# Introduction

Healthcare is generating an unprecedented amount of data due to the rise in digital technology [[1,2]](https://paperpile.com/c/wYkhtl/MXki+oSx6). This data, ranging from patient records to complex genetic information, holds value for various studies, including those related to real-world evidence. However, the sheer volume of this data makes manual chart generation and updating increasingly impractical. As such, automation is becoming essential for efficient data interpretation in healthcare.

As healthcare increasingly digitizes, the sector is inundated with a complex array of data that professionals and researchers must make sense of. While visualization tools exist, they often don't address the specific needs of healthcare data or scale well with big data challenges [[3,4]](https://paperpile.com/c/wYkhtl/eqi7+IuQ8). Moreover, the manual effort involved in using these tools remains significant. Therefore, there's a growing demand for an automated and scalable solution capable of simplifying the generation and deployment of relevant visualizations.

## Objectives

The aim of this project is to conceive, architect, develop and evaluate Visual Viper (VV), a Python library aimed to automate the creation of data visualizations in the healthcare sector. This work will provide a description of each phase, from initial requirement gathering and system architecture design to coding, testing, and evaluation. Limitations will be discussed, along with suggestions for future enhancements.

To provide a comprehensive understanding of the scope of this project, the following objectives are enumerated:

* Conduct an initial requirement analysis to identify the specific needs and constraints that VV aims to address.
* Outline the architecture of VV while adhering to best practices in software development.
* Implement the designed architecture of VV, emphasizing its modular and extensible nature.
* Apply and critically analyze software development methodologies such as object-oriented programming and test-driven development in the creation of VV, considering their impact on the code's quality, maintainability, and extensibility.
* Implement, test, and evaluate the features that VV offers for data retrieval, transformation, and visualization, with a specific focus on retrieving data from Google Sheets, creating Forest Plots, and deploying visualizations to Miro Board and Google Drive.
* Conduct performance testing on VV to assess its efficiency and scalability, especially when handling large healthcare datasets.
* Review and identify areas of improvement within the current version of VV, setting the stage for future iterations and enhancements.
* Assess the tool's success in automating the data visualization process in healthcare research, measuring its effectiveness in facilitating scientific communication.

The project seeks to fill a critical gap in the existing tools for automating the generation of healthcare data visualization. By automating the often labor-intensive and complex process of generating custom visualizations, VV aims to significantly improve the efficiency of scientific communication in healthcare. It also introduces a modular and extensible architecture, enabling the library to adapt to diverse data sources and evolving visualization needs, thereby extending its lifespan and relevance.

## Thesis Overview

This thesis is organized as follows:

* **Introduction**: This section encompasses the background of the study, problem statement, purpose, research objectives, and justification. It serves to establish the context and significance of the research.
* **Background**: This section provides a concise assessment of the literature relevant to data visualization.
* **Methodology**: This section details the methodologies adopted. It covers aspects like requirement analysis, user stories, and scope of the project. Core principles such as modularity, extensibility, and usability are also elaborated in separate subsections.
* **Development Approach**: Offers an overview of the development approach and programming paradigms employed. It discusses possible development approaches and discusses the use of Object-Oriented Programming and Test-Driven Development.
* **Development Environment and Tools**: In this section, the various tools utilized during the development process are covered in detail. This includes aspects of Docker containerization, version control through GitLab, and CI/CD pipelines. Furthermore, this section elucidates the build automation process and discusses the Makefile in detail.
* **System Architecture**: Provides an in-depth description of the system’s architecture. It discusses the high-level architecture, key classes, component interactions, and data flows among components.
* **Implementation Details**: This section delves into the technical nuances of the project's implementation.
* **Workflow Demonstration**: This section provides a demonstration of how the VV system operates in a real-world context. The aim is to convey both the utility and the user experience of the system.
* **Evaluation Results**: This section analyzes VV's performance on the specific use cases of Google Sheets data retrieval, Forest Plots creation, and deployment to Miro Board and Google Drive.
* **Discussion**: This section serves as a platform to review the research findings and to propose future recommendations.
* **Conclusion**: Summarizes the research and outlines the contributions made by the study.
* **References**: Lists all sources cited throughout the document.