184413
Deepam Rai
Pro-EDA
Project Documentation
III year BCA
Group Project: 2020-21

Done By,
Saikrishna B.
Parth Maheswari
Sainath Reddy
Deepam Rai

Guide:
Dr. Sampath Lonka,
Asst. Proffessor,
Mathematics and Computer Science,
SSSIHL

Dedicated to our Beloved Mother Sai....

ACKNOWLEDGEMENT

We are grateful to the institute for incorporating such helpful and mind expanding project in our curriculum. We express our heartfelt gratitude to our guide Dr. Sampath Sir for guiding us through this project and also we thank our senior brothers Raghava and Akhil Sai for giving us the advices as to how to go with the project and also their project demonstration which is akin to ours. And we are grateful to all whose name we missed but who directly or indirectly helped us to bring this project to life.

We are forever indebted and grateful to the Creator of Universe, our beloved mother Sai.

CERTIFICATE

Sri Sathya Sai Institute of Higher Learning
(Deemed to be University)
Department of Mathematics & Computer Science

This is to certify that this project report entitled Pro-EDA being submitted by Saikrishna B, Sai Nath Reddy, Parth Maheswari and Deepam Rai III year Bchelor of Computer Applications is a record of bonafide project work carried out by them under my supervision and guidance during the academic year 2020-21 in the Department of Mathematics and Computer Science, Sri Sathya Sai Institute Of Higher Learning, Muddenahalli campus.

Place: .....
Date: .....

.....
Dr. Sampath Lonka
Project Guide

CONTENTS

1. Pro EDA

Pro-EDA is a group project done by III year students of BCA, MDH Campus, SSSIHL under the guidance of Dr. Sampath Lonka, Asst. Professor of Mathematics and Computer Science at MDH Campus, SSSIHL. This project aims to create a web-app with easy user interface where the user can provide a datafile and get the relevant plots based upon different parameters of the data.

In the web-app we are able to supply a datafile from which the numerical and categorical variables are extracted and all of which can be appropriately selected and drawn into the desired graphs. Various graphs such as area chart, bar chart, histogram, pie chart, line chart, column chart and scatter plot.

Other than the charting graph it also features extra tools to help the users such as notes, feedback mechanism and themes.

The user can also see the guide page which shows how to use the web-app in the about page sections. And there only the other informations of the project are also given.

2. TECHNOLOGIES USED

The technologies used for the project are as follows:

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 Vuetify:

Vuetify is a Vue UI Library with ready made handcrafted Material Components.

Vuetify is a complete UI framework built on top of Vue.js. The goal of the project is to provide developers with the tools they need to build rich and engaging user experiences. Unlike other frameworks, Vuetify is designed from the ground up to be easy to learn and rewarding to master with hundreds of carefully crafted components from the Material Design Specification.

Since its initial release in 2014, vue.js has grown to be one of the most popular JavaScript frameworks in the world. One of the reasons for this popularity is the wide use of components which enable developers to create concise modules to be used and re-used throughout their application. UI Libraries are collections of these modules that implement a specific style guideline and provide the necessary tools to build expansive web applications.

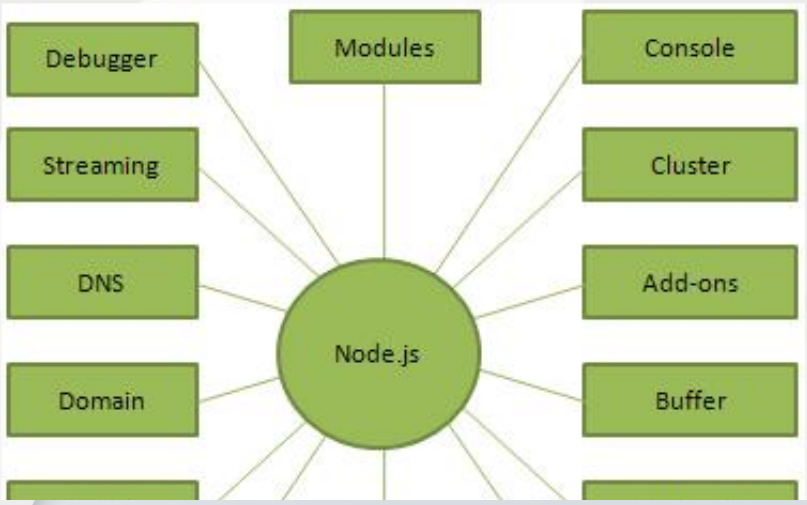
The reason we chose was because it has got large community and plenty of tutorials on youtube and other platforms.

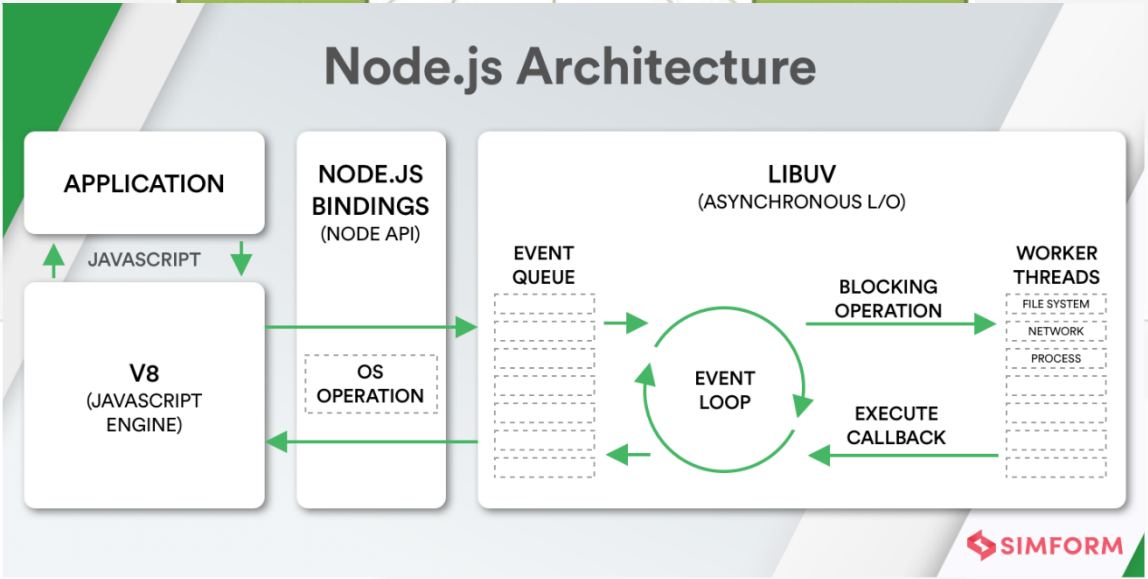
3 Node JS

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.

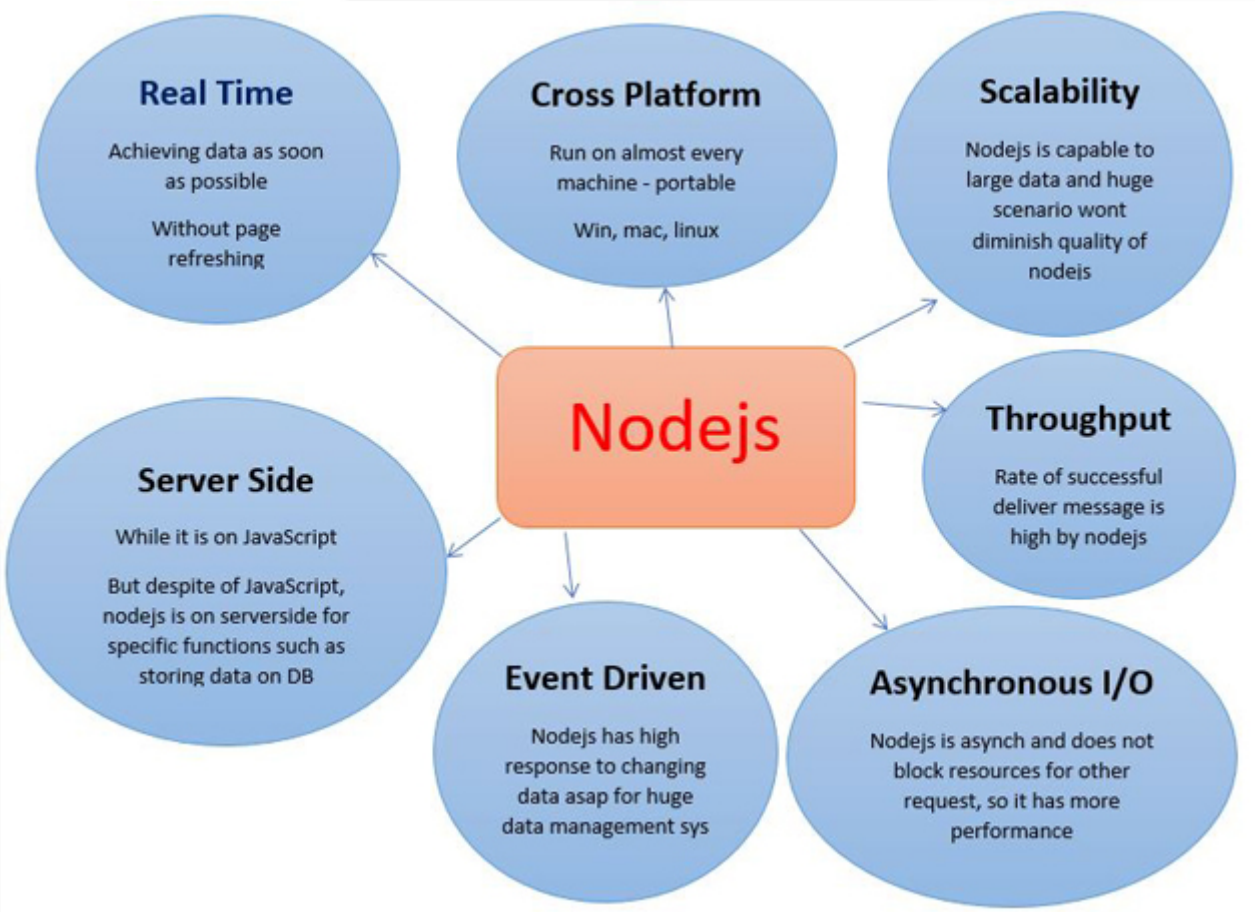




One of the biggest offerings of Node Js is its package management system, **npm**. It is one of the largest package library managers out there, and there are millions of absolutely free packages offered with regular updates and a slew of features. It is a main staple of full stack developers as it allows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

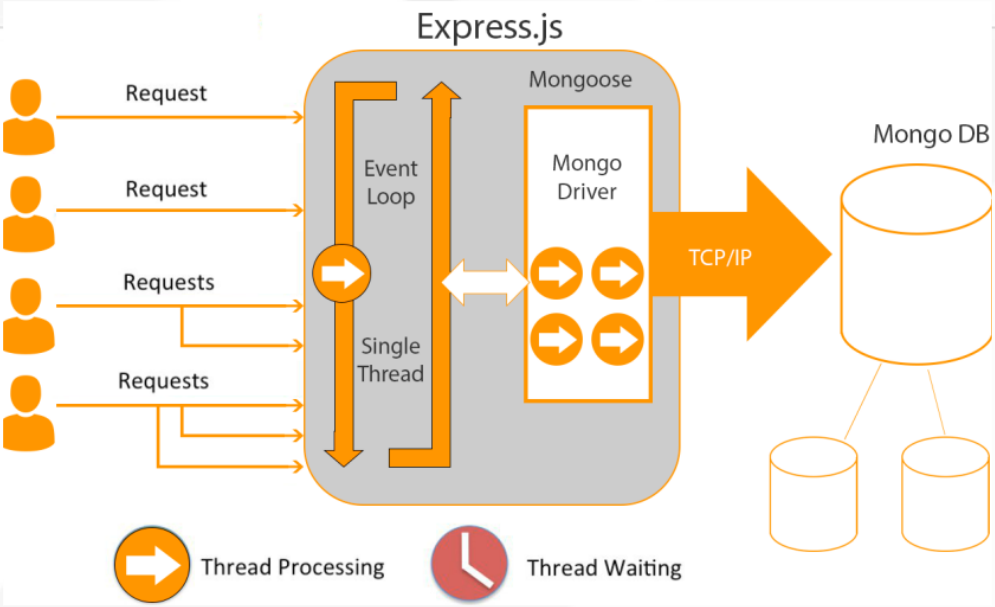
4 Express JS

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 usable 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.



5 MongoDB

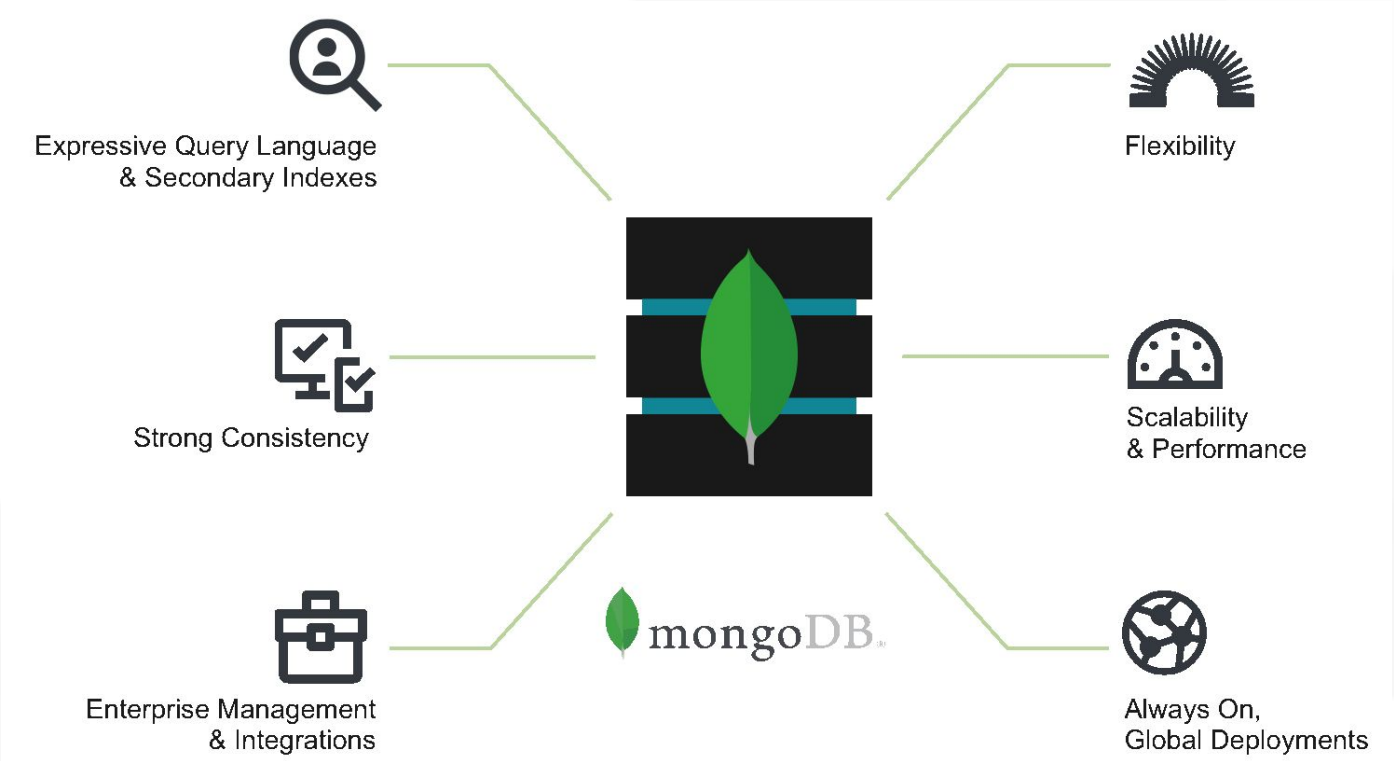
MongoDB is a general purpose, document-based, distributed, NoSQL database built for modern application developers. These days, most developers are used to thinking in terms of objects, its properties and data. This DB stores its records in the collections in a similar manner, using JSON. This makes it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



6 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.

7 **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 data-set and analyze 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 data set, 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.

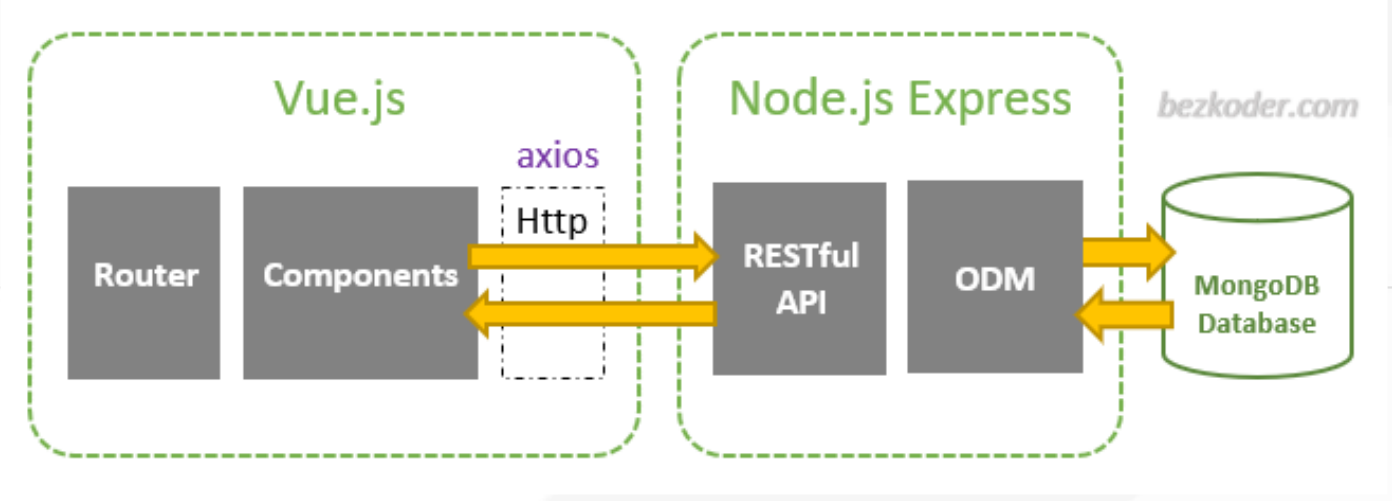
Many of these technologies that were used were gibberish to us when we first started the project development. Slowly through the help of the internet community, the expertise shared through endless tutorials in the form of YouTube videos or Blog posts, and our own interest to make this project better 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.

Express JS

Google Developers Charts

node JS

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.



. 3. THE DEVELOPMENT PLATFORM

The specifications of the platform used for project development is as follows:

3.1 System specs:
Processor: Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz 2.00 GHz
RAM: 4.00 GB
Product Type: Laptop

3.2 Operating System:
Edition: Windows 10 Home
Version: 20H2
OS build: 19042.928

3.3 Browser: Firefox
version: 88.0
platform: (64-bit)

3.4 project code editor: vscode
Version: 1.55.2 (user setup)
Commit: 3c4e3df9e89829dce27b7b5c24508306b151f30d
Electron: 11.3.0
Chrome: 87.0.4280.141
Node.js: 12.18.3
V8: 8.7.220.31-electron.0
OS: Windows_NT x64 10.0.19042

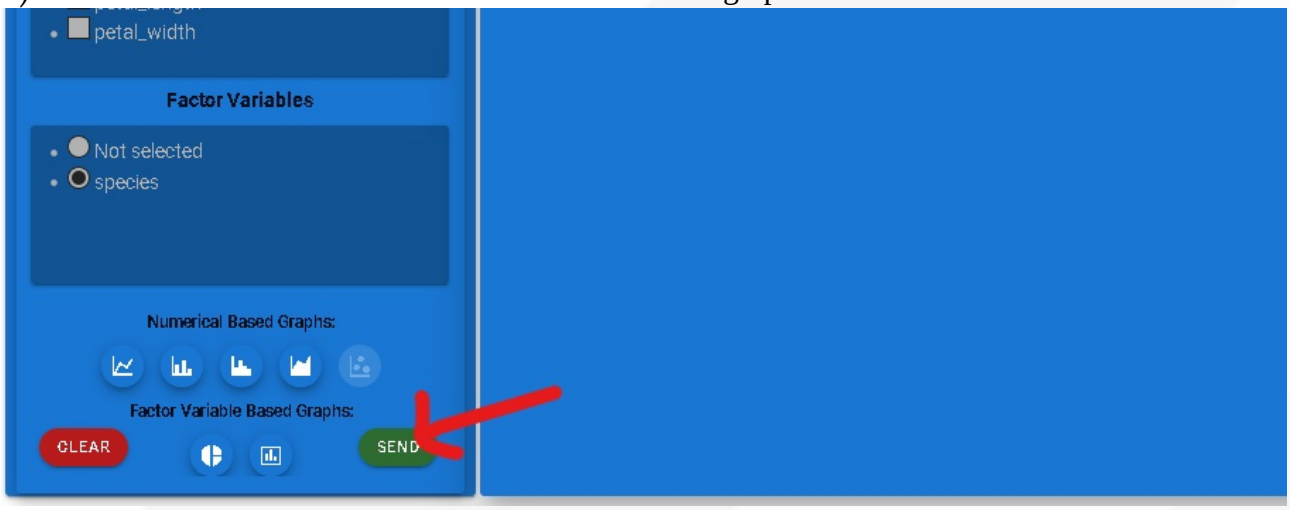
4. WORKFLOW: HOW TO USE?

The steps to be followed by the user are really simple as given below:

1) Input Dataset into the app. And click on "START ANALYSIS"

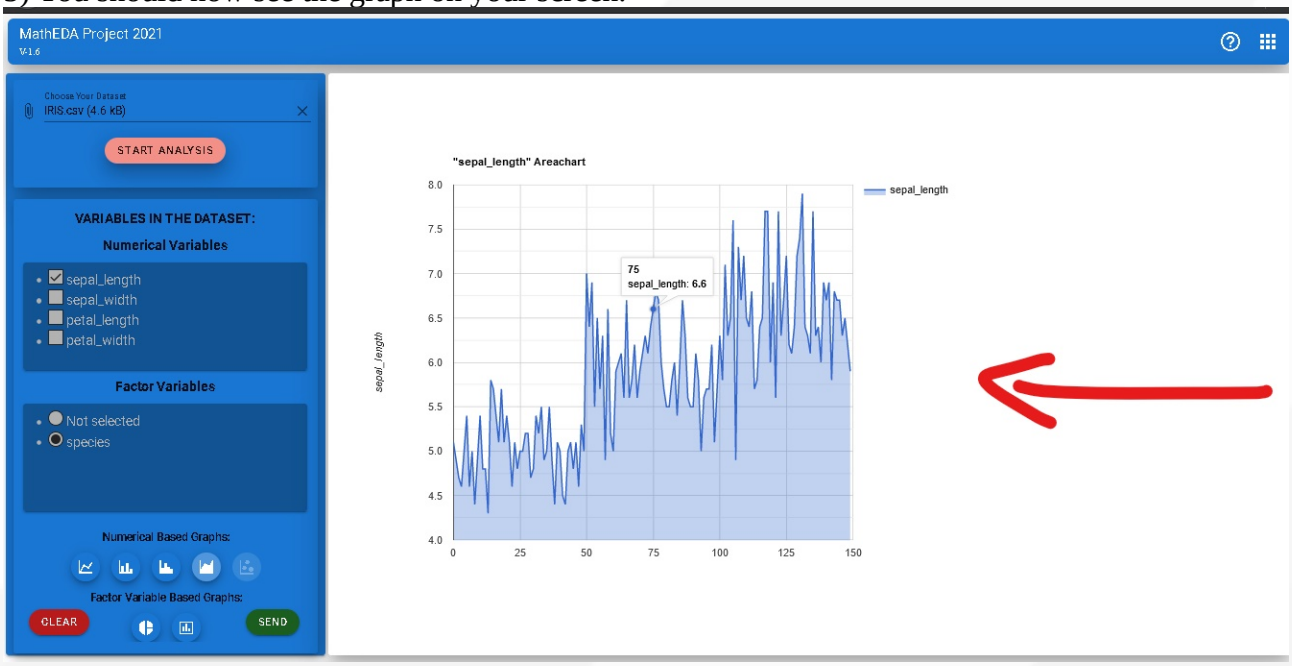
2) Choose the variables to analyse. Your graph options will be suited to your variable selection choice.

3) Click the "SEND" button to send the variables to the graph selector.



4) Choose the type of graph you want to set up.

5) You should now see the graph on your screen.



5. User Interface (UI)

5.1 UI of the main site

The UI of the whole page is maintained in view Home.vue which is further displayed by App.vue as per the general requirement of vue framework. All parts of projects are broken down into components which can be nested too.

The Home.vue holds the two major components of the web-app: TopDashboard.vue and MainScreen.vue respectively. The TopDashboard is responsible for the top bar in the app displaying the Project name, button for about pages and also a button for extra tools. And the MainScreen is the main playground responsible for user data inputs, graphs options and the displaying of created graph too.

[Image of webapp showing the topbar and mainscreen]

Code:

```
<template>
  <!-- This is the main page
  It contains top-dashboard and the main-screen( which holds items for
  graphing) -->
  <div class="home" style="padding: 4px;">
    <v-container fluid style="height:96%;">
      <v-row>
        <v-col style="padding: 2px; margin-bottom: 10px;">
          <TopDashboard @changeTheme="mtheme=$event" :ptheme="mtheme"
          :version="version" />
        </v-col>
      </v-row>
      <v-row style="height:100%;">
        <v-col style="height:100%; padding:0;">
          <MainScreen :ptheme="mtheme" />
        </v-col>
      </v-row>
    </v-container>

  </div>
</template>

```

[Template section of the Home.vue showing TopDashboard and MainScreen]

Instead of cluttering a single component the features are divided and put into the relevant components. As such a component has to just worry about its immediate child components and not its inner workings and inner placements of its own children components.

Main site also exhibits a link button on top dashboard which routes to the about pages. There user can find information about the project, the way of using it, technologies used, etc. The detailed description of which can be found on sections dedicated to them.

5.1.1 Communication among the components:

The components communicate with each other using the feature of props and emit functions provided by the vue framework. Props are used when a component wants to communicate with its child component and emit functions are used when the component wants to send some message to its immediate parent component. Of course the child component should have declared the props sections in its script part and the parent component must have a binded emitted message catcher coded into the component call.

A simple example can be seen by the way themes are applied:

[Image of sending the ptheme prop to MainScreen component]

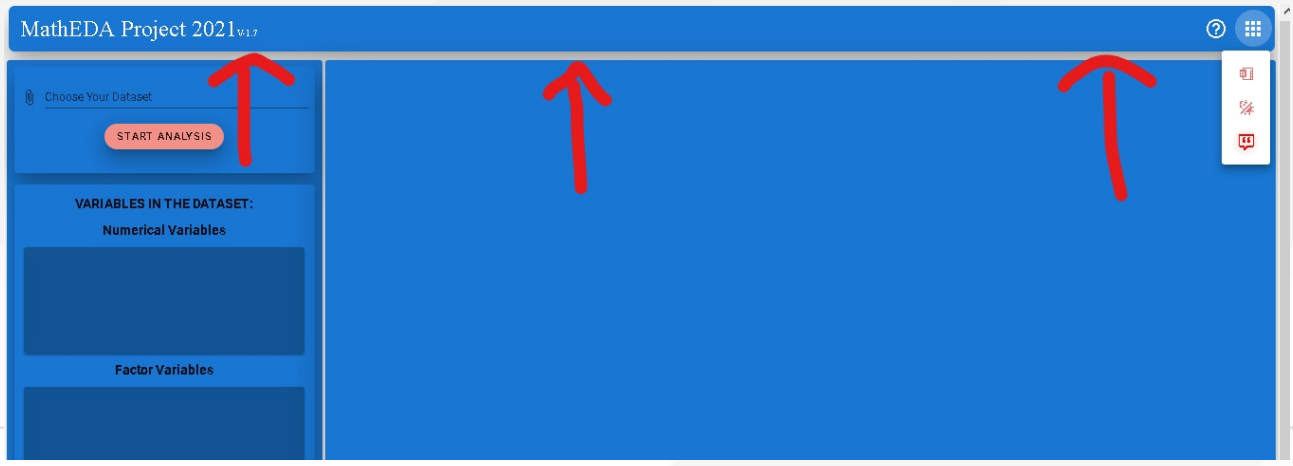
[Image of component declaring the prop 'ptheme' in script]

5.2 Flow of data:

The only major data flow that occurs is that of the datafile given by the user based upon which the graph is drawn. The data is first taken as a file in the left bar, then its analysed and as per the options chose by the user graph is drawn based upon the data.

5.3 Top navigation bar router

This is the top bar displayed in main home page. It basically exhibits the project name and hold other functional links and buttons that are not directly related to the plotting and charting of graphs. It contains a button with (?) icon which links to the about pages where user can find useful informations about the project and ways of using it. This top bar also holds a "three vertical dots" button which opens the extra-tools that are provided by the project to the user. The extra tools being a temporary note editor, theme changer and feedback button.



[Image of top dashboard]

Code:

```
<v-toolbar :color= ptheme dark rounded="lg" elevation="24" >

  <!-- name and version -->
  <v-toolbar-title>
    <span class="projectName">MathEDA Project 2021</span>
    <span class="subheading">V-{{ version}}</span>
  </v-toolbar-title>

  <v-spacer></v-spacer>

  <!-- link to about page -->
  <v-btn color= white icon to="/about" x-large title="About"
  class="dashButtons">
    <v-icon>mdi-help-circle-outline</v-icon>
  </v-btn>

  <!-- tools ( noted, themes) -->
  <ExtraTools @changeTheme= "changeTheme($event)"/>
</v-toolbar>
```

[Code for top-dashboard]

In terms of code TopDashboard is just a 'v-toolbar' which holds sub components and a button with a link and icon. The project name is displayed as 'v-toolbar-title'.

5.4 About

The top dashboard contains a button with (?) icon which routes to the about pages which through individual sections gives useful informations as listed:

i. About Page: Tells about the project.
ii. How to use?: Guide with screenshots to tell user how to use the web-app.
iii. Technologies used: The technologies upon which the project is built.
iv. Future Prospect Page: The possible improvements in the future.
v. Creators Page: Gives information about the creators of the web-app.

[Image of about (?) icon in topdashboard]

5.4.1 About page

This page tells about the project. The reason it was created and the purpose it was created for.

5.4.2 Creators page

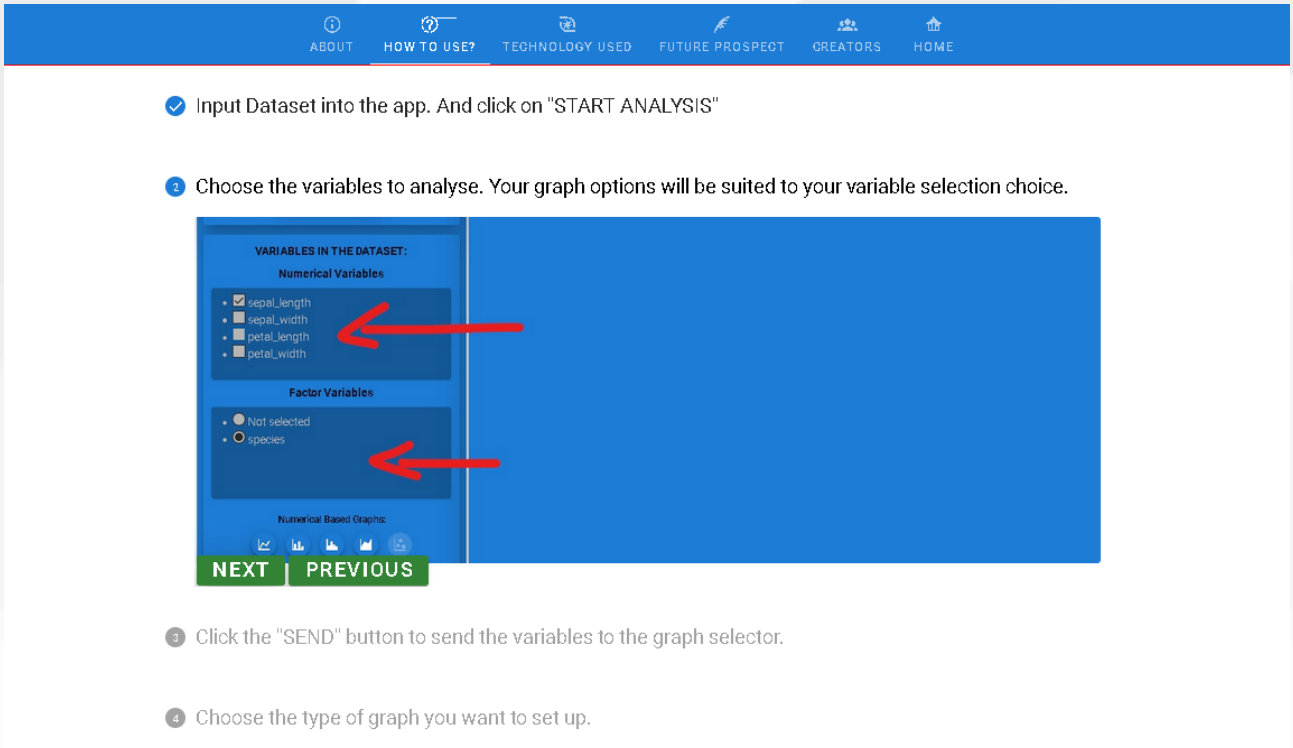
This page tells about the creators. As this web-app was built as a III year group project, the creators and the guide's informations are show cased here in this page. The individual information are showed in form of profile cards where the user can see more details by clicking upon the profile cards.
[Image of Creators Page]

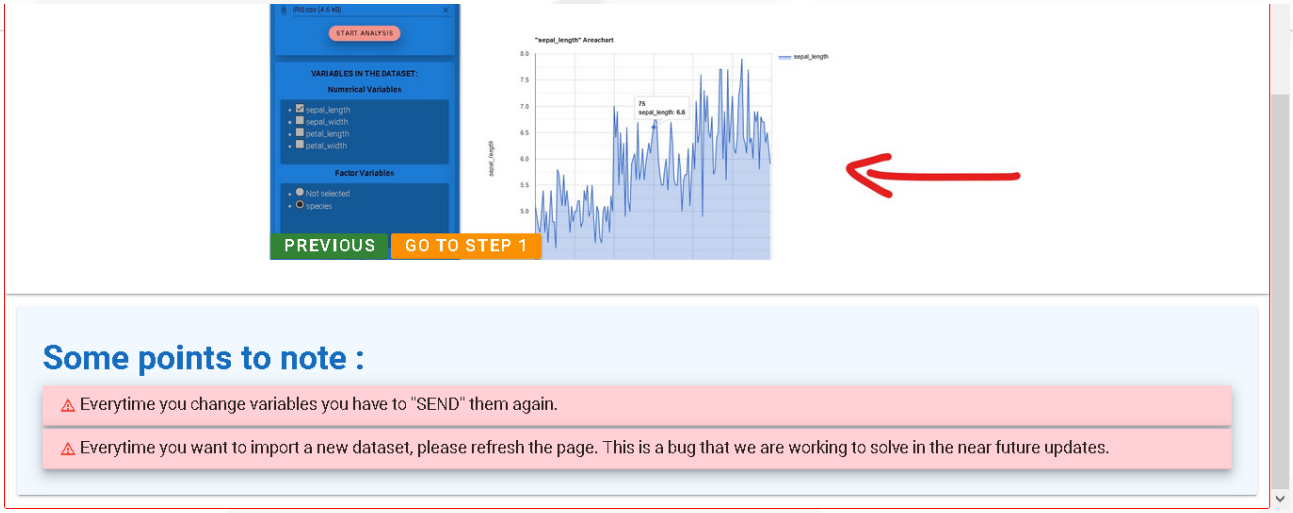
5.4.3 Future prospect page

This page gives the possible improvements on the project in the future. Of course the list is not exhaustive thus this page only features few of the prospects.

5.4.4 How to use page

This is a help page for user where the way using the web-app is shown step-by-step using screenshots. Further some tips and points to be remembered by the user is given at the bottom of the page.





[Images of How to Use Page]

Though called page, these about-pages are implemented using 'v-tab' and used as a component and as such no routing or page loading takes place. For the step-by-step demonstration the vuetify component 'v-stepper' and its relevant child components are used.

5.4.5 Technologies page

This is technologies page which gives the list of the technologies that were used to build the project and thus upon which the project stands. The technologies are segregated based upon the role they play such as backend, frontend, server connections, etc.

The list is given as:

i) Frontend:
Vue framework + Vuetify Component Library

ii) Server Platform:
Node.js

iii) Server Backend Connection:
Express + Axios Library

iv) Backend Database:
MongoDB (No-SQL database)

v) Graphing Library:
Google Charts

vi) Parsing Libraries:
BodyParser(DB objects to JSON)
papa-parse(CSV file to JSON)

5.5 Extra tools:

Other than the prime feature of drawing the graphs the web-app also provides other extra tools to ease the utilization for the users. This extra tools

can be accessed by pressing the ‘vertical three dots’ icon at the top right side on the top dashboard of the main home page.

The extra tools provided are listed in the following sections.

5.5.1 Notes:

This is a very simple temporary notes service provided by the web-app. The user can write basic texts on this and it stays consistent till the loading, reloading or refreshing of the page.

5.5.2 Themes:

The web-app can adapt itself to the different themes as desired by the user and this facility is provided by this extra tool. This can be accessed from the extra tools section and when its main button is pressed, a list of themes drops down from where user can choose and click to change the theme of the whole web-app.

As desired the background of the graph area isn’t affected by the theme change once a graph is drawn, before that however it is fully active.

5.5.3 Feed back:

The web-app also gives the basic functionality to provide feedback to the admins. This opens a form which asks for the user name, registration number, email address and the feedback.

This feature is connected to the backend and the feedback is stored on the server for later access and review.

5.6 Colour palette & fonts

The web-app UI is built around simple fonts and colour palettes. They are chosen as browser safe and hence presents no complications of being rendered falsely on some browsers while functioning properly on others.

The basic colour palettes are controlled via themes which affects the whole of the webpage, and which is accessed through the extra tools section. There are a list of colours given to choose from in the theme section namely: sky, greeno, pinko, mango, ocean, dark theme. User just have to select one and click on it to see the desired colour flourish throughout the webpage.

6. Graphing

This is the core purpose of the web-app and hence of prominent importance. From the datafile given by the user numerical and categorical parameters are extracted and then options for various parameters, graph types permutations are presented to the user. When the user chooses and finalizes the graph is drawn on the central pane of the web-page.

Majority of the operations and processing are done on the background. On the backside multiple technologies work in sync to provide the end output. First the datafile supplied by the user is parsed into the supported version of the web-app. Then the corresponding parameters and data values are analysed and extracted. After the user gives their desired selection of choices they are sent to google charts, the library used to render the graphs, which in turn returns the graph, and which is rendered by UI on the main screen.

6.1 Interconnecting Graph and UI

The project on the whole can be divided into two broad divisions: the UI and the Graphing part. The graphing part is all about the logic used in the drawing the graph from the supplied fields and parameters. While User Interface just focuses on providing the easy and elegant interface to the users.

In the project the conjunction of these two broad divisions are majorly done on the component MainScreen.vue. For the sake of easy distinction and maintenance the whole logic of graphing is isolated in the MainScreen component, which further consists of input taking left-bar and output graphing area. Then on these logic input and output elements the UI is wrapped around and basic wiring of themes is done to maintain the consistency with the other parts of the web-app.

6.2 Wrapping UI design on Graphing tools:

All of the options for graphs and the plotting/charting of graphs is done by the MainScreen.vue component and these is where lies the main graphing logic too. The screen covering of this MainScreen.vue in home page is major one and in itself its divided into two major sections:

6.2.1 Leftbar

This is mainly the input area for plotting the graph. The datafile receiving, selection of graph options and the type of graph to plot itself is all given in this component. The user provides the datafile and then analyzes the data when a list of parameters are shown. Then the user can register the selected parameters by pressing send and then plot a graph by clicking on the relevant graph

options. The segregation is done between numerical and categorical parameters and appropriately presented.

6.2.2 Graphing-Area

This is the area where the graphs are drawn. The user inputs taken from the left bar are processed and when the user selects and clicks on list of relevant graph options graphs are drawn on this area.

7. FUTURE SCOPE

As the statistics forms the major field of these project and its main purpose being the data visualizations, it sees an in-exhaustive list of improvements and add-ons as we see the emergence of data visualization. In this nascent stage it is only able to provide the graph drawings of some of the simplest graphs. The implementation of very basic concepts of statistics would also empower this project immensely and with ever increasing field of data science there isn't gonna be any short comings of feature list to add to the project.

The project can be joined with the more stable backend database to provide more stable and secure user database.

It can be improved to support the many more platforms and all screen sizes.

The basic features present themselves can be improved such as simultaneous drawing of multiple graphs which would allow the user to better compare the graphs and extract more rich information from there in.

The possible improvements for the project is also given in the Future Prospect page on the about pages in the web-app itself.

The definite is with more tomorrow there will be more scope to enrich the project.

8. CONCLUSION

Though this project in itself is not big, vivacious and scintillating with the higher data visualization concepts, it was a first step to build such project for as we see there are few site offering this service and that too on paid terms. And there is no denying the significance of such tools in the world of data science. As such this project has been successful in at least initiating the creation of such tool on our own. It has been successful in incorporating the basic functionalities and has created a base on which further development can be done. As such there is also scope of massive evolvement of this project.

9. REFERENCES

From used technologies official home pages, documentations and communities:

1. Vue Framework
2. Vuetify Component Library
3. Node.js
4. Express
5. Axios Library
6. MongoDB (No-SQL Database)
7. Google-Charts
8. Papa-Parser (CSV file to JSON)
9. BodyParser (DB objects to JSON)

From the Q/A internet sites:

1. Stackoverflow
2. Superuser

Tutorial sites:

1. youtube
2. w3schools
3. freecodecamp
4. tutorials point

Thank You!

We are forever indebted and grateful

to our beloved Mother Sai...