Funded by the European Union's Internal Security Fund - Police (2014-2020)

Internal Security Fund – Police - Programme (2014-2020)

823701 - YoungRes - ISFP-2017-AG-RAD

Strengthening European Youngsters Resilience through Serious Games - YoungRes

YoungRes Visualization Tool's CODE Documentation

By
Saroar Jahan (Team Oulu)

April 12, 2021

Funded by the European Union's Internal Security Fund - Police (2014-2020)

Index

1 INTRODUCTION

This document mainly contains code documentation of the YoungRes visualization tool, as part of a same-named project funded by the European Union, whose goal is to improve youngsters’ resilience by means of the inclusion of video games in the learning process. The YoungRes Visualization tool represents the frontend part of the monitoring system that will work in the context of the YoungRes videogame API.

We have already developed YoungRes successfully; therefore, frontend code documentation is required to understand the system in the insight of developer perspective to further develop and deploy the project.

In this documentation, we will cover technology that has been used for development purposes, installation, development environment, brief details of the application file description, and deployment of the project.

2 DEVELOPED VISUALIZATION TOOL

The developed visualization tool will use the YoungRes game's API and help teachers understand the youngsters' overall performance during the gameplay. The tool allows performing primarily two types of analysis, which can be divided as:

1. **Microanalysis (focused on a single group):** This analysis will allow the user to understand a single group's performance.

2. **Macroanalysis (focused on two groups):** This analysis will allow the user to compare two different groups' performances.

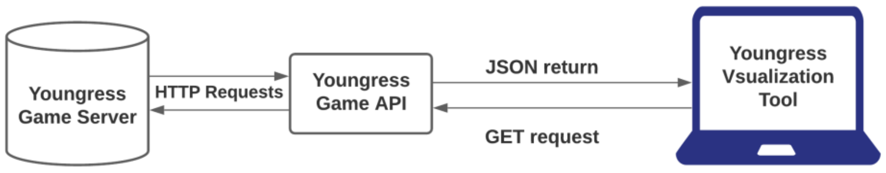


Figure 1: System basic architecture.

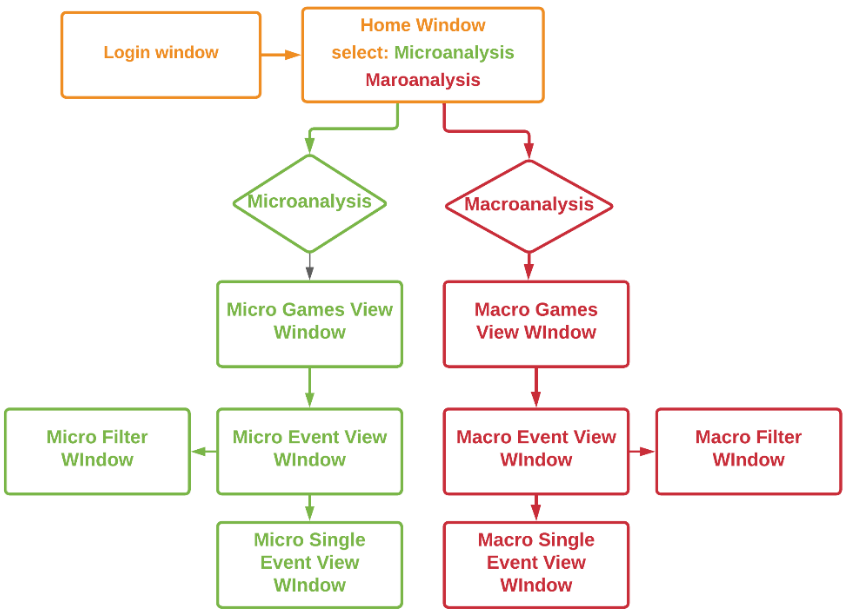


Figure 2: System developed windows/webpages.

To fulfill the task, we have developed several windows/webpages in our application (e.g., Fig 2), as stated below:

1. **Login window:** The visualization tool's first window is the login window, which takes input username and password, verifies it from the API calls, and returns special headers token to access the rest of the application.

2. **Home window (analysis type selection):** This window has options for selecting the analysis type: Microanalysis or Macroanalysis. This also considers as the home window since this the first window after successful login.

3. **Two types of 'Games View' window based on Microanalysis or Macroanalysis:** This window allows viewing chapter details, general game information, and selecting game version and chapter id, required for subsequent window (Events View window).

4. **Two types of 'Event View' window based on Microanalysis or Macroanalysis:** An event represents a situation in the videogame where the students' game interactions are recorded. To visualize students' decisions that all students have taken, this Event View has been created.

   There are two different kinds of events in the game:

   a) **Multichoice events:** In this type of Event, the students can choose an answer from a set of possible options.
   b) **Temporal events:** In this type of Event, the information recorded is the student's time to complete the task.

5. **Two types of 'Single Event View' window based on Microanalysis or Macroanalysis:** This window presents individual event information options with graphical and textual representation (i.e., event description, highlights, possible choices, percentages. etc.).

6. **Two types of 'Filter option' based on Microanalysis or Macroanalysis:** This window presents options for filtering events with the parameter of student gender (male, female), age, group id, country, city, etc.

3 TECHNOLOGY USED

For the development of the visualization tool, we have analyzed several state-of-the-art practices. Finally, we have used HTML, CSS (framework Bootstrap), and JavaScript (libraries and framework). This section will briefly introduce the technology we have used, development workflow, and development server.

1. **HTML:** HTML is the standard markup language for presenting information on Web pages. All web-based platforms that accessible by a web browser contain HTML. Since our developed tools is a web-based application, therefore we have used HTML as markup language.

2. **CSS:** HTML alone cannot design anything except presenting the information on the web page. Therefore, CSS is used to define web pages' styles, including the design, layout, colors, font family, and display variations for different devices and screen sizes. Here, we have a used BootStrap which is a CSS framework.

3. **JavaScript:** JavaScript is a scripting or programming language that allows the implementation of complex features on web pages. Every time a web page does more than display static information, there is expected JavaScript programming. However, using only vanilla JavaScript, this complex project would not be development friendly. Therefore, we have used a complete JavaScript framework, namely Vue.js.

   Vue.js[1]: Vue (pronounced, like view) is a progressive open-source framework for building user interfaces like Angular/React. It has its own ecosystem and core libraries (e. g., routing, State Management, etc.) and easy to integrate with other libraries.

   Inside Vue we have used two other JavaScript libraries:

[1] https://vuejs.org/v2/guide/

a) **Axios[2]:** Axios is a JavaScript library for connecting with an API endpoint. We have used Axios to fetch YoungRes API data and update the visualization tools continuously.

b) **Chart[3]:** Chart.js is a simple yet flexible JavaScript library for visualizing data. Our project has a bar chart and scatters plot to display; we have used this Chart.js library.

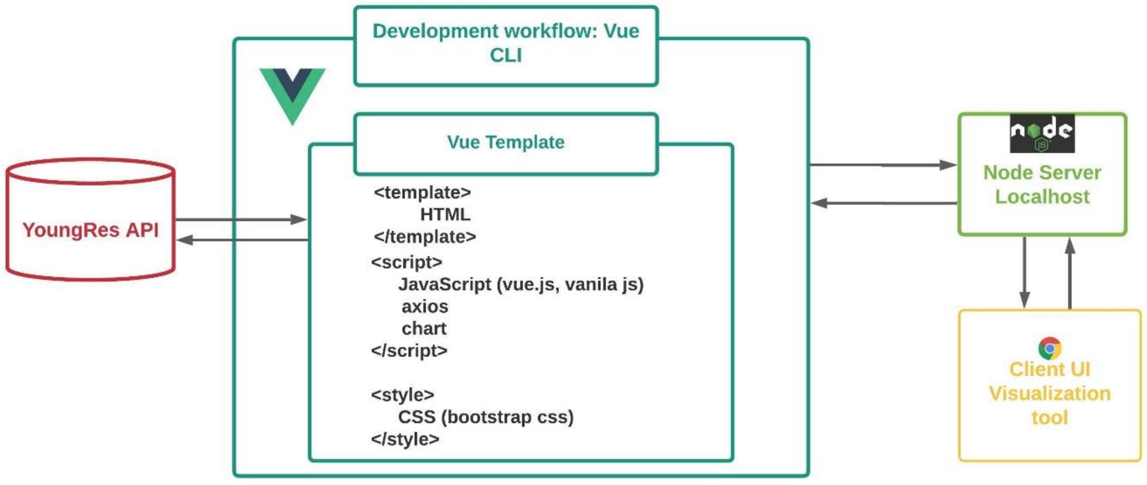


Figure 3: Localhost development workflow with Vue CLI and Node sever.

4. Development workflow (Vue CLI): We have used a dedicated workflow for our project for a couple of reasons:

a) Project needs many components to be built,

b) Different libraries and packages need to import,

c) A workflow that will hold Vue.js and all other resources and can be open by IDE or Text editor to write code,

d) and finally, when the project is done, we need to bundle everything together for deployment.

[2] https://github.com/axios/axios
[3] https://www.chartjs.org/

For this purpose, we have used Vue CLI[4]. This will help write our code by following a specific template guideline provided by Vue CLI, and after that, it has a dedicated command to build the project to ready for deployment.

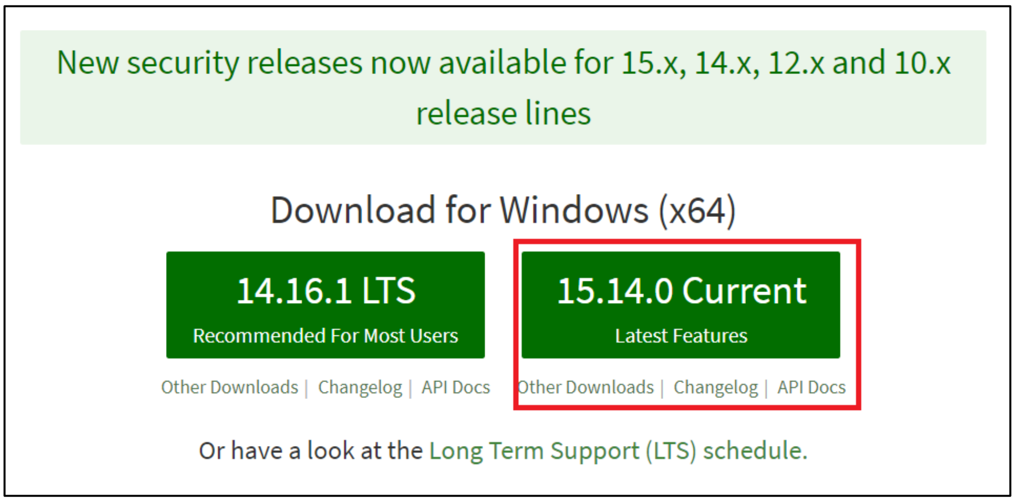
5. **Development Server (node.js):** When the development workflow (Vue CLI) is installed in our machine, it requires a browser to run our developed tool that sends HTTP requests for communication. Therefore, we need an additional local server that understands HTTP requests and runs our project on the client-side. Here we have not developed any server; instead, we have used pre-built node.js for this purpose.

4 QUICK START

This section will discuss the installation of the development server and Vue CLI and the running of the project source code that is available in the GitHub repository.

4.1 Installation of Development Server

For creating a localhost server, we will use node.js. We do not need to write any server-side code; instead, installing the current version of node.js on our machine will be suitable. Available resources could be found here.[5]


New security releases now available for 15.x, 14.x, 12.x and 10.x release lines

Download for Windows (x64)

14.16.1 LTS
Recommended For Most Users
15.14.0 Current
Latest Features
Other Downloads | Changelog | API Docs
Other Downloads | Changelog | API Docs

Or have a look at the Long Term Support (LTS) schedule.

Figure 4: Recommend latest version of node.js to download.

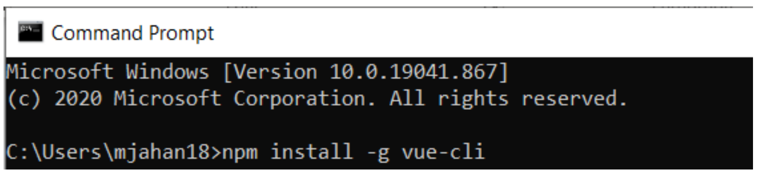
[4] https://cli.vuejs.org/
[5] https://nodejs.org/en/

After the successful installation of node.js, we will be able to use the npm command from our command prompt.

4.2 Installation of Vue CLI[6]

After installing node.js in our machine, we will now install the development workflow (Vue CLI) with the command npm install -g vue-cli.

To do so, open the command prompt for windows and run the command npm install -g vue-cli; this will install Vue CLI globally on our machine (instruction example presents in Fig-5).



```
Microsoft Windows [Version 10.0.19041.867]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\mjahan18>npm install -g vue-cli
```

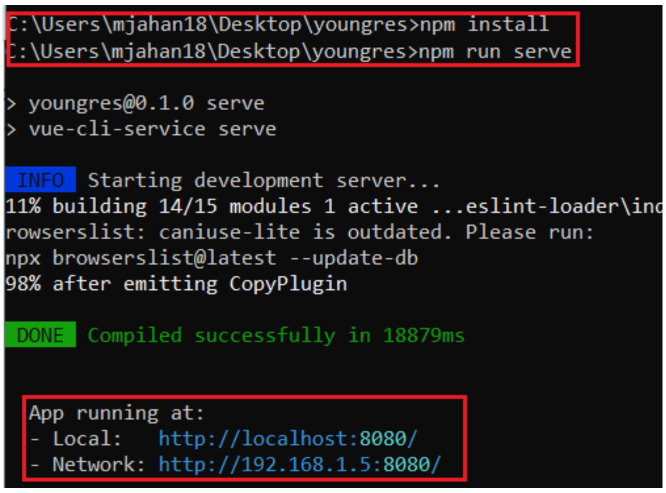
Figure 5: Command for installing Vue CLI globally.

4.3 Project Setup and open with Browser

To set up the project, follow the steps as described below:

a) First, clone YoungRes project from the GitHub repository here[7],
b) Unzip the file and browse that directory with the command prompt. Here we have saved our project file on our desktop with the repository name 'youngres'.
c) Run npm install; this will install the required dependencies for this project from the package.json file (instruction examples shown in Fig-6).

[6] https://cli.vuejs.org/
[7] https://github.com/saroarjahan/youngres



```
:\Users\mjahan18\Desktop\youngres>npm install
:\Users\mjahan18\Desktop\youngres>npm run serve

> youngres@0.1.0 serve
> vue-cli-service serve

 INFO  Starting development server...
11% building 14/15 modules 1 active ...eslint-loader\inc
rowserslist: caniuse-lite is outdated. Please run:
npx browserslist@latest --update-db
98% after emitting CopyPlugin

 DONE  Compiled successfully in 18879ms


  App running at:
  - Local:   http://localhost:8080/
  - Network: http://192.168.1.5:8080/
```

Figure 6: Command for installing dependencies and running the project.

d) Finally, to run the project, use the command npm run serve. If the project runs successfully, it will provide a URL to open it with the browser (recommend Google Chrome browser).

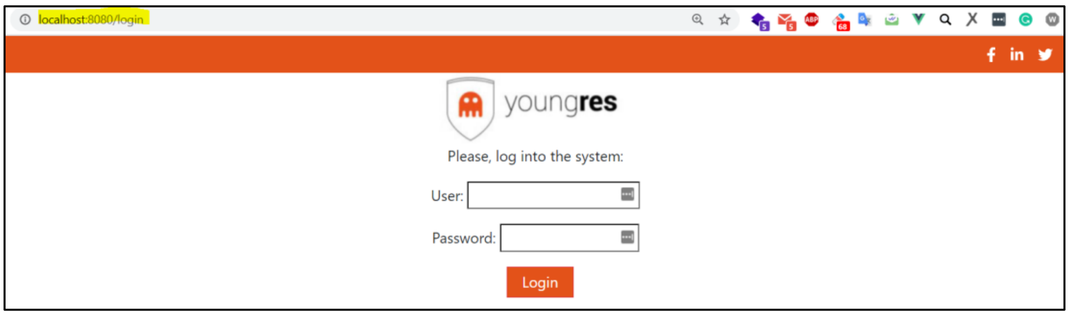


Figure 7: Frontend look of the login window after the first-time project run.

4.4 Project open with Text Editor

To open this project for development, any text editor will work that suitable for editing HTML, CSS, and JavaScript. Here we recommend SubmileText2[8] editor. SublimeText also has a Vue syntax highlighter which will help to understand Vue syntax. Installing syntax highlighter is optional. But here is the procedure to install syntax highlighter for SublimeText2:

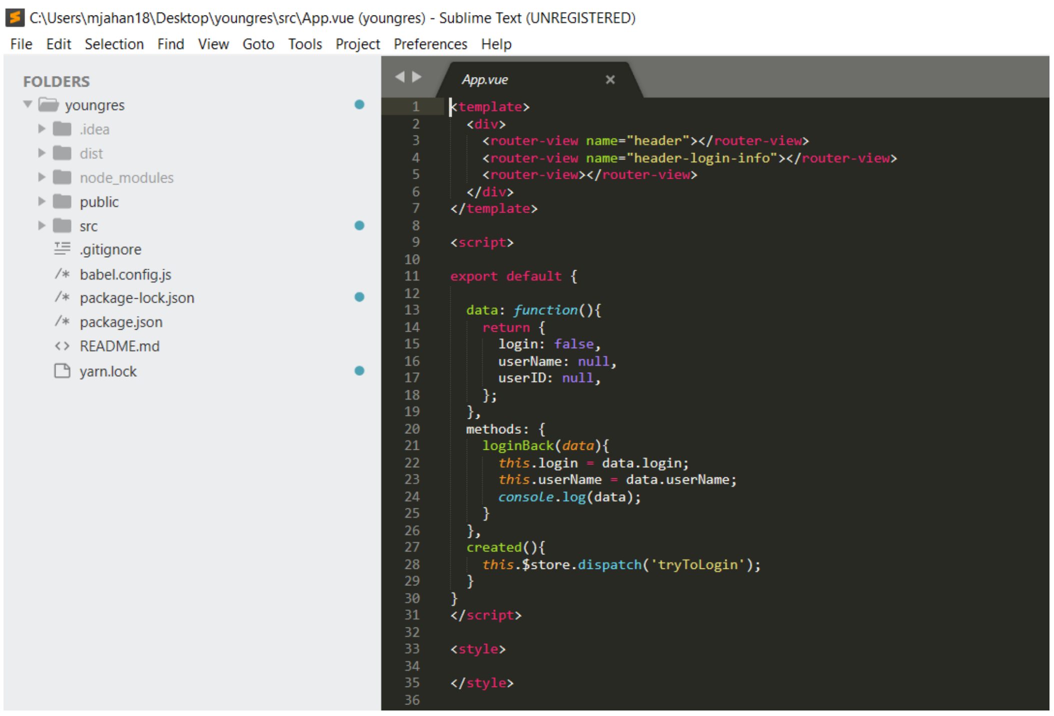
Install the Package Control package in Sublime:

[8] https://www.sublimetext.com/2

1. Installation from sublime menu bar
   a. Open the 'Tools' menu
   b. Select ‘Install Package Control’ press enter.

Install Vue Syntax Highlighting package:

a) Press Command-Shift-P (Mac OS X) or Ctrl-Shift-P (Windows) to open the Command Palette.
b) Start typing Package Control and select Package Control: Add Repository
c) Paste this on the textbox field that appears -> https://github.com/vuejs/vue-syntax-highlight.git
d) Press Command-Shift-P (Mac OS X) or Ctrl-Shift-P (Windows) again and start typing Package Control
e) Select Package Control: Install Package. It should list all available packages.
f) Search for Vue Syntax Highlight and select it for actual installation.

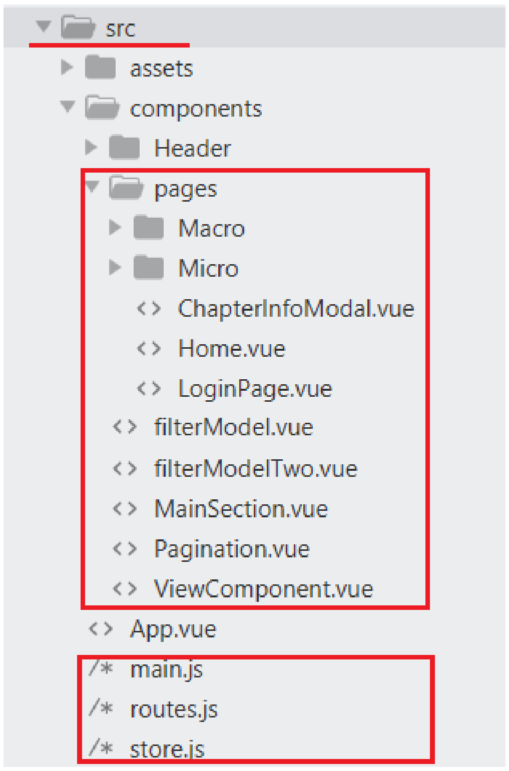

C:\Users\mjahan18\Desktop\youngres\src\App.vue (youngres) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
FOLDERS
youngres
.idea
dist
node_modules
public
src
.gitignore
babel.config.js
package-lock.json
package.json
README.md
yarn.lock
App.vue

```
<template>
  <div>
    <router-view name="header"></router-view>
    <router-view name="header-login-info"></router-view>
    <router-view></router-view>
  </div>
</template>

<script>

export default {

  data: function(){
    return {
      login: false,
      userName: null,
      userID: null,
    };
  },
  methods: {
    loginBack(data){
      this.login = data.login;
      this.userName = data.userName;
      console.log(data);
    }
  },
  created(){
    this.$store.dispatch('tryToLogin');
  }
}
</script>

<style>

</style>
```

Figure 8: Project repository open with SublimeText2 text editor.

5 PROJECT FILES DESCRIPTION

In this section, we will describe the file system of our project. The ‘src’ is the directory where we have placed all of our required files.



```
src
  assets
  components
    Header
    pages
      Macro
      Micro
      ChapterInfoModal.vue
      Home.vue
      LoginPage.vue
    filterModel.vue
    filterModelTwo.vue
    MainSection.vue
    Pagination.vue
    ViewComponent.vue
  App.vue
  main.js
  routes.js
  store.js
```

Figure 9: Inside look of ‘src’ folder.

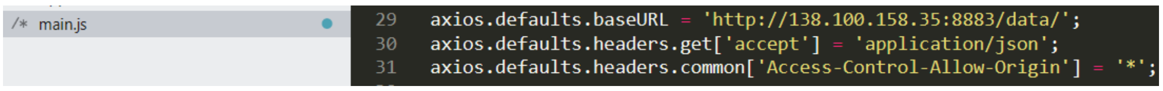
5.1 Assets

Assets directory contains the images and other graphical files (e.g., logo of YoungRes).

5.2 Global JS file

1. **store.js:** here we have stored some global variables state. Example if user login and logout this state of user data will update globally into client browser.
2. **router.js:** Vue Router is the official router for Vue.js. It deeply integrates with Vue.js core to make building Single Page Applications. In this file, we have declared the route path of our application.
3. **main.js:** In this file, we have imported Vue built-in libraries and other additional libraries that have been used in this project. We have imported font-awesome library for the icons,

bootstrap for CSS and Axios for API call. Besides, here we have globally declared the YoungRes API path (e.g., Fig 10).



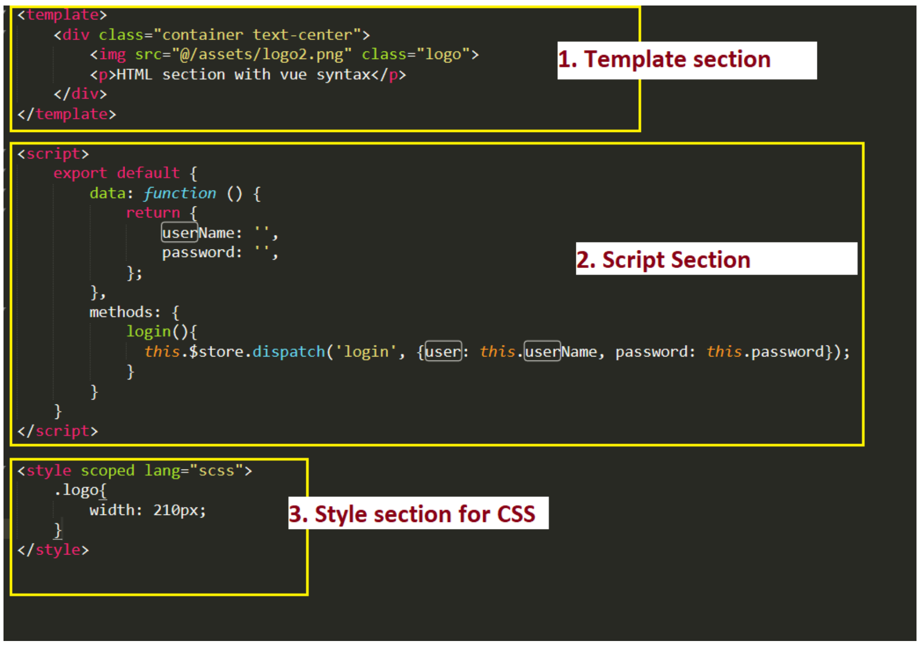
```
axios.defaults.baseURL = 'http://138.100.158.35:8883/data/';
axios.defaults.headers.get['accept'] = 'application/json';
axios.defaults.headers.common['Access-Control-Allow-Origin'] = '*';
```

Figure 10: YoungRes API URL declaration inside main.js

5.3 Components directory

In the repository, we have created template files for each of our windows; the template file format is dot(.)vue (e.g., <file_name>.vue). In the Micro directory, we have kept template files related to macroanalysis, and the Macro directory has stored the files for Macroanalysis. We also have some global templates example, header, login page, filters, etc. Every template file primarily has three main sections as follow (e.g., Fig.11):

a) Template: template section used for writing HTML tag and Vue related syntax,
b) Scripts: scripts section used for declaring data type and method/function related to template, and
c) Style: style section used for CSS code.



```
<template>
    <div class="container text-center">
        <img src="@/assets/logo2.png" class="logo">
        <p>HTML section with vue syntax</p>
    </div>
</template>

<script>
    export default {
        data: function () {
            return {
                userName: '',
                password: '',
            };
        },
        methods: {
            login(){
              this.$store.dispatch('login', {user: this.userName, password: this.password});
            }
        }
    }
</script>

<style scoped lang="scss">
    .logo{
        width: 210px;
    }
</style>
```

Fig. 11: A Vue template example; the entire tool has followed the same template

architecture.

5.3.1 Header

Header template is the top section of every window. It contains information about the logo, logged username, and option for logout (e.g., Fig. 12).

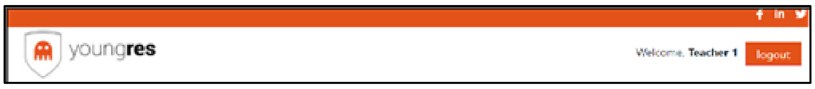


Figure 12: Header section.

Header template file location:

a) components/Header/LoginInfo.vue
b) components/Header/Header.vue

5.3.2 Login window

The visualization tool's first window is the login window, which takes input as a username and password, verifies it from the API calls, and returns special headers token that required for successful login.

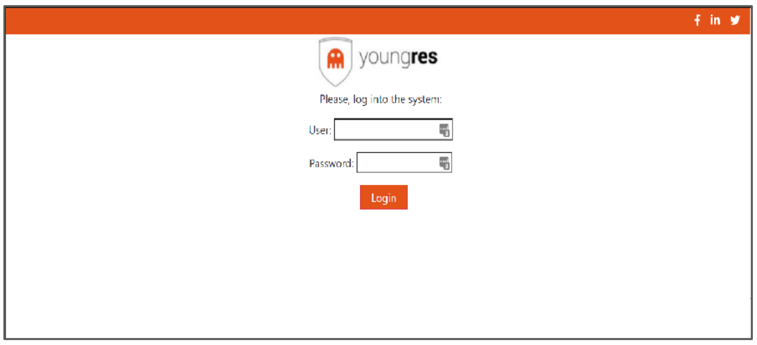
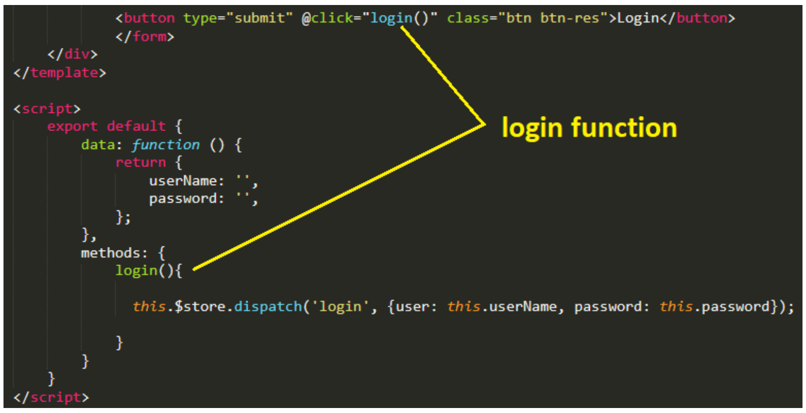


Figure 13: User Login window.

Login template file location: components/pages/LoginPage.vue

Code Instruction: login() function used for controlling login into the system (e.g., Fig14).

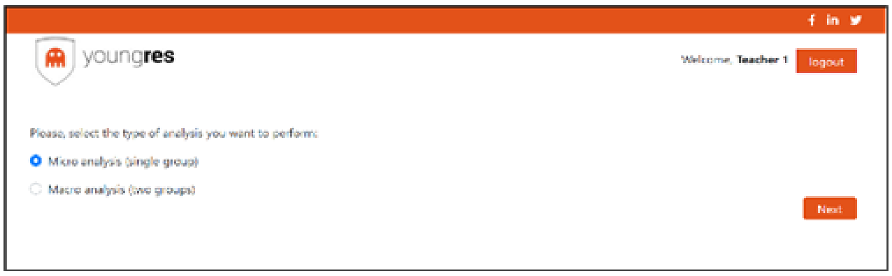


```
            <button type="submit" @click="login()" class="btn btn-res">Login</button>
            </form>
    </div>
</template>

<script>
    export default {
        data: function () {
            return {
                userName: '',
                password: '',
            };
        },
        methods: {
            login(){

              this.$store.dispatch('login', {user: this.userName, password: this.password});

            }
        }
    }
</script>
```

Figure 14: login function used for controlling access to the system.

5.3.3 *Home (analysis type selection) window*

This analysis type selection window is the first window after successful login to the system. When the user has logged in, they will be asked to choose between two different analysis types: microanalysis or macroanalysis. After selecting the preferred radio-button user, click the ’Next’ button from the right-left corner to go to the next window, which is called Games View. Almost all the next window has a ’Home’ button that will bring the user back to this window; therefore, we named it as a 'home' window.

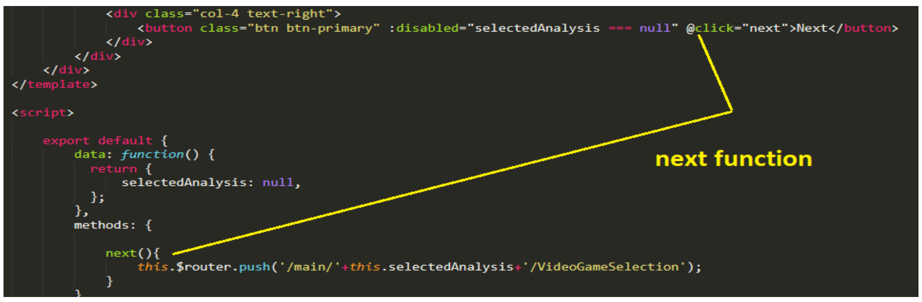


Please, select the type of analysis you want to perform:

[✓ Micro analysis (single group)]

Macro analysis (two groups)

Figure 15: Home window allows a user to select what kind of analysis they want to perform, Micro/Macro analysis.



```
            <div class="col-4 text-right">
                <button class="btn btn-primary" :disabled="selectedAnalysis === null" @click="next">Next</button>
            </div>
        </div>
    </div>
</template>

<script>

    export default {
        data: function() {
          return {
              selectedAnalysis: null,
          };
        },
        methods: {

            next(){
                this.$router.push('/main/'+this.selectedAnalysis+'/VideoGameSelection');
            }
        }
```

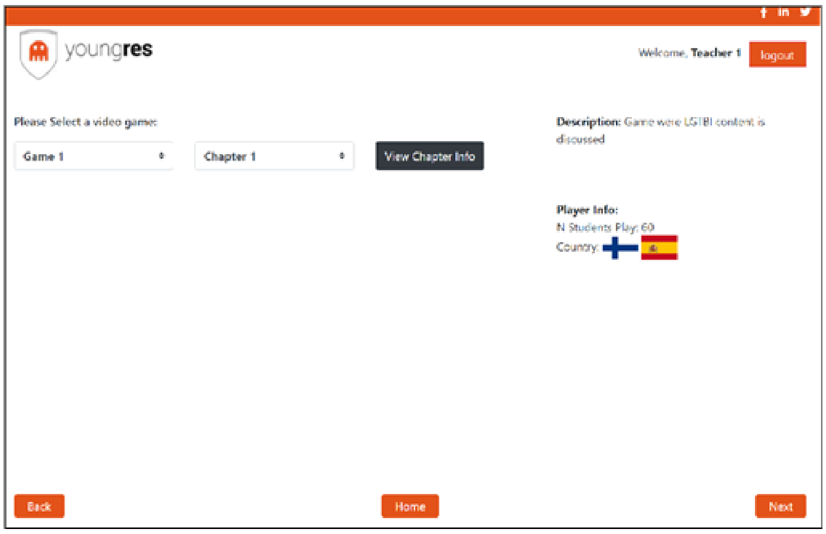
Figure 16: Home window code snippet, next() function redirect next window.

Template file location: components/pages/Home.vue

Code Instruction: JavaScript function- next() used for redirecting to Games View window.

5.3.4 *Games View*

There are two type of Game View window **(i)** Games View window for microanalysis and **(ii)** Games View window for macroanalysis. User will be redirect one of them based on their analysis type that has been selected from previous ‘Home’ window.



Please Select a video game:

Description: Game were LGTBI content is discussed

Player Info:
N Students Play: 60
Country:

Figure 17: Game Selection window for microanalysis.

Game view template file location:

a. components/pages/Micro/VideoGameSelection.vue (for Microanalysis)
b. components/pages/Macro/VideoGameSelectionGroup.vue (for Macroanalysis)

Code Instruction: Like other Vue templates in this porject, it also has three sections, Template, Script, and Style. Here, we highlight some of the important functions from the Script section:

a. data() : is the variable declaration section, here it store data about the country, flag, chapter, description, events, selected chapter, selected game, selected game version, number of play, etc.
b. mounted(): use Axios library to get API call /descriptions/games.
c. chapterInfo(): open new window for game chapter details.
d. next(): redirect to the next Event View window.
e. home(): redirect to ‘home’ window.
f. back (): redirect to previous window.
g. selectGame(): select game version.
h. selectChapter(): select chapter.
i. loadGameData(): load game data e.g., description, game code, number of players, chapters, countries etc.

5.3.5 Events View

There are two types of Event View window **(i)** Events View window for microanalysis and **(ii)** Events View window for macroanalysis. Users will redirect one of them based on their pre-selection from the 'home' window. This 'Events View' window has been created to visualize students' decisions that all students have taken.

There are two different kinds of events in the game:

a) **Multi-choice events:** The students can choose an answer from a set of possible options. Graphical representation of all multi-choice events from a chapter shown in Fig. 18.

b) **Temporal events:** In this case, students do not have to select an answer, but they have to conduct a specific task during a period. In this type of Event, the information recorded is the student's time to complete the task Time Event (Fig. 19).

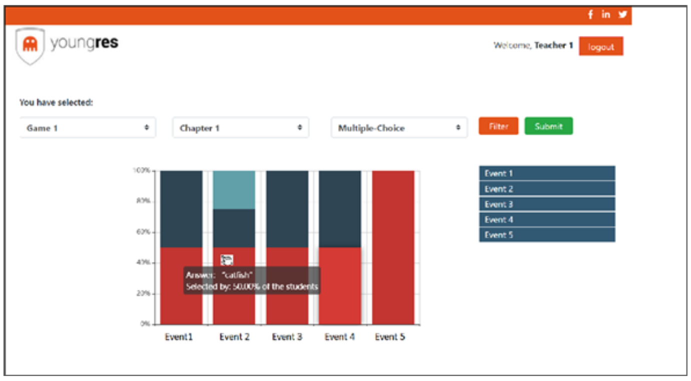


Figure 18: Multi-choice Events View window for microanalysis.

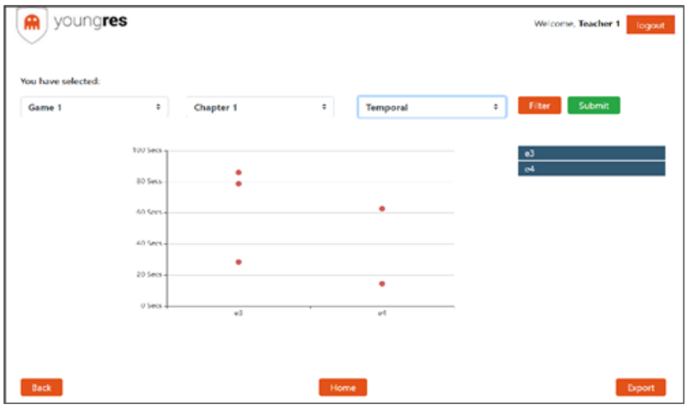


Figure 19: Time Events View window for microanalysis

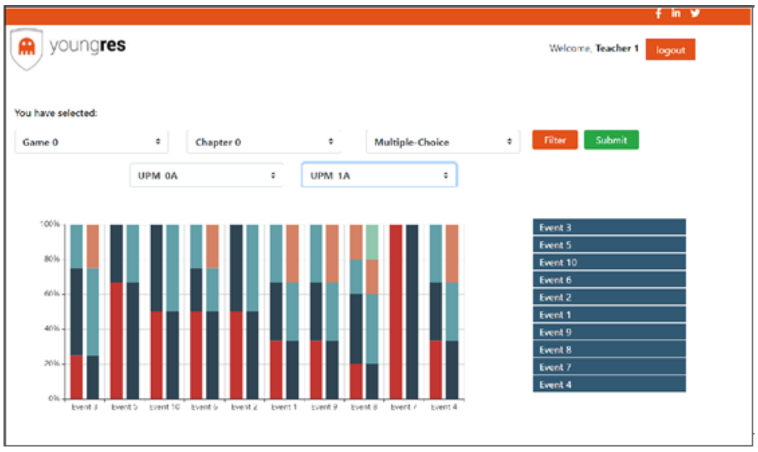


Figure 20: Multi-choice Events View window for macroanalysis.

Events View template file location:

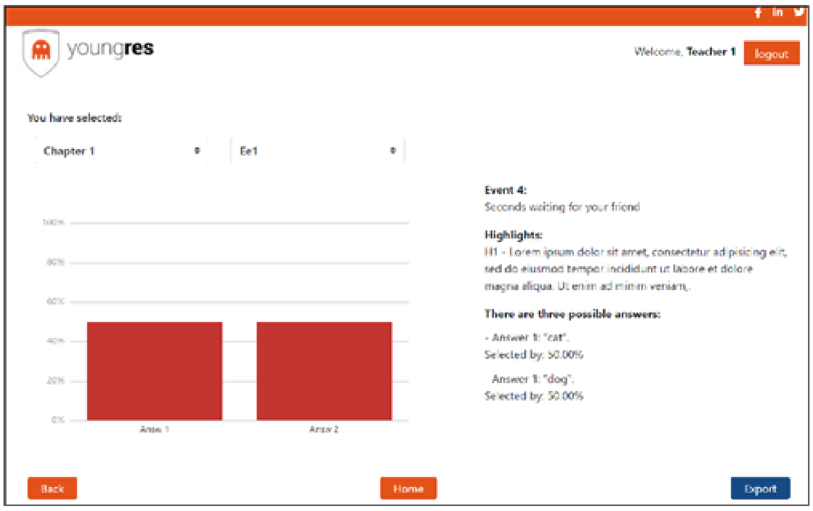
a. components/pages/Micro/MicroAnalysis.vue (for Microanalysis)
b. components/pages/Macro/MacroAnalysis.vue (for Macroanalysis)

Code Instruction: In scripts section, it has many functions, some important are mention below:

a. data() : It store data about choice, event decisions, event type, game, version, result, chapters, student filter, group filter, and others.
b. mounted(): used Axios library get three different API call descriptions/games", filters/group and filters/student.
c. chapterInfo(): collect chapter details about game, version, and chapter.
d. filter(): load filter data, student filter and group filter.
e. home(): redirect to 'home' window.
f. back (): redirect to previous window.
g. selectGame(): select game id.
h. selectChapter(): select chapter id.
i. loadData(): load game data based on game id
j. selectChoice(): select event type , multichoice or temporal.
k. dataAnalysis(): analysis data based on event type multichoice or temporal.
l. barChartLoad() : represent data into bar graph or scatter plot.

5.3.6 *Single Event View*

There are two types of Single Event View window (i) Single Event View window for microanalysis and (ii) Single Event View window for macroanalysis. This window will present options for displaying individual event information with graphical and textual representation (i.e., event description, highlights, possible choices, percentages. etc).



Event 4:
Seconds waiting for your friend

Highlights:
H1 - Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam,.

There are three possible answers:

- Answer 1: "cat".
Selected by: 50.00%

Answer 1: "dog".
Selected by: 50.00%

Figure 21: Single Event View window for microanalysis.

Single Event View template file location:

a. components/pages/Micro/EventView.vue (for Microanalysis)
b. components/pages/Macro/GroupEventView.vue (for Macroanalysis)

Code Instruction: Some of the important functions from the Script section:

a. data() : it store data about result, description, highlights, answers, chart data, chapters, choice, decisions, game, version, chapter, mean value, Standard Deviation and many more.
b. mounted(): use Axios library to get API call descriptions/games.
c. home(): redirect to ‘home’ window.
d. back (): redirect to previous window.
e. selectGame() : select game id.
f. selectChapter(): select event id.
g. selectChoice(): select event type , multichoice or temporal.
h. dataAnalysis(): analysis data based on event type multichoice or temporal.
i. barChartLoad() : represent data into bar graph or scattered plot.
j. exportOpt(): print report as pdf file.
k. changeChapter(): select chapter from dropdown menu.
l. changeEvent(): select event from dropdown menu.

m. loadData(): load and return event data e.g., highlights, event description, possible choices etc.
n. getMeanValue(): calculation of mean value.
o. getStd(): calculation of standard deviation.

5.3.7 FILTER OPTION

There are two types of Filter window **(i)** Filter for microanalysis and **(ii)** Filter for macroanalysis. This window can filter events with student gender (male, female), age, group id, country, city, etc.

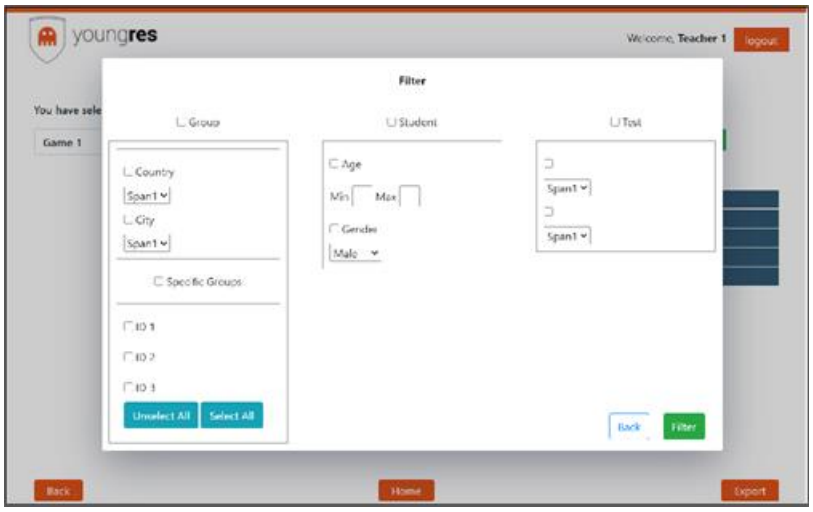


Figure 22: Filter Option pop-up window.

Filter template file location:

a. components/filterModel.vue (for Microanalysis)
b. components/ filterModelTwo.vue (for Macroanalysis)

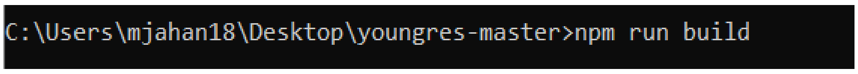
Code Instruction: Some of the important functions from the Script section as mention below::

a. data() : it handle and store data about country, city, group, age, sex, minimum age, ,maximum age, student, country list, city list, group id list, selected country, selected city and others.

b. limitCheck(): check student age limit.
c. filterUpdate(): update data based on filter applied.

6 DEPLOYMENT

To deploy YoungRes application, we need to build it first. To build, we have used the command npm run build.

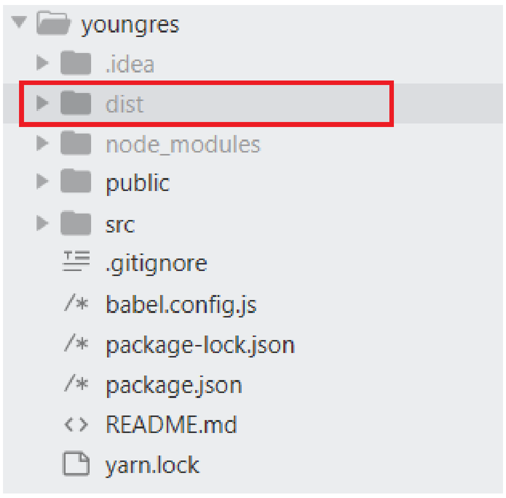


```
C:\Users\mjahan18\Desktop\youngres-master>npm run build
```

Figure 23: Command for building deployment file.

The built file will be stored in the 'dist' directory, which will be ready to be deployed. The deployment flies have only HTML, CSS, image and JavaScript files that are deployable to any standard server.

We have also built the application, and deployment files can be found here[9] in GitHub.



```
youngres
  .idea
  dist
  node_modules
  public
  src
  .gitignore
  babel.config.js
  package-lock.json
  package.json
  README.md
  yarn.lock
```

Figure 24: Deployment files available inside the 'dist' directory.

[9] https://github.com/saroarjahan/yougres_deployment_ready_files

7 CONCLUSION

This document discussed the code documentation of the visualization tool, which has been completed. Here we have described the technology used for development, development workflow, installation of the project, brief file description, and project deployment.