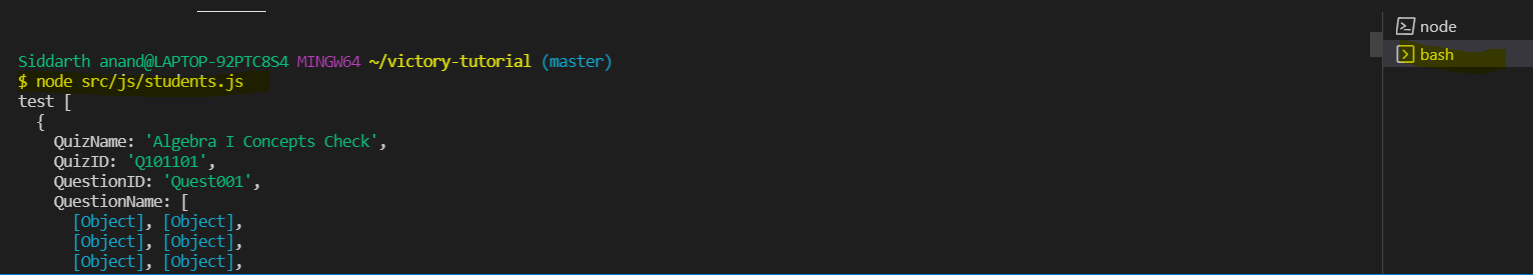
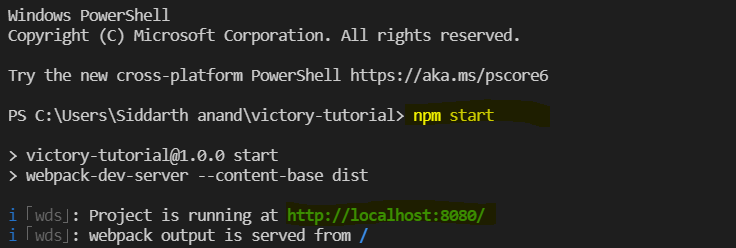
**Current ETL Process**

**Notes on ETL**

* There are two scripts: ***students.js*** (which is where the mock datasets are generated and manipulated into the required outputs), and ***client.js*** (which is where the visualizations are rendered using the outputted data)
* The outputs will be generated every time the students.js file is run. To run this script, open a Git Bash terminal, then run the command “*node src/js/students.js*” (see image below). If the file is moved to another path, change the command accordingly



* Related to the previous point, if the datasets are updated, the changes won’t be reflected in client.js until the students.js file is run again
* To run the client.js script, open a powershell window and enter the command “*npm start*”. A link should appear (http://localhost:8000/ or something similar). Opening that link will allow the user to preview the visualizations in a browser.



* If the client.js script gets modified and then saved, the page will refresh and reflect any changes made. If there is a syntax issue, the script will fail to compile and the page will be blank. If the code compiles but the page is still blank, it means that there is a logical issue (e.g., components are not compatible or the dataset that is referred to in the visualization code is written incorrectly or does not exist within the path )

**Dataset Creation**

**2.A.1 - Fields**

The idea was to have the data stored in 3 separate, linked tables / collections: Students, Teachers, and Courses.

**2.A-1.1 - Students**

**Student ID**: The unique identifier of the student

**Registration Details**: The students’ registration details, such as their registration / creation date, status (registered or not), permission levels

**Email**: The students’ email address (maybe their school address if applicable)

**Name**: The students’ name

**Age**: The students’ age

**Password**: The students password

**Last Updated**: The date the account was last updated

**Year / Grade**: The year that the student is in

**Course Details**: A subset of the course details for courses the student has signed up for. This would include common course details (Name, ID, Difficulty, Academic Term, an estimate for completion, Average Course Ratings), as well as specific course details. This latter set would consist of unit details (see Courses section for more information on the Unit Details).

**Country / City**: The country / city that the student resides in. Having this information might help LRNG.IO understand if there are differences in learning styles / lesson plans usage. It can also help LRNG.IO design templates that cater to that specific country (at this stage, the product might only roll out in Scotland, but the option to provide data for more than one country should be considered).

**Academic Year**: The academic year (e.g., 6th year, 7th year) that the student falls under. Aside from record-keeping purposes and helping educators decide the level of complexity required for the lesson plans, knowing the grade might tie in towards LRNG.IO’s gaming objectives. For example, it might be the case that the number of badges and points are higher for 6th year students compared to 7th year students, so these features could be highlighted more often for 6th grade students and less often for older students.

**School**: The school that the student is enrolled in. This could tie in with gaming objectives, as some schools might find the game elements more appealing than others. It could also tie into understand how much the learning styles and lesson plans differ from school to school.

**Demographic Data**: This could include income levels, qualification levels of parents / guardians, ethnicity, gender, postcodes or similar fields to capture location, gender and so on

**2.A.1.2 – Teachers (currently unused)**

**Teacher ID**: The unique identifier of the teacher

**Name**: The teachers’ name

**Email**: The teacher’s e-mail address

**Age**: The teacher’s age

**Password**: The teacher’s password

**Registration Details**: The teacher’s registration details, such as their registration / creation date, status (registered or not), permission levels

**Last Updated**: The date the account was last updated

**Qualification Details**: A list of all of the teacher’s qualifications (e.g., degree, highest qualification level, years of experience)

**Ratings (Average / Median)**: Average / median student feedback ratings of the teacher

**Course Details**: A subset of the course details. The teachers would only have common course details (Name, ID, Difficulty, Academic Term, an estimate for completion, Average Course Ratings)

**2.A.1.3 - Courses**

**Course ID**: The unique identifier of the course

**Name**: The name of the course

**Term**: The academic term that the course is offered in (e.g., 2020-2021)

**Course Rating**: The average course rating

**Time Needed to Complete (Approx.)**: The approximate time needed to complete the course materials

**Difficulty**: The average difficulty rating for the course

**Unit Details**: A nested field with the details of the course units. A course can have more than one unit. Each unit would have the following fields: Unit ID, Name, a Read Status flag to mark completion, time spent on the unit, teaching media (text, audio, visual), and nested quiz/assignment details (e.g., name of the assignment / quiz, quiz ID, the grade the student received, the time spent completing the assignment, a mastery / proficiency level based on the score, and misc. quiz details such as the questions, the answer choices per question, the selected answer, and the correct answer.)

**Victory Notes**

**Getting Started**

1. Install NPM and Node.JS (if you have not done so): Go [here](https://nodejs.org/en/download/) and follow the instructions to install Node.JS and NPM
2. Install Victory: Follow the steps [here](https://formidable.com/open-source/victory/docs/). Run “*npm install*” and “*npm install victory*” in the powershell window to install Victory packages
3. Import React/Victory Packages: Write the following lines (“*import React from ‘react’*”, “*import ReactDOM for ‘react-dom’*” and “*import \* as V from ‘victory*’”) at the top of the relevant JavaScript file (in this case, client.js) to import the React and Victory packages

**Introductory Notes on Victory Syntax**

1. The key principle behind Victory is that you wrap components inside other components in order to render them.
2. Within each component, you can change certain parameters. Examples of parameters include fill colour, size of the data points, stroke colour, label values/formatting, spacing between the data and axis, etc. You can also nest components within other components (e.g., a labelling component under a Chart component).
3. Any parameters that you apply within a nested component only apply to that component. For example, if you change the size parameter within VictoryBar, it only applies to VictoryBar and not VictoryChart.
4. If there are parameters/components that are only compatible with certain components, it will throw an error (i.e., it will highlight the line(s) in **red**) if you add that parameter to non-compatible components. For example, if a VictoryLabel component is only usable with VictoryChart and not usable within VictoryPie, it will throw an error if you write the parameter within the VictoryPie component.

Parameters (Fill, Grid, Stroke, Labels, Dataset, etc.)

Chart of Choice (Bar Plot, Scatter, Histogram, etc.)

Victory Chart

**Key Victory Components to Use**

***VictoryChart***: This is the main component that you use to render visualizations. Wrap your chart type under VictoryChart. NB: some parameters can only be applied to VictoryChart (e.g., a Title, Axis Formats, orientation of bars in a bar chart, etc.).

**Standard Plotting Components**

***VictoryBar***: Lets you make a bar chart

***VictoryBoxPlot***: Lets you make a box and whisker plot

***VictoryHistogram***: Lets you make a histogram

***VictoryScatter***: Lets you make a Scatterplot

***VictoryPie***: Lets you make a Pie Chart

***VictoryArea***: Lets you make an Area Chart

***VictoryStack***: Lets you stack multiple charts together in the same view. Useful for making visualizations like Stacked Bar Charts/Column Charts

***VictoryPolarAxis***: Lets you make polar charts

***VictoryGroup***: Similar to Victory Stack

**Components That Can Be Added or Modified**

***VictoryLabel***: Lets you add text. Most common use case is adding a Chart Title

***VictoryAxis***: Lets you add and modify the axes. You can nest two VictoryAxis components under the same chart, and use the dependentAxis parameter (explained below) to specify which axis you are modifying/listing

***VictoryTooltip***, ***VictoryZoomContainer***: Lets you add interactive components, such as showing datapoints when you move your mouse over the data point, zooming in or out, etc. (tooltip as per my initial exploration seems to be compatible with certain chart components)

***VictoryLegend***: Lets you add a chart legend

**Standard Parameters**

***colorScale***: Lets you list a group of colours to apply to your chart. Useful when there is more than one category of data

***grid***: Lets you format the grid (e.g., gridline colours per axis, rules for displaying gridlines, etc.)

***labels***: Lets you format how the chart labels look

***size***: Lets you format the size of your chart elements (e.g., scatterplot datapoint size)

***stroke***: Lets you change the border colour. Can be used to stylize the grid and labels as well

***strokeWidth***: Lets you change the border thickness. Can be used to stylize the grid and labels as well

***fill***: Lets you change the colour of your visualization elements (e.g., bar colours in a bar chart, scatterplot points in a scatterplot, etc.). Can be used to stylize the grid and labels as well

***fillOpacity***: Lets you change the opacity of the fill. Lower values indicate higher transparency

***style***: Lets you encompass formatting elements such as fill, stroke, fillOpacity, etc.

***dependentAxis*** (only for VictoryAxis): Used to classify the VictoryAxis component as the y-axis. All modifications to parameters will then be applied solely to the y-axis

***domainPadding***: Determines how much further away from the axis the data is

theme: Applies a format for the chart, such as the font type of the text

***fontSize***: Lets you specify the size of a font. Useful for labels, titles, tick values, etc.

***tickValues***: Lets you specify what values should be displayed on the ticks

***tickFormat*** (only for VictoryAxis): Lets you specify how the tick values are displayed (e.g., round to the nearest integer, only showing values that are divisible by 10, etc.)

***width***: Lets you specify the chart width

***height***: Lets you specify the chart height

**Visualization Technical Specifications**

The specifications are listed by **learning outcome and associated visualizations**. They are meant to explain **what parameter values were used, and what components / parameters were modified during the course of the project**. The tables contain the original specifications, as well as any updates made following the discussion with LRNG.IO (highlighted in **BOLD**).

**Terminology and Notes**

**Visualization/Chart Type**: Examples include stacked bar charts, pie charts, box and whisker plot, etc.

**Fill**: This refers to the fill colour of the primary chart components. The component types vary by chart type (e.g., for a scatterplot, the fill would refer to the colour of the datapoints; for a bar chart, the fill would refer to the colour of the bars).

**Stroke/Stroke Width**: Stroke refers to the colour of the border. Stroke width refers to the thickness of the border.

**Colour Coding**: The colours used for the fills and strokes and their respective hexadecimal values have been provided.

**Category 1: Grade Distribution**

**Outcome** ***(Teacher)***: To help teachers check how their students performed on a quiz / assignment for a particular unit/course. This could be used to assess whether they have the right level of difficulty

**Outcome** (***Student)***: To help students track how they performed on a quiz / assignment compared to their peers. The student view should also highlight the student’s individual grade/score so that they can easily see where they rank compared to their peers

**Visualization(s)**: Scatterplot (both a student and teacher view), Box and Whisker Plot (both a student and teacher view), Histogram (see Table 1.1 for relevant specifications)

**Table 1.1 - Grade Distribution Visualizations Specifications**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Visualization** | **Data Presented** | **Features** | **Colours** | **Misc.** |
| ***Scatterplot (student view and teacher view)*** | Students IDs are plotted on the x-axis, scores are plotted on the y-axis | Dynamic scaling on the y-axis is necessary to ensure that multiple datasets can be used, and to ensure that the overall findings/insights are not distorted (e.g., lowest values can be seen as much worse than they actually are due to a small scale). The current formula that I am using to adjust the axis scale is:  *Minimum axis value: Minimum score\*0.99*  *Maximum axis value: Maximum score\*1.01*  **UPDATE: Y-AXIS RANGE HAS BEEN CHANGED FROM 0-100 SO THAT USERS CAN COMPARE QUIZZES MORE EASILY.** | The scatterplot points have a size of 5, with a maroon fill (#C43A31) and a black stroke (#000000) of width 1.  For the student view, their score is highlighted using a magenta fill (#FF00FF) with a purple stroke (#800080) of width 2 to make it stand out more easily. | The dataset is first filtered by Quiz ID/Quiz Name  **UPDATE: AN INTERACTIVE ZOOMING COMPONENT HAS BEEN ADDED, SO THAT USERS CAN ZOOM IN AND OUT ON THE DATASET OF INTEREST.** |
| ***Box and Whisker Plot (student view and teacher view)*** | The quiz name is displayed on the x-axis, and the scores are displayed on the y-axis | The y-axis range is automatically fitted | The boxes have a light blue fill (#ADD8E6), and the whiskers have a black fill (#000000). Every component has a stroke width of 4. | The student view has data for one quiz at a time. The teacher view has quiz data for a given course. |
| ***Histogram*** | The quiz scores are plotted on the x-axis | The y-axis range is automatically fitted | The bars have a dark pink fill (#BC5090), and a black stroke (#000000) of width 1 | Dataset is first filtered by Quiz ID/Quiz Name |

**Category 2: Progress Tracking**

**Outcome (*Student*):** To help students track their progress and compare it to their peers

**Outcome (*Teacher*)**: To help teachers track how many students are going through the materials. This can be used to identify students that are lagging behind (if any)

**Visualization(s)**: Progress Bar (individual student), Average Progress Bar (all students), Circular Progress Bar (individual students) (see Table 1.2 for relevant specifications)

**Table 1.2 - Progress Tracking Visualizations Specifications**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Visualization** | **Data Presented** | **Features** | **Colours** | **Misc.** |
| ***Progress Bar (individual student)*** | The Student ID for a particular course is plotted on the y-axis. There are two values plotted for the specified student: completed progress and remaining progress | The completion percentage is displayed on the right of the remaining progress bar in order to account for formatting consistency as the values increase over time. | The fills used are green for completed progress and light green for remaining progress (#008000 and #90EE90 respectively) to convey the image of a bar being filled. Both bars have a black stroke (#000000) of width 1. | A stacked bar chart has been used to make the progress bar visualization  **UPDATE: HOVER FUNCTIONALITY HAS BEEN ADDED TO THE PROGRESS BARS. WHEN THE USER HOVERS OVER THE BARS, THEY CAN ALSO SEE THE EXACT NUMBER OF UNITS READ. THE DATA THAT IS PRESENTED CAN BE CHANGED IN THE FUTURE.** |
| ***Average Progress Bar (all students)*** | All course IDs are plotted on the y-axis. There are two values plotted for each course: completed progress and remaining progress | The completion percentage is displayed on the right of the remaining progress bar in order to account for formatting consistency as the values increase over time. | The fills used are green for completed progress and light green for remaining progress (#008000 and #90EE90 respectively) to convey the image of a bar being filled. Both bars have a black stroke (#000000) of width 1. | A stacked bar chart has been used to make the progress bar visualization.  In future, median progress completion rate can be implemented as well |
| ***Circular Progress Bar (individual student)*** | The Student ID for a particular course is displayed in the Title, with the Course Name. There are two values plotted for the specified student: completed progress and remaining progress | The completion percentage is displayed in the middle of the chart | The fills used for completed progress is green (#008000). No fill was chosen for the remaining progress. This was done to make the chart easier to read and to give more focus to completed progress.  **UPDATE: THE REMAINING PROGRESS PORTION OF THE CHART NOW HAS A LIGHT GREEN FILL (#90EE09).** | A pie chart/donut chart was used to make the circular progress bar visualization. |

**Category 3: Question Answer Visualizations**

**Outcome (*Teacher*):** To help teachers assess whether the questions are designed to the educator's / teacher's expectations, and to identify potential flaws in the design of the answers/questions

**Visualization(s)**: Pie Chart, Bar Chart (see Table 1.3 for relevant specifications)

**Table 1.3 – Question Answer Visualizations Specifications**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Visualization** | **Data Presented** | **Features** | **Colours** | **Misc.** |
| ***Pie Chart for Question Answer Selection*** | A quiz question is assumed to have four answer choices: Answers A, B, C and D. The chart contains the number of students that selected each answer for that given question. The Quiz Name and Question No. are displayed on the title | The total number of selections per answer is displayed on top of each section, next to the Answer Selection (e.g., A:12) | 4 distinct colours were used for each answer: very dark blue for A, navy for B, purple for C and red for D ("#003F5C", "#58508D", "#BC5090", and "#FF6361" respectively). The colours were chosen to make the categories easily distinguishable (i.e., provide contrast). The pie chart has a black stroke (#000000) of width 1. | The quiz data is filtered at quiz/question no. level before being fed into the visualization.  **UPDATE: THE CORRECT ANSWER HAS BEEN HIGHLIGHTED IN INDIGO (#4B0082), WITH A FONT SIZE OF 17 INSTEAD OF 13 TO MAKE IT EASIER TO SEE.** |
| ***Bar Chart for Question Answer Selection*** | A quiz question is assumed to have four answer choices: Answers A, B, C and D. The chart contains the number of students that selected each answer for that given question. The Quiz Name and Question No. are displayed on the title | The total number of selections per answer is displayed on top of the bars, next to the Answer Selection (e.g., A:12) | The bars all have a maroon fill (#C43A31), and a black stroke (#000000) of width 2. | The quiz data is filtered at quiz/question no. level before being fed into the visualization. |

**Category 4: Course Feedback Visualizations**

**Outcome (*Teacher*):** To help teachers identify any areas of concern with respect to course quality, access, difficulty, etc.

**Visualization(s)**: Radar Chart (see Table 1.4 for relevant specifications)

**Table 1.4 – Course Feedback Visualizations Specifications**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Visualization** | **Data Presented** | **Features** | **Colours** | **Misc.** |
| ***Radar Chart for Course Feedback Scores*** | The idea is that a course has various parameters (e.g., quality, difficulty, access to materials, etc.), each scored by the students at the end of the course. The chart shows the average score for each parameter for a given course. | For the sake of simplicity, each parameter has been assigned a value from 1 to 5. | Both the fill and stroke are gold (#FFD007). The fill has an opacity of 0.2 to make it easier to see the axes and the scores. | A combination of an area chart and polar chart was used to design this visualization. The scores are filtered by Course Name before being fed into the visualization. |

**Next Steps**

1. Similar coding logic can be used at different levels of granularity (quiz level, skill level, etc.) or aggregated at different levels as well (e.g. multiple classes, etc.)
2. In the long-run, the data needs to be shifted to a database such as MongoDB
3. Filters are hard-coded in the script. In the future, they need to be selectable by the user via a feature such as a drop-down menu.
4. Every visualization is stored on the same page. In future, these need to be stored on different pages.
5. Certain visualizations were not created due to time constraints, namely the skill proficiency chart. This is one of the designs that should be added as soon as possible. Another view that would be useful to build would be the user journey view, with emphasis on making it engaging and informative (listing skills acquired).
6. As LRNG.IO starts to collect more data on usage, it should consider adding the most often used visualizations in a default dashboard, with various customization options available per user