MINISTRY OF EDUCATION, CULTURE AND RESEARCH OF REPUBLIC OF MOLDOVA
TECHNICAL UNIVERSITY OF MOLDOVA
FACULTY OF COMPUTERS, INFORMATICS AND MICROELECTRONICS
DEPARTMENT OF SOFTWARE ENGINEERING AND AUTOMATICS

ChartLab: A modern approach to Data Visualization.

Project Report

Mentor: conf. univ. dr., Dumitru Ciorba
Students: Anastasia Tiganescu, FAF-231
Daniela Cojocari, FAF-231
Janeta Grigoras, FAF-231
Maxim Isacescu, FAF-231
Vladimir Vitcovschii, FAF-231

Chișinău, 2025

Abstract

The project titled ”ChartLab” was developed by students Anastasia Tiganescu, Daniela Cojocari, Janeta Grigoras, Maxim Isacescu, Vladimir Vitcovschii, from the Technical University of Moldova. This paper includes two main chapters: Problem Analysis and Research, and Implementation and Evaluation, followed by Conclusions and Bibliography.

ChartLab represents a unique DSL that has the purpose to ease the process of data visualisation and analysis. The general objectives include a feature to automatically create graphs and charts, which will not require any knowledge from the user. The domain specific language will only require a CSV, JSON or XML file, from which it will generate an appropriate graph. As tools and methodologies we use Python libraries for visualisation of our charts, Zotero for referencing, JIRA for task management and Latex for document formatting.

Keywords: chart, graph, automatization, visualisation, analysis.

Content

Introduction

This is ChartLab, an innovative solution designed to modernize the approach to data visualization and analysis. This paper provides an in-depth exploration of ChartLab's key features and its potential to transform the way data is being viewed and interpreted.

We begin by addressing the prevalent issue of inaccessible and ineffective tools for designing correct and visually appealing charts, despite the growing need for data visualization. ChartLab emerges as the solution to this challenge, providing customizable graph representation, and dynamic data processing to enhance the usability and accessibility of data visualization. The Target Group section categorizes our audience into 5 main groups: data analysts, scientists, managers, software engineers, and students. User personas like John, Henry, Olivia, Jacob, and William offer insightful feedback guiding ChartLab's feature development to cater to each group's unique needs and motivations. We then go into John's story, a main data analyst who is currently a part of a small side project of a big company. Through John's experience, we show how ChartLab's feature enable John to improve his skills and maximize his productivity. Next, we compare ChartLab's features to already existing data visualization tools, bringing out its advantages.

In the next chapter, we present the overall structure of our project. We start by describing the DSL's components. The presented grammar includes chart commands for 9 types of visualization options: bar chart, grouped bar chart, stacked bar chart, line graph, histogram, area chart, scatter plot, bubble chart, and pie chart. Using ANTLR helped us to build the lexer and parser efficiently. Last but not least, the interpreter has access to two components: reader and plotter, which are useful for extracting the necessary data from files and rendering different chart types.

Moving forward, the web components section describes the server, which is based on Flask, and the client part, which is represented by a webpage written with the help of Nextjs. We finish with a system overview, which includes a visual representation of all the main parts of ChartLab.

Problem Analysis and Research

Problem Description and Problem Analysis

Problem Description

Our world is ruled by data, it is what we collect as a result of observing the world and all of its facets, and data visualisation could be considered a shared language between people, businesses, organisations, nations. It aids in sharing the data and the knowledge gained from it. However, despite its importance in today's world, we still aren't proficient in "speaking" this shared language, or we can't make it accessible to everyone. Why is that such a problem?

Firstly, data is everywhere. Not only this, but according to Scott Berinato in his book Good Charts, we can actually call information, nowadays, a "megacosm" - an entire universe of data we have to handle. But data means nothing when we don't have the fitting tools to convert raw numbers into useful insights [1]. Therefore, the right visualization tools are essential to help us visualize and interpret data as efficiently and, preferably, as aesthetically-pleasing as possible.

Moreover, according to Stephen Few in his article "Data Visualization Past, Present and Future", data visualisation is still largely misunderstood, and apparently, "too often undermined by the very vendors that produce and sell visualization software"[2]. The very goal is to clearly display information for the understanding of as many people as possible. The book "Graphing Statistics and Data", says that the process of creating charts or graphs is an art in itself, having really specific rules on the basic principles of visual perception and cognition[3].

Unfortunately, many of the current trends in data visualisation act in opposite of these rules. Not only this, but also much of the visualisation software existent in the market today is not accessible to each and every individual interested in work with data [4]. So, they are either too advanced and technical, hard to learn, designed to be understood only by computer graphicists and programmers, or needing immense amounts of effort from people not in these fields. They could even be too basic and not do their job well. This is what keeps researchers or any other people from this technology.

To conclude, the role of data in today's world and the need for the right tools to handle this data is acknowledged. However, we recognize the lack of accessible software for designing accurate charts and graphs, which is a true problem in this day and era.

Domain Analysis

A DSL is a programming language focused on a specific problem domain, which it aims to solve through appropriate notations and abstractions. In the study "Domain-Specific Languages: A Systematic

Mapping Study" DSLs are considered to be a flourishing area of study in the field of Software Engineering, and significant components of "software development methodologies such as: Generative Programming Product Lines, Software Factories, Language-Oriented Programming, and Model-Driven Engineering" [5].

One might argue that there is no need to introduce an entire new DSL for solving a problem, when there are alternatives such as XML - which translates data in a machine to a form more human-readable - or JSON - which is less verbose than XML, since XML is considered to be more machine-readable than human-readable - according to the book "Implementing Domain-Specific Languages with Xtext and Xtend" [6]. However, DSLs provide optimizations that general-purpose languages lack. DSLs are prone to less errors [**?**], lead to increased productivity, and can help in writing code faster. Moreover, according to [7], DSLs are "small, more declarative than imperative, and more attractive than General-Purpose Languages (GPL)".

Hundreds of DSLs exist today, in various domains. Figure 1.1 highlights the range of domains DSLs has a significant impact on, but they are even more widespread than that: science, medicine, robotics, and finance. Some of the most well-known ones are HTML, BNF, SQL, MATLAB, GNU Octave, etc [8].

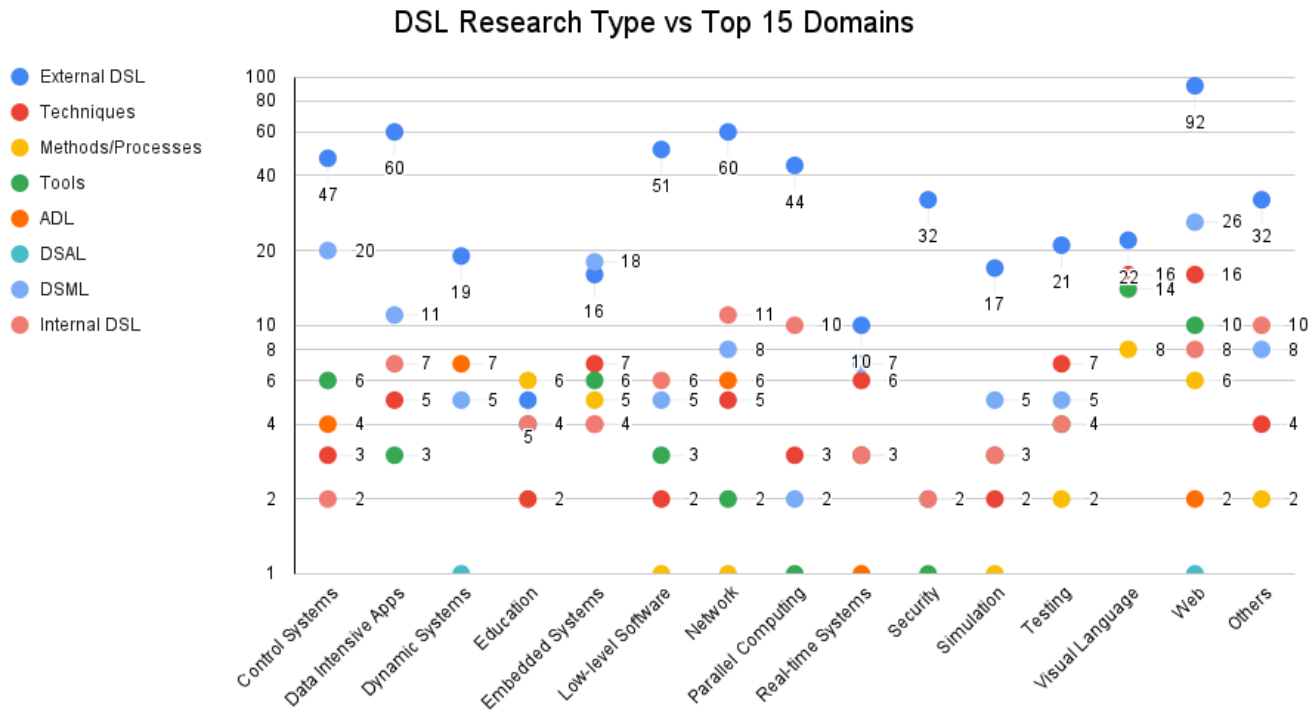

Figure 1.1: DSL Research Type VS Top 15 Domains

The most common tools for data visualization are bar graphs, line graphs, pie charts and network graphs. But there are many lesser-known ways in which you can represent the data more interactively and unusually, like: area graphs, waterfall charts, gauge charts and funnel charts.

According to James A. DuPlass in his article "Charts, Tables, Graphs, and Diagrams: An Approach

for Social Studies Teachers" students who were presented material with and without graphic displays provide convincing evidence that comprehension was improved for those who were taught with graphics [9].

A study by Kennedy and Hill (2017) revealed that data visualizations awaken a wide range of feelings in people who engage with them [10], therefore it is essential to have a tool for constructing data efficiently.

While there is no specific data about how many people have created a graph or chart at least once, it is undeniable that they are widely used in schools by both teachers and students, in economics, in mathematics and in science.

Moreover, data is pervasive in modern society. The International Data Corporation (IDC) predicts the "Global Datasphere" will grow from only 45 Zettabytes in 2019 to 175 Zettabytes in 2025 [11]. This significant increase leads to difficulties in processing and analyzing data. On top of that, a study conducted by Richard Alleyne reveals that we are now exposed to 5x more data daily than in 1985 [12]. As a result, this data overload creates an imminent obstacle in driving meaningful insights, and can lead to missed opportunities and misinformed decisions.

Another point worth mentioning is that data stored in charts is easier to understand and is more visually appealing to the human eye. A study entitled "A Mind's Eye" done by researchers at the University of Rochester reveals that more than 50% of the human brain is devoted to processing visual information [13]. This, once again, highlights that information presented visually is more likely to be comprehended and retained. Furthermore, in an analysis titled "Blinded with science: Trivial graphs and formulas increase ad persuasiveness and belief in product efficacy," authored by Aner Tal and Brian Wansink, it is mentioned that when a scientific claim is presented in words 68% of those reading it will consider the information to be truthful. However, when the same claim is presented in a simple graph, the percentage raises to 97% [14]. So, having information presented visually not only makes the understanding of the facts easier, but also increases its credibility.

In conclusion, data visualization play a pivotal role nowadays. As the amount of data generated and consumed grows, so does the need for effective tools to present this information in a way that's accessible for each individual.

Problem Statement

This analysis led to the following problem statement: there is a lack of software for designing theoretically correct and visually pleasing charts, easy to use for each and every individual.

Solution Proposal

In response to the enlisted challenges of creating a technically accurate and readable graph, we propose the ChartLab, a domain specific language aimed for real-time chart construction. This tool provides

several features that enable the user to produce a unique and customized representation of their data.

One of the features of the domain specific language is graph type selection. The user will be able to choose from a variety of chart types, a specific one that he wants to implement, like: bar, line, pie, histogram, area, etc. Moreover, in constructing the graph, in the future, he will have the possibility to customize his graph by choosing the color, font and size of the graph.

In addition to these features ChartLab supports dynamic data processing. Whether the data is stored in a CSV, JSON or XML file, the program will automatically process the data and will transform it into a visually comprehensive chart. This implies that users can import data from external sources rather than manually inputting it. Furthermore, ChartLab provides exporting options, which consists of PNG, SVG and PDF. It allows the user to save the chart in various formats ensuring flexibility for different use cases.

In a nutshell, the domain specific language - ChartLab is an optimal tool for graph creation, that allows the user to have a unique representation of their data and dynamic data processing. Together, all these features enhance the creation of a technically accurate and visually pleasant chart.

Figure 1.2: ChartLab Logo

Data Analysts - this group is the most interested in qualitative data representation. They have got many different tools to solve some of the issues, but the tools require a higher knowledge or are not flexible and not generalized [15]. That is why our solution strongly focuses on this target group.

Scientists - share the same target as the data analysts, but with the results of their calculations in a clear and easy to understand chart. The graphs can significantly increase the speed of experiment processing [16].

Managers - utilize data visualization for decision-making, reporting, and strategy. They may not need any statistics compared to data analysts but need sharp and interactive graphs that reveal the insights. Therefore, it is extremely essential to have a tool which will facilitate extracting clear insights.

Software Engineers - need data visualization tools to debug, monitor system performance, and view logs. A data visualization tool integrated in their development workflow can contribute significantly to increased productivity, as it will help them to discover system bottlenecks, trace errors, and explore trends in real time.

Students - require simple yet effective charting tools for academic projects and presentations. The learning experience could greatly be affected by the visualization of materials [17].

User personas

John represents the target group of data analysts. His goal is to ease hardships with visualisation of the data and shorten the time that he spends on the work but increase efficiency.

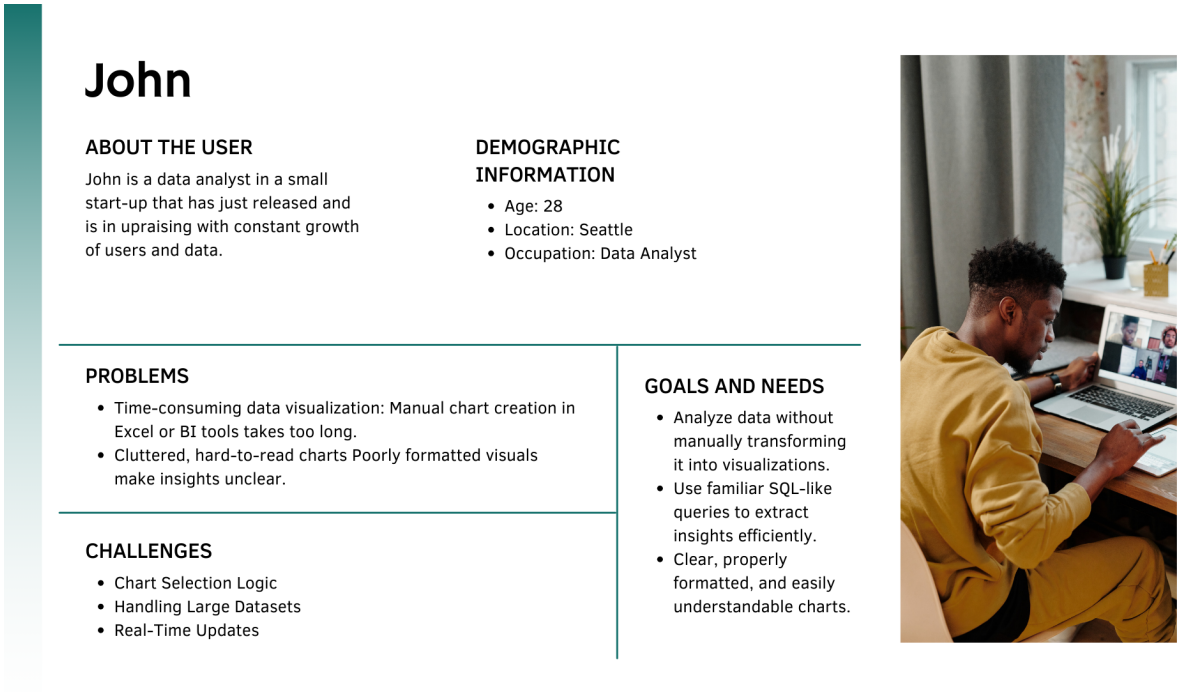

Figure 1.3: User Persona 1: John

Henry represents the target group of scientists. His goal is the visualization with the data he obtained through calculation and simulation right away, for further analysis.

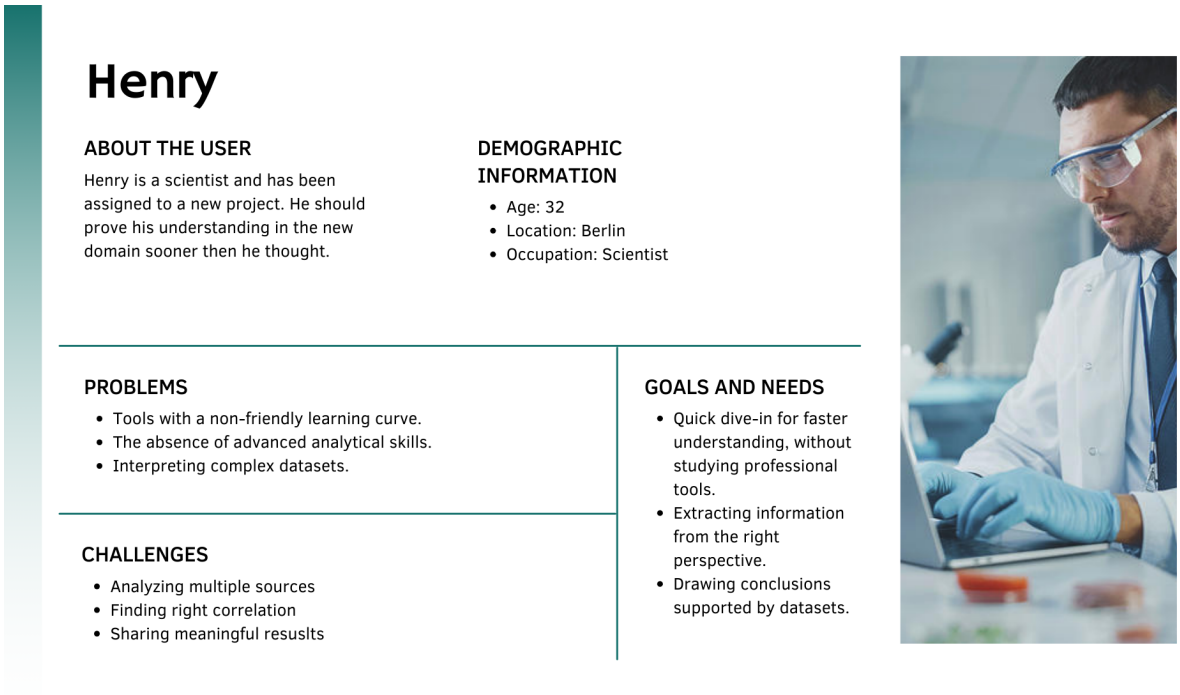

Figure 1.4: User Persona 2: Henry

Olivia represents the target group of managers. Her goal is to track the trends within the company and visualize them.

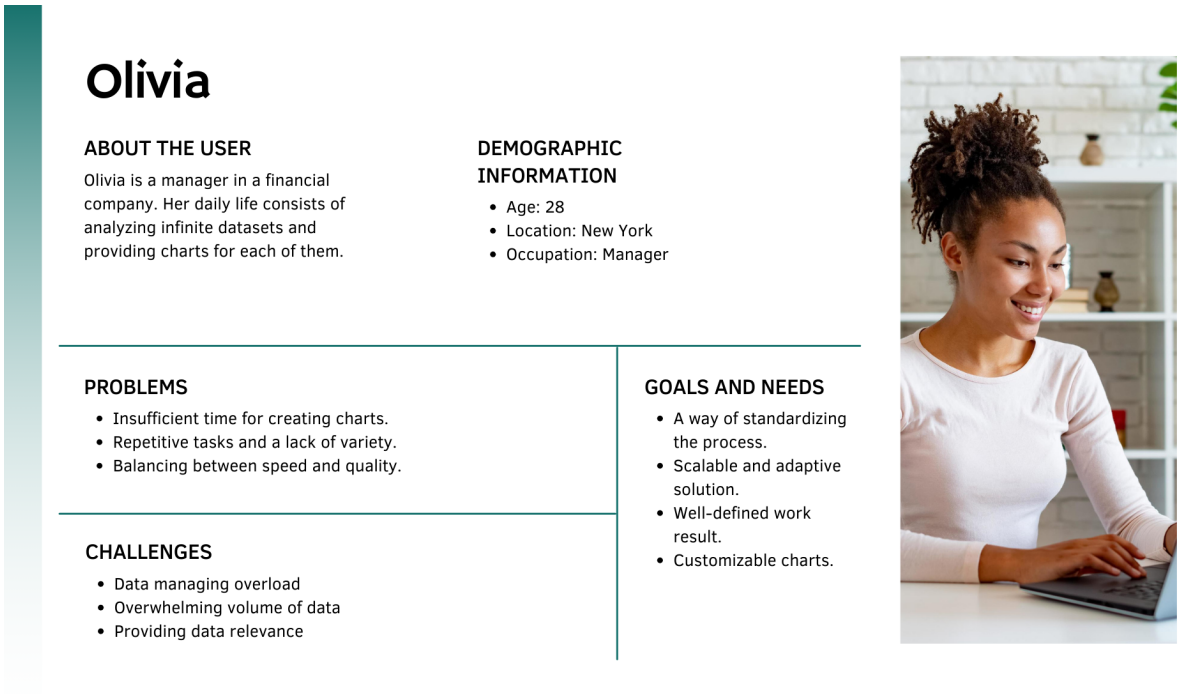

Figure 1.5: User Persona 3: Olivia

William represents the target group of software engineers. His goal is to use data visualization to track performance, fix bugs, and analyze logs, ensuring stable and scalable applications.

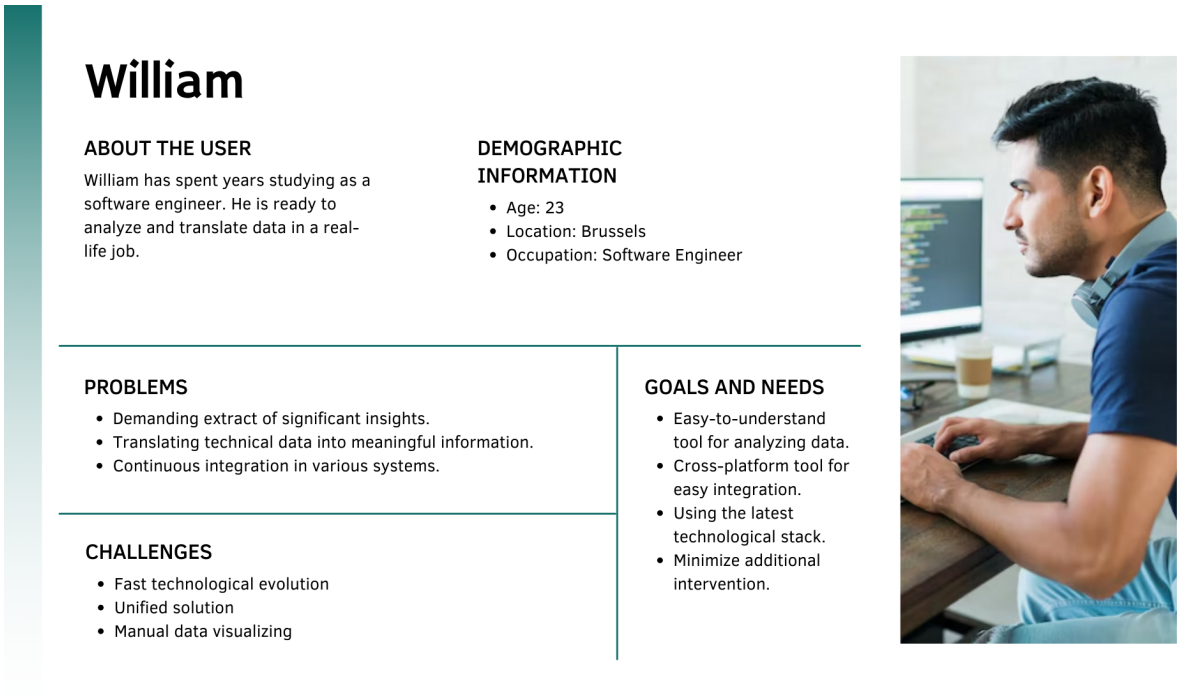

Figure 1.6: User Persona 4: William

Jacob represents the target group of students. His goal is to obtain experience by working with data and making it clear for his future studies or work.

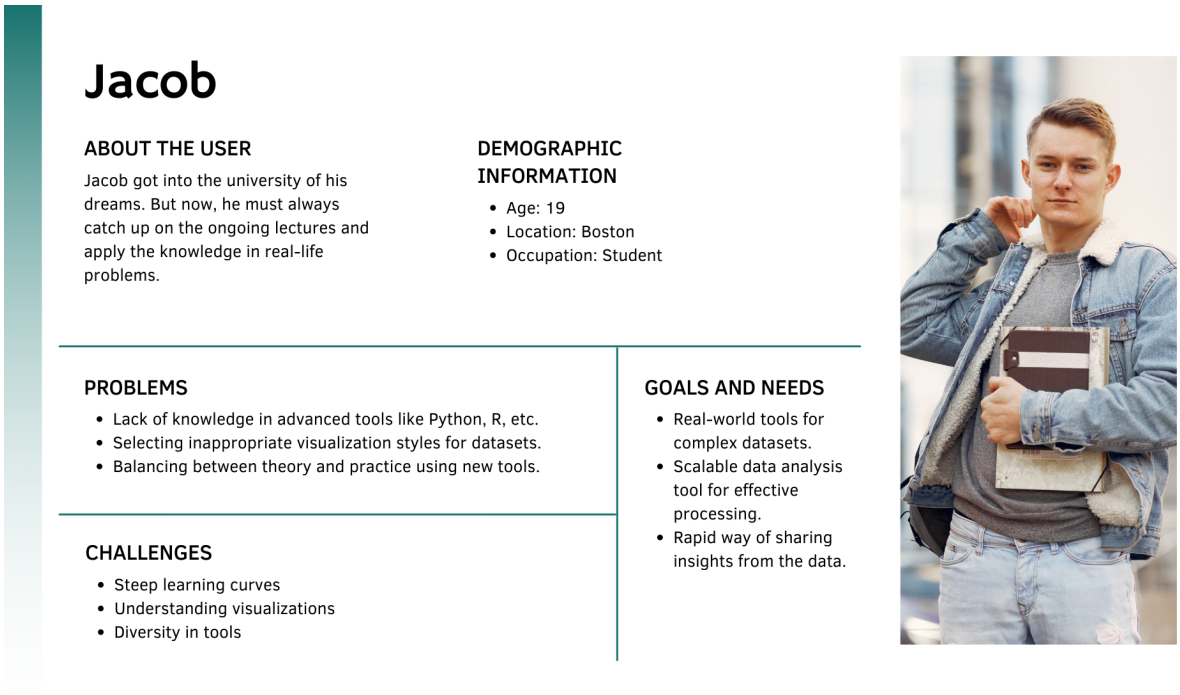

Figure 1.7: User Persona 5: Jacob

User story

Meet John – he is a main data analyst in a small side project of a big company. Lately his project has gained an increased popularity due to some circumstances. John doesn't know the reason behind this sudden growth and is tasked to prepare a comprehending data analysis.

John doesn't know why these tendencies take place and how they are related. Furthermore, he can't decide on what to begin his analysis. The results should also be easy to understand and eye catching for the audience to understand and for the company to make the strategy for the future. Finally, the analysis should be easily continued while the trends are in place.

Considering the problem, John decides to use ChartLab to give him a hint of idea through the automatically generated best in place charts that show the correlation between data. Giving the start to his analysis, it also gives him the beautiful and correct charts. Lastly, the charts can be updated by rerunning the script once again.

Comparative Analysis

Effective data analysis and visualization require the latest technologies to gain accurate insights. While any mentioned service or tool offers advanced visualization possibilities, we should focus on different aspects of using them.

ChartLab offers many features, that allow us to visualize datasets in the most convenient way. As a domain-specific language, our main competitor is Python with its libraries, such as Matplotlib, Seaborn, Bokeh, Altair, etc., which are very handy in the data analysis and visualization domain. But it has drawbacks to be considered. The offered libraries require explicit code writing to visualize even basic correlations. This highly complex way of use makes it useful only for specialized people, such as software engineers, who also handle how data is fed to the tools manually. Because of this manual data selection, the number of data processing and data cleaning tools has increased, and so has the need of another set of skills [18]. From these considerations, it's not popularized and used in businesses as much as other competitors [19]. ChartLab intends to simplify this process by using a built-in data extraction mechanism that adapts dynamically to the input data, so no additional knowledge is required.

Tableau, Redash, Apache Superset, and especially Microsoft Excel are some tools more recognized by people who intend to spend less time possible and obtain impactful results. Tableau and Microsoft Excel are the most used by data analysts and managers [20], for their flexibility and popularity. But that comes with a price, which is high complexity, as a result of its overwhelming GUI, and actual cost of the services. They are not open-source, so we are forced to accept any conditions that come with it [21]. On the other hand, we have Redash and Apache Superset, which are free to use and open-source. This gives us the possibility to modify its functionality, in case of necessity, and easily switch us to another tool without any

financial considerations. Although they satisfy most of the user needs, they suffer from a lack of community and are still hard to use, offering SQL interaction and the same overwhelming GUI.

Speaking of customizable visualization, Excel provides only a basic set of charts , while Tableau and Apache Superset promote interactive dashboards, which are expandable, but introduce a new layer of abstraction on the use of the services [22][23][24].

ChartLab takes the best features from both ”worlds”. It is open-source and free to use, so anyone can use it for any use, and the GUI itself is cut completely. Our main goal is to offer our users an easy way of interacting, without the painful number of buttons on the screen, and so our single way to create and customize the visual part is the DSL, which supports multiple kinds of charts, user-friendly syntax, and even a help mechanism for selecting the right visualization. All of these would be available online so that the user won't get hands dirty with any installation processes and unknown errors. Choosing the right solution is crucial for effective analysis in situations when we aim to achieve specific goals, and so we need a specific approach to materialize our vision.

	ChartLab	**Microsoft Excel**	**Tableau**	**Redash**	**Apache Superset**
Accesibility	GUI + DSL	GUI + DSL	GUI	GUI	GUI
Multiple Data Sources	✓	✓	✓	✓	✓
Flexible/ Customizable	✓	✓	✓	✓	✓
Simple to use	✓	✗	✗	✗	✗
Licensing and Cost	Open-source and Free	Proprietary and Paid	Proprietary and Paid	Open-source and Free	Open-source and Free

Table 1 - Comparative Analysis

Implementation and Evaluation

DSL Components

Grammar

The grammar for our Domain-Specific Language (DSL) defines a syntax that allows users to easily generate data visualizations using accessible natural language commands.

Design Decisions and Structure

The grammar was developed in EBNF format and it follows a structured approach, starting with the root rule.

The root rule $< program >$ consists of one or more $< statement >$s followed by EOF. A $< statement >$ can either be a charting command or a control structure like a loop.

```
<program> ::= <statement>+ EOF

<statement> ::= <loop> | <command>
```

Each $< command >$ begins with the sequence of keywords $with...from...chart$: followed by a $< chartFunction >$, which defines the type of visualization to generate.

```
<command> ::= "with" <data> "from" <table> "chart:"   <chartFunction> [ <
    block> ]

<block> ::= "{" <statement>+ "}"
```

Key Grammar Components

1. **Chart Commands:**

   The $< chartFunction >$ rule defines 9 visualization options, depending on the visualization the user wants to access. The supported visualizations are:

   - **Bar Chart:** Compares a single variable across multiple cases.
   - **Grouped Bar Chart:** Compares a variable split into subsets across various cases. It contrasts subsets within and across individual cases, not useful for comparing total values for cases.
   - **Stacked Bar Chart:** Compares the value of one variable split into 2 or more subsets, across various cases. Best for comparing totals across cases and subsets within cases.
   - **Line Graph:** Compares continuous variables for one or more cases.
   - **Histogram:** Compares two variables, with one segmented into ranges that function like the cases in a bar graph. Best for comparing segments within continuous data sets.
   - **Area Chart:** Compares two continuous variables for one or more cases.
   - **Scatter Plot:** Compares two variables at multiple data points for a single case. Best for showing distribution of data, when there is no clear trends, or when focus is on outlying data

points.

- **Bubble chart:** Compares three variables at multiple data points for a single case. It emphasizes the relationship between the third variable(the bubbles) and the first two.
- **Pie Chart:** Shows the proportion of a single variable for multiple cases (when you compare one segment to a whole).

Each type of visualization follows a predefined syntax rule. For example, in the case of the Grouped Bar Chart, the syntax looks like this:

```
<chartFunction> ::=
                    | "compare" <var> "split by <subgroup> "for" <
                      cases>
                    | "compare" <var> "grouped by" <subgroup> "for" <
                      cases>
                    | "differences within" <subgroup> "for" <cases>
```

The derivation tree of some sample code looks like this:

```
    with sales from orders chart:
        compare revenue for regions

    with sales from orders chart:
        compare revenue for regions

    while ( x < b ) {
        with sales from orders chart:
            compare revenue for regions
    }
```

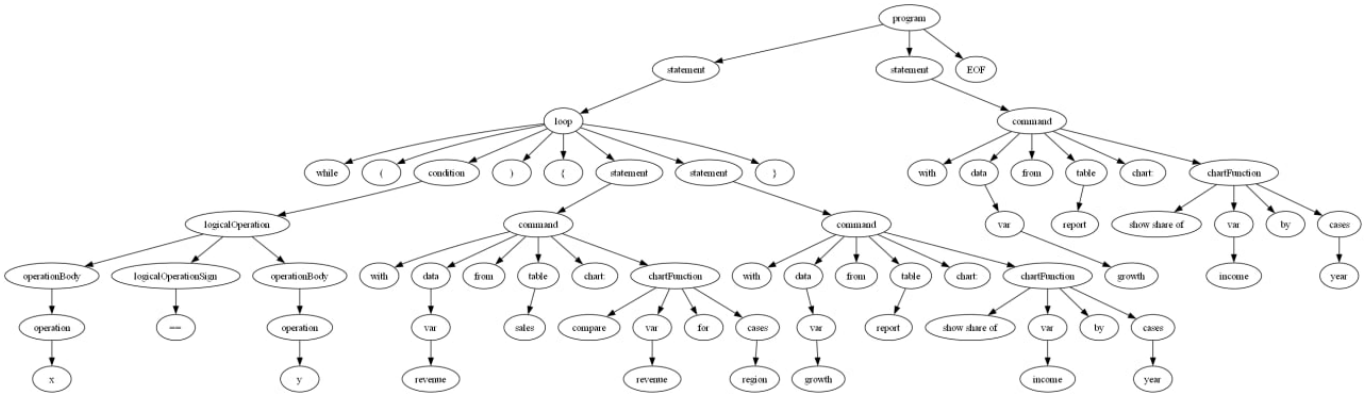

Figure 2.1: Derivation Tree Example

2. **Variables and identifiers:**
   - `<var>`(variable): represents actual columns from the dataset (the tables) given as input by the user;
   - `<continuousVar>`: represents a list of one or more continuous variables for line/area charts;
   - `<cases>`: also represents a columns from the dataset, however it usually represents categories on the y-axis in visualizations;
   - `<subgroup>`: it is also a column from a table, but used in the context of specific charts/graphs, it

enables stacked or group visualizations.

3. **Control Structures**

They enhance our DSL's flexibility.

- Loops make the language flexible and allow charting logic to repeat under specific conditions.

```
<loop> ::= "while" "(" <condition> ")" ":"
```

- Loops are controlled using conditions, which use logical and arithmetic operations:

```
<condition> ::= [ "!" ] [ "(" ] { "!" } <logicalOperation> { ("and
   " | "or") [ "!" ] <logicalOperation> } [ ")" ]

<logicalOperation> ::= [ "(" ] <operationBody> [ ")" ] <
   logicalOperationSign> [ "(" ] <operationBody> [ ")" ]

<operationBody> ::= [ "(" ] <operation> [ ")" ] { <operationSign>
   [ "(" ] <operation> [ ")" ] }

<operation> ::= <identifier> [ <operationSign> <identifier> ]

<operationSign> ::= "+" | "-" | "*" | "/"
<logicalOperationSign> ::= "<" | "<=" | ">" | ">=" | "==" | "!="
```

ANTLR Tool

We used ANTLR to generate our parser and lexer due to the efficiency and the features it offers. ANTLR (Another Tool for Language Recognition) is a powerful parser generator that simplifies lexer and parser development. So, instead of developing the lexer and the parser on our own, we only had to convert our BNF grammar into ANTLR.g4 format and let ANTLR generate the corresponding lexer and parser. ANTLR also supports multiple target languages, which again proved to be really useful for us, using Python.

ANTLR allows defining both the lexer and parser in a single grammar file, but we decided to split the grammar into two separate ANTLR.g4 files. ChartLexer.g4 focuses only on token definitions (keywords, operators, identifiers), whereas ChartParser.g4 defines the structure and rules of actual DSL commands using the tokens in ChartLexer.g4.

Lexer

The lexer tokenizes the input DSL commands, meaning it breaks into meaningful symbols, called tokens, which are then passed to the parser.

The generated lexer maps each token to an internal numeric ID:

```
WITH = 1
FROM = 2
CHART = 3
COMPARE = 4
DIFFERENCES = 5
...
IDENTIFIER = 56
NUMBER = 57
WS = 58
```

When the lexer encounters "with" in the input, it assigns it WITH (1), "chart:" becomes CHART (3), and "revenue" becomes an IDENTIFIER (56). If the lexer encounters an unexpected symbol (e.g., @),

it throws an error. The lexer skips spaces and newlines using the rule:

```
WS = 58  # Skipped automatically
```

If an identifier contains invalid characters (e.g., ”sales-data” instead of ”sales_data”), the lexer rejects it.

Parser

The parser takes the tokenized input from the Lexer and checks if it follows the correct syntax. If it does, it constructs a parse tree that represents the input's structure. If not, it reports an error. It expects every command to follow this structure:

```
with <data> from <table> chart: <chartFunction>
```

This ensures:

- ”with” always starts the command.
- ”from” specifies the data source.
- ”chart:” introduces the type of visualization.

The parser analyzes the token stream and creates a structured parse tree.

Interpreter

The interpreter has three main parts and is interacting with another two components. The interpreter is divided into the following parts:

- **Tree traversal:** recursively visits each node.
- **Node interpretation:** decides what node type we are working with.
- **A set of interpretation methods:** each method will interpret a specific node type

Each of the interpretation methods have access to two components which help with specific tasks:

- **Reader component:** following the Adapter design pattern, it extracts the necessary data. For now, our target interface is CSV, but later, we'll shift to a table-based schema approach. This transition will allow us to implement other specific formats (adaptees) smoothly, such as JSON or XML. Our Reader component, the adapter, processes the data and passes it to the interpreter component (client).
- **Plotter component:** responsible for rendering different charts type using external visualization libraries.

  The Plotter component now offers full chart support: all 9 charts mentioned in the Solution Proposal are now implemented and supported. Each chart type follows a coherent and clear pattern of how commands must be written and how they generate output, according to our updated DSL grammar.

  During the realisation of our project, we also switched from static plots using Seaborn to fully interactive ones using Plotly. This now allows viewing detailed data point information by hovering over the chart and adapting the chart to the user's liking for a deeper exploration (zoom, dynamic resizing, etc.)

Web Components

Server

The server is based on Flask, a back-end framework on Python. It provides a RESTful API for communication to the client and smoothly implements additional features, such as loading and saving templates, loading data input, and running scripts.

Client

The Client part is represented by a webpage written on Nextjs with the use of TailwindCSS for styling. The UI library that was mostly used for components is Headless UI. It is a one-page web application with the following components: Text Editor, Run Button, Add Files Button, Templates, Output Screen.

After adding a CSV file and writing code, on the click of the `Run` button, the request will be sent to the server. The response will be the image displayed in the output screen.

System Overview

The client component handles user interaction. The requests arrive in the main component through the server component, where the language processing starts. ANTLR component provides lexer and parser, which generate the Abstract Syntax Tree and is passed to the interpreter component. By traversing the tree, the interpreter is helped by Reader and Plotter components, which execute extracting data and rendering plots. Described dataflow can be observed in the component diagram in Figure 2.2.

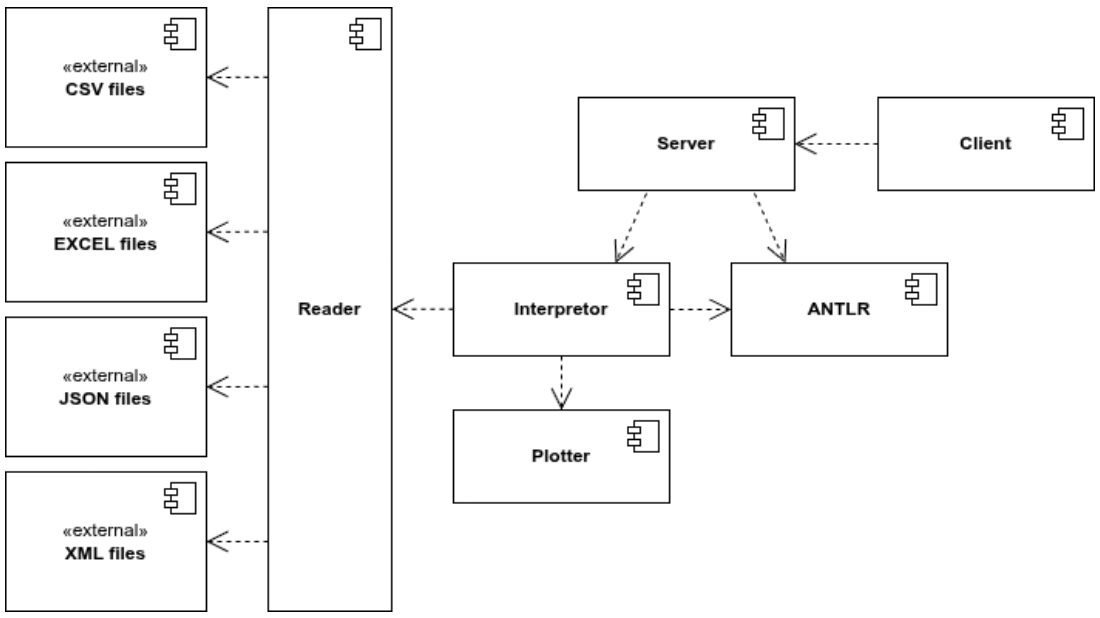

Figure 2.2: Component diagram of the ChartLab system

Implementation results

To evaluate the final product, each supported chart type was tested by writing DSL commands and generating results based on sample datasets. Below are some examples of the results obtained.

1. **Example 1: Bar Chart**

   The bar chart was obtained using this command:

```
with sales from orders chart:
   compare revenue for regions
```

   The output generated by the command is displayed in Figure 2.3. It illustrates a bar chart comparing revenue across distinct regions. Each bar represents one category, and its height corresponds to the value associated with that category. This visualization enables direct comparison of magnitudes between groups, highlighting differences or similarities within the dataset.

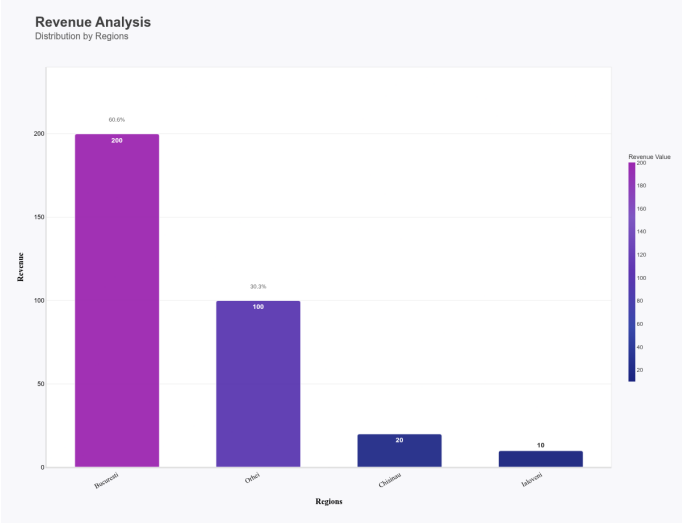

Figure 2.3: Bar Chart

2. **Example 2: Grouped Bar Chart**

   The grouped bar chart was obtained using this command:

```
with data from industry chart:
    compare salary split by gender for company
```

   The resulting chart is presented in Figure 2.4. The grouped bar chart compares salaries across different companies, emphasizing the differences between male and female salaries in each company.Thus, it highlights contrasts between subgroups within each group.

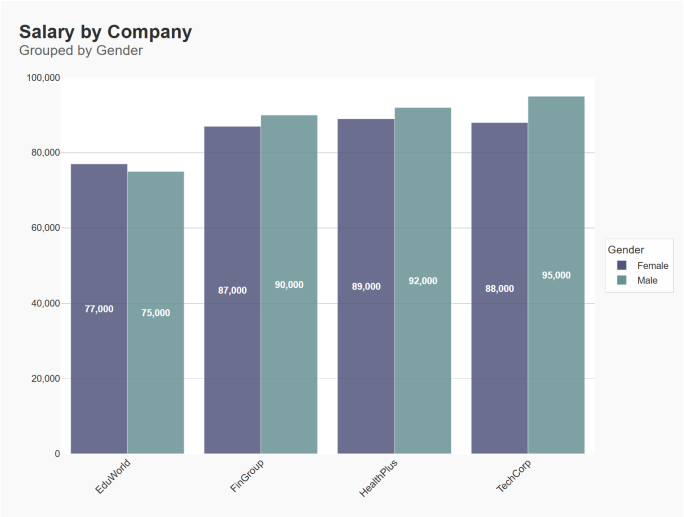

Figure 2.4: Bar Chart Grouped

3. **Example 3: Stacked Bar Chart**

   The stacked bar chart was obtained using this command:

```
with data from sales chart:
    show sales subgroups region for product
```

   The resulting chart is presented in Figure 2.5. The stacked bar chart displays sales amounts for different products, with each bar segment representing a region. This makes it easy to visualize both the total and the contribution of the subgroups to that total.

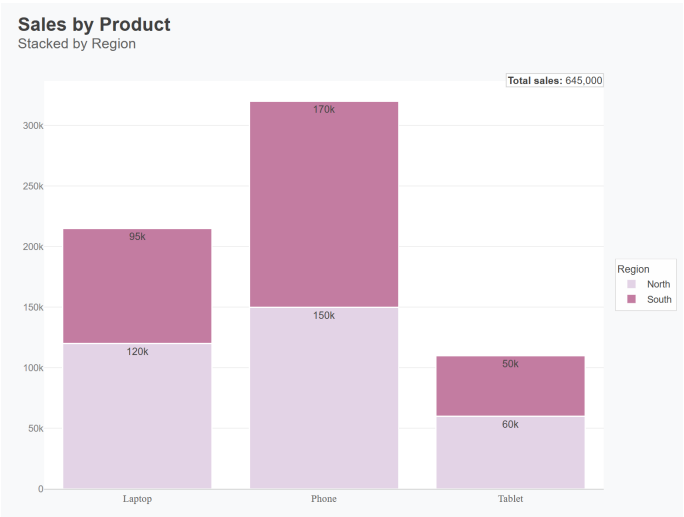

Figure 2.5: Bar Chart Stacked

4. **Example 4: Histogram**

   The histogram was obtained using this command:

```
with data from scores chart:
   show frequency of score step 5
```

   The result generated by the command above can be viewed in Figure 2.6. It illustrates a histogram that

shows how values are distributed according to scores. In this case, the data was organized into eight columns while respecting the five-step interval that the user specified in the command. Each bar's height indicates the frequency of scores within its range, giving a clear picture of how the scores are distributed throughout the dataset.

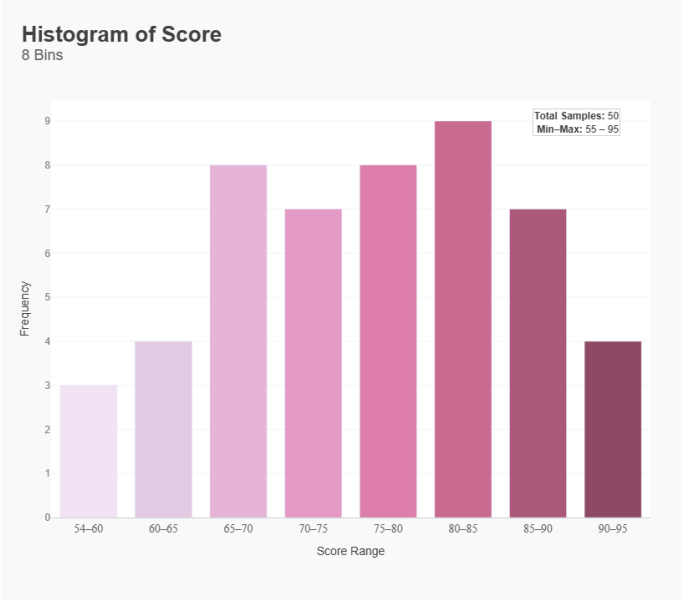

Figure 2.6: Histogram

5. **Example 5: Pie chart**

The pie chart was obtained using this command:

```
with data from sales chart:
   show proportion of sales by region
```

The output generated by the command above is displayed in Figure 2.7. It presents a pie chart that illustrates how the dataset is divided among distinct categories, in this case 2. The size (angle and area) of each slice indicates the proportion or percentage of the total dataset belonging to its corresponding category, the numbers themselves are also provided along with the graphical representation, for even more precise evaluation, if it's necessary.

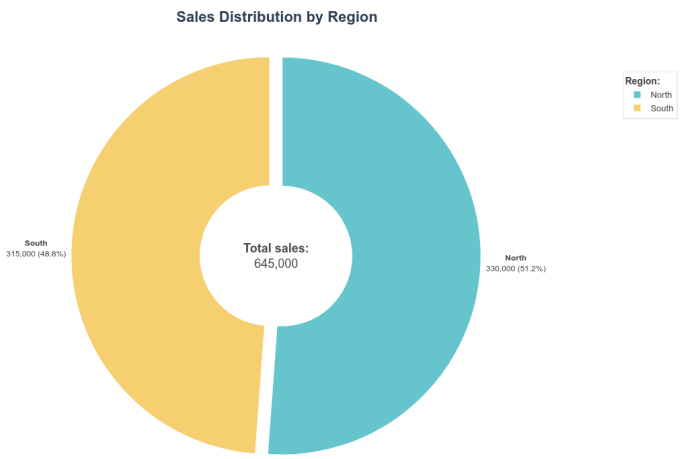

Figure 2.7: Pie chart

6. **Example 6: Scatter plot**

The scatter plot was obtained using this command:

```
with data from studentexam chart:
   pattern of StudyHours and ExamScore
```

The resulting scatter plot can be viewed in Figure 2.8. The figure displays the relationship between the number of study hours and the corresponding exam scores. The upward trend of the points indicates a positive correlation, with higher scores typically obtained by students who studied more. The ranges of both variables and the total number of observations are also showed.

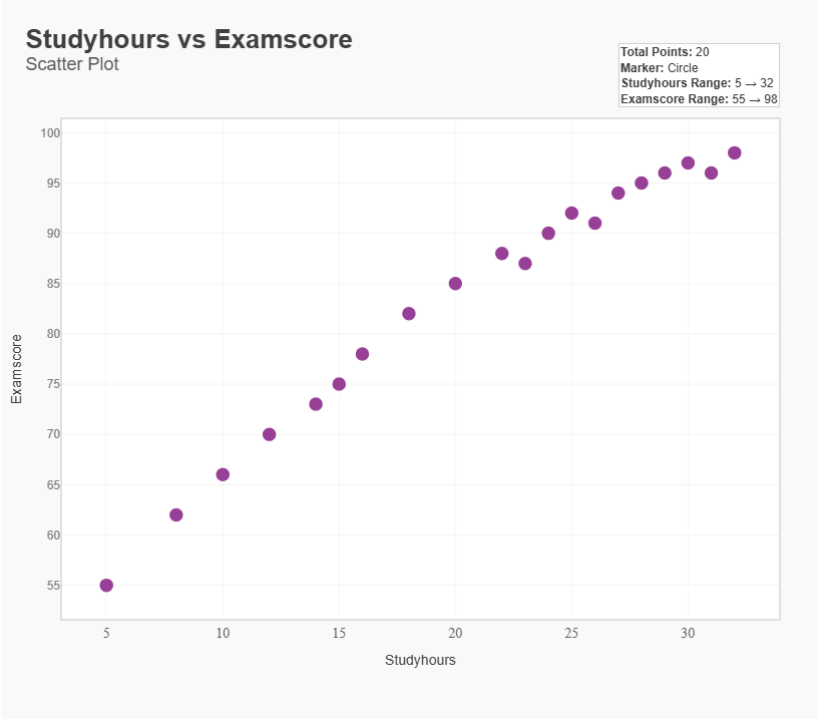

Figure 2.8: Scatter plot

7. **Example 7: Bubble chart**

The bubble chart was obtained using this command:

```
with data from economy chart:
   bubble of GDP, LifeExpectancy, Population for Country
```

The plot generated by the above command is presented in Figure 2.9. It displays a bubble chart that compares countries based on their GDP, Life Expectancy, and Population using data from an economy chart. The horizontal axis represents GDP, while the vertical axis shows life expectancy. Each bubble corresponds to a country, with its size indicating the population and its color distinguishing the region or category. When hovering over a bubble, detailed information about the country, including its exact values, is displayed for closer analysis.

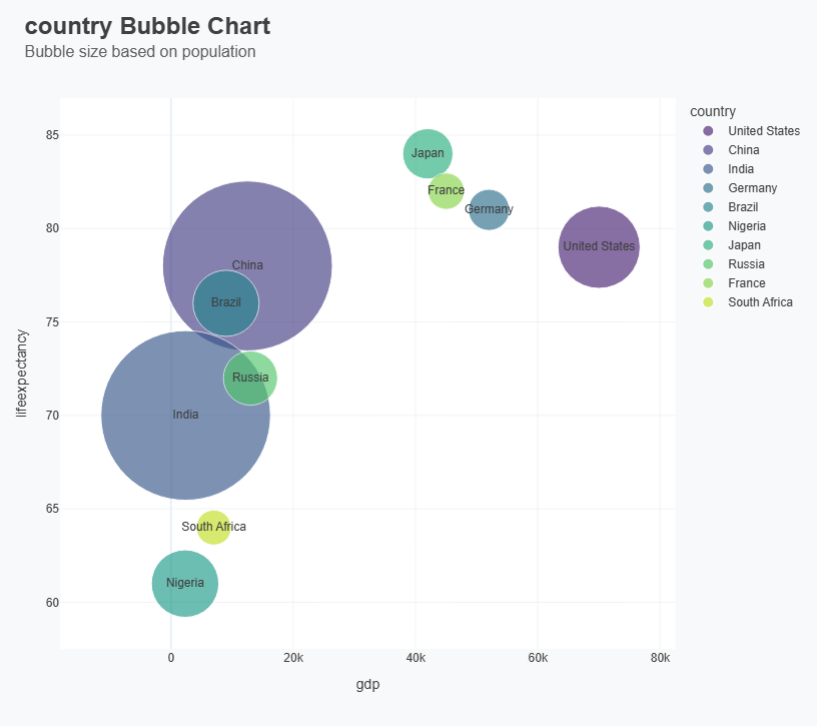

Figure 2.9: Bubble chart

8. **Example 8: Area chart**

The area chart was obtained using this command:

```
with sales from electronics chart:
   stacked trend of year for revenue from category
```

The result generated by the command is displayed in Figure 2.10. It illustrates an area chart depicting the trend of sales over a continuous 5 year period of several products. The filled area beneath the line emphasizes cumulative volume or intensity, with the height of the shaded region at any point indicating the magnitude of the value at that specific interval. This visualization highlights overall patterns, peaks, and troughs across the timeline.

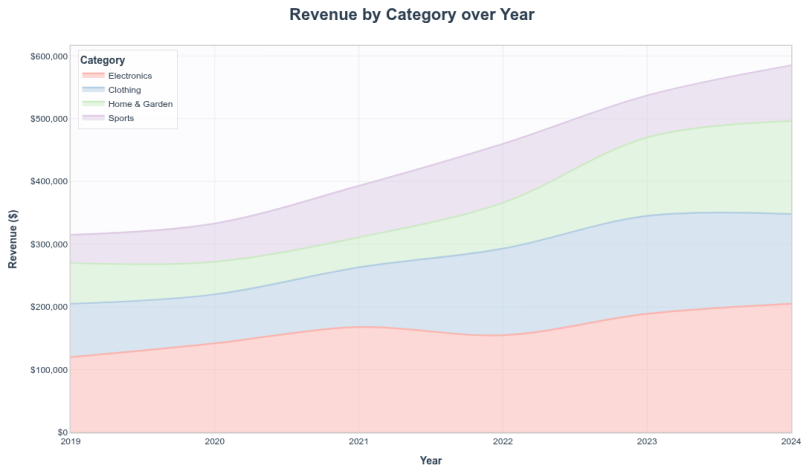

Figure 2.10: Area chart

9. **Example 9: Line graph**

The line graph was obtained using this command:

```
with data from spotify chart:
   show correlation between despacito and date
```

The chart resulting from the command is shown in Figure 2.11. It presents a line graph that illustrates the changes in Despacito's performance over time based on Spotify chart data. The horizontal axis represents the dates, while the vertical axis indicates a performance metric such as streaming counts or chart position. A smooth, pale, burgundy line connects the values, clearly showing the trend in popularity. When hovering over the graph, detailed information for each point is displayed, allowing for a closer examination of the data.

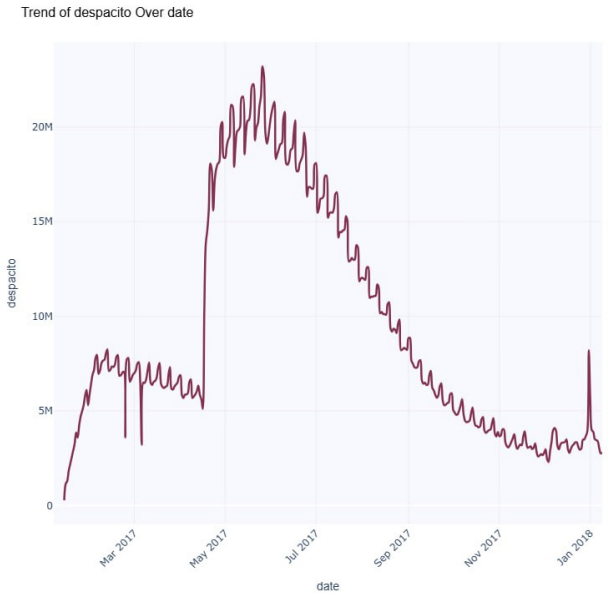

Figure 2.11: Line graph

Conclusions

In conclusion, our Domain Specific Language (DSL), created in a structured and user-friendly way, paired with a visually appealing and straightforward UI, represents a powerful tool for anyone looking to create graphs or charts. ChartLab is a data analysis and visualization tool which manages to simplify the complex word of data visualization. In this way, it makes the process of creating charts or graphs accessible to anyone, regardless of their technical background.

Moreover, it simplifies without compromising correctness. Thus, it is also a tool which preserves theoretical accuracy, helping users choose the appropriate visualization type for their data.

A key strength of our tool lies in its simplicity of use - users can easily upload their files and rely on natural-language commands to generate the chart that fits best. Unlike traditional tools, it isn't mandatory to know which type of chart you want, that's the beauty of ChartLab. Instead, through a keyword-based approach, it ensures the best visualization is chosen for the user's data.

While ChartLab has definitely proved its potential, we recognize there is room for improvement. We aim to enhance the UI, implement chart history, which makes it possible for the user to access his previously created charts, provide more charting options, more versatility and make it more intelligent and user-friendly, to provide the best data visualization experience.

The project was divided into 2 repositories for Backend [25] and Frontend [26].

Appendix

Project Plan

Living in a fast paced world, humans tend to optimize every process that consumes more time than needed. One of them is the creation of graphs and charts, which requires both knowledge and time. Therefore, here comes ChartLab DSL, which is aiming to ease this process and optimize it.

The main features of this project consists of graph type selection, chart history and dynamic data processing. The user has the option to choose the type of chart he wants or he can just provide the CSV or JSON file and based on the data, the domain specific language will automatically generate a certain graph. Moreover, the user will be able to see his previous charts by looking at the chart history.

One of the key milestones is designing an appropriate language suitable for every individual. Therefore, the user won't need to have an advanced knowledge both in programming and chart creation. Another one is the visual representation of the graphs. In order to achieve success we need to be able to make better visually attractive and diverse graphs then the existing standard solutions. The risk here consists of not being able to integrate a simplified language with advanced graph visualization.

To achieve a good outcome, we establish weekly task assignments and deadlines to every member of our team:

- 03.02.2025 – 07.02.2025: Selecting the Domain and identification of a specific problem
- 10.02.2025 – 14.02.2025: Domain analysis and problem definition. PBL Problem Map presentation.
- 17.02.2025 – 21.02.2025: Literature analysis (gathering a minimum of 20 references)
- 24.02.2025 – 28.02.2025: Project plan and team contract.
- 03.03.2025 – 07.03.2025: Formal grammar description (BNF).
- 10.03.2025 – 14.03.2025: Lexer/parser initial implementation (abstract-syntax tree).
- 17.03.2025 – 21.03.2025: Initial implementation of the interpreter and front-end.
- 24.03.2025 – 04.04.2025: Preparations for midterm. Reviewing the project report and creating the presentation.
- 07.04.2025 – 18.04.2025: Improving the interpreter and front-end.
- 21.04.2025 – 25.04.2025: Writing the scientific paper for submitting to the Technical scientific conference of undergraduate, master and PhD students.
- 28.04.2025 – 02.05.2025: Finishing the design of the front-end and interpreter. Technical demonstration of the language and additional tools/libraries/frameworks.
- 05.05.2025 – 09.05.2025: Final draft of the project report.
- 12.05.2025 – 16.05.2025: Final presentation.

As a primary tool we consider using the Python libraries in order to generate unique and creative graphs. Moreover, these libraries will serve as support and inspiration to our DSL. Another important technology that we are using is JIRA, which allows us to track every issue

and assignment, and provides an agile project management. In addition to this, in order to have a structured report, we use Latex, which is an advanced document preparation that allows us to meet scientific standards. And to improve reference management we use Zotero. In addition to these tools, we also consider using ANTLR which is a powerful parser generator for reading, processing, executing, or translating structured text or binary files.

In conclusion, we have structured a very well-defined project plan that helps us ensure an efficient process of developing the ChartLab DSL. By breaking down the project into specific tasks, we managed to make our work more structured and therefore an increasing percentage of success. With external tools like Python libraries, JIRA, Latex and Zotero we believe that we'll succeed in our goals.

PBL PROBLEM MAP

a)	Present the problem situation in 5-7 sentences.	Data is everywhere nowadays, it rules our world. And data visualization is essential to transform the data into useful insights for people, businesses, organisations. Data means nothing when we don't have the fitting tools for data visualisation. Existent tools in this domain are either too difficult to use, inaccessible to everyone, can lead to confusion in comprehending data, or could even be theoretically incorrect.
b)	Formulate the problem – 1 statement.	There is a lack of software for designing theoretically correct and visually pleasing charts, accessible for each and every individual.
TYPE OF QUESTION	**EXAMPLES OF QUESTIONS**	**SHORT ANSWER**
Who…?	1. Who has a problem? 2. Who is involved in the problem situation? 3. Who suffers from the problem? 4. Who should take part in solving the problem?	1. Data analysts, scientists, managers, software engineers, and even students. 2. The problem involves the categories mentioned above – those affected by the problem, and the ones who are able to solve it – software engineers, stakeholders, etc. 3.People who might draw wrong conclusions from the data, (business owners, students, etc.), who waste time creating graphs (people not in IT Industry, managers, etc.). 4. Software engineers should become the main contribution to the solution.
What …? Which…?	5. What happened? 6. What is the problem? 7. What led you to identify the problem (reasons, causes, evidence, arguments, findings, survey results, etc.)? 8. What are the causes that affect those involved? 9. What are the needs of those targeted? 10. What are the resources (financial, human, logistical, time, etc.) necessary to solve the problem?	5. During the Database course we realised the challenge of effective data visualization, and the lack of efficient tools. This highlighted the need for a more user-friendly solution. 6. The lack of software for designing theoretically-correct and visually-pleasing charts, accessible for each and every individual. 7. -> According to the article "Data Visualization Past, Present and Future ", data visualisation is still largely misunderstood, and "too often undermined by the very vendors that produce and sell visualization software" -> The International Data Corporation (IDC) predicts the "Global Datasphere" will grow from only 45 Zettabytes in 2019 to 175 Zettabytes in 2025. -> A study conducted by Richard Alleyne reveals that we are now exposed to 5x more data daily than in 1985. -> A study by the University of Rochester reveals that more than 50% of the human brain is devoted to processing visual information.

		-> In the analysis titled "Blinded with science: Trivial graphs and formulas increase ad persuasiveness and belief in product efficacy," it's shown that 68% of people believe a scientific claim when presented in words, but this percentage rises to 97% when the same claim is shown in a simple graph. 8. Many data visualization tools are complicated and hard for non-experts to use. They can also take a lot of time for professionals to work with. Moreover, there are not enough simple and easy-to-use tools that fit the different needs of groups like data analysts, students, managers, and scientists. As data grows quickly, individuals and organizations struggle to manage, understand, and find useful insights in it. 9. To simplify the extraction of the insights from the datasets by using a more efficient and less time-consuming tools. 10. For solving the problem, we need to develop a product which will meet the growing requirements. It will require a dedicated team of software engineers (human resources), financial resources and around 3-4 months to make a fully integrated and usable tool.
Where…?	11. Where did the problem arise (location, area, stage of the process of the problem arising)?	11. The problem arose globally, in academia, business, the tech industry.
When…?	12. When did the problem arise? 13. When was the problem identified? 14. When is it planned to be resolved?	12. This has been an issue for many years, but became prominent in the recent decades with the rise of influence of data in all industries. 13. The problem has been addressed in multiple research papers and articles over the years. However, we observed it especially while working closely with data during our Databases course, when we gained insights on real-life practices of data analysis. 14. We are planning on developing a working prototype by mid-2025 (May – June).
Why…?	15. Why is there a need to solve the problem?	15. Data means nothing without the right and accessible tools to interpret it. The wrong tools lead to confusion, lost insights and slowing down of productivity.
How...?	16. How (in what way) can the problem be solved? 17. How will the elimination of the problem be determined?	16.The problem can be solved by developing software for creating charts that has simple, intuitive syntax, and which is based on theoretically-correct concepts. 17. By prompting users to test it and give feedback, which would hopefully mention its simplicity of use and obtaining correct and nice charts.
How much...?	18. How big is the problem?	18. The problem is significant because, nowadays, data is present in every aspect of our lives.

	19. How much is the quantity/quality/number of people etc. affected? 20. How long will it take to resolve the problem? 21. How many parties need to be involved in resolving it?	19. This problem affects a huge amount of people, millions of data analysts, business owners, researchers, students worldwide – everyone who depends on data in their daily work. Then, the quality of data analysis directly and strongly influences business decisions, scientific discoveries and productivity in workspaces. 20. It will take around 3-4 months to develop a minimum viable product (MVP) and present it to the general product. Additionally, perfecting the product and adding additional feature may take another few months. 21. To resolve this problem, at least these 3 key parties must be involved: developers (software engineers), end users (those affected by the problem, which will later test it), and stakeholders (investors, sponsors, etc who will support the product's development).
Other questions you found answered (optional)	22. 23.	22. 23.
	How do you see solving this problem (in terms of deliverable/product/result)?	Solving the problem will involve a real-time chart creation software, where the user will provide the data and his really specific needs: what he wants to obtain through his data visualisation. Then, the program will provide one or more technically accurate and visually pleasant graph options the user can choose from. The user can make minor changes, but the workload falls on the software which provided this straight-forward solution to our problem.
	Conclusions, findings, reflections...	To solve this problem, our team proposes a DSL (Domain Specific Language), that will simplify the process of analysing data and will provide clearer data visualisation.

UNIVERSITATEA TEHNICĂ A MOLDOVEI

Team Contract

ChartLab

Semester: 4
Project theme: Data Analysis and Visualization
Project period: February, 2025 - May, 2025
ECTS: 8
Supervisory team: Supervisors: 1 and 2; Mentors: Andreea Ioana Bujac; Romeo V. Turcan
Project group number: 4
Students:
Anastasia Tiganescu, FAF-231
Daniela Cojocari, FAF-231
Janeta Grigoras, FAF-231
Maxim Isacescu, FAF-231
Vladimir Vitcovschii, FAF-231

1. Team Roles & Responsibilities:
    a. All tasks are distributed dynamically in the team to ensure all team members get to choose what they want to work on, based on their expertise and availability, to ensure efficiency and balance.
    b. The team will track each task on Jira.
2. Communication Plan:
    a. Primary Communication Platform: Teams, Telegram.
    b. Meeting Frequency: Bi-weekly,on Tuesday and Thursday, 3 PM.
    c. Meeting Medium: In-Person, online only in exceptional situations.
3. Decision-Making & Conflict Resolution:
    a. All decisions are to be made collectively, with every team member expressing their opinion.
    b. If no consensus is reached on a specific topic, we take the problem to our mentor.
4. Code Standards:
    a. Code must follow best coding-practices, must be properly commented.
    b. Version control is maintained using GitHub.
5. Project Timeline & Milestones:
    a. All tasks and deadlines correspond to the detailed Project Plan crafted for our project.
6. Ethical Considerations & Academic Integrity:
    a. Absences must be communicated in advance.
    b. All tasks must be high-quality, original and properly cited when necessary.
    c. Plagiarism or extensive use of AI machines won't be tolerated.
    d. In case of detected plagiarism or use of AI machines, the team is at risk of not qualifying for the final exam. In this case, the team has the full right to exclude the guilty member/members from the team.

By signing this document, each member of the group confirms participation on equal terms in the process of writing the project. Thus, each member of the group is responsible for all contents in the project.

Chișinău, 2025

Bibliography

[1] Scott Berinato. *Good Charts, Updated and Expanded: The HBR Guide to Making Smarter, More Persuasive Data Visualizations*. Harvard Business Press, August 2023.

[2] Stephen Few. Data Visualization - Past, Present, and Future.

[3] Anders Wallgren. *Graphing Statistics & Data: Creating Better Charts*. SAGE, June 1996.

[4] L. A. Treinish, J. D. Foley, W. J. Campbell, R. B. Habor, and R. F. Gurwitz. Effective software systems for scientific data visualization. *ACM SIGGRAPH Computer Graphics*, 23(5):111–136, December 1989.

[5] Tomaž Kosar, Sudev Bohra, and Marjan Mernik. Domain-Specific Languages: A Systematic Mapping Study. *Information and Software Technology*, 71:77–91, March 2016.

[6] Lorenzo Bettini. *Implementing Domain-Specific Languages with Xtext and Xtend*. Packt Publishing Ltd, August 2016.

[7] Leandro Nascimento and Daniel Viana. A Systematic Mapping Study on Domain-Specific Languages. 2012.

[8] Georg Hinkel, Henning Groenda, Lorenzo Vannucci, Oliver Denninger, Nino Cauli, and Stefan Ulbrich. A Domain-Specific Language (DSL) for Integrating Neuronal Networks in Robot Control. In *Proceedings of the 2015 Joint MORSE/VAO Workshop on Model-Driven Robot Software Engineering and View-based Software-Engineering*, MORSE/VAO '15, pages 9–15, New York, NY, USA, July 2015. Association for Computing Machinery.

[9] Charts, Tables, Graphs, and Diagrams: An Approach for Social Studies Teachers: The Social Studies: Vol 87 , No 1 - Get Access. https://www.tandfonline.com/doi/pdf/10.1080/00377996.1996.10114492, February 2025.

[10] The Feeling of Numbers: Emotions in Everyday Engagements with Data and Their Visualisation - Helen Kennedy, Rosemary Lucy Hill, 2018. https://journals.sagepub.com/doi/10.1177/0038038516674675, February 2025.

[11] David Reinsel, John Gantz, and John Rydning. The Digitization of the World.

[12] Welcome to the information age – 174 newspapers a day. https://www.telegraph.co.uk/news/science/science-news/8316534/Welcome-to-the-information-age-174-newspapers-a-day.html, February 2025.

[13] Rochester Review :: University of Rochester. https://www.rochester.edu/pr/Review/V74N4/0402_brainscien February 2025.

[14] Blinded with science: Trivial graphs and formulas increase ad persuasiveness and belief in product efficacy - Aner Tal, Brian Wansink, 2016. https://journals.sagepub.com/doi/abs/10.1177/0963662514549688, February 2025.

[15] Top 24 tools for data analysis and how to decide between them. https://www.stitchdata.com/resources/data-analysis-tools/.

[16] Allan Franklin and Slobodan Perovic. Experiment in Physics. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2023 edition, 2023.

[17] Eliza Bobek and Barbara Tversky. Creating visual explanations improves learning. *Cognitive Research: Principles and Implications*, 1:27, December 2016.

[18] Data Cleaning & Data Preprocessing for Machine Learning. https://encord.com/blog/data-cleaning-data-preprocessing/, February 2025.

[19] Top 6 Data Visualization Tools for Business Professionals. https://online.hbs.edu/blog/post/data-visualization-tools, January 2021.

[20] Top 3 Data Visualization Tools for Business Analytics 2025 – 365 Data Science. https://365datascience.com/trending/top-data-visualization-tools-business-analytics/, February 2025.

[21] Microsoft bundles Office AI features into Microsoft 365 and raises prices | The Verge. https://www.theverge.com/2025/1/16/24345051/microsoft-365-personal-family-copilot-office-ai-price-rises, February 2025.

[22] Visualizing Data in Excel. https://www.datacamp.com/tutorial/visualizing-data-in-excel, February 2025.

[23] Dashboard Tools. https://www.tableau.com/dashboard, February 2025.

[24] Creating Your First Dashboard | Superset. https://superset.apache.org/docs/using-superset/creating-your-first-dashboard/, February 2025.

[25] Anastasia Tiganescu, Daniela Cojocari, Janeta Grigoras, Maxim Isacescu, and Vladimir Vitcovschii. CharLab-Backend-Repository. https://github.com/NeValetik/ChartLab-ChartingDSL.

[26] Anastasia Tiganescu, Daniela Cojocari, Janeta Grigoras, Maxim Isacescu, and Vladimir Vitcovschii. ChartLab-Frontend-Repository. https://github.com/NeValetik/chartLabFront.