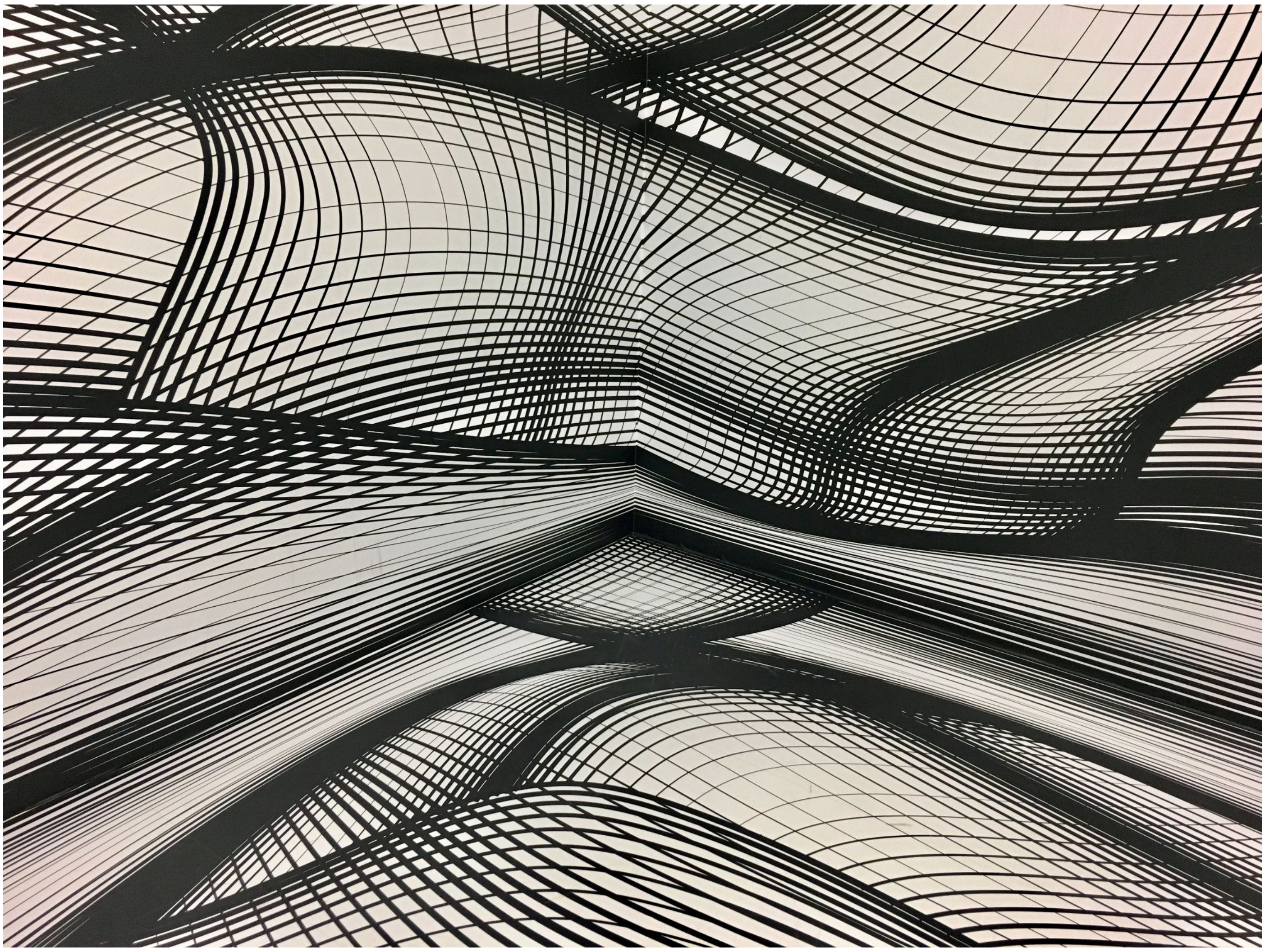
|| *Aum Sri Sai Ram* ||



ProEDA

Final Year Capstone Project

FOR

Bachelor’s of Computer Applications

BY

SAIKRISHNA B

184407

III BCA

Sri Sathya Sai Institute of Higher Learning,

Muddenahalli Campus

A humble offering at the divine lotus feet of our dearest Bhagawan,

who has been with us and guided us throughout the duration of the project.

This is His project.

We are just the instruments he had chosen for doing this.

ACKNOWLEDGEMENTS

Firstly, I would like to offer my most sincere, heartfelt gratitude to the ever present, ever loving Lord, Bhagawan Sri Sathya Sai Baba, without whose grace and blessings, I would not have been able to do this project, or anything else for that matter.

I would like to thank the Deputy Director, Sri Sathya Sai Institute of Higher Learning, for approving the project idea, and his assistance throughout.

I would like to thank my Project Guide, Dr. Sampath Lonka, for his guidance on the project content. His knowledge on the mathematics behind the visualization, and his prompt replies to any doubts raised really helped us during the development phase.

I would like to thank my fellow team members, Sainath Reddy, Parth Maheshwari, and Deepam Rai, for their collaboration and team-working spirit. It was a incredibly fun and eye opening experience working together with them.

In view of the bigger picture, I would like to extend my thanks to the internet community, the people in forums like the vue js forum, and stack overflow, for their help and support, replying to every single question that was raised due to ignorance in the new technologies used in this project.

CERTIFICATE

This is to certify that this project, titled ProEDA, submitted by Saikrishna B, 184407, III BCA, Department of Mathematics and Computer Science, Sri Sathya Sai Institute of Higher Learning, Muddenahalli Campus, is a bonafide record of the original work done under my supervision and guidance as a course requirement for the degree of Bachelors of Computer Applications during the year 2020-2021.

……………..…………………………….

Dr. Sampath Lonka,
Asst. professor,
Dept. of Mathematics and Computer Science,
Sri Sathya Sai Institute of Higher Learning,
Muddenahalli Campus

Place: Chennai, Tamilnadu

Date: 26th April 2021

CONTENTS

1. Introduction
2. Technologies used
    2.1 Vue JS
    2.2 Node JS
    2.3 Express JS
    2.4 MongoDB
    2.5 Google Charts API
    2.6 Papa Parse
3. Development Environment
4. ProEDA
    4.1 Homepage
    4.2 Components
    4.3 Workflow
    4.4 Graphing
5. Code samples
    5.1 File structure of the client side
    5.2 Example of a Graph Component
    5.3 The Router for the top navigation bar
6. More Information Page
7. Extra Tools
8. Project Future Prospects
9. Conclusion
10. References

Introduction

Since time immemorial, man has been looking for a way to bring a visual representation of the facts and information that he collected around him. In this day and age where data is blowing up into insane proportions and data analytics is required very much, the need for visualizing data into clear concise visual formats such as colorful and informative charts and graphs has never been higher.

Thus, this project is named Professional Exploratory Data Analysis (**ProEDA**), a dashboard of tools and options that provide an easy and enjoyable experience of seeing your data come to life. It is very user friendly, and allows you to choose exactly the data variables and trends you want to analyse, whether it is the break up of how you spend your day doing different activities in a simple pie chart, or whether it is the correlation between the age of a patient and his blood pressure in a complex scatter plot.

It allows the user to upload his or her dataset file (.csv) from their computer and start the analysis, giving out the classification of the numerical and factor variables present in the dataset. Then, based on the selection of the variables to be analysed, and the graph options for the visualization, it will then chart the data onto the graph charting area.

This tool is extremely lightweight because it uses heavily optimised technologies in the form of the MEVN(MongoDB, Express JS, Vue JS, Node JS) stack, more details to be provided later in this document.

This project is done as part of the credited paper in the VI semester of the Bachelors of Computer Applications course in Sri Sathya Sai Institute of HIgher Learning, Muddenahalli campus in collaboration with:

- Sai Nath Reddy, 184402, III BCA
- Parth Maheshwari, 184403, III BCA
- Deepam Rai, 184413, III BCA

Technologies Used

1. Vue JS

Vue (pronounced **view**) is a progressive frontend framework based on the Javascript Language. It is open source, and is one of the fastest growing technologies that is becoming popular for developing interactive, beautiful and responsive apps. It is a model-view-view-model (MVVM architecture) framework that is mainly used to build single page applications and futuristic user interfaces.

It was created by Evan You, a senior developer working at Google on Angular, another frontend framework and a product of Google. He wanted to extract the features that he liked in Angular, but to make a more lightweight but feature intensive framework. The first code commit of the project was made in July 2013, in its GitHub repository. It was then released as a public product the following February.

In Vue, everything is regarded as a component. It makes the code really easy to scale, understand or replicate because of its inherent modularity. Each component comes bundled with three facets:

- HTML – the `<template>` tag
- JS – the `<script>` tag
- CSS – the `<style>` tag

Thus, each component is independent, and just has to be embedded in a parent component or such to be used on the page. This doesn't mean the flow of data between components is hindered. That is smoothly carried out by features like event emitting(Child to parent data flow), and props for passing data(Parent to Child data flow).

Advantages:

Simplicity – The purpose behind Vue was to make Component based web development as easy as possible for the devs, and this was achieved. Vue is also very optimized performance wise as the components have very small overheads due to the bundling up of HTML, JS and CSS into one file.

Easy Integration – Vue code can be easily integrated into other technologies like Angular or React. Thus, it can be used to improve existing web apps very easily. Also, as it is based on the MVVM architecture, it can easily handle HTML blocks too.

Community Support - The support for the platform is impressive. For instance, in 2015, every query on the official platform was answered. Similarly, more than 1,400 issues were resolved on GitHub with an average time of less than 13 hours. In 2018, the support continued to impress as every query was diligently answered. In fact, more than 6,200 problems were resolved with an average resolution time of only six hours. To support the community, there are consistent release cycles of updated information. In addition, the community continues to grow and evolve with the backend support of developers.

Drawbacks:

Safe to say, there are a few drawbacks too. For example, the development team has very little support from corporates for the last few years, so Vue has been widely used for personal or small scale projects only. Stability has also been a constant issue since its inception. But this is manageable for small scale applications, as the issues can be resolved quickly.

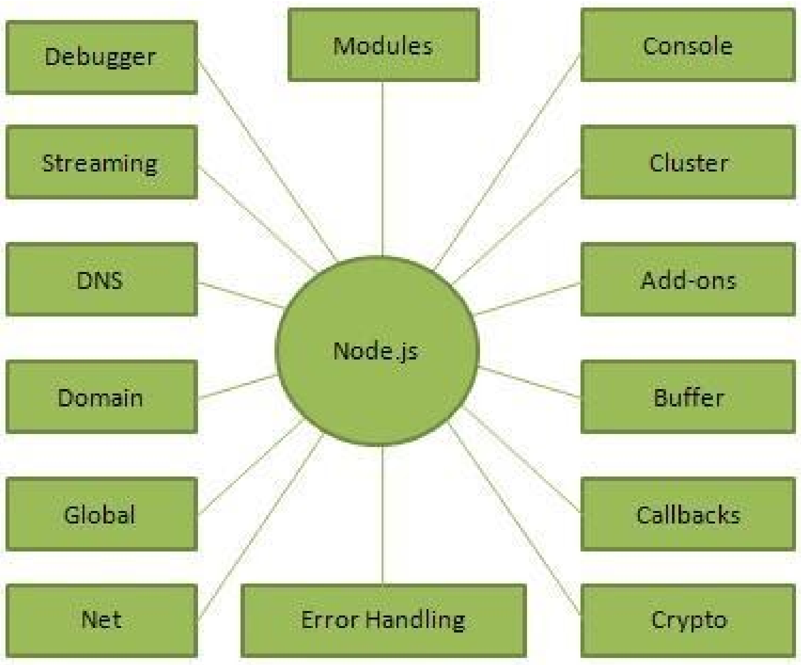
We chose Vue because of its simplicity. The learning curve of the framework was comparatively flatter and it also suited the scale of our application, while providing the benefits of a component based framework.

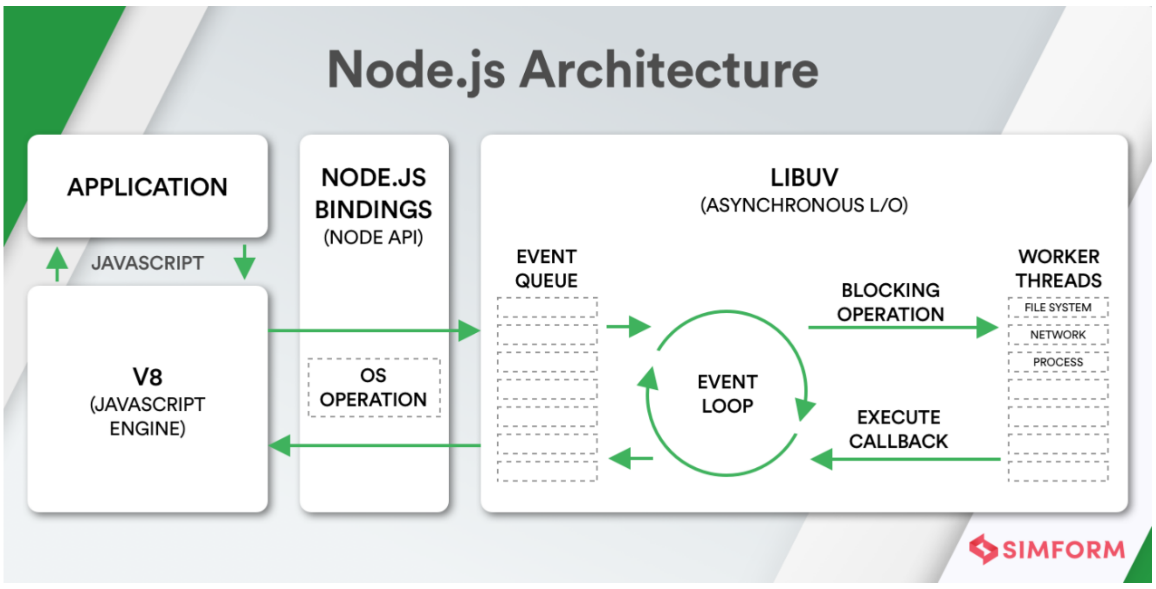
2. **Node JS**

Node JS is an asynchronous, event-driven, server side platform, based on Google Chrome's Javascript Runtime, that can be used to build heavily scalable applications easily. It is one of the main components of most Javascript based full web stacks. It is largely popular due to the availability of extensive and exhaustive Javascript modules that simplify the building of the connecting platform between the server and the frontend client.

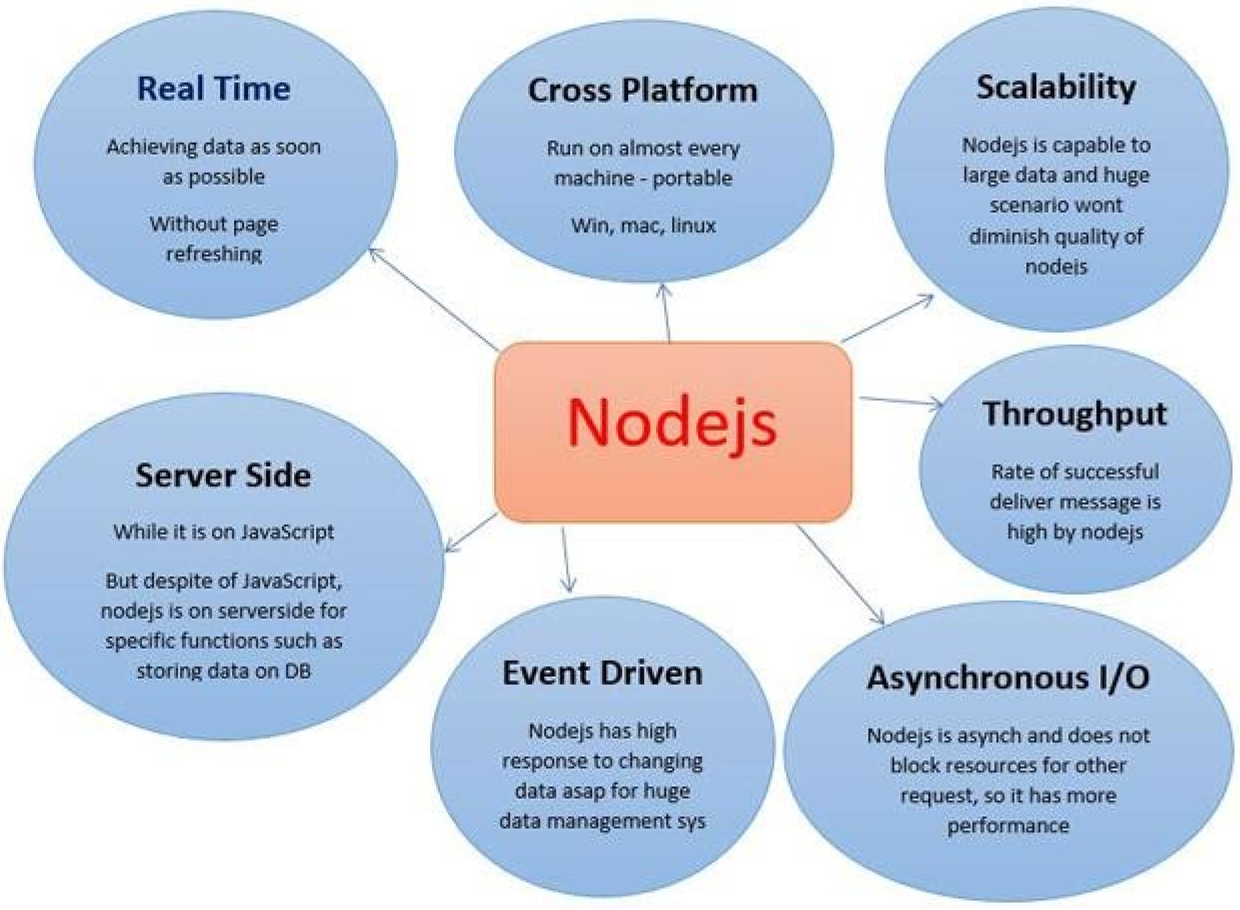
It is blazing fast because it is asynchronous. It never waits for an API call to return the requested data. It executes it and moves on to the next call, using its callback system to capture the return of the previous call. It is also heavily optimized because it uses a single threaded event based system. Thus no deadlocks can be caused in the OS, unlike many other server platforms which use the multithread based concurrency system.

Node JS is used extensively on many commercial and heavy corporate projects such as, eBay, GoDaddy, Yahoo!, Uber, PayPal, to name a few. Its reliability and widespread popularity can be gauged based on its usage in these giants' applications.





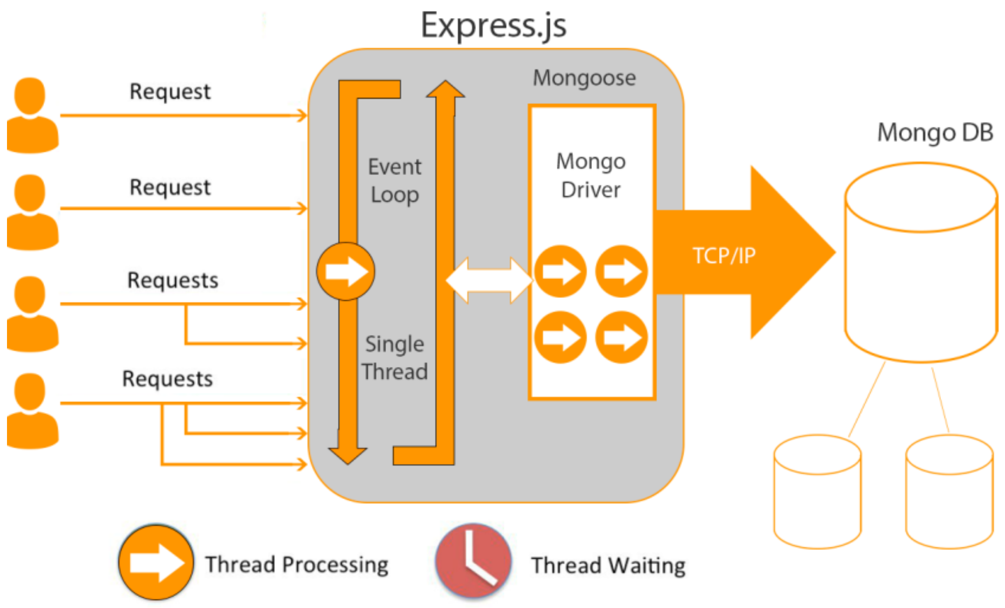
One of the biggest offerings of Node Js is its package management system, **npm**. It is one of the largest package library managers out there, and there are millions of absolutely free packages offered with regular updates and a slew of features. It is a main staple of full stack developers as it allows easy integration of libraries and other tools into the app.


Nodejs
Real Time
Achieving data as soon as possible
Without page refreshing
Cross Platform
Run on almost every machine - portable
Win, mac, linux
Scalability
Nodejs is capable to large data and huge scenario wont diminish quality of nodejs
Throughput
Rate of successful deliver message is high by nodejs
Server Side
While it is on JavaScript
But despite of JavaScript, nodejs is on serverside for specific functions such as storing data on DB
Event Driven
Nodejs has high response to changing data asap for huge data management sys
Asynchronous I/O
Nodejs is asynch and does not block resources for other request, so it has more performance

3. **Express JS**

Express JS is a minimal and flexible Node JS application that provides robust features for web and app development. It is a very thin layer that offers fundamental development libraries without obscuring the features of Node JS itself. It is also very light, thus heavily improving performance. It is used to build APIs and offers easily useable middleware to the full stack.

These days, wherever Node JS is used as a server side platform, Express JS is often used as the middleware. It is really easy to work with Express and to make it handle the connection between the Client side requests and the server side responses.



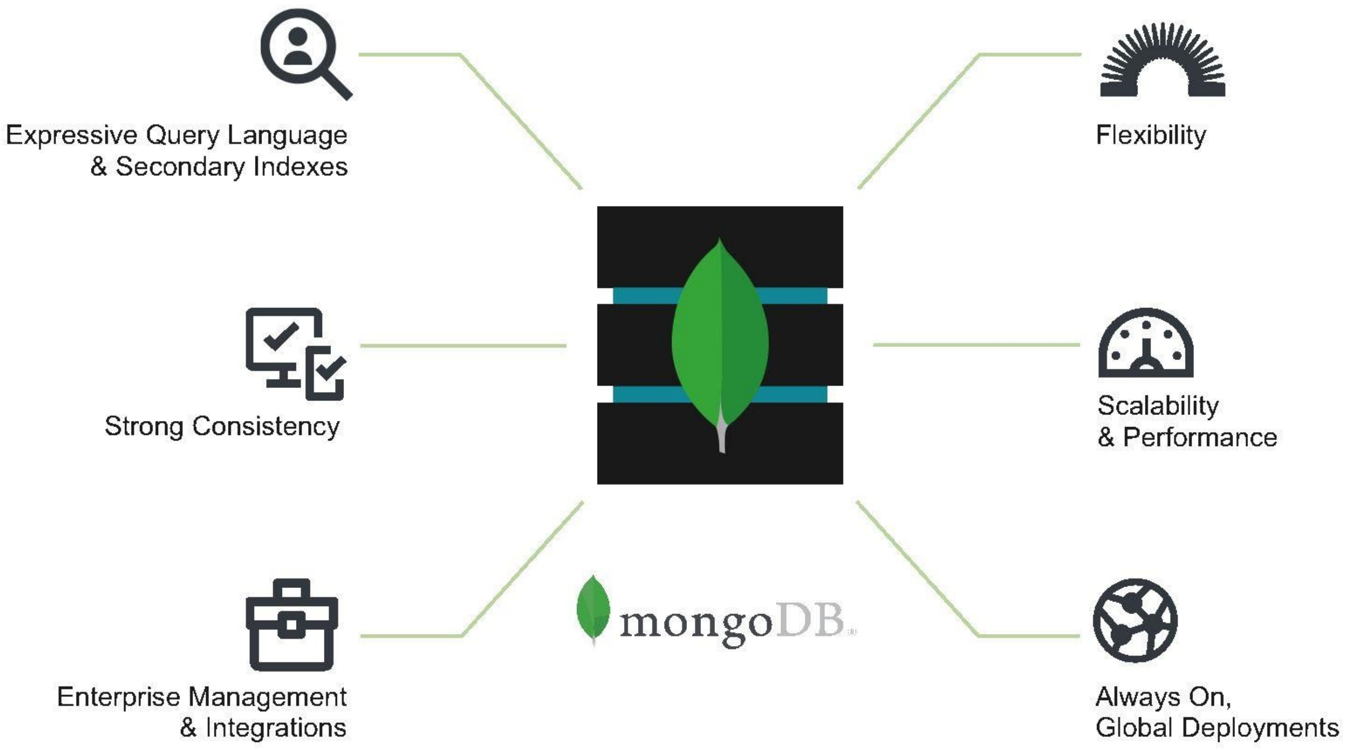
4. MongoDB

MongoDB is a general purpose, document-based, distributed, NoSQL database built for modern application developers. These days, most developers are used to thinking in terms of objects, its properties and data. This DB stores its records in the collections in a similar manner, using JSON. This makes it so easy and comfortable for programmers to send and receive the data from the DB.

A NoSQL database(*not SQL, not only SQL, etc.*) is a non relational database, meaning the data isn't stored in relations, a.k.a rows/columns format. This type of database gives maximum flexibility to the developer. Developer Comfort became a higher priority than storage space since the late 2000s, due to reducing costs in storage mediums. Thus, the interest in NoSQL databases started to increase.

People think that since there are no row column relations between the data, there is no storage of data of the relationships between them. This is wrong. They just store it in a different way. Many actually prefer NoSQL to relational databases in terms of relationships, because the data doesn't have to be split in between the tables like primary and secondary keys.

Another advantage is the feature of cloud sync. The MongoDB Atlas cloud database service offers maximum comfort to developers, as there doesn't have to be any local database or server service running to connect to the DB. The connection happens through a URL provided by the cloud service according to your IP and it is then used by Express JS and Node JS to connect to the URLString.


Expressive Query Language & Secondary Indexes
Flexibility
Strong Consistency
Scalability & Performance
Enterprise Management & Integrations
mongoDB
Always On, Global Deployments

5. **Google Charts API**

The main mechanism of the whole project, the charting of the selected data and the graph type is done via the google charts API. The Google charts API is a rich collection of HTML 5 Canvas based charting tools that are really easy to use and modify to need. They take in data in a very flexible manner. The key feature that sets this API apart from the countless other Charting APIs are the sheer number of chart types supported. There is even support for making your own custom chart type.

The charts are responsive by nature and come with animations bundled in stock. The numerous chart options make customizing the charts output down to the exact details that are required for the exact application very easy. Another key feature of the API is the level of the documentation. The forums and the community around it is very active and supportive for newbie users.

Though this API could've been integrated with the **Vue** frontend using just vanilla Javascript, there was a vue wrapper for the charts API found on GitHub called vue-google-charts, which allowed the Charts to be mounted to the template section of the parent component in the form of a child component. The `chartData` and the `chartOptions` could then be passed to the component in the form of props and its own native data in the `data()` section of the `<script>` part of the parent component.

6. **Papa Parse**

   Papa parse is a JS library that allows easy parsing of CSV files into formats required for data analysis by the app, formats that are more conducive for the devs to extract the data from the dataset and analyse it.

   Initially, the parsing had been implemented from ground up natively, but it wasn't proper and done to a level enough to integrate with the Google Charts API. Thus, in search of a good parser, this library was found. It gives numerous options to customize the options in parsing, such as reading the first row as the table headers(as usually the first column contains the data variables), selecting the delimiter used in the dataset, whether the output parsed data should be an array of arrays or JSON objects. These options really helped to bring the parsed data to a level of easily importing into the charts API.

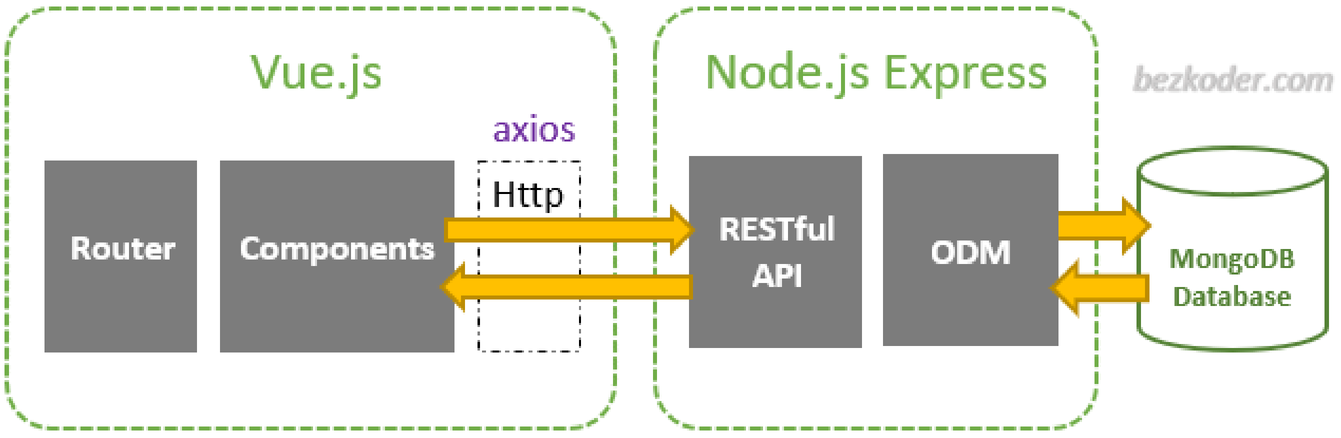
   This too could've been implemented as vanilla Javascript but was done by a vue wrapper found on GitHub called vue-papa-parse.

Many of these technologies that were used were gibberish to us when we first started the project development. Slowly through the help of the internet community, the expertise shared through endless tutorials in the form of YouTube videos or Blog posts, and our own interest to make this project better with newer technologies, we were able to integrate these technologies fairly well. That said, there are many places that we can use these tools to further improve our project. The exposure that we gained from using these full stack tools to program this app is invaluable as, though carried out in a very small scale and with questionable formal software engineering practices, this is industry-related and industry-relevant experience that can be carried forward and onward.

Thus, all the technologies put together, the basic four – Vue JS, Express JS, Node JS, and MongoDB – form what is commonly called, the VENM stack. This contains a frontend framework, a backend Database, a server side platform, and a middleware connecting API.

Some other popular stacks used are MERN(React JS instead of Vue), MEAN(Angular JS instead of Vue), and LAMP(Linux, Apache, MySQL, Perl/PHP/Python).

One of the major advantages of the VENM, MERN or MEAN stacks are that they use Javascript end-to-end in the stack. This makes life extremely comfortable for web developers as they do not have to struggle with learning and switching to and from different languages, and with them, their thinking processes and syntax issues.



Development Environment

- **Hardware:**

  -> Dell Inspiron 3505 laptop

  -> RAM: 8 GB

  -> ROM: 512 GB

  -> Processor: AMD Ryzen 5 3500U, Quad core, Radeon Vega 8 Graphics
- **Software:**

  ->OS: Ubuntu 20.04 Focal Fossa

  ->Code Editor: VS Code 2019

  ->Package Manager: npm

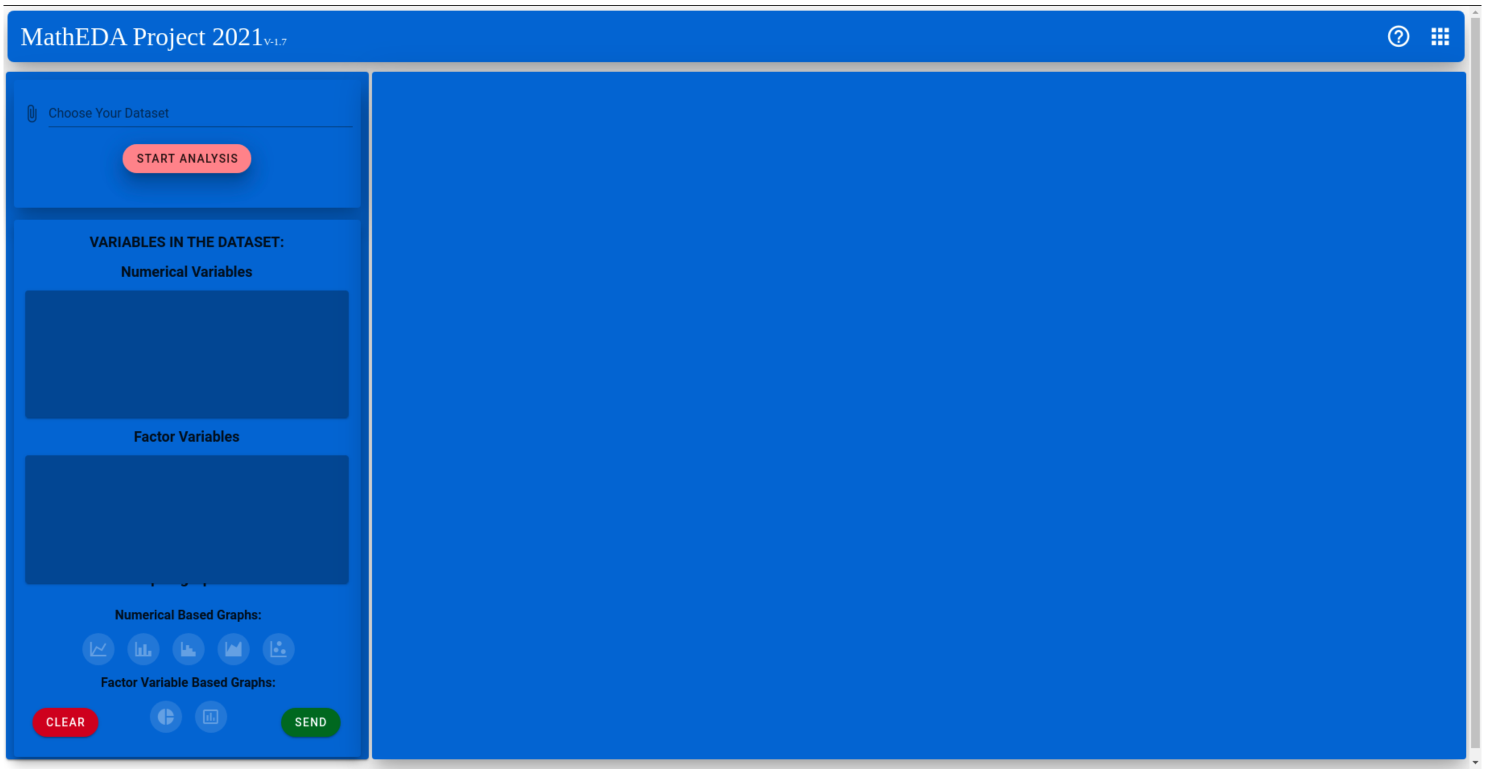
  ->Browser: Google Chrome

  ->REST API Pseudo Client for testing Backend: Postman, Insomnia

Ubuntu was used as it was easy to use npm and the bash terminal easily for installing packages and running the development server for local testing. WSL with Windows 10 could've been used to achieve the same bash shell capabilities, but it was too much hassle to set the permissions for the bash shell there to install the packages and later use them in development.

ProEDA

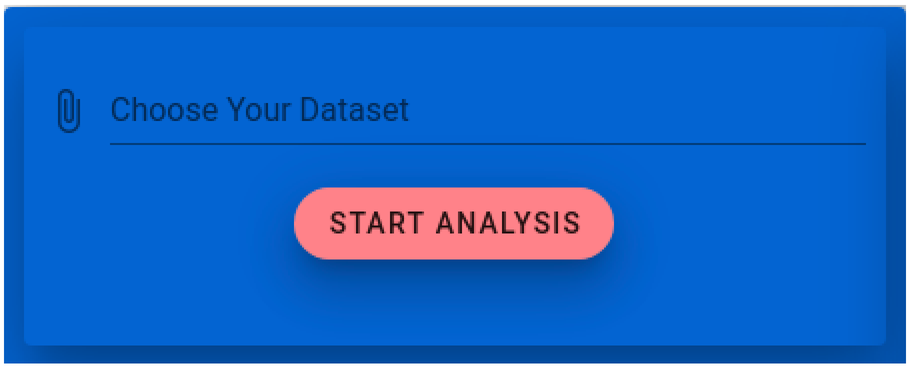
Homepage:



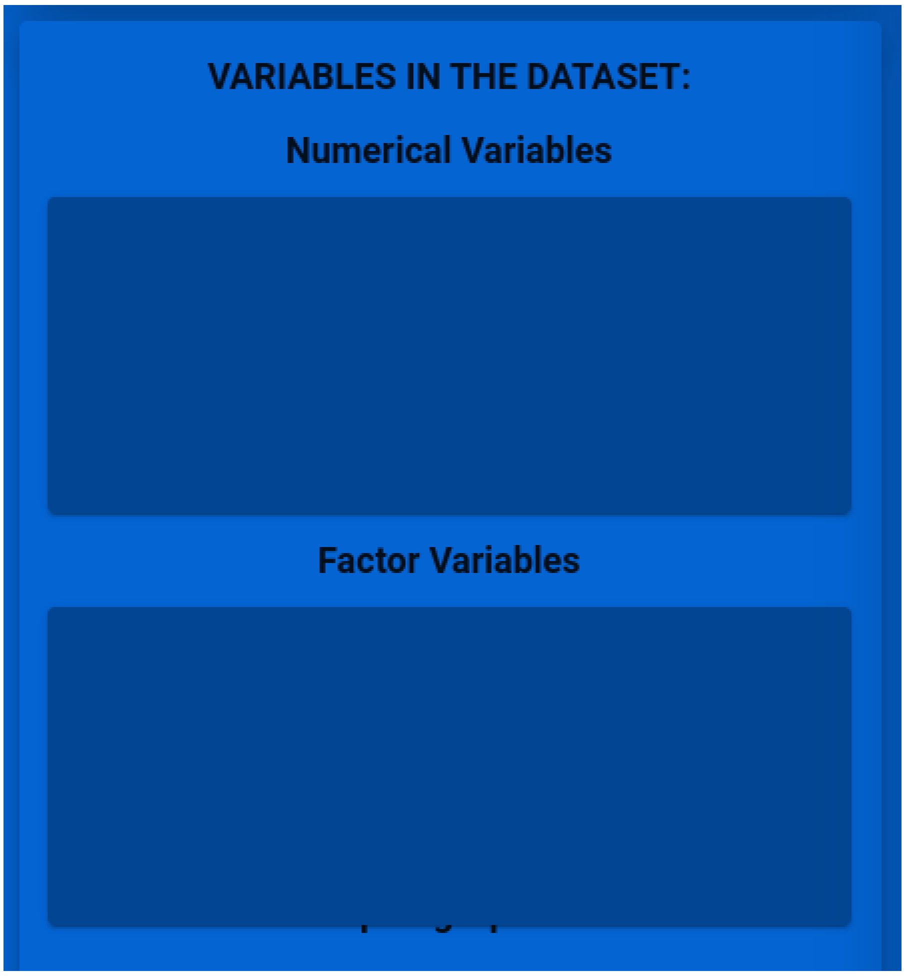
This is the screen that greets the user as the app is loaded. It is a very presentable, no frills, clean interface that is intuitive for the user to use. On the left bar, you have the dataset input area, the variable selector and the graph selector.

Components:

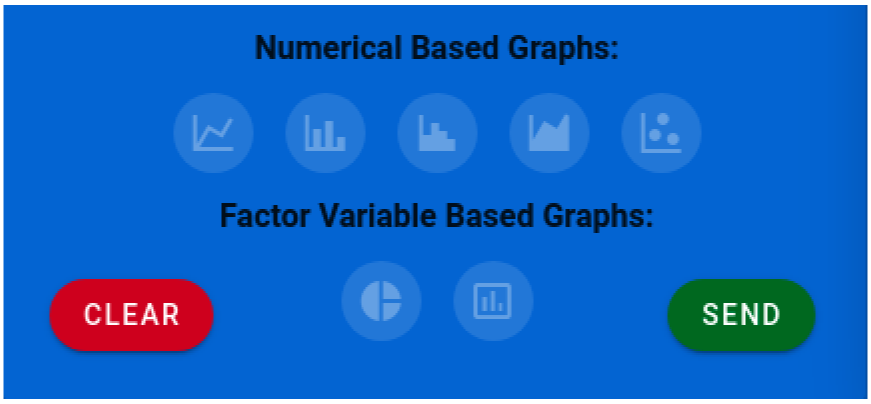
Dataset Input section (from local computer):



Variable Selector Section:



Graph Selector Section:

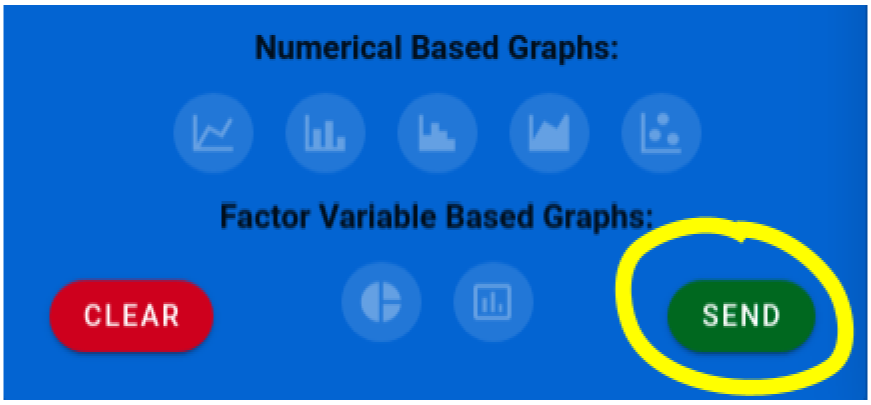


Workflow:

The workflow for the user to graph the data is as follows:

1. Select the Dataset from the machine. When the “Choose your dataset” option is clicked, it will open up a file explorer to select the .csv file from the computer. This is then loaded to the browser.

2. Click the “Start Analysis” button. This will then in the background emit an event that sends this file to the csv file parser (Papa Parse JS library). The file will be parsed, aking the delimiter to be commas, and the first row to be the variables. The parsed data will be then sent back as an Array of arrays, with each sub array representing a row of the dataset. The data value can be accessed by using the row number, and the index of that particular variable as the column number.
3. Choose the required Variables. Once the parsed data reaches the app, it will segregate the variables into numerical ones and categorical ones, and they will be displayed appropriately. Two numerical variables can be selected at once(for the scatter plot), but only one factor variable can be selected at once. According to the variables chosen, the graph options will be enabled or disabled.
   - If one numerical variable is selected, the line, bar, column and area charts will be enabled.
   - If one factor variable is selected, the pie and histogram chart options will be enabled.
   - If two numerical variables are selected, the factor variable selection is not considered and the scatter plot option will be enabled.
4. **<u>Click the “SEND” button BEFORE SELECTING THE GRAPH TYPE.</u>** This lets the app know that the variables have been selected and that the graph can be plotted by selecting the graph type. If the **“SEND”** button is **NOT CLICKED,** the app will not recognise the selection of the variables.



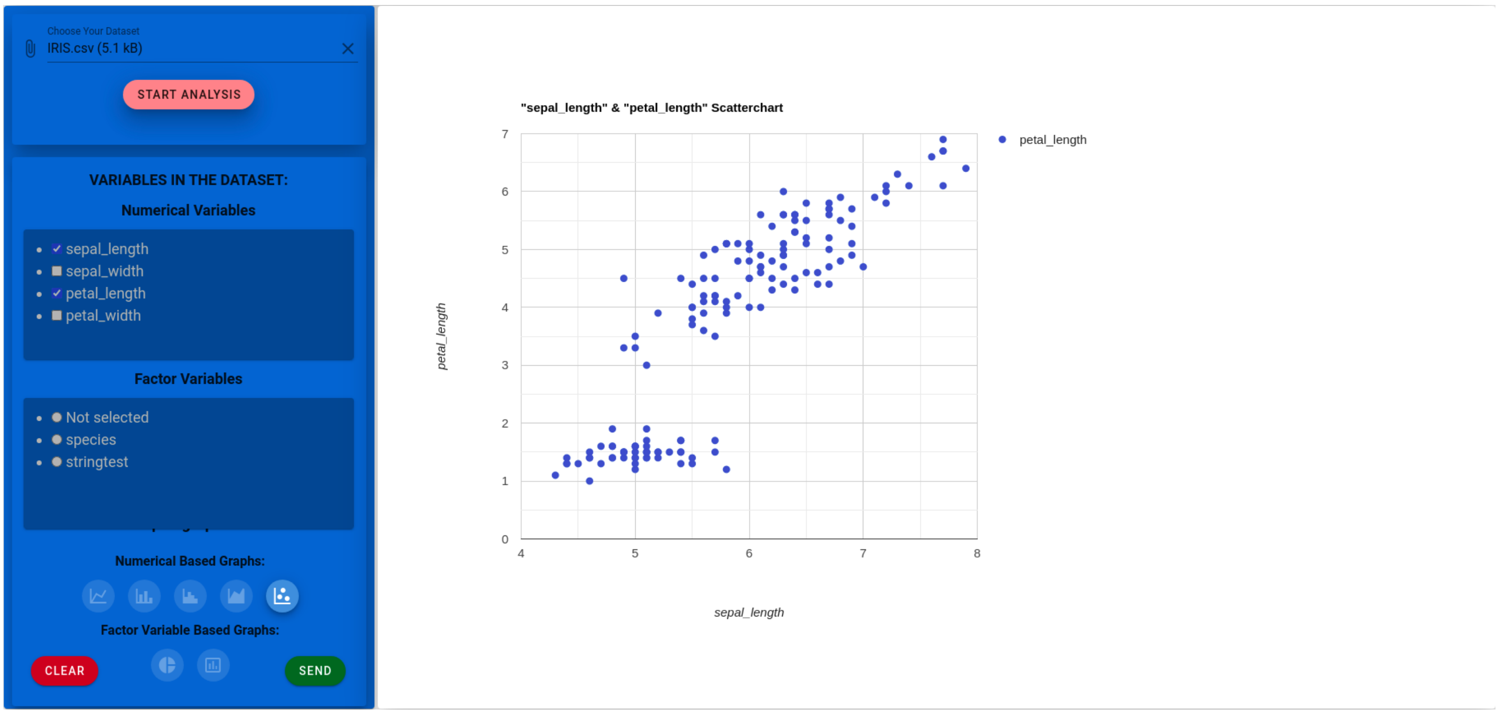
5. Choose the graph type. When the mouse cursor is hovered on the type bubbles, their names will appear in the tool tips.

6. Use the “CLEAR” button to clear the graphing area.

*****NOTE***:**

IF YOU WANT TO USE THE DATASET AGAIN, PLEASE REFRESH THE PAGE AND INPUT THE DATASET FRESHLY. DO NOT INPUT ON TOP OF AN EXISTING DATASET. THIS IS DUE TO TECHNICAL REASONS IN THE BACKEND.

Example: Scatter plot between the sepal_length and the petal_length in the IRIS dataset



Hovering on the data point in the graph, irrespective of the chart type, will give you details of that data point.

Graphing:

The graphing was implemented using the Google charts API. Though this could’ve been implemented directly using vanilla JS, it was implemented using a vue wrapper called vue-google-charts. This wrapper allowed the chart to be embedded in the parent component as a child component.

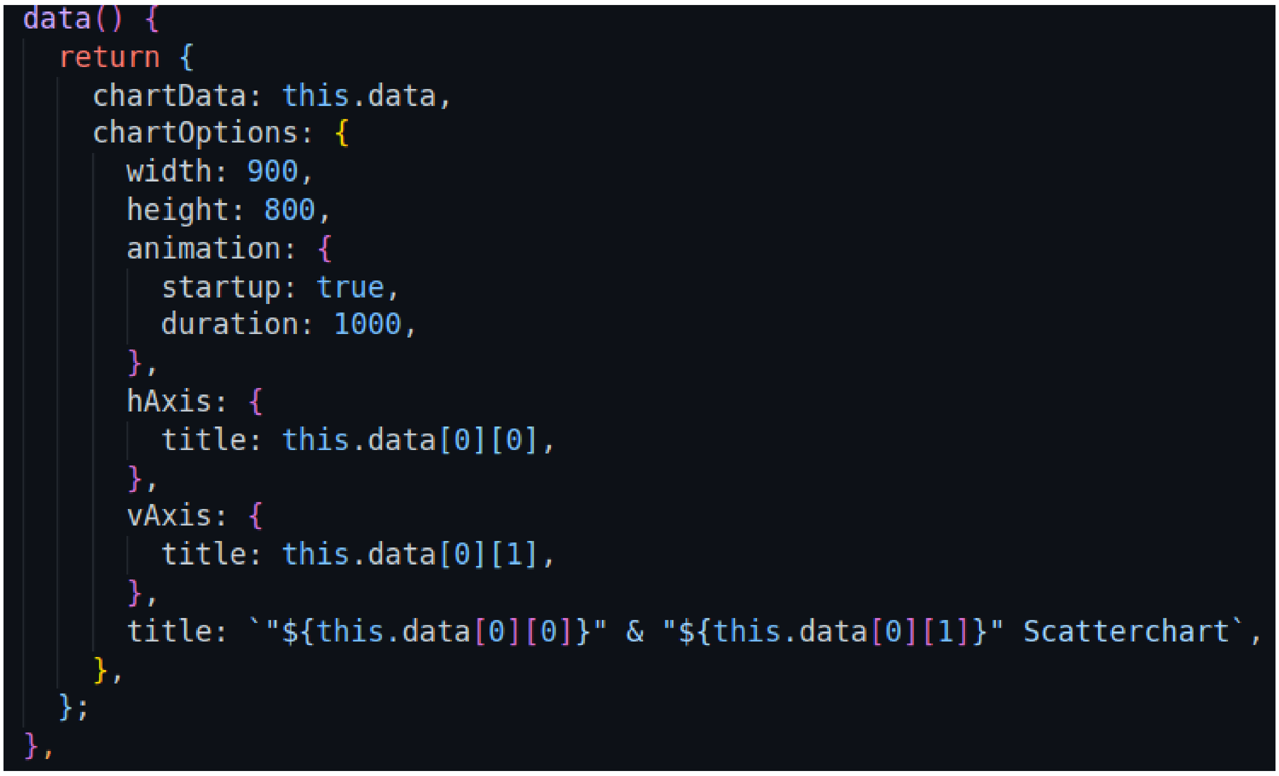
```
import Vue from 'vue'
import VueGoogleCharts from 'vue-google-charts'

Vue.use(VueGoogleCharts)
```

```
import { GChart } from 'vue-google-charts'

export default {
  components: {
    GChart
  }
}
```

The component was embedded like this:

```
  <GChart
    type="ColumnChart"
    :data="chartData"
    :options="chartOptions"
  />
```

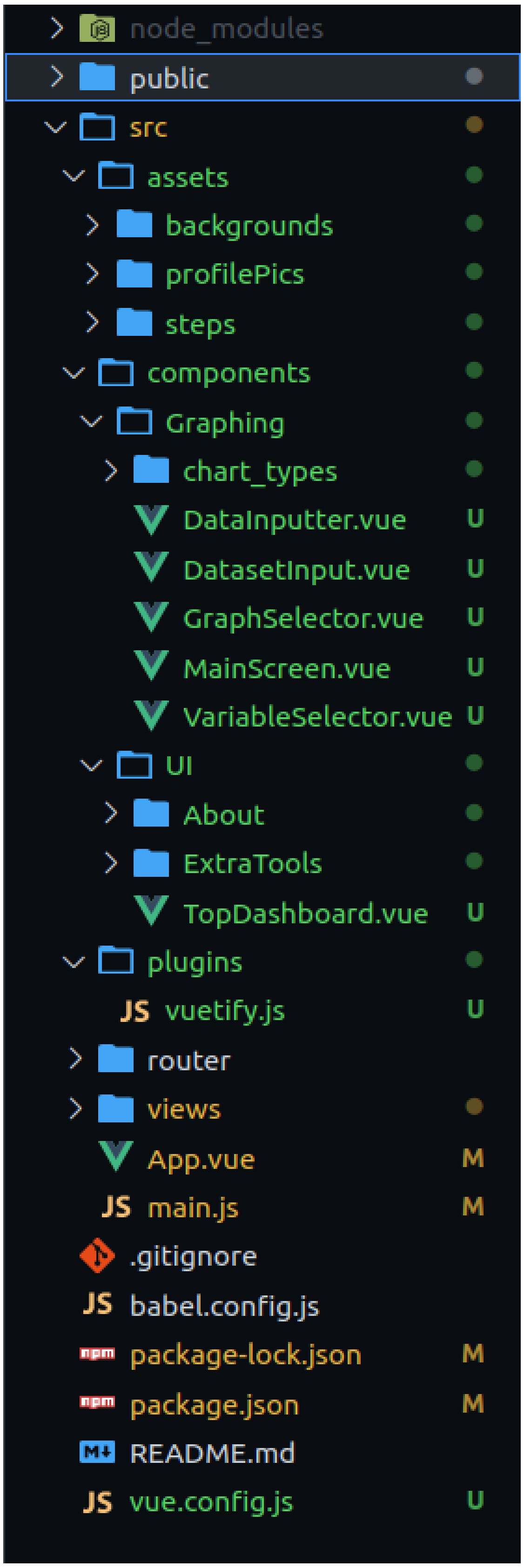
The chartData and the chartOptions were updated in the graph component's data() attribute:



```
data() {
  return {
    chartData: this.data,
    chartOptions: {
      width: 900,
      height: 800,
      animation: {
        startup: true,
        duration: 1000,
      },
      hAxis: {
        title: this.data[0][0],
      },
      vAxis: {
        title: this.data[0][1],
      },
      title: `"${this.data[0][0]}" & "${this.data[0][1]}" Scatterchart`,
    },
  };
},
```

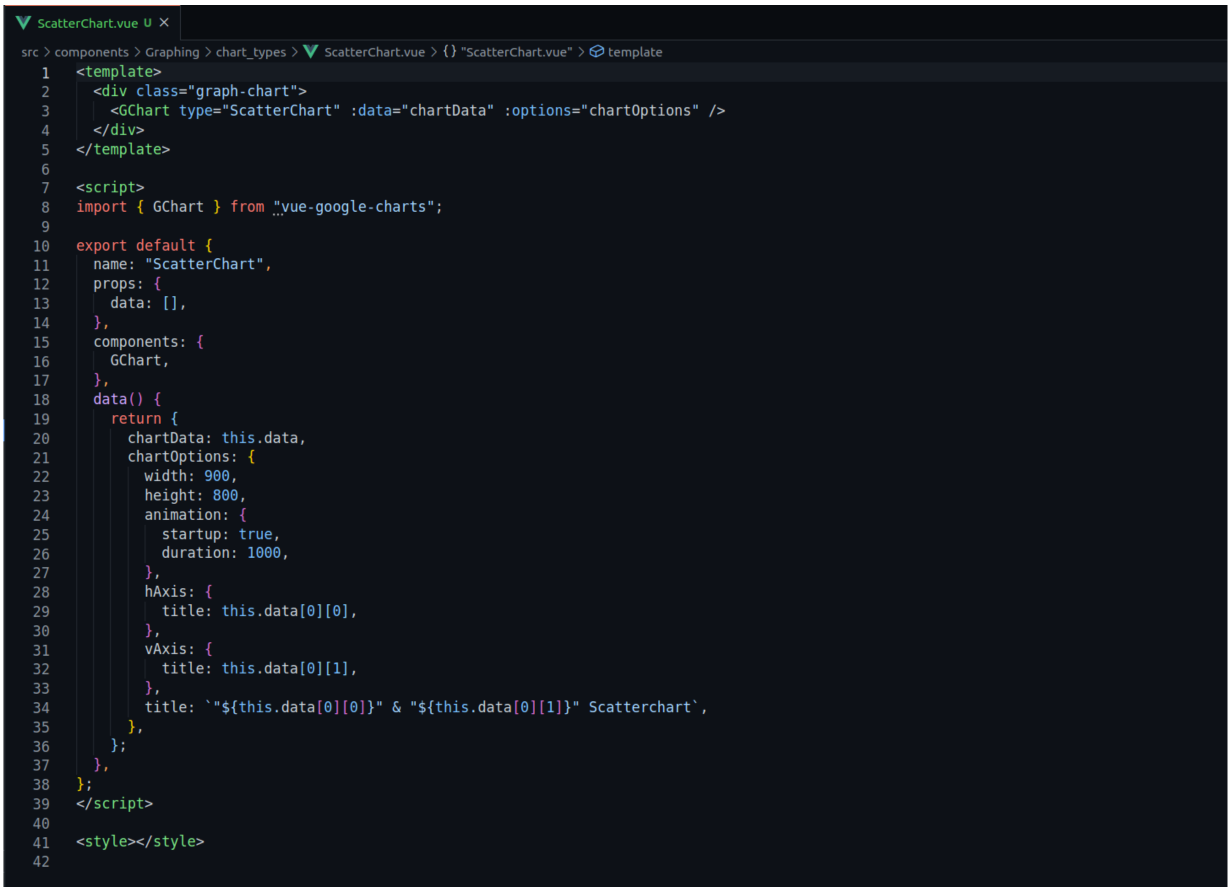
Code Samples:

File structure of the Client side:



```
> node_modules
> public                      ●
v src                         ●
  v assets                    ●
    > backgrounds             ●
    > profilePics             ●
    > steps                   ●
  v components                ●
    v Graphing                ●
      > chart_types           ●
        DataInputter.vue      U
        DatasetInput.vue      U
        GraphSelector.vue     U
        MainScreen.vue        U
        VariableSelector.vue  U
    v UI                      ●
      > About                 ●
      > ExtraTools            ●
        TopDashboard.vue      U
  v plugins                   ●
      vuetify.js              U
  > router
  > views                     ●
    App.vue                   M
    main.js                   M
  .gitignore
  babel.config.js
  package-lock.json           M
  package.json                M
  README.md
  vue.config.js               U
```

Example of a graph component:

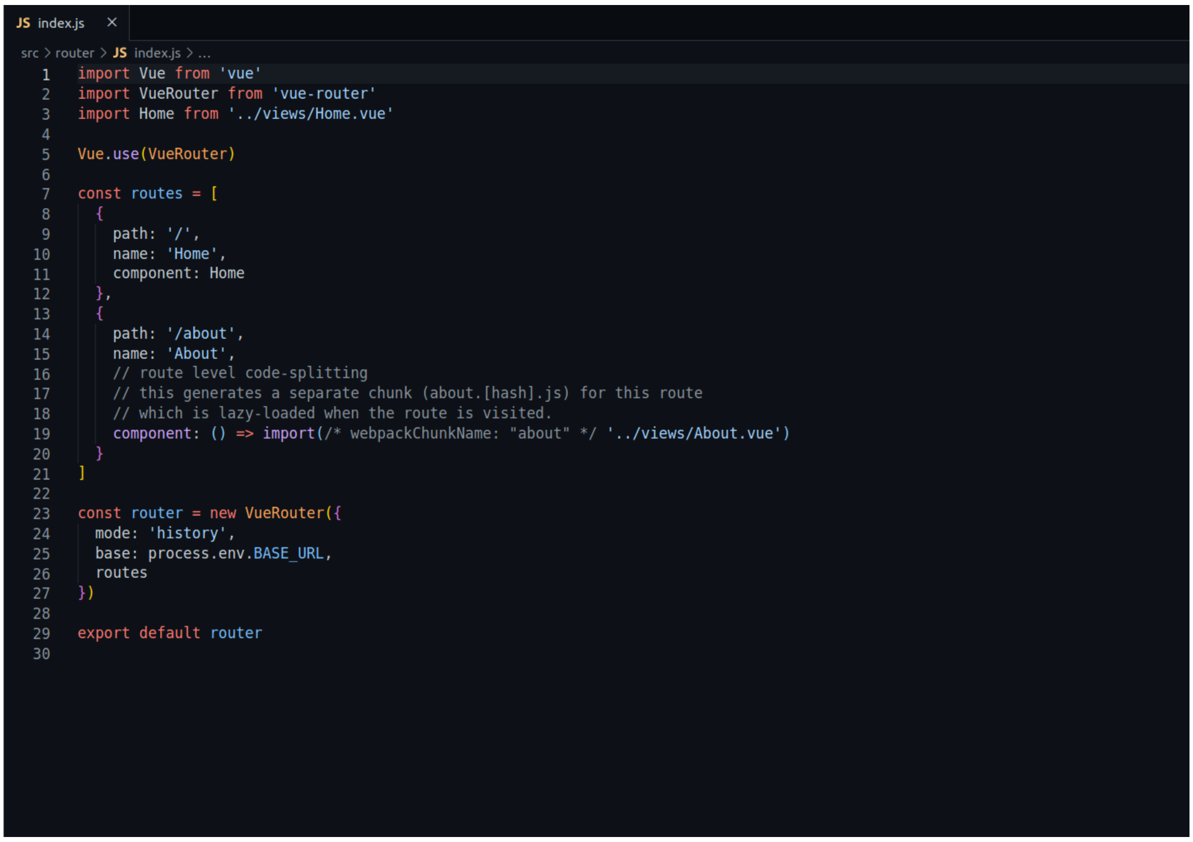


ScatterChart.vue U ×

src > components > Graphing > chart_types > ScatterChart.vue > {} "ScatterChart.vue" > template

```
<template>
  <div class="graph-chart">
    <GChart type="ScatterChart" :data="chartData" :options="chartOptions" />
  </div>
</template>

<script>
import { GChart } from "vue-google-charts";

export default {
  name: "ScatterChart",
  props: {
    data: [],
  },
  components: {
    GChart,
  },
  data() {
    return {
      chartData: this.data,
      chartOptions: {
        width: 900,
        height: 800,
        animation: {
          startup: true,
          duration: 1000,
        },
        hAxis: {
          title: this.data[0][0],
        },
        vAxis: {
          title: this.data[0][1],
        },
        title: `"${this.data[0][0]}" & "${this.data[0][1]}" Scatterchart`,
      },
    };
  },
};
</script>

<style></style>
```

The Router for the top navigation bar:



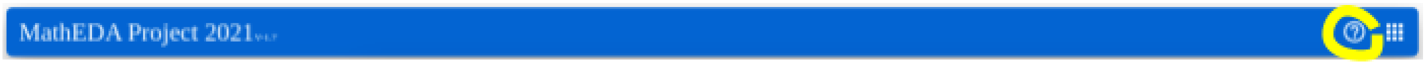
index.js ×

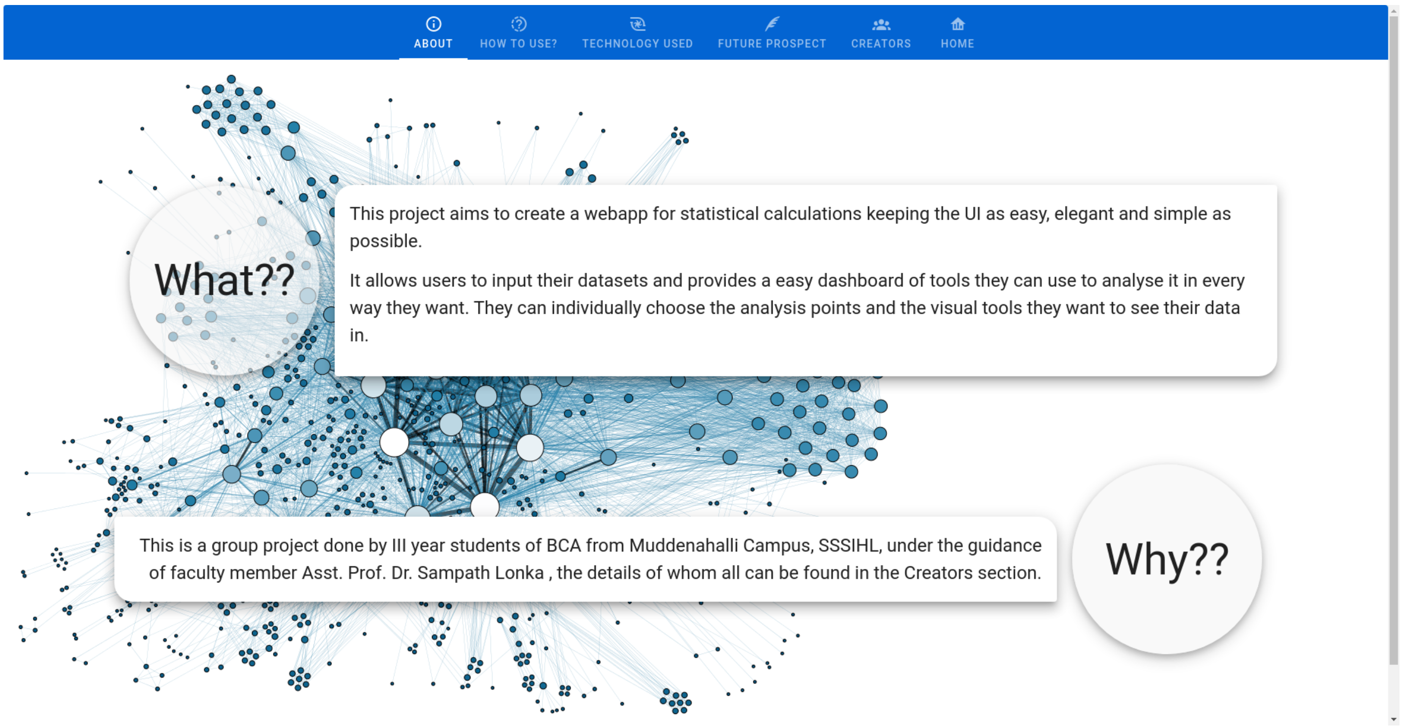
src > router > index.js > ...

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/Home.vue'

Vue.use(VueRouter)

const routes = [
  {
    path: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/about',
    name: 'About',
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () => import(/* webpackChunkName: "about" */ '../views/About.vue')
  }
]

const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes
})

export default router
```

More information page:

This page contains subpages that give more information about this project, its purpose, its creators, a tutorial on how to use it, upcoming feature updates, etc. It can be clicking on the ? symbol button on the top navigation bar in the right corner next to the options button.

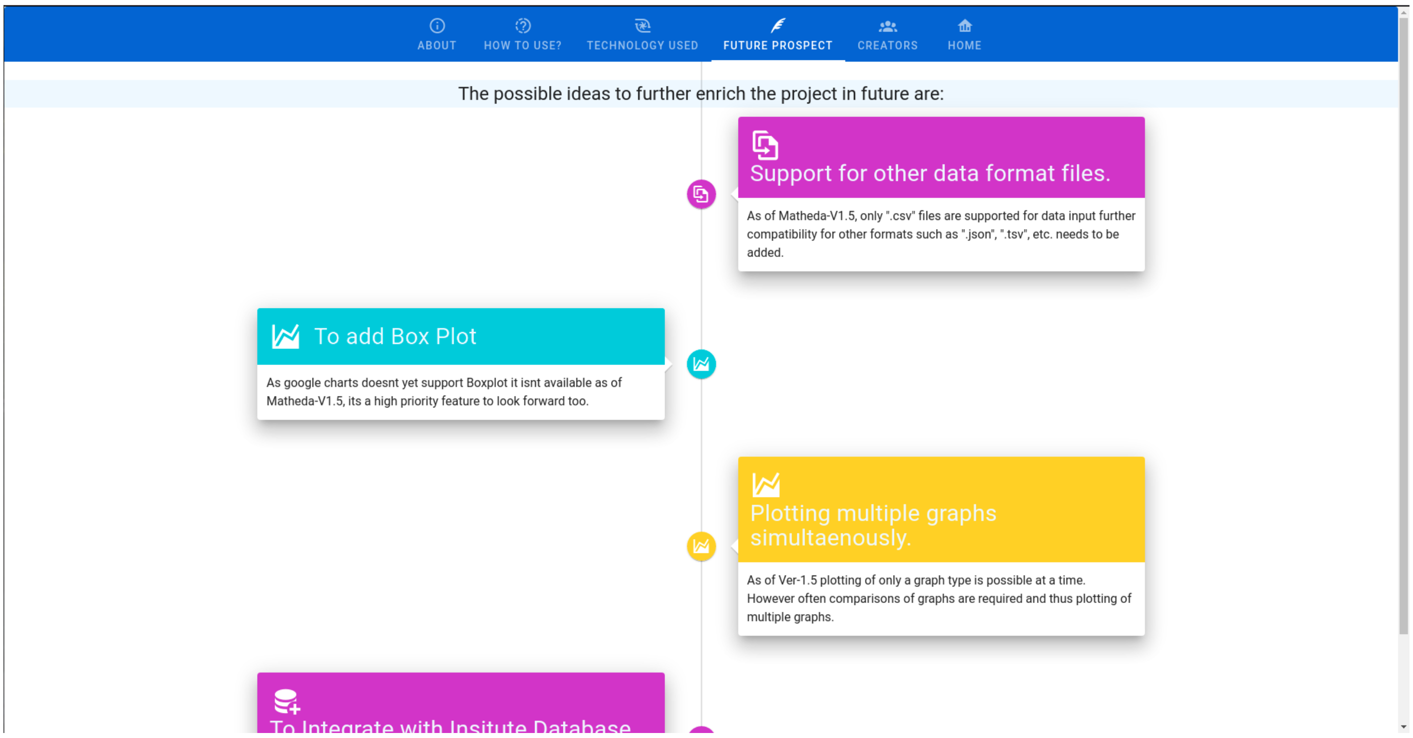




This project aims to create a webapp for statistical calculations keeping the UI as easy, elegant and simple as possible.

It allows users to input their datasets and provides a easy dashboard of tools they can use to analyse it in every way they want. They can individually choose the analysis points and the visual tools they want to see their data in.

This is a group project done by III year students of BCA from Muddenahalli Campus, SSSIHL, under the guidance of faculty member Asst. Prof. Dr. Sampath Lonka , the details of whom all can be found in the Creators section.



The possible ideas to further enrich the project in future are:

Support for other data format files.

As of Matheda-V1.5, only ".csv" files are supported for data input further compatibility for other formats such as ".json", ".tsv", etc. needs to be added.

To add Box Plot

As google charts doesnt yet support Boxplot it isnt available as of Matheda-V1.5, its a high priority feature to look forward too.

Plotting multiple graphs simultaenously.

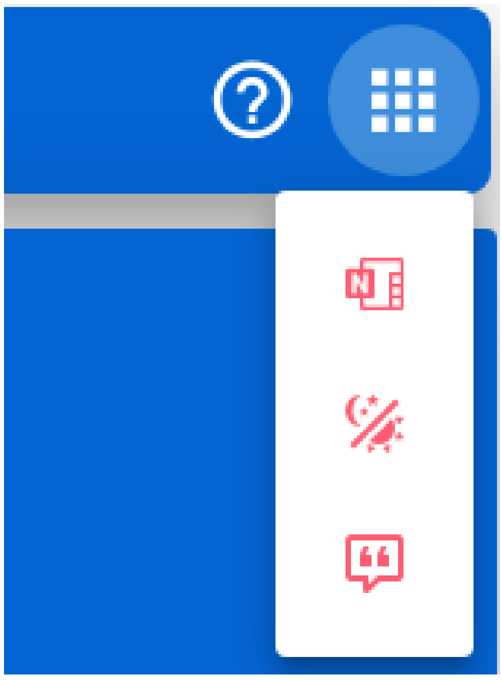
As of Ver-1.5 plotting of only a graph type is possible at a time. However often comparisons of graphs are required and thus plotting of multiple graphs.

To Integrate with Insitute Database

Extra Tools:

Apart from the graphing mechanism, some other tools are given for the user's convenience. These are:

- Theme selector: There are many themes available. The user can select the color scheme of his own choice. This gives the app a certain degree of personalization and vibrance.
- Notes Taker: This is a useful utility for the user to take some notes, maybe jot some things that pop up in his mind. This feature being inbuilt allows the user to stay on the same page, instead of using another app for taking notes.
- Feedback support: The user can enter his details and his feedback. This would then be, on clicking "submit", be forwarded to the backend to store the feedback. The administrators of the app have a separate page that allows them to view the feedback submitted.



Project Future Prospects

This is not the end of the road for this project. There is so much scope and potential to keep developing this project to make it more user friendly, more feature intensive, better integrated with the system in our campus.

One of the main scope areas is the inclusion of a login system based on the existing login style of Regn number and password. The student can have more customization options and preference saving. He can also have the facility of storing analysis charts for later use.

Another scope area is the inclusion of a blog for the app. It can contain articles pertaining to data analysis and data visualization, based on the emerging trends in this every growing modern field.

Moreover, other graph types could also be included, so that there is no dearth of options for any use case that the user might need the dashboard app for.

One more feature that could be added is the inclusion of graph options that is currently not offered to the user. The options for the graphs and the charts are hardcoded by the development team currently. In the next version, even that dimension of flexibility could be given to the user to use.

Last but not the least, a main feature that could be added is the inclusion of support for uploading the dataset file from Google Drive or Dropbox, as these are the platforms used by many people to store their data online for access anywhere.

Conclusion

This project isn't without its own limitations. The UI could be further improved for easier user usage. The charting could be further optimized for more clarity and better performance with regard to big datasets. But this is a humble attempt to make a full stack JS web application that makes data visualization simpler for the user, with holistic goodness. The project will be further developed and supported and many more better versions will be put out in the future that keeps on this basic idea of a Professional Exploratory Data Analysis (ProEDA) dashboard app.

References

The following references are for the benefit of those who are reading this document, and want to:

- know more about the technologies used in this Full Stack web app
- know more in detail to add improvements to this app
- maintain this project in the future

Vue JS - https://vuejs.org/v2/guide/

Node JS - https://nodejs.org/dist/latest-v14.x/docs/api/

Express JS - https://expressjs.com/en/4x/api.html

MongoDB - http://mongodb.github.io/node-mongodb-native/3.6/api/

Google Charts - https://developers.google.com/chart/interactive/docs/quick_start

vue-google-charts - https://www.npmjs.com/package/vue-google-charts

Papa parse - https://www.papaparse.com/docs

vue-papa-parse - https://www.npmjs.com/package/vue-papa-parse

In addition to this, some other blogs, resources, forums and youtube channels that were used to refer to or learn the technologies used are mentioned below:

Traversy Media YT channel - https://www.youtube.com/user/TechGuyWeb

Web Dev Simplified YT Channel - https://www.youtube.com/channel/UCFbNIlppjAuEX4znoulh0Cw

The coding train YT Channel - https://www.youtube.com/channel/UCvjgXvBlbQiydffZU7m1_aw

CSS Tricks Site - https://css-tricks.com/

And most importantly, the two tools that are indispensable to any programmer or developer,

Google - https://www.google.com/

Stack Overflow - https://stackoverflow.com/