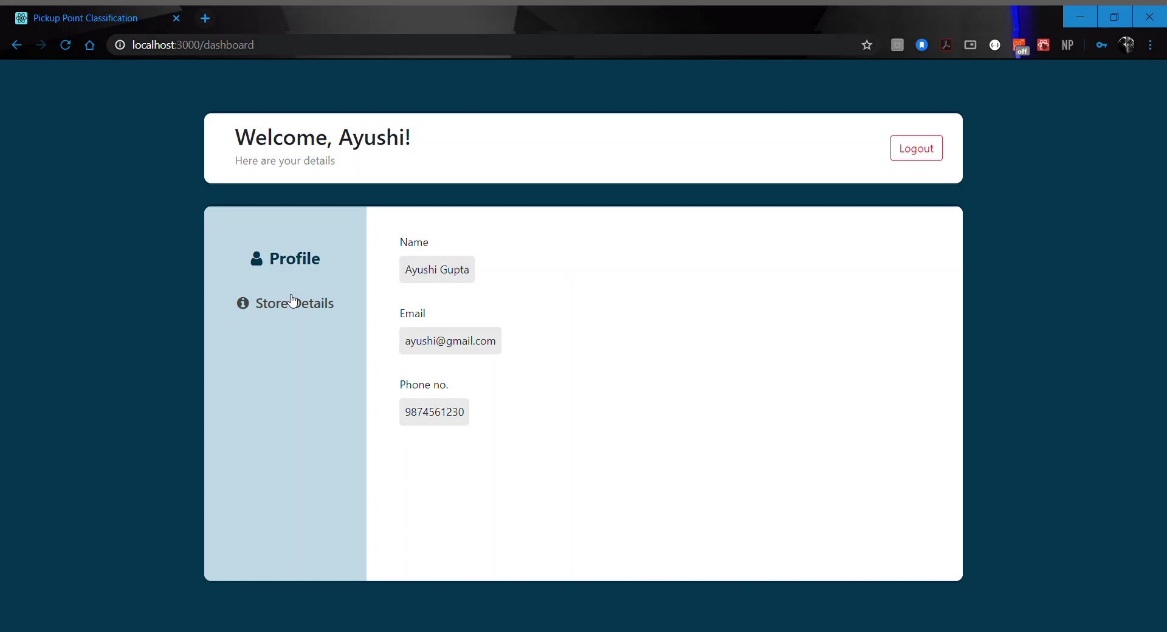
1. ***Login to Dashboard routing/CSS issues***



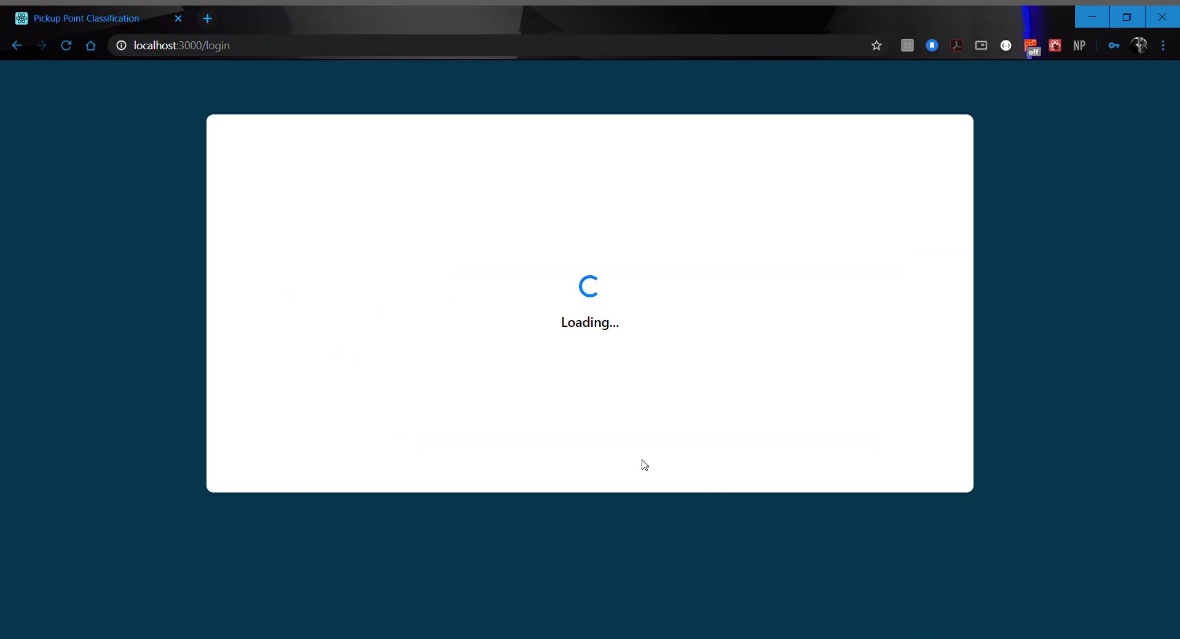
* + Dashboard pages were being loaded into the pre-set code. The sign-up form code was routed using a static wrapper for the pages developed, assuming all the other code would follow suit and fit into the design without informing/discussing.
  + This led to issues in loading the dashboard component as it is a different component which should not have the pre-sets of the signup page, UI/UX logic wise, namely header with register/login buttons, and the form box.
  + Status: **Fixed;** *replaced the form-wrapper in App.js with an HOC(Higher Order Component), for code reusability and efficiency.*

**

1. ***Updated Dashboard icons.***
   * While shifting the code from folder /Ayushi\_Gupta/Frontend/Dashboard to /Project/frontend/Components/Dashboard, icons were not transported.
   * Status: **Fixed;** *added the icon call at index.html to be imported successfully at the Dashboard.*

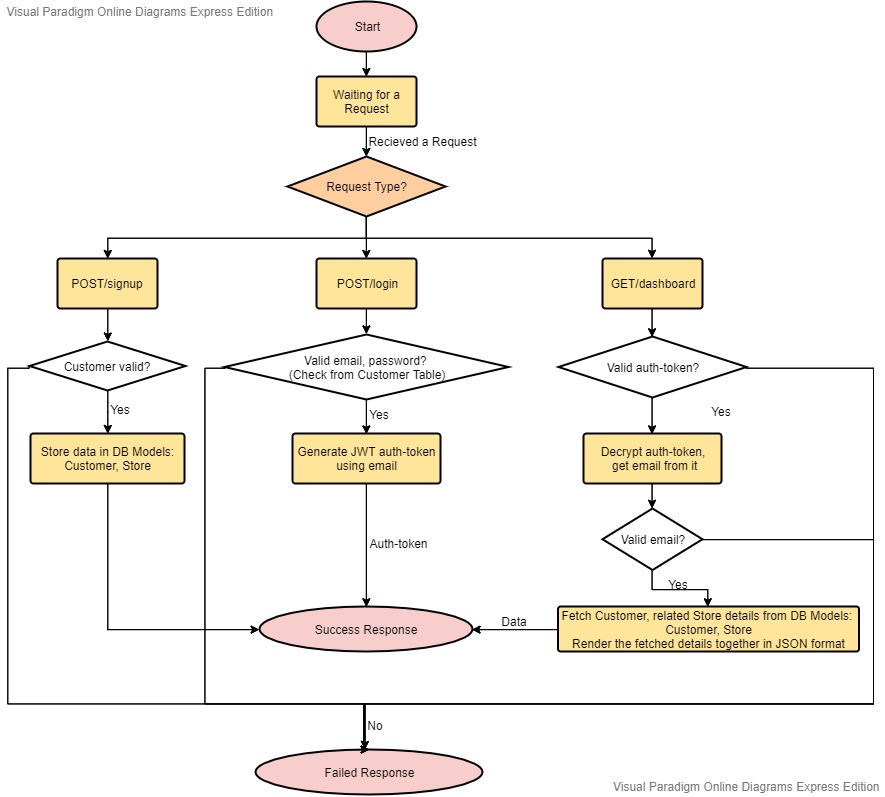
**

1. ***Added Spinner to Dashboard, loaded while data is being fetched from the backend.***

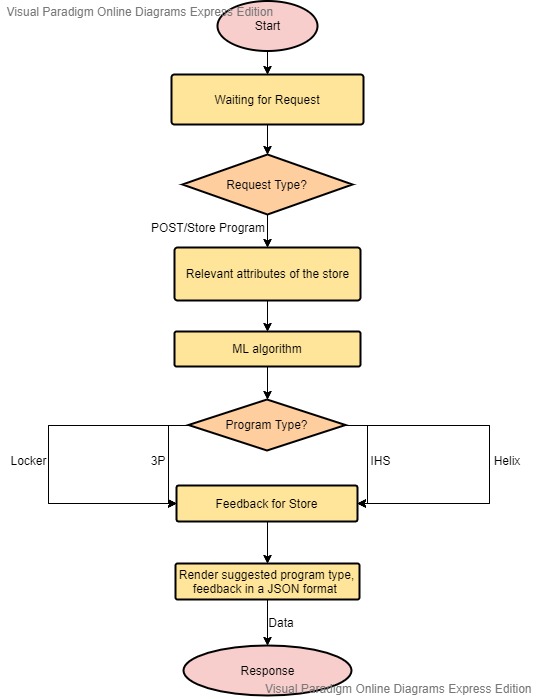
******

1. ***Updated Dashboard with GET api call to get user’s personal and store details.***
   * Updated statically inserted values with dynamically fetched values from the api/token value.
   * Updated Welcome, {name} to that of the user firstname.
2. ***All users’ data being sent, along with the passwords during api call.*** 
   * All the users’ records are being dumped during dashboard api call.
   * Not secure
   * Reported the issue to the backend.
3. ***Added successful Logout auth.***
   * Updated logout function to successfully terminate user session by removing the authToken from cache/localStorage.
   * Redirected logout to login page.
4. ***Tested end-to-end of the application built so far.***
   * The routing and frontend display is working perfectly. [Application complete, frontend wise]
   * Some issues were observed in the data flow while server data fetch. Reported the same to the teammates. Issue has been resolved now.

**Present Flow Diagram**



**Remaining Flow Diagram**



**Technical Challenges faced**

* Resolving the issue of first-name last-name not being stored at the backed: resolved.
* Resolving the issue of user data not being stored at the backend from Confirm.js: resolved.
* Identifying issues in routing form login to dashboard: resolved.