earliest visualization libraries in Python, is known for its flexibility and extensive control over plot customization. It supports a wide range of charts, including line plots, bar plots, and 3D visualizations, making it ideal for scientific research that requires precision. However, its complexity can be a barrier for users new to data visualization, and it lacks built-in interactivity, limiting its use in modern web applications [1]. Despite these challenges, *Matplotlib* remains a cornerstone in scientific data visualization due to its precision and versatility [2]. *Seaborn*, developed as an extension of *Matplotlib*, simplifies the process of generating statistical visualizations. It integrates closely with *Pandas* and provides a high-level interface that automates many of the complex steps required to produce aesthetically pleasing plots. *Seaborn* is particularly suited for exploratory data analysis due to its ability to create complex multi-faceted plots with minimal code [2]. *Plotly* is distinct from both *Matplotlib* and *Seaborn* due to its focus on interactivity. Its plots, which are interactive by default, allow users to zoom, hover, and toggle data points, making it ideal for web-based applications. *Plotly* supports a wide variety of chart types, including choropleth maps and 3D scatter plots, making it a versatile tool for dynamic data exploration. However, its reliance on JavaScript can introduce performance issues when handling large datasets [3], [4].

The role of media in shaping public perception of crime, particularly homicides, is significant. Media coverage often creates a distorted image of homicides, which influences public opinion and potentially affects policymaking. Research shows that homicide reporting is frequently inconsistent with actual statistics. For instance, a study of Australian media representations found that while offender characteristics were often accurate, the representation of victims and circumstances surrounding the crimes were skewed, leading to public misrepresentation. [5]. Further studies from England and Wales suggest that newspapers tend to disproportionately highlight certain types of homicides, such as those involving sexual assault or extreme violence. This selective reporting contributes to skewed public perceptions of homicide risk, focusing on sensational cases that do not reflect statistical realities. These media narratives often construct a version of homicide which can influence social and political responses that are not based on factual data [6]. Media coverage also plays a crucial role in raising public awareness about specific types of crime. Studies have demonstrated that high-profile media coverage of crimes against children heightens public sensitivity to such issues and can lead to stronger advocacy for child protection measures [6]. However, the nature of media reporting on domestic homicides is notably less prominent. Research has found that domestic-related homicides are reported less prominently than non-domestic ones, receiving fewer front-page articles, and shorter articles, which may downplay the seriousness of domestic violence [7]. In cases where homicides are committed by mentally disordered offenders, media coverage often sensationalizes these events, exacerbating public stigma against mental illness. A study on New Zealand media found that such cases were more likely to feature dramatic reporting, contributing to a skewed perception of the mentally ill as dangerous individuals [8]. The overall impact of media on public perceptions of homicide is substantial. For instance, a study of Chicago homicide reporting patterns found that while juvenile homicide rates had declined, media coverage of such cases increased. This over-reporting, particularly of atypical cases, led to heightened public fear of crime, further distorting public perceptions [9].

The *Homicide Media Tracker* aims to address these discrepancies by systematically collecting and analysing media reports on homicides. By providing transparent and data-driven insights into media coverage, the platform can reduce distortions in public perception and foster more informed discussions around crime and justice.

3. METHODOLOGY

3.1. Platform Development and Data Entry

The platform was developed using *Dash*, a Python web application framework for building analytical web applications. *Dash* was chosen for its ease of integration with the Python ecosystem, particularly in handling data visualizations and interactivity, both critical for the success of this project. Data is manually entered through a custom dashboard, where users input metadata such as victim demographics, crime location, and nature of the homicide. The data is stored in a *PostgreSQL* database, accessed via the *SQLAlchemy* Python library for smooth interaction between the web app and database**.** The database holds all homicide-related data, including news URLs, victim and perpetrator details, and crime specifics. A dedicated table handles duplicates to ensure data cleanliness. Data can be queried, inserted, and updated via Dash callbacks using the *psycopg2* library and *SQLAlchemy*, keeping the system dynamic and interactive. Built-in mechanisms validate and maintain data integrity. Callback functions detect duplicate entries based on customizable criteria and handle errors during CSV uploads or form inputs. Errors are flagged to users with detailed messages to ensure clean data entry.

3.2 Data Visualization

The data visualization section of the app is designed to provide users with a dynamic and interactive way of selecting and displaying different plots. The interface is structured using the *dash-bootstrap-components* library to ensure a clean and intuitive layout. At the core of this section is a set of dropdown menus. The first dropdown allows users to select a broader plot category, such as 'Homicides Over Time' or 'Geographical Distribution.' Once a category is chosen, the second dropdown is dynamically updated with specific plot types tailored to that category, such as line plots or bar charts. This setup makes it straightforward for users to navigate through various visualization options, ensuring that the right tools are always available based on the type of insight they wish to explore. The callback functionality underpinning the app's interactivity is carefully structured to manage both the dynamic updating of plot-type options and the rendering of visualizations. When a user selects a category from the first dropdown, a callback function dynamically updates the second dropdown to show only the plot types relevant to that category. This ensures that users are only presented with valid options, simplifying the visualization process. Another callback handles the actual rendering of the selected plots. Once a user chooses both a category and a plot type, the app queries the PostgreSQL database, processes the relevant data using *Pandas*, and generates the appropriate visualization. This mechanism allows for a wide range of plot types, from time series visualizations plots to geographical insights and several others. In addition to the predefined plot categories, there is a separate section that caters to custom visualizations, specifically for bar graphs. In this section, users can select the x-axis from a list of available columns in the dataset while the y-axis is currently fixed as 'count', reflecting the number of unique entries. This feature provides significant flexibility for users who wish to go beyond the standard plots and create visualizations based on their specific needs.

3.3 Open-Source Implementation and Community Engagement

The platform is designed to be fully open-source, encouraging community collaboration and improvement. All code and resources will be made publicly available, where users can contribute by adding features or optimizing the existing system, ensuring that the project remains adaptable to future needs and can integrate improvements from a broad range of developers and researchers. The decision to use Python and its associated libraries was driven by the availability of open-source tools that foster community involvement, making it possible for users with different technical backgrounds to modify and improve the platform.

4. EXPERIMENTAL SETUP

Three versions of the application were developed, each utilizing one of the aforementioned data visualization libraries. To simulate real-world data, a Python script was used to generate CSV files containing fields identical to those in the *homicide_main* database, populated with randomized but contextually relevant data. Five CSV files were created, containing 10, 100, 1000, 10 000, and 100 000 entries, respectively. The applications were run and the bar graphs were generated 10 times on each dataset. The averages of the running memory consumption as well as the peak memory consumption were recorded and plotted, while the size of the serialized data was also recorded and plotted. Each library was assessed based on four key metrics: memory consumption, serialized data size, community support, and robustness of available chart types.

4.1 Memory Consumption and Data Serialization

The memory footprint of each library was evaluated using *tracemalloc*, a Python module that tracks memory allocations. Each library was tested on a bar graph representing the number of homicides over time as it was the only plot existing within the application that was common to all libraries, as well as it being the most common plot type to represent data. This was done to assess how efficiently they handle data. The aim was to ensure that the platform remains functional, even on low-resource machines. Memory consumption patterns across all three libraries are largely independent of the specific machine specifications. This is because Python's memory allocation, follows deterministic models based on the size and complexity of the dataset being processed. The backend architecture of these libraries ensures that they follow predictable memory usage patterns, with any variations in machine specifications affecting execution time rather than memory footprint [11]. Additionally, due to Python's Global Interpreter Lock (GIL), memory access and manipulation in these libraries is largely sequential, further reducing the impact of hardware variations on memory consumption.

The size of serialized data is crucial as it directly impacts storage requirements and data transmission efficiency. Smaller serialized data ensures faster loading times and reduces bandwidth consumption, making the platform more accessible and responsive, particularly for users with limited hardware capabilities

4.2 Community Support and Feature Set

As an open-source project, the platform relies on libraries that are actively maintained and have strong community support. Each library's GitHub repository was analysed for activity, crucial to avoid dependencies on libraries that may become obsolete. The libraries were also compared based on their ability to generate a wide range of visualizations needed for the platform. *Plotly*, for instance, was favoured for its interactivity, allowing users to explore the homicide data dynamically, while *Matplotlib* and *Seaborn* were assessed for their ability to create detailed, publication-quality static plots.

5. RESULTS AND DISCUSSION

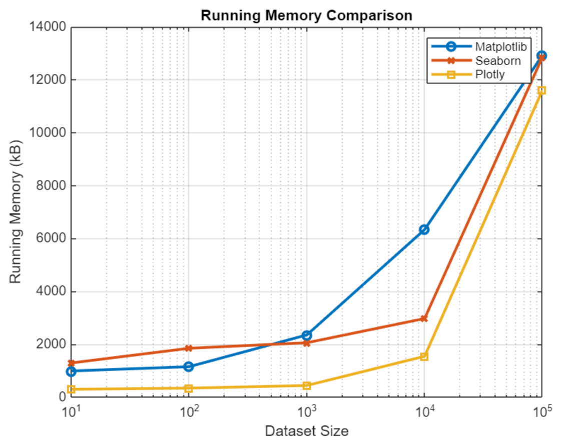


Figure 5.1: Plot Showing Running Memory Comparison

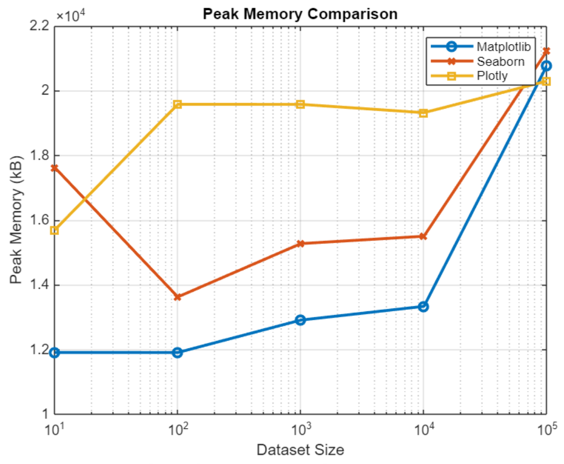


Figure 5.2: Plot Showing Peak Memory Comparison

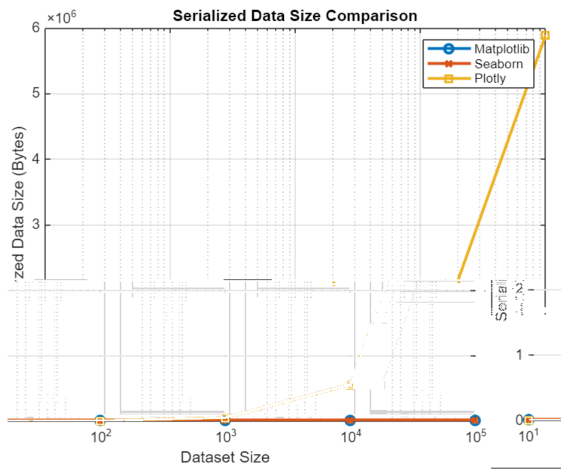


Figure 5.3: Plot Showing Serialized Data Size Comparison

5.1 Memory Consumption Analysis

The plots produced in *Figures 5.1- 5.3* are logarithmic with regards to the x-axes. The exponential nature of the plot found in *Figure 5.1* infers a relatively linear increase in memory usage with an increase in dataset size. To model the relationship between dataset size and memory usage, a simplified linear growth equation can be used:

$$M(n) = c.n + k \qquad (1)$$

Where *M(n)* is the memory consumption in kilobytes, *n* is the number of data-points, *c* is the memory used per data point and *k* is the baseline memory overhead for initialization. A zero entry dataset was used to find *k* for each of the libraries and was found to be 737,84 kB for *Matplotlib*, 690,28 kB for *Seaborn*, and 277,42 kB for *Plotly*. To find *c* for each of the libraries, consumption was measured again after adding one more entry to each dataset. The difference between that reading and the previous reading was then taken. The average of these metrics across all datasets were taken, yielding *c* to be 31,148 kB for *Matplotlib*, 22,13 kB for *Seaborn* and 15,135 kB for *Plotly*, yielding:

$$M(n) = 31{,}148n + 737{,}84 \qquad (2)$$

$$M(n) = 22{,}13n + 690{,}28 \qquad (3)$$

$$M(n) = 15{,}135n + 277{,}42 \qquad (4)$$

However, it is important to note that *equations 1-4* would only be used to model an approximation of the running memory consumption of the libraries.

Matplotlib demonstrates an increase in running memory consumption as the dataset size grows. For the smallest dataset, *Matplotlib* consumed approximately 1,146 kB, whereas for the largest dataset, it utilized 12,918 kB. While this is a significant jump, it reflects the library's extensive use of backend rendering processes, which involve maintaining data points and rendering details in memory. The peak memory usage, however, remains high across all dataset sizes, from 11,900 kB (small dataset) to over 20,700 kB (large dataset). This overhead reflects rendering model, which stores complex plot configurations even for simple plots, leading to high memory overhead [11].

Seaborn, also shows an increase in memory usage as the dataset size grows. However, its running memory consumption is slightly higher than *Matplotlib* for small datasets indicating that *Seabor* more advanced plotting capabilities come with a memory cost. For large datasets, *Seaborn's* memory usage becomes inefficient, with peak memory usage exceeding 21,200 kB for the largest dataset. The inefficiency in peak memory, even for small datasets, indicates that internal operations, especially those related to statistical plotting, require more memory for intermediate calculations [10].

running memory consumption is substantially lower than both *Matplotlib* and *Seaborn* for small datasets, however, its memory usage grows rapidly with larger datasets. ability to keep running memory lower for small datasets is due to its web-based rendering model, which offloads much of the rendering work to the front-end, reducing Python's in-memory footprint [11]. However, this front-end model leads to high peak memory, reaching 20,294 kB for the largest dataset. This indicates that while Plotly is more efficient in real-time memory handling, it still requires significant resources during the initial rendering of complex plots, likely due to its state management for interactive.

5.2 Serialized Data Size Analysis

The serialized data size for *Matplotlib* remained relatively constant across datasets, ranging from. This stability reflects static rendering model, where plots are outputted as basic vector or raster images, which do not require storing large amounts of data for interactions or detailed plot states. As such, *Matplotlib* is particularly well-suited for generating publication-quality plots or static reports, where the size of the data file is less critical and more focused on visual clarity [1]. *Seaborn* showed a slightly larger increase in serialized data size, from 14,978 bytes to 18,614 bytes. The increase in serialized size can be attributed to automatic handling of statistical elements such as confidence intervals and regression lines, which result in additional data being stored. While this additional data enhances the richness of visualizations, it also makes *Seaborn* somewhat less efficient for web-based applications or cases where lightweight data storage is crucial [12]. *Plotly* exhibited the most substantial increase in serialized data size, from 524 bytes (10 rows) to a striking 5,894,357 bytes (100,000 rows). This vast growth in serialized size is due to support for interactive web-based visualizations, which require storing complex data structures such as interaction states, user event bindings and detailed plot metadata. While interactivity makes it highly suited for web applications and dashboards, the significant increase in serialized data size can introduce performance bottlenecks in environments with limited bandwidth or storage capacity [5]

5.3 Implications for Accessibility

The Homicide Media Tracker is to be accessible to users across different hardware setups, especially in resource-constrained environments. The findings on memory consumption and serialized data size are crucial for achieving this goal. *Matplotlib* and *Seaborn* are better suited for users with lower-spec hardware or limited bandwidth due to their smaller memory footprint and stable serialized data sizes. *Plotly*, while offering advanced interactivity, may pose challenges for users with limited computing resources or internet speeds if the app is operating on a server, suggesting that it should be reserved for users requiring in-depth interactive exploration of the data.

Table 5.1: Github Activity For Each of the Libraries

Library	Contributors	Used By	Forks	Stars
Matplotlib	1473	1.3m	7.6k	20.1k
Seaborn	249	308k	2.5k	16.1k
Plotly	200	477k	1.9k	12.5k

The GitHub activity of the three libraries demonstrates their varying degrees of adoption, maintenance, and community support. *Matplotlib* has the most significant contributor base, highlighting its long-standing presence and widespread community support. The GitHub statistics reflects its popularity for technical, customizable visualizations that require precision and detail. The extensive community behind *Matplotlib* ensures continued development and feature improvements, making it a reliable choice for static and publication-quality visualizations. Although *Seaborn* has less contributors, it is still well-maintained, receiving regular updates. The GitHub statistics show strong community engagement, albeit at a smaller scale compared to *Matplotlib*, while *Plotly* appears to be the smallest of the three libraries in terms of community support.

Table 5.2: Natively Supported Plot Types for Each Library

Plot Type	Matplotlib	Seaborn	Plotly
Line	x	x	x
Histogram	x	x	x
Bar	x	x	x
Scatterplot	x	x	x
Boxplot	x	x	x
Contours	x	x	x
Filled Polygons	x	o	x
Spectrogram	x	o	x
Violin	x	x	x
Pairplot	o	x	o
Heatmap	o	x	x
Regression	o	x	o
Polar	x	o	x
3D	x	o	x

Each library offers a different range of natively supported plot types. *Matplotlib* supports the widest range of plot types, from basic line and bar charts to advanced visualizations such as contours, 3D plots, polar plots, and spectrograms. Its versatility makes it the go-to tool for detailed, technical visualizations required in scientific, engineering, and mathematical fields. Its ability to handle such a broad variety of plots allows users to produce highly customized and precise visual representations of complex data [1]. *Seaborn*, though limited compared to *Matplotlib*, specializes in statistical visualizations such as pairplots, *heatmaps*, and regression plots. These features make *Seaborn* particularly useful for generating quick, high-level insights during the exploratory phase of data analysis. However, its lack of support for more complex or specialized plots limits its use in advanced technical visualizations. Its strength lies in its ability to create appealing statistical plots with minimal configuration [2]. *Plotly* excels in its support for interactive visualizations, particularly 3D plots and choropleth maps, offering interactivity by default. Its focus on web-based applications and dashboards makes it an invaluable tool for dynamic data exploration. While it supports all common plot types, its interactivity sets it apart, allowing users to interact with data directly in the visualization through zooming, hovering, and clicking. *Plotly* offers the most flexibility for dynamic, interactive visualizations, while *Matplotlib* is the best choice for detailed, static outputs. *Seaborn*, though powerful for statistical analysis, is more limited in scope and interactivity.

6. CONCLUSION

The Homicide Media Tracker successfully integrates Python's visualization libraries into an open-source platform capable of analysing and presenting homicide-related data in an accessible and scalable manner. Considering the goal of making the Homicide Media Tracker accessible to as many users as possible, particularly in resource-constrained environments, the final choice must balance performance, community support, and usability. *Matplotlib* excels in its vast range of plot types and strong community support but is limited by its higher memory consumption and lack of interactivity. It's best suited for static, publication-quality visualizations and users with higher-spec hardware. *Seaborn*, while user-friendly for statistical analysis, is limited in its versatility and becomes inefficient with larger datasets, making it less ideal for this application. *Plotly* offers the best compromise for this context. It provides interactivity essential for engaging users with dynamic data and is relatively memory-efficient for small to medium datasets. Its growing community and focus on web applications align well with the platform's goals of accessibility and scalability. Although it may face challenges with large datasets, its ability to offload tasks to the front-end and its focus on interactivity make it the most suitable choice for the Homicide Media Tracker in bridging the digital divide. Thus, for this context, *Plotly* emerges as the most suitable library, offering the right balance between interactivity, memory efficiency, and modern usability required to make the platform accessible and engaging for a wide range of users, even those with lower-end hardware.

REFERENCES

[1] N. Ari and M. Ustazhanov, "Matplotlib in python," 2014 11th International Conference on Electronics, Computer and Computation (ICECCO), pp. 1-6, 2014
[2] A. H. Sial, S. M. A. S. Rashdi, and A. H. Khan, "Comparative Analysis of Data Visualization Libraries Matplotlib and Seaborn in Python," Int. J. Adv. Trends Comput. Sci. Eng., 2021
[3] D. Probst and J. Reymond, "FUn: A framework for interactive visualizations of large, high-dimensional datasets on the web," Bioinformatics, vol. 34, pp. 1433-1435, 2018
[4] I. Stancin and A. Jović, "An overview and comparison of free Python libraries for data mining and big data analysis," 2019 42nd Int. Conv. Inf. Commun. Technol., Electron. Microelectron. (MIPRO), pp. 977-982, 2019
[5] E. M. Waters, C. Bond, and L. Eriksson, "Examining the Accuracy of Print Media Representations of Homicide in Australia," *Current Issues in Criminal Justice*, vol. 29, pp. 137-153, 2017
[6] M. Peelo, B. Francis, K. Soothill, J. Pearson, and E. Ackerley, "Newspaper reporting and the public construction of homicide," *British Journal of Criminology*, vol. 44, no. 2, pp. 256-275, 2004
[7] J. S. Wong and C.-W. Lee, "Extra! Extra! The Importance of Victim–Offender Relationship in Homicide Newsworthiness," *Journal of Interpersonal Violence*, vol. 36, pp. 4186-4206, 2018.
[8] B. McKenna, K. Thom, and A. Simpson, "Media Coverage of Homicide Involving Mentally Disordered Offenders: A Matched Comparison Study," *International Journal of Forensic Mental Health*, vol. 6, no. 1, pp. 57-63, 2007
[9] J. G. Boulahanis and M. J. Heltsley, "Perceived Fears: The Reporting Patterns of Juvenile Homicide in Chicago Newspapers," *Criminal Justice Policy Review*, vol. 15, no. 2, pp. 132-160, 2004
[10] Blanco, A. F., Bergel, A., & Alcocer, J. P. S. (2022). Software Visualizations to Analyze Memory Consumption: A Literature Review. ACM Computing Surveys, 55, 1-34.
[11] M. L. Waskom, "Seaborn: Statistical data visualization," Journal of Open Source Software, vol. 6, no. 60, 2021, doi: 10.21105/joss.03021.
[12] C. Sievert, Interactive Web-Based Data Visualization with R, Plotly, and Shiny, Chapman and Hall/CRC, 2020.

Appendix A

REFLECTION ON GROUP WORK

Muhammad Salman Bux

This group project was completed by Syed Anas and I, with both of us equally contributing to the development of the Homicide Media Tracker platform. Our project aimed to create a tool capable of collecting, storing, and analyzing homicide-related data from media reports. This reflection will discuss the scope of the investigation, the work distribution, time management, team communication, challenges encountered, and a review of our group performance.

Project Scope

The project scope involved developing the Homicide Media Tracker platform to enable comprehensive data collection and visualization. Our tasks included dashboard development, visualization implementation, database design. Despite the challenges posed by the scale of the project, our collaborative efforts in discussing and defining the scope ensured that all essential tasks were identified and completed on time. Given the complexity of the project, constant communication and shared insights were crucial for the timely and effective completion of our goals.

Work Distribution

The workload was split evenly between us. I focused more on the dashboard development and visualization, ensuring that the platform provided interactive and user-friendly visualizations. Syed took charge of the database design and backend integration, creating a robust

Appendix B

PROJECT PLAN FOR AN OPEN-SOURCE TOOL FOR HOMICIDE MEDIA ANALYSIS

Syed Mohammad Khizar Anas (2166010), Muhammad Bux (2130436)

School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa

Abstract: This project presents the design and development of the data collection and data analysis tools for a homicide media tracker for humanities researchers who are researching homicide media. The primary objectives are to develop a user interface (UI) for researchers to input information regarding homicide cases and news articles that cover those corresponding cases, to create data visualisation tools that give researchers more insight into homicide cases and to make the homicide media tracker easily accessible to people. The project leverages information engineering principles to create the homicide media tracker a user-friendly website and to develop the homicide media tracker using open-source tools so that it is accessible and free to the researchers. The anticipated outcomes will significantly help researchers in storing and archiving homicide media data and in analysing the data. The project is expected to be completed in 6 weeks, 1 week ahead of the 7-week deadline, providing slack time for any unforeseen delays. This project showcases the interdisciplinary collaboration between engineering and humanities to address the storage of news articles on homicide cases.

Key words: User Interface (UI), newspaper, homicide, Python, Prototype

1. INTRODUCTION

This report presents a comprehensive plan for a sevenweek investigation project commencing on the 26*th* of August and concluding on the 10*th* of October. The project aims at the development of data collection and analysis tools for a homicide media tracker. This is a critical tool for humanities researchers who are inexperienced in handling and representing data.

The primary objective is to design and develop a homicide media tracker website that can store and archive data on homicide cases that are captured by news articles in South Africa. This will be done by creating a UI that researchers will use to enter data on homicide cases that are captured in news articles present on the internet and newspapers. Moreover, data visualisation tools will be designed and developed which will help researchers out in data representation.

The anticipated outcome of this project is expected to help researchers capture and store data on homicide cases in newspapers and news articles in a systematic and fast way without copyright infringement and the data can be exported on any machine in CSV format. Moreover, it will also help them find news articles and newspapers on homicide cases that have happened in South Africa in the past more easily and efficiently. Furthermore, it will allow researchers who are not good at data handling to visualise the data more easily and efficiently.

Section 2 outlines the project scope in which the project background, objectives, assumptions and constraints will be discussed. Section 3 addresses research risks and mitigation strategies. Section 4 identifies the necessary resources, including personnel and equipment. Section 5 details the methodology used to design and develop the homicide media tool. Section 6 outlines project milestones and deliverables. Section 7 discusses the project implementation strategies. Section 8 talks about the project management strategies. After that, the next section concludes the report.

2. PROJECT SCOPE

The scope outlined below includes the project background, objectives, constraints, and assumptions. *2.1 Project Background*

In the current digital age, the accessibility, organization, and analysis of large datasets, particularly from news archives and other document sources, have become increasingly important. However, significant challenges exist in accessing and systematically collecting this data, especially regarding news reporting on homicide cases. Many news archives are locked behind paywalls, restricting access for researchers and the general public. This limitation creates substantial barriers for those without the financial resources to subscribe to multiple news services. Even when accessible, commercial news publications often provide fragmented and incomplete data, making systematic analysis difficult.

The inconsistent nature of data collection further complicates this issue. News is not consistently collected or archived, and various organizations that offer news clipping or archive services are often unreliable. Articles can be lost due to changes in website URLs or the closure of news outlets, leading to incomplete datasets. News coverage does not always accurately represent the reality of crime, with certain types of homicides, particularly those involving male victims, being more frequently reported. Male homicides make up 85% of all murder cases. In South Africa, the high volume of murder cases makes comprehensive coverage unattainable. Less than 20% of female homicides are reported in the news, and coverage is often concentrated in specific areas, neglecting others.

Despite these limitations, news reports provide valuable qualitative insights that cannot be obtained from police reports or statistical data. They offer context, mode, and consequences of crimes, which are crucial for a deeper understanding of homicide cases. However, news creation is qualitative and lacks the systematic approach needed for comprehensive analysis. Additionally, many researchers worldwide lack funding and cannot afford paid news services, hindering their ability to conduct thorough research and analysis.

2.2 Project Objectives

The primary objectives of the project are to develop a platform designed as a homicide media tracker capable of capturing data and metadata from homicide-related news articles reaching as far back as 100 years, as well as the capability to visualize the data utilizing a userfriendly, customisable dashboard. The platform is to be available to as many people as possible, therefore necessitating the use of purely open-source tools.

2.3 Project Assumptions

The following assumptions have been made:

- It is assumed that the database provided is sufficiently filled with accurate information as a starting point.
- The data is sufficiently pre-processed and appropriately labelled, therefore suitable for analysis.
- The data provided complies with all legal and ethical standards.

2.4 Project Constraints

The investigation is limited to several constraints, which are outlined in the following:

- The project is to be completed within a 7week time frame, therefore limiting the extent of features and functionality which can be implemented.
- The platform is to be developed using open-source tools, further limiting functionality.
- The platform must be able to run locally on machines. This limits the ability to use cutting-edge techniques due to computational infeasibility.

3. RISKS

These are the potential risks in this investigation project:

3.1 Low Data Quality and Lack of Availability

There may be limited access to news archives due to paywalls or restricted access policies. There may also be inconsistent, incomplete or incorrect data from news sources. These may hinder the platform's ability to provide accurate and meaningful insights. Mitigation strategies would include implementing robust data cleaning and preprocessing methods, as well as using multiple sources to cross-verify information.

3.2 Compliance Risks

There are no risks related to copyright infringement as the tool will just be used to capture data from articles and store it, rather than capturing the entire work of the author.

3.3 Dependency Risks

Third-party tools and libraries could become obsolete, therefore causing delays in project completion due to needing to search for and implement alternative measures. Mitigation strategies would include choosing well-supported and widely-used technologies, along with having contingency plans for replacing critical dependencies.

3.4 Validation Risks

The strict time frame in which the project needs to be completed may lead to insufficient time for testing should delays take place. This could lead to undetected issues surfacing during public use. Mitigation strategies would include allocating adequate time and resources for thorough testing, and the use of automated testing tools where possible.

4. REQUIRED RESOURCES

4.1 Hardware

- **Computing**: A computer with sufficient CPU and GPU resources is essential for the training of machine learning models and the execution of feature selection algorithms.

4.2 Software

- **Python**: The Python programming environment will be used for data analysis, feature extraction, data visualisation and the application of machine learning libraries. The Python programming language will also be used to create the UI of the website. It will also be used to develop the server required to run the website on.
- **Libraries and Toolkits**: Python offers extensive libraries for data analysis, feature extraction, UI design and development, server development and machine learning, which will be utilized throughout the project.

4.3 Data

- **News articles and newspaper on homicide cases**: Access to 5000 homicide cases that are covered by various new publications is required. These entries of data will be essential to check if the website is functioning properly and if all functions are available. This is crucial in the development of the project.

4.4 Development Tools

- **Version Control**: GitHub will be used for version control to manage code, track changes, and facilitate collaborative development.
- **Communication Tools**: Communication will be maintained through email, Microsoft Teams, and WhatsApp to ensure efficient collaboration and review processes.

4.5 Diagramming and Documentation Tools

- **Draw.io**: This free online tool which will be used for creating diagrams and visual representations of project workflows and results.
- **MATLAB**: MATLAB's built-in diagramming capabilities will also be leveraged. The university will provide the necessary licenses.
- **LaTeX**: Access to online LaTeX editors for documentation and report generation. The university provides free Wi-Fi, which supports the use of these tools.

4.6 Training and Tutorials

- **Machine Learning Tutorials**: Tutorials and instructional materials for the use of machine learning libraries in Python can be obtained from YouTube or other Web resources to ensure that all team members are proficient in the necessary techniques and tools.

5. PROJECT METHODOLOGY

The investigation project will employ a hybrid development process that combines the Agile Software Development Process and the Waterfall Development Process. This hybrid approach is chosen to leverage the strengths of both methodologies which provide a balance between flexibility and structure. This suits the nature of this project.

The Agile Software Development Process is characterised by its iterative and flexible nature, which allows for continuous improvements and adaptation throughout the project. This will minimise risks when adding new functionality to the project. This is because it will allow the project team to work in sprints which will be one week long[1]. In these sprints, the project team will design and develop a prototype of the homicide media tracker and do testing on it to check for bugs. If bugs are found then they will be resolved in the next iteration which will happen in the next sprint and the required functionality will be added to the prototype.

On the other hand, the Waterfall Development Process is a linear and systematic approach that involves progressing through a series of distinct stages, from the initial requirements gathering to final product delivery[2]. This approach provides a clear structure and road map for the project, ensuring that all necessary steps are completed logically and thoroughly.

Using these two approaches combined will allow the project team to proceed through the project easily and efficiently. A flowchart is presented below which summarises the methodology process.

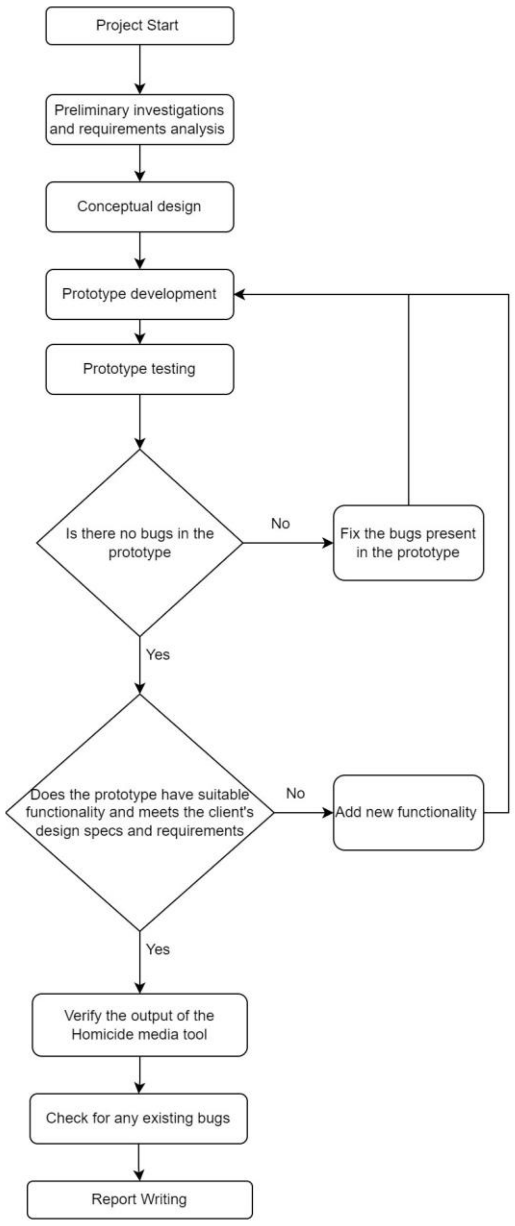

Project Start
Preliminary investigations and requirements analysis
Conceptual design
Prototype development
Prototype testing
Is there no bugs in the prototype
No
Fix the bugs present in the prototype
Yes
Does the prototype have suitable functionality and meets the client's design specs and requirements
No
Add new functionality
Yes
Verify the output of the Homicide media tool
Check for any existing bugs
Report Writing

Figure 1 : Flowchart showing the methodology process.

5.1 Preliminary investigation and requirements

The first stage of the project involves a detailed study of existing apps and websites that store or archive news articles and identifies potential areas of improvement in these apps/websites. Moreover, different open-source libraries in Python will be explored to design and develop the prototype. Alongside this, the specific requirements, constraints and assumptions for the homicide media tracker will be defined by having meetings with the client. The scope of the project and deliverables will be defined as well. Access to the database will be given by the client with 5000 data entries in it.

5.2 Conceptual Design

This phase begins with the development of a preliminary design for the homicide media tracker which is informed by the findings from the initial step taken in the project. The design will integrate innovative solutions to enhance the design of existing homicide media archives/websites. The focus of this stage is to understand the system's functions and the relationships between its components.

5.3 Prototype development and initial testing

In this phase, a prototype of the homicide media tracker will be developed, incorporating all key design elements and operational parameters defined during the planning stage. There will also be a focus on data scraping and processing scripts, as well as the development of the layout and interactive elements of the dashboard. Initial validation tests will be conducted on this prototype to verify the initial functionality that is implemented and adherence to the design specifications. Moreover, the back end and front end will be connected, ensuring seamless data flow and interaction. This phase will also involve the implementation of callbacks and dynamic updates within the dashboard to reflect real-time data analysis *5.4 Iterative testing and improvements*

Following the completion of the preliminary prototype, the project will transition into an iterative process of testing, feedback, and refinement. Each iteration will thoroughly test the homicide media tool's functionality and efficiency. This will be done to check for system bugs and to see if the functionality implemented is operating correctly. Bugs existing in the system will be fixed. If there are no bugs, then new functionality will be added to the prototype which adheres to the requirements of the project. This will then become the next prototype. Rigorous testing will be done again to check if any new bugs were introduced due to the implementation of new functionalities in the website. Each modification and its impacts will be meticulously documented, contributing to the comprehensive reporting of the project's progression and outcomes.

Potential risks will be identified and managed proactively throughout the project life cycle. Regular risk assessment meetings will be held to identify new risks, evaluate their impact, and develop mitigation strategies.

5.5 Final evaluation, verification and reporting

Upon completion of the iterative testing and improvement phase, a final round of comprehensive testing will be conducted on the refined prototype. Specific details will be obtained from the client to determine the overall criteria for a successful product that can be validated. Another round of testing will be done to verify there are no bugs present in the final prototype and it adheres to the client's requirements.

Comprehensive documentation will be maintained throughout the project, including design documents, code comments, and user manuals. This will ensure that the platform is maintainable and extensible in the future. Regular progress reports will be prepared and shared with stakeholders, providing transparency and accountability.

All findings, methodologies, insights and bugs will be compiled into a comprehensive investigation report. This report will detail the investigation process, summarise the results, discuss any observed bugs or anomalies in the system, and propose recommendations for future research.

6. PROJECT MILESTONES

To track the project's progress and to ensure that project completion will be on time, a set of milestones has been chosen. The project is projected to be completed in 6 weeks which is 1 week ahead of the 7 weeks deadline. This allows for slack time to accommodate any unforeseen delays that may arise. Table 1 below, shows the list of milestones.

Table 1 : Milestone Estimation

Milestone	**Deadline**
Meet with researchers	26 August
Literature Review Complete	28 August
Become familiar with database	30 August
Begin prototype application development	2 September
Prototype complete	12 September
Testing commences	12 September
Testing complete	15 September
Start corrective actions	16 September
Corrective actions complete	21 September
Meeting with researchers	2 October
Poster preparations begin	5 October
Poster Complete	9 October
Open Day	10 October

7. PROJECT IMPLEMENTATION

The homicide media tool will be designed and developed using Python. In Python, the Dash library will be used which is an open-source framework specifically designed for building interactive web-based applications and dashboards [3]. It seamlessly integrates with Plotly's graphing library to provide a rich set of visualization capabilities, making it an excellent choice for creating complex and data-driven dashboards. Dash applications are web servers that run Flask, communicate with JSON packets over HTTP requests, and update web pages via React.js, offering a powerful and flexible way to develop interactive, user-friendly interfaces[4].

8. PROJECT MANAGEMENT

To facilitate the execution of the project, the processes outlined in the Project Methodology section are further decomposed into specific sub-tasks. These subtasks, which are guided by the milestones established in the milestone section which are detailed in the Work Breakdown Structure found in Appendix A. Moreover, the project's progression is monitored and managed in line with the Gantt chart in Appendix B specifically developed for this project.

8.1 Task distribution

The project will be managed collaboratively between the two primary developers, Muhammad Bux and Syed Anas, who will share the workload evenly to ensure efficiency and thorough coverage of all tasks and sub-tasks.

In the initial phase, Syed will be responsible for setting up the meetings with the client. These meetings will be held regularly to discuss project progress, address any challenges, and ensure alignment with the project objectives. Communication will be maintained through daily stand-ups and weekly review sessions, ensuring continuous feedback and adjustment of tasks as needed.

After the client meeting, the literature review and development of design specifications will be conducted in tandem by both Syed and Muhammad. The dual involvement will help in the thorough understanding and exploration of the research done on existing solutions and their shortcomings. It will also make it easier and more efficient to formulate the design specifications of the homicide media tracker.

After the design specifications are established, both team members will contribute equally to the design and development of the prototype. Collaboration at this stage ensures a fusion of skills and expertise, culminating in an initial prototype that meets the established design specifications.

Syed will do the testing on the initial prototype. On the other hand, Muhammad will design and develop a new functionality that will be added to the prototype which will form the next prototype. Muhammad will also test for bugs in the new prototype. While Muhammad is testing for bugs, Syed will start the development process for the next functionality that needs to be added to the prototype forming the next prototype. He will then check for any bugs present in the new prototype. This will continue until the final prototype is developed which will be under the design specifications and will have no bugs in it.

Both Syed and Muhammad will try and identify potential risks during the testing phase for the homicide media tracker. If any risks are found then a meeting will be held between the two teammates in which the impact of the risk on the project will be discussed and a mitigation strategy will be drafted and implemented.

The data analysis and report drafting will be done individually. After this Syed and Muhammad will work together to create a video presentation of the project.

9. CONCLUSION

The Homicide Media Tracker project aims to enhance the systematic collection and analysis of homiciderelated news data by developing an open-source digital platform. Leveraging Plotly's Dash for an interactive dashboard and advanced machine learning models for data analysis, the platform will provide valuable insights into homicide reporting patterns. Through a collaborative approach between Muhammad and Syed, rigorous testing, proactive risk management, and thorough documentation, the project ensures a reliable, user-friendly, and extensible tool. This initiative will significantly enhance data accessibility, foster community engagement, and support informed research and policy-making in the realm of crime data analysis.

REFERENCES

[1] I. wrike. "The Agile Software Development Life Cyle." https://www.wrike.com/agile-guide/ agile-development-life-cycle/, 2024. Accessed: 14 July 2024.

[2] M. McGuire. "Top 4 software development methodology." https://www. synopsys.com/blogs/software-security/ top-4-software-development-methodologies. html, 2024. Accessed: 14 July 2024.

[3] Plotly. "Dash documentation and user guide." https://dash.plotly.com/, 2024. Accessed: 14 July 2024.

[4] F. S. Ltd. "Dash Bootstrap Components." https://dash-bootstrap-components. opensource.faculty.ai/, 2024. Accessed: 14 July 2024.

Appendix A

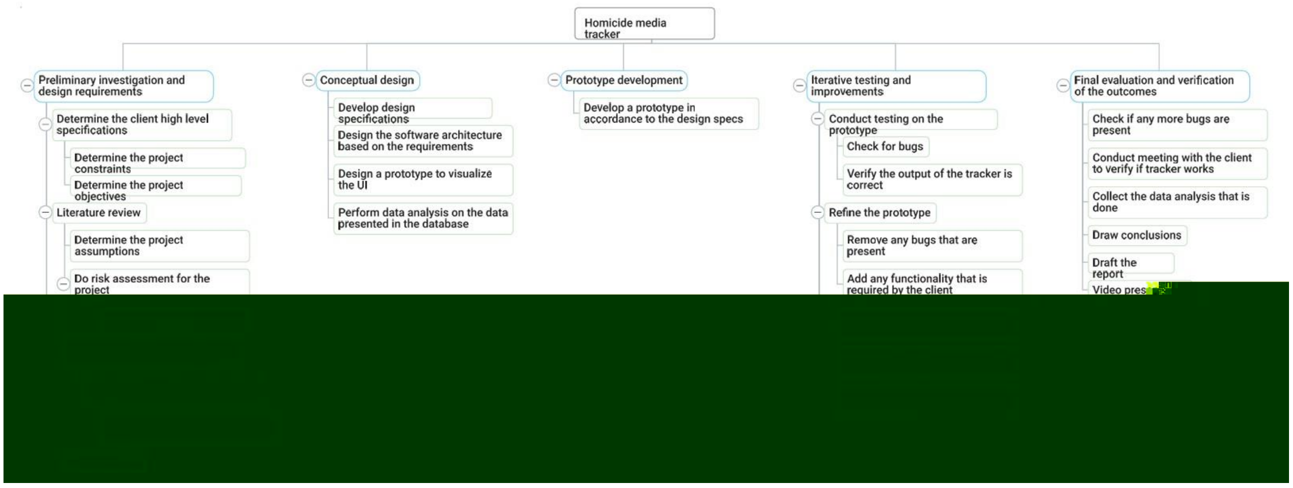
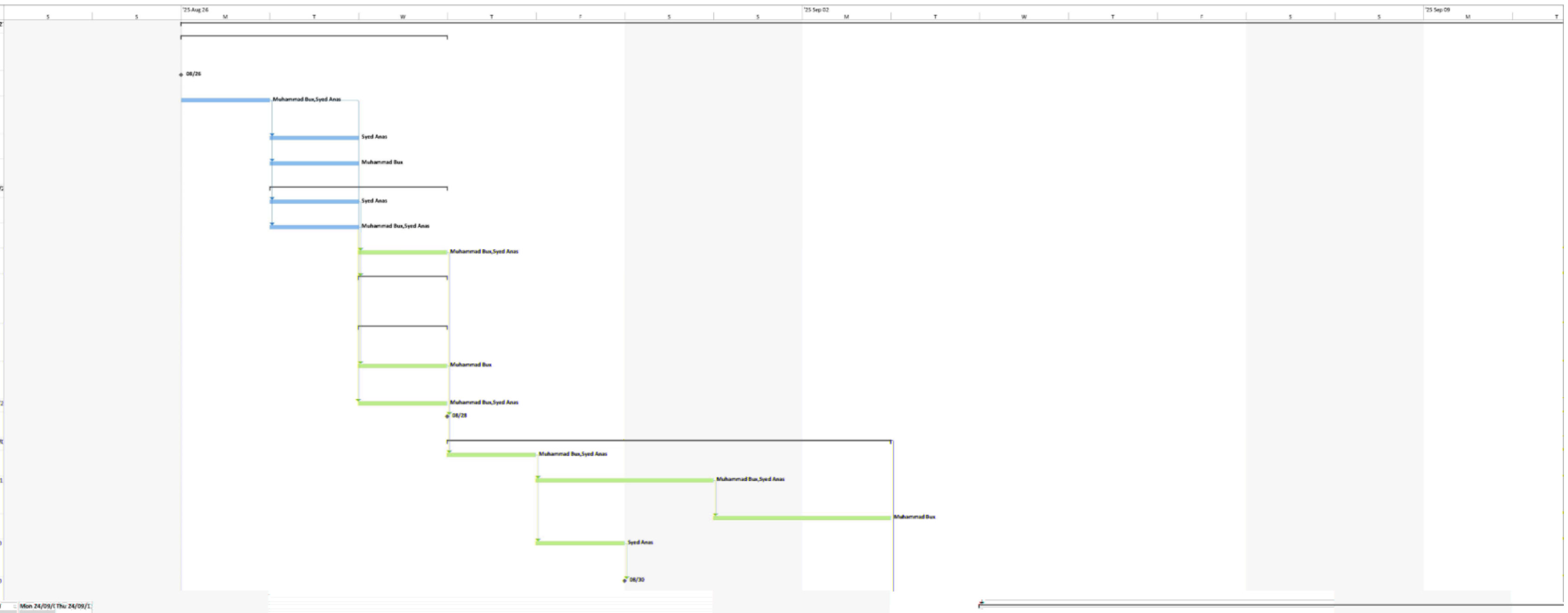

Homicide media tracker
Preliminary investigation and design requirements
Determine the client high level specifications
Determine the project constraints
Determine the project objectives
Literature review
Determine the project assumptions
Do risk assessment for the project
Conceptual design
Develop design specifications
Design the software architecture based on the requirements
Design a prototype to visualize the UI
Perform data analysis on the data presented in the database
Prototype development
Develop a prototype in accordance to the design specs
Iterative testing and improvements
Conduct testing on the prototype
Check for bugs
Verify the output of the tracker is correct
Refine the prototype
Remove any bugs that are present
Add any functionality that is required by the client
Final evaluation and verification of the outcomes
Check if any more bugs are present
Conduct meeting with the client to verify if tracker works
Collect the data analysis that is done
Draw conclusions
Draft the report
Video pres

Figure A1 : Work breakdown structure of the project plan

ID	Task Mode	Task Name	Duration	Start	Finish
1		**Homicide media trac**	**62,33 d**	**Mon 24/08/2**	**Sun 24/10/2**
2		**Preliminary investigation and design requirements**	**3 d**	**Mon 24/08/26**	**Wed 24/08/28**
3		Meet up with the researchers	0 d	Mon 24/08/26	Mon 24/08/26
4		Determine the client high level design requirements	1 d	Mon 24/08/26	Mon 24/08/26
5		Determine the project constraints	1 d	Mon 24/08/26	Tue 24/08/27
6		Determine the project objectives	1 d	Mon 24/08/26	Tue 24/08/27
7		**Literature revie**	**2 d**	**Mon 24/08/2**	**Wed 24/08/2**
8		Determine the project assumptions	1 d	Mon 24/08/26	Tue 24/08/27
9		Do risk assessment for the	1 d	Mon 24/08/26	Tue 24/08/27
10		Devise mitigation strategies	1 d	Tue 24/08/27	Wed 24/08/28
11		**Do research on the hardware and software requirements**	**1 d**	**Tue 24/08/27**	**Wed 24/08/28**
12		**Research what coding language**	**1 d**	**Tue 24/08/27**	**Wed 24/08/28**
13		Define the libraries that will	1 d	Tue 24/08/27	Wed 24/08/28
14		Client sign	1 d	Tue 24/08/2	Wed 24/08/2
15		Literature review complete	0 d	Wed 24/08/28	Wed 24/08/28
16		**Conceptual design**	**5 d**	**Wed 24/08/2**	**Mon 24/09/0**
17		Develop design specifications	1 d	Wed 24/08/28	Thu 24/08/29
18		Design the software architecture based on the	2 d	Thu 24/08/29	Sat 24/08/31
19		Design a prototype to visualize the UI	2 d	Sat 24/08/31	Mon 24/09/02
20		Perform data analysis on the data presented in the database	1 d	Thu 24/08/29	Fri 24/08/30
21		Become familiar with the database	0 d	Fri 24/08/30	Fri 24/08/30
22		**Prototype develop**	**10 d**	**Mon 24/09/0**	**Thu 24/09/1**
23		Begin prototype development	0 d	Mon 24/09/02	Mon 24/09/02
24		Develop a prototype in accordance to the design	10 d	Mon 24/09/02	Thu 24/09/12
25		Complete prototype development	0 d	Thu 24/09/12	Thu 24/09/12
26		**Iterative testing and improvements**	**16,99 d**	**Thu 24/09/12**	**Sun 24/09/29**
27		Testing commen	0 d	Thu 24/09/1	Thu 24/09/1
28		**Conduct testing on the prototype**	**3 d**	**Thu 24/09/12**	**Sun 24/09/15**
29		Check for bug	1 d	Thu 24/09/1	Fri 24/09/13
30		Verify the output of the tracker is	2 d	Fri 24/09/13	Sun 24/09/15
31		Testing compl	0 d	Sun 24/09/1	Sun 24/09/1
32		**Refine the protc**	**6 d**	**Sun 24/09/1**	**Sat 24/09/21**
33		Remove any bugs that are present	1 d	Sun 24/09/15	Mon 24/09/16
34		Start corrective actions	0 d	Mon 24/09/16	Mon 24/09/16
35		Add any functionality that is required by the client	5 d	Mon 24/09/16	Sat 24/09/21
36		Corrective actions complete	0 d	Sat 24/09/21	Sat 24/09/21
37		Do Retests to find any bugs that are present	1 d	Sat 24/09/21	Sun 24/09/22
38		*Add additional functionality and remove any bugs*	6 d	Sun 24/09/22	Sat 24/09/28
39		Do a risk analysis to verify if any risk is present	1 d	Sat 24/09/28	Sun 24/09/29
40		**Final evaluation and verification of the outcomes**	**27,35 d**	**Sun 24/09/29**	**Sun 24/10/27**
41		Check if any more bugs are present	2 d	Sun 24/09/29	Tue 24/10/01
42		Conduct meeting with the client to verify if tracker	1 d	Tue 24/10/01	Wed 24/10/02
43		Meeting with the researcher	0 d	Wed 24/10/02	Wed 24/10/02
44		Collect the data analysis that is done	1 d	Wed 24/10/02	Thu 24/10/03
45		Draw conclusion	2 d	Thu 24/10/0	Sat 24/10/05
46		Poster preparati	0 d	Sat 24/10/05	Sat 24/10/05
47		Prepare for oper	4 d	Sat 24/10/05	Wed 24/10/C
48		Poster complete	0 d	Wed 24/10/0	Wed 24/10/C
49		Open day	0 d	Thu 24/10/1	Thu 24/10/1
50		Draft the report	9 d	Thu 24/10/1	Sat 24/10/19
51		Video presentat	8 d	Sat 24/10/19	Sun 24/10/2

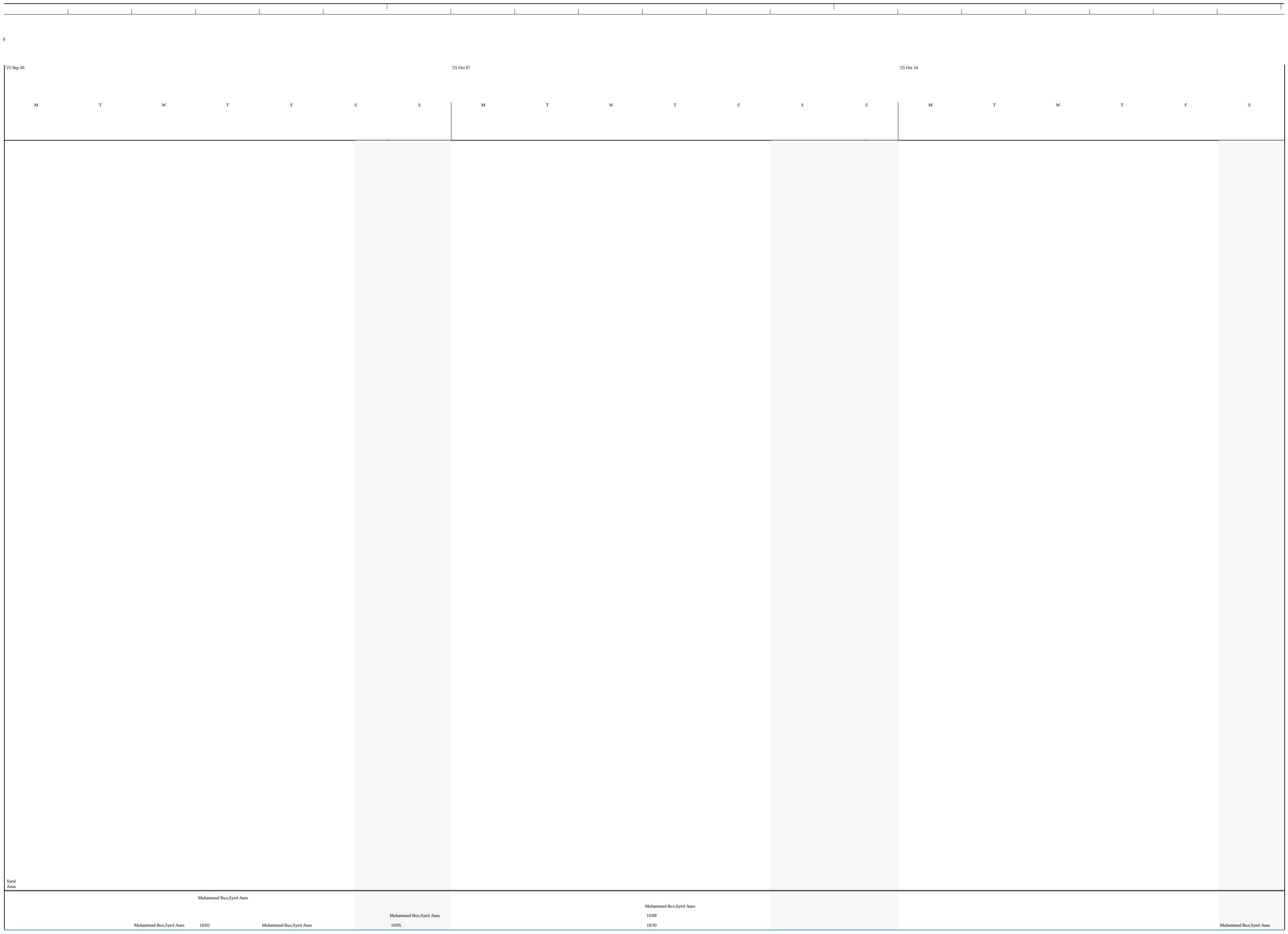


Project: Homicide media tracker Date: Mon 24/07/15								
Task	Milestone	Project Summary	External Milestone	Inactive Milestone	Manual Task	Manual Summary Rollup	Start-only	Deadline
Split	Summary	External Tasks	Inactive Task	Inactive Summary	Duration-only	Manual Summary	Finish-only	Progress
								Manual Progress

Project: Homicide media tracker Date: Mon 24/07/15	Task	🟦	Milestone	◆	Project Summary	⊢⊣	External Milestone	◆	Inactive Milestone	◆	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	◆	Manual Progress	▬
	Split	⋯	Summary	⊢⊣	External Tasks	▬	Inactive Task	▭	Inactive Summary	⊢⊣	Duration-only	▬	Manual Summary	⊢⊣	Finish-only	]	Progress	▬		

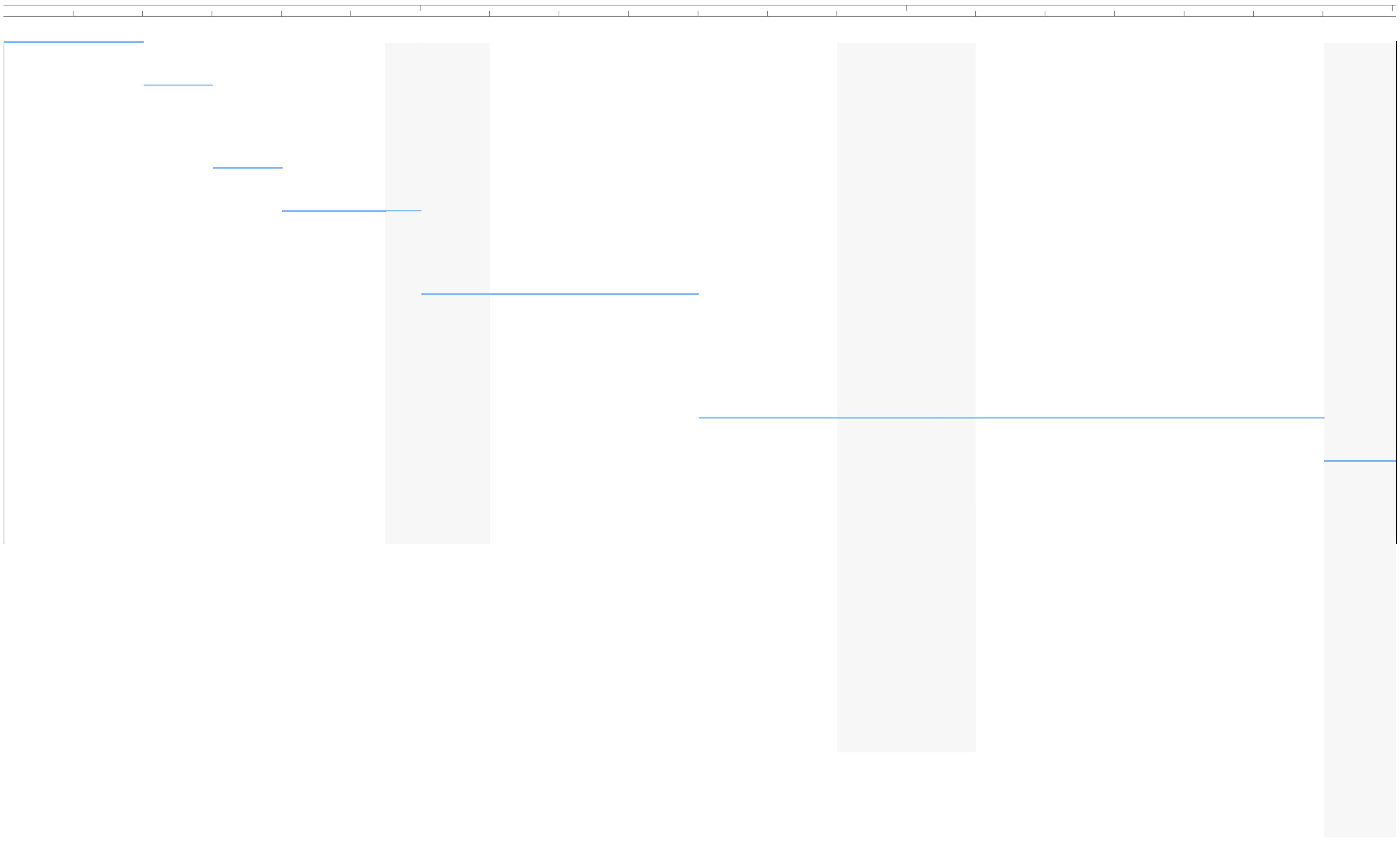
'25 Sep 23

S S M T W T F S S



Project: Homicide media tracker
Date: Mon 24/07/15

Task	▬	Milestone	◆	Project Summary	┌─┐	External Milestone	◆	Inactive Milestone	◇	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	⬇	Manual Progress	▬
Split	⋯	Summary	┌─┐	External Tasks	▬	Inactive Task	▭	Inactive Summary	┌─┐	Duration-only	▬	Manual Summary	┌─┐	Finish-only	]	Progress	▬		



Project: Homicide media tracker	Task	▬	Milestone	◆	Project Summary	⊓	External Milestone	◆	Inactive Milestone	◇	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	◆	Manual Progress	▬
Date: Mon 24/07/15	Split	······	Summary	⊓	External Tasks	▬	Inactive Task	▭	Inactive Summary	⊓	Duration-only	▬	Manual Summary	⊓	Finish-only	]	Progress	▬		

S	'25 Oct 21 M	T	W	T	F	S

Project: Homicide media tracker Date: Mon 24/07/15	Task	▬	Milestone	◆	Project Summary	┌─┐	External Milestone	◆	Inactive Milestone	◇	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	◆	Manual Progress	▬
	Split	- - - - - -	Summary	┌─┐	External Tasks	▬	Inactive Task	▭	Inactive Summary	┌─┐	Duration-only	▬	Manual Summary	┌─┐	Finish-only	]	Progress	▬		

Project: Homicide media tracker Date: Mon 24/07/15	Task	▬	Milestone	◆	Project Summary	⊢⊣	External Milestone	◆	Inactive Milestone	◇	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	⬇	Manual Progress	▬
	Split	‐ ‐ ‐ ‐ ‐ ‐	Summary	⊢⊣	External Tasks	▬	Inactive Task	▭	Inactive Summary	⊢⊣	Duration-only	▬	Manual Summary	⊢⊣	Finish-only	]	Progress	▬		

S

Project: Homicide media tracker Date: Mon 24/07/15	Task	▬	Milestone	◆	Project Summary	⊢⊣	External Milestone	◆	Inactive Milestone	◇	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	◆	Manual Progress	▬
	Split	⋯	Summary	⊢⊣	External Tasks	▬	Inactive Task	▭	Inactive Summary	⊢⊣	Duration-only	▬	Manual Summary	⊢⊣	Finish-only	]	Progress	▬		

Muhammad Bux,Syed Anas

S S

Project: Homicide media tracker Date: Mon 24/07/15	Task	▬	Milestone	◆	Project Summary	⊏⊐	External Milestone	◇	Inactive Milestone	◇	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[
	Split	┈	Summary	⊏⊐	External Tasks	▬	Inactive Task	▭	Inactive Summary	⊏⊐	Duration-only	▬	Manual Summary	⊏⊐	Finish-only	]
	Deadline	◆	Manual Progress	▬												
	Progress	▬														

Project: Homicide media tracker	Task	▬	Milestone	◆	Project Summary	▬	External Milestone	◆	Inactive Milestone	◆	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	◆	Manual Progress	▬
Date: Mon 24/07/15	Split	------	Summary	▬	External Tasks	▬	Inactive Task	▭	Inactive Summary	▬	Duration-only	▬	Manual Summary	▬	Finish-only	]	Progress	▬		

'25 Oct 28'25 Nov 04

Task	▬	Milestone	◆	Project Summary	⊢⊣	External Milestone	◆	Inactive Milestone	◇	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	◆	Manual Progress	▬
Split	------	Summary	⊢⊣	External Tasks	▬	Inactive Task	▭	Inactive Summary	⊢⊣	Duration-only	▬	Manual Summary	⊢⊣	Finish-only	]	Progress	▬		

M T W T FM T W T F

Project: Homicide media tracker Date: Mon 24/07/15	Task	▬	Milestone	◆	Project Summary	⊓	External Milestone	◆	Inactive Milestone	◇	Manual Task	▬	Manual Summary Rollup	▬	Start-only	[	Deadline	◆	Manual Progress	▬
	Split	⋯	Summary	⊓	External Tasks	▬	Inactive Task	▭	Inactive Summary	⊓	Duration-only	▬	Manual Summary	⊓	Finish-only	]	Progress	▬		

Appendix C

Cohort 9

Week 1: Meeting Minutes

Meeting Date: 28/08/2024
Cohort Participants: Akiva Levitt, Joseph Kaplan, Edgar Ngoepe, Phuti Mokgehle, Muhammad Bux, Syed Anas, Shen Reddy, Dylan Baker, Edward Ndlala and Lefa Mofokeng
Meeting Chair: Akiva Levitt
Minute Taker: Joseph Kaplan

Overview: Introduction meeting introducing each other, discussing each project overview and finalising admin details. Group agreed to meet every Monday at 9-10 from now on.

Group 11: Edgar Ngoepe and Phuti Mokgehle

- Project Objective: Determining the optimum route up a sport or bouldering climbing wall
- This week: Visit the CityRock gym, capture high-quality images of the climbing wall from multiple angles and create a 3D Representation of the Climbing Wall.

Group 19: Akiva Levitt and Joseph Kaplan

- Building a VS Code extension that allows new programming C++ students to learn how pointers work, by visualizing the code provided.
- We have investigated various ways of going about doing the project, using JavaScript for backend, with Python to run Clang to interpret C++ code.
- Launch a VSCode extension that can be downloaded (1 step before an MVP)
- This week: Need to figure out how to launch a VSCode extension and subsequently decide the best way to take in the C++ code as input and output the corresponding visual representation of the pointer logic in the code

Group 22: Muhammad Bux and Syed Anas

- Building an open-source platform for tracking information retrieved from media about homicide cases and producing insights using various data visualization methods
- Database has been built previously, supervisor needs to provide access
- Using plotly dash to build an interactive dashboard for the data
- This week: get access to database and integrate database with the dashboard

Group 41: Shen Reddy and Dylan Baker

- Project Summary: Benchmarking TinyML systems. We will be comparing the performance of ML algorithms for anomaly detection in laptops and microcontrollers. We will also be comparing the energy consumption of both systems.
- Some research has been conducted into potential microcontrollers for the project as well as potential algorithms that can be used.
- This week: we will continue with our research and will aim to find datasets for anomaly detection as well as design an anomaly detection system for the laptop solution.

Group 47: Edward Ndlala and Lefa Mofokeng

- Project summary: investigation into power usage of IT Equipment based on varying computational load.
- Building algorithms that solves soduko and ticktactoe then measure Their computational power.
- This week: Research how the smart powerr distribution unit (PDU) device operates. Researching algorithms and soduko. This week is based on understanding the PDU and the algorithms

Week 2: Meeting Minutes

Meeting Date: 02/09/2024
Cohort Participants: Akiva Levitt, Joseph Kaplan, Edgar Ngoepe, Phuti Mokgehle, Muhammad Bux, Syed Anas, Shen Reddy, Dylan Baker, Edward Ndlala and Lefa Mofokeng
Meeting Chair: Phuti Mokgehle
Minute Taker: Edgar Ngoepe

Overview: 2nd meeting each group shared what they worked on the previous week, plans for the upcoming week and any challenges encountered the previous week

Group 11: Edgar Ngoepe and Phuti Mokgehle

- We successfully visited the City Rock gym and captured high-quality images of the climbing wall from multiple angles.
- However, we faced challenges when attempting to create a 3D representation of the climbing wall. The software we obtained presented installation issues, and when we managed to execute the programs, our PCs struggled to handle the computations, with some tasks taking more than seven hours to complete.
- Plans for the Week: Finish the 3D reconstruction and start modelling the climber with the idea of developing a stickman.

Group 19: Akiva Levitt and Joseph Kaplan

- We have successfully published a VSCode extension, and it is now live. We successfully decided on and have implemented clang as the compiler that converts the C++ code into an AST that is stored in a JSON format. The visualiser aspect takes this JSON format and visualises basic pointer functionality using D3.js based on the AST.
- We need to create the google forms and the tutorial that will allow users to test our software. Our goal is to integrate the tutorial and questionnaires into a single streamlined environment so that it is plug and play for users.
- We have been struggling to figure out how to correctly integrate a video with audio into our VSCode extension. We think that there are VSCode settings that prevent this from being possible, but we will work more on this.

Group 22: Muhammad Bux and Syed Anas

- Last week: we were able to get access to the database and the data. We downloaded and initialised the database and found that some functionalities of the database do not work like adding data into the database, deleting data from the database and editing data in the database.
- This week: This week we will try to figure out how to make the functionalities that does not work in the database to work. If we cannot make the functionality work, then we will design

and develop the database from scratch and input the data provided to us. Moreover, we will do some research on user-friendly dashboard online which will help us design our dashboard this week.

- Challenges faced: The challenge we faced is the misconception we had about the project, but that misconception was cleared with the meeting with our supervisor. But this made us a bit behind on our project according to the project plan.

Group 41: Shen Reddy and Dylan Baker

- last week we investigated different options for our microcontroller and have come to a decision on which one we'll be using. We also investigated different machine learning algorithms that could be used for anomaly detection in our system, such as CNNs and KNNs. We also started development on the laptop/pc ML system.
- This week we plan to order and obtain our microcontroller, continue with development on our laptop system, as well as obtain multiple dataset sources for image anomaly detection
- Currently we have not encountered any challenges yet with our project development

Group 47: Edward Ndlala and Lefa Mofokeng

- We have successfully implemented the backtracking algorithm and constant propagation algorithm. We made an extension cord for the smart PDU device.
- This week, we will be benchmarking the project matrixs. Test the developed algorithms and get some insights on how we can change the computational power
- We will also be acquiring login details for access to the smart PDU metrics system and run test measurements to determine its capabilities
- Encountered challenges regarding acquiring exact baseline values of memory usage, cpu usage and drive storage.

Week 3: Meeting Minutes

Meeting Date: 09/09/2024
Cohort Participants: Akiva Levitt, Joseph Kaplan, Edgar Ngoepe, Phuti Mokgehle, Muhammad Bux, Syed Anas, Shen Reddy, Dylan Baker, Edward Ndlala and Lefa Mofokeng
Meeting Chair: Muhammad Bux
Minute Taker: Syed Anas

Overview: This is the third meeting. In it each group shares the progress they have made in their project in the last week. The group also discusses their plans for the project for the week and lastly the challenges that are faced by each group in the project are discussed.

Group 11: Edgar Ngoepe and Phuti Mokgehle

- Progress this Week:
- We successfully created the basic stickman model and developed the 3D climbing wall. These components are ready for integration to simulate a climber's movement on the wall.
- Challenges:
- Simulating dynamic movements with high realism remains a challenge. We need to make further enhancements to the physics-based model to achieve more accurate results.
- Plans for the Week:

- Explore different machine learning algorithms or models to optimize the climbing route, focusing on determining the most efficient paths based on simulated movements.
- Integrate the stickman with the 3D climbing wall to allow us to visualize the climber's movements on the wall.

Group 19: Akiva Levitt and Joseph Kaplan

- Last week:
- We created videos that will help users learn about pointers and guide them through using the software. Adding the videos to the VS Code extension with audio was very difficult however we eventually got it working. We also improved the UI of the software making it more dynamic and user friendly.
- This week:
- Need to fix functionality regarding visualizing arrays and pointers to arrays. Need to finialise the UI look and feel. Need to finalise the Google forms and questionnaires along with the flow of the tutorial. We would like to finish functionality so we can get ready to start experimenting with our product.

Group 22: Muhammad Bux and Syed Anas

- Last week: We built our own database from scratch as the database provided was not functional. Moreover, in the database, three tables were built, relationship between the tables were developed and data in a csv file was imported into the database. Furthermore, the dashboard was designed and developed and the export to a csv file functionality was developed and implemented.
- Challenges: One of the challenges that we faced last week was implementing the insert functionality in the dashboard as it was giving an error therefore data cannot be inputted from the dashboard. Moreover, as three tables are used in the database therefore; to show all the data in the database on the dashboard as one table is a challenge.
- This week: We will work on the insert functionality on the dashboard to make it work. Other functionalities such as the delete, edit and import from a csv file will be developed and implemented into the dashboard. Also, a one table database will be developed to allow for easy representation of the whole data on the dashboard. Lastly, investigation will be done to see which Machine Learning techniques need to be implemented on the data.

Group 41: Shen Reddy and Dylan Baker

- Last week: We worked on implementing a CNN algorithm to perform basic image anomaly detection and were able to successfully implement it. We were also able to implement TensorFlow to work on our laptop GPUs instead of the CPUs. We also looked at other potential algorithm options to implement other than CNNs, such as K-Nearest Neighbours. We also briefly looked at options for energy measurement.
- This week: We will continue to investigate options for algorithms, continue implementing algorithms and start on our energy measurement system design.
- Challenges: So far, the only challenge we faced was getting the right version of TensorFlow to work with the Windows CUDA toolkit, but we managed to resolve the issue.

Group 47: Edward Ndlala and Lefa Mofokeng

- Last week:

- We have successfully implemented the two new algorithms which are dancing link and genetic. Moreover, we did benchmark for the smart PDU and worked on automatically logging the benchmarking of the cpu usage and memory usage
- This week.
- We will try to test the algorithms and metrics. We will develop a sudoku generator. We will also develop a revised work breakdown structure.
- Challenges:
- The challenges that we faced are that we had constraints in logging the consumption on one pc since it couldn't access the PDU metrics page. Also, we couldn't log other PC metrics on the computer as well as it was slow to retrieve the data we required.

Week 4: Meeting Minutes

Meeting Date: 16/09/2024
Cohort Participants: Akiva Levitt, Joseph Kaplan, Edgar Ngoepe, Phuti Mokgehle, Muhammad Bux, Syed Anas, Shen Reddy, Dylan Baker, Edward Ndlala and Lefa Mofokeng
Meeting Chair: Shen Reddy
Minute Taker: Dylan Baker

Overview: 5th week of the investigation project. The groups briefly discussed what they did in week 3, what their plans are for this week, and if they are struggling with any aspect of the project.

Group 11: Edgar Ngoepe and Phuti Mokgehle

- Progress This Week: We successfully:
- Explored two different machine learning algorithms (A* search and Dijkstra’s pathfinding algorithms) to determine the optimal climbing route.
- Integrated the stickman with the 3D climbing wall to visualize the climber's movements on the wall.
- Challenges: No changes arose for the past week.
- Plans for the Week:
- Simulate dynamic movements of the climber/stickman with high realism.
- Make further enhancements to the physics-based model to achieve more accurate results.

Group 19: Akiva Levitt and Joseph Kaplan

- Last week we fixed the pointer functionality regarding pointers and arrays, made each element of the array a separate block and allowed for the incrementing and decrementing of array pointers. We fixed dereferencing functionality for various pointer operations. We added error handling for C++ code that doesn’t compile. We also improved the UI, flow and questions for the tutorial and experiment part of our extension. We also added a few more tutorial videos.
- This Week: We need to run a pilot session with a user and our software so that we can get feedback for finalising our software, the UI, and the tutorial flow. Then we will hopefully be able to start testing our software and send out messages asking people to test it. We need to research Cronbach's alpha and p values relating to data and statistical importance.

Group 22: Muhammad Bux and Syed Anas

- Past week: We got the dashboard working. We investigated multiple data visualization techniques and implemented some of them into our dashboard. We implemented a bar graph and line graph which shows the homicides that took place over time. Also, a choropleth was implemented to show which provinces, the homicides have taken place. Moreover, we developed the add data, delete data and export to CSV format functionality in the dashboard.

- Challenges faced: The challenge that we are currently facing is to find the investigative element in our project. The investigative element of the project needs to be different for both partners which is suggested by Dr. Yuval.

- This week: More research will be done on different types of data visualization techniques and how we can implement them in our dashboard. Moreover, some other functionality such as finding duplicates, delete data manually and import to CSV file format will be implemented in the dashboard. Furthermore, more research will be conducted to find the investigative element of the project.

Group 41: Shen Reddy and Dylan Baker

- Last week we successfully implemented a version of a Convolution Neural Network (CNN) algorithm, for our PC system, to detect defects in images of screws and to detect clouds in the sky. We also obtained our microcontroller for the project and started investigating the software IDE for it. We implemented a basic demo application on the microcontroller just to understand how the system worked.
- This week we need to implement 2 more ML algorithms for our PC system (potentially KNN, SVM or Isolation Forests). We also need to try get an ML model on the microcontroller and we also need to devise an energy measurement system for both of our systems.
- The only challenge we faced was implementing code on the microcontroller due to deprecated methods in the software IDE.

Group 47: Edward Ndlala and Lefa Mofokeng

- Executed 2 more algorithms and recorded their execution time for sudoku boards of different difficulties

- Recorded baseline Cpu time and memory usage using the performance monitor

- Used the Hardware Monitor software to record baseline cpu tenperature
- Created the sudoku board generator.

- Automatically logging the metrics in c++ and HW monitor.

- Constraints included being unable to record values overtime as HW Monitor only logs data at specific time and not over a period.

- This week we will focus on testing the algorithms based on different difficulty and the number of sudoku boards.

Week 5: Meeting Minutes

Meeting Date: 23/09/2024
Cohort Participants: Akiva Levitt, Joseph Kaplan, Edgar Ngoepe, Phuti Mokgehle, Muhammad Bux, Syed Anas, Shen Reddy, Dylan Baker, Edward Ndlala and Lefa Mofokeng
Meeting Chair: Edward Ndlala
Minute Taker: Lefa Mofokeng

Overview: 6th week of the investigation project. The groups discussed the progress they have made in the past week, the issues they encountered and what their objectives are in the following week.

Group 11: Edgar Ngoepe and Phuti Mokgehle

- Progress this week: We integrated a basic stickman model capable of climbing the wall.
- The model successfully simulates hand and foot movements as it ascends the climbing wall.
- The A* search algorithm has been completed, and it successfully determines the optimal climbing routes based on the designated holds.
- Challenges: One of the challenges we encountered was the appearance and movement of the stickman.
- Plans for the coming week: Our next focus is on perfecting the stickman's appearance and movement.
- We will compare the two implemented algorithms (A* search and Dijkstra) and finalize our analysis on the most efficient route-finding algorithm for the climbing wall.

Group 19: Akiva Levitt and Joseph Kaplan

- Last Week: we ran a pilot on two people with our software, got great feedback and then implemented the changes necessary. Thereafter we ran our software experiment on 5 people so far and have seen great results showing that the software improves user understanding of pointers. We did research into Conbachs alpha and p values on how we will determine the validity of our experiment.
- This week we will be finding participants to partake in our research experiment testing out our software.

Group 22: Muhammad Bux and Syed Anas

- Last week: We added more functionality into our dashboard such as being able to export the database tables as csv, import csv into a table in the database, being able to find duplicates in the table and delete those duplicates as well as keeping a record of the deleted duplicates by creating a duplicates table and storing it there. We also implemented 9 visualisation plots to see the correlation between the different fields in the table.
- This week: We have a meeting with Dr. Yuval Genga in which we will show a demo of our dashboard and depending on his criticism and advise on future improvements which we will add into our project. We will also do more research on the investigative element of the project and work on it this week.

- Challenges faced: The only challenge we are facing right now is time management. Even though we have added most of the functionalities in our project and visualisation techniques, we still might have to add a few more depending on Dr. Yuval Genga's suggestions. While we also need to do the investigative element of the project and work on it to get results for our project.

Group 41: Shen Reddy and Dylan Baker

- Last week we were able to implement a basic computer vision model using a pure CNN algorithm as well as an autoencoder CNN algorithm on our laptop systems. We were able to compress the computer vision model and put it on our microcontroller. We also started developing software to display the results of microcontroller models on the LCD screen on the microcontroller.
- This week we need to compress the autoencoder algorithm and put it on the microcontroller, and we need to start benchmarking the performance on microcontroller. We also need to develop one more ML algorithm for both systems.
- The only challenge we faced was being able to use the LCD display screen. We need to investigate more online tutorials to help us get the screen properly set up for use.

Group 47: Edward Ndlala and Lefa Mofokeng

- Last week: Switched to HWinfo instead of HW Monitor which produces hardware metrics overtime and stores, data in a .csv file
- Recorded the baseline temperatures of CPU Package and Cores
- Tested the execution of the algorithms, the speed, memory usage and CPU consumption on different difficulties.
- No challenges of note in the prior week.
- This week: we are doing extensive testing and measurement. We will be running each algorithm on 2 PCs and collecting and analyzing the data.

Week 6: Meeting Minutes

Meeting Date: 30/09/2024
Cohort Participants: Akiva Levitt, Joseph Kaplan, Edgar Ngoepe, Phuti Mokgehle, Muhammad Bux, Syed Anas, Shen Reddy, Dylan Baker, Edward Ndlala and Lefa Mofokeng
Meeting Chair: Joseph Kaplan
Minute Taker: Akiva Levitt

Overview: Each group discussed the achievements and accomplishments of last week as well as the plans for the next week and the problems encountered. It seems all projects are making good progress and reaching the end of development, moving onto the investigation part of the project.

Group 11: Edgar Ngoepe and Phuti Mokgehle

- Progress this week: Integrated a CNN for hold detection and scoring based on shape and size.
  Optimized the pathfinding algorithm by incorporating the climber's reach, the distance between holds, and hold scores.
  Compared the performance of A* and Dijkstra algorithms for route optimization.Challenges:
- Challenges: The CNN accuracy is currently 68%. We plan to improve it by expanding the dataset.

- Plans for the coming week: Increase the dataset size and perform hyperparameter tuning to improve CNN accuracy. Finalize the integration of CNN-based hold scoring into both the A* and Dijkstra algorithms.Continue improving the stickman model after route optimization.

Group 19: Akiva Levitt and Joseph Kaplan

- last week: we recruited various students from computer science and engineering degrees from 1st to 4th year, to join in our experiment. We help an online experiment where we broke people up into breakout rooms and allowed each student to do the experiment at their own pace while being able to call for assistance. So far, we have had 14 students partake in our experiment. The results look promising. The average of the first test is around 66% and the average of the second test is 86%. So there has been an improvement after using our software.
- This week we will be trying to get more people to do our experiment and consolidating out results

Group 22: Muhammad Bux and Syed Anas

- Last week: We finished up on cleaning our dashboard so it looks neat and our dashboard is now ready to be shown on open day. We implemented a custom visualise technique where the user can define the x and y axis for the plot and what plot they would want to see the visualisation in. We also did research on the investigative element of the project. We found two investigative element.
- This week: The two investigative element that we researched are that Syed Anas(me) is working on implementing the different database designs (normalization, indexing strategies, and query optimization) and how it will impact the performance of data entry, updates, and data retrieval on our dashboard. My partner (Salman Bux) is working on comparing different plotting libraries (e.g., Plotly, Matplotlib, Bokeh) to evaluate which is most efficient or provides the best user experience for data visualization in our dashboard.
- Challenges faced: The challenge we faced was deciding which is the best investigative topic in our project. As we had to see which investigative topic will give us the best results and good enough results which we can write in our reports.

Group 41: Shen Reddy and Dylan Baker

- Last Week we were able to adapt our basic computer vision model to work for anomaly detection. We tested the model on a laptop using a dataset of manufactured screws and were able to get an accuracy of around 77%. We were also able to compress the new model on run it on our microcontroller and have it accurately identify anomalies in our dataset.
- This week we need to optimise our 3 algorithms to try increase our accuracy to the 80% range as well as ensure that they can work for multiple datasets. We will need to recompress the models again and run them on the microcontroller. We will also be taking energy usage measurements for our 2 systems.
- We have not encountered any major challenges in the past week.

Group 47: Edward Ndlala and Lefa Mofokeng

- 1. Took temperature distribution for solving one board.
- 2. We had a setback that forced us to review our algorithms and methodology. This is due to us encountering inconsistent results with the algorithms' memory records.

- 3. This week we will be testing and performing analysis of the captured data using our updated methodology and reviewed DLX

- Challenges faced: The challenge we faced last week was cleaning the data set that was provided to us by Dr. Broadie but we were able to be done with it in the weekend. Except for that we did not face any other challenges.

Group 41: Shen Reddy and Dylan Baker

- Last week we were able to successfully run all of our algorithms on both our PC and microcontroller systems. We then collected performance metrics for both systems. Our results (in the form of a confusion matrix) showed that we were able to achieve the same ML performance metrics on both systems.
- This week we need to finish collecting our energy measurements for both systems and finish printing our poster for the open day.

Group 47: Edward Ndlala and Lefa Mofokeng

Last week:

- we reviewed our algorithms and did some refactering
- We implemented web API which can extract information from the Smart pdu faster.

This week: ρ M δ

- analysing the results, we will look at creating our poster and preparing for presentation day.
- Challenges faced: the challenge we fast was that at first, we only logged at 1 minute. The web api can log at 1 second. This affected our testing results that we got. Thus we had to start testing afresh.

Week 8: Meeting Minutes

Meeting Date: 1[illegible]/10/2024
Cohort Participants: Akiva Levitt, Joseph Kaplan, Edgar Ngoepe, Phuti Mokgehle, Muhammad Bux, Sy

! ρ

- Last week: We developed the homicide media tracker and implemented all the required functionality. While implementing the functionality, we found bugs which we then fixed. We ironed out the dashboard and made it look professional for open day. Lastly, we designed and printed out our poster for open day.
- This week: We are now finalising the testing and the results we are getting from it. After that we will do the report for the project in the week.
- Challenges: Last week, the challenges we faced were fixing some of the bugs in our code as well as finalising the testing methodology as our metrics to test our investigative element in the project needs to be independent of the machine we are running the dashboard on, therefore some of our testing methodology was not compiling to that need.

Group 41: Shen Reddy and Dylan Baker

- Last week we created our poster and presented it at the open day. We also spent some time revising some of the energy measurements we had taken for our microcontroller system.
- This week we are going to take one more set of energy measurements to ensure the integrity of the results we captured last week. Both of us will continue to work on our individual reports.

Group 47: Edward Ndlala and Lefa Mofokeng

- Last week we took final Power measurements to ensure that they are consistent with previously taken measurements and to account for any false readings such as when the PC is in its sleep mode while running the algorithms. We then created our poster and presented it at the open day using the final results.
- This week we are going to perform a broader analysis of our results and discuss them.
- Both of us will continue to work on our individual reports.

Appendix D

Python Script to Build PostgreSQL Database

```
import psycopg2
import psycopg2.extras
from config import config

def copy_from_csv(conn, cursor, table_name, csv_file_path):
    #Open the csv file
    with open(r'C:\Data Generation\homicide_news_10000.csv', 'r', encoding='ISO-
8859-1') as file:
        #copy data from the csv file to the table
        cursor.execute('SET datestyle = "ISO, DMY";')
        cursor.copy_expert("""COPY homicide_news (news_report_url,
                    news_report_headline,
                    news_report_platform,
                    date_of_publication ,
                    author,
                    wire_service ,
                    no_of_subs,
                    victim_name ,
                    date_of_death,
                    race_of_victim,
                    age_of_victim ,
                    place_of_death_province ,
                    place_of_death_town,
                    type_of_location ,
                    sexual_assault ,
                    mode_of_death_specific ,
                    robbery_y_n_u,
                    perpetrator_name ,
                    perpetrator_relationship_to_victim ,
                    suspect_arrested ,
                    suspect_convicted,
                    multiple_murder,
                    intimate_femicide_y_n_u ,
                    extreme_violence_y_n_m_u ,
                    notes)
        FROM STDIN WITH CSV HEADER DELIMITER ';'
        """, file)
        print(f"Data copied successfully to homicide_news_data.")


def connect():
    connection = None
    csr = None
    try:
        params = config()
        print('Connecting to the PostgreSQL database ...')
        connection = psycopg2.connect(**params)
        csr = connection.cursor(cursor_factory=psycopg2.extras.DictCursor)

        csr.execute('CREATE EXTENSION IF NOT EXISTS "uuid-ossp";')
        # Drop the table if it already exists

        csr.execute("DROP TABLE IF EXISTS homicide_news CASCADE")

        create_script_homicide = '''CREATE TABLE homicide_news (
                        article_id SERIAL PRIMARY KEY,
                        news_report_url VARCHAR(255),
                        news_report_headline VARCHAR(255),
                        news_report_platform VARCHAR(255),
                        date_of_publication DATE,
                        author VARCHAR(255),
                        wire_service VARCHAR(255),
                        no_of_subs INT,
                        victim_name VARCHAR(255),
                        date_of_death DATE,
                        race_of_victim VARCHAR(255),
                        age_of_victim INT,
                        place_of_death_province VARCHAR(100),
                        place_of_death_town VARCHAR(255),
                        type_of_location VARCHAR(255),
                        sexual_assault VARCHAR(255),
                        mode_of_death_specific VARCHAR(100),
                        robbery_y_n_u VARCHAR(10),
                        perpetrator_name VARCHAR(255),
                        perpetrator_relationship_to_victim VARCHAR(255),
                        suspect_arrested VARCHAR(255),
                        suspect_convicted VARCHAR(255),
                        multiple_murder VARCHAR(10),
                        intimate_femicide_y_n_u VARCHAR(10),
                        extreme_violence_y_n_m_u VARCHAR(10),
                        notes VARCHAR(1000)
                        )'''
        csr.execute(create_script_homicide)
        print("homicide_news Table created successfully in public")
        copy_from_csv(connection, csr, 'homicide_news_data', r'C:\Data
Generation\homicide_news_10000.csv')


        connection.commit()
```

```python
    except(Exception, psycopg2.DatabaseError) as error:
        print(error)
    finally:
        if csr is not None:
            csr.close()
        if connection is not None:
            connection.close()
            print('Database connection terminated.')

if __name__ == "__main__":
    connect()
```

Appendix E

Python Code for the Application

```
import dash
from dash import dcc, html
from dash.dependencies import Input, Output, State
from dash import callback_context
import psycopg2
import dash_bootstrap_components as dbc
import pandas as pd
import plotly.express as px
from sqlalchemy import create_engine, text
import json
import io
import base64
from psycopg2 import pool
import traceback
import altair as alt
import time
from dash import dash_table


#import main
with open("za.json") as f:
    geojson_data = json.load(f)

# Database connection
conn = psycopg2.connect(
    host="localhost", port="5432", database="homicide_main",
    user="postgres", password="deadlocked6111234")

engine =
create_engine("postgresql://postgres:deadlocked611@localhost:5432/homicide_main")
# Define options
provinces = {
    'Western Cape': ['Cape Town', 'Stellenbosch', 'George'],
    'Eastern Cape': ['Port Elizabeth', 'East London', 'Grahamstown'],
    'Gauteng': ['Johannesburg', 'Pretoria', 'Soweto'],
    # Add more provinces and towns here
}

race_options = [
    {'label': 'African', 'value': 'African'},
    {'label': 'White', 'value': 'White'},
    {'label': 'Coloured', 'value': 'Coloured'},
    {'label': 'Indian/Asian', 'value': 'Indian/Asian'},
    {'label': 'Other', 'value': 'Other'}
```

```
]

relationship_options = [
    {'label': 'Family', 'value': 'Family'},
    {'label': 'Friend', 'value': 'Friend'},
    {'label': 'Acquaintance', 'value': 'Acquaintance'},
    {'label': 'Stranger', 'value': 'Stranger'},
    {'label': 'Other', 'value': 'Other'}
]

bool_options = [
    {'label': 'Yes', 'value': 'Y'},
    {'label': 'No', 'value': 'N'}
]

app = dash.Dash(__name__, external_stylesheets=[dbc.themes.BOOTSTRAP],
suppress_callback_exceptions=True)



# Navbar
navbar = dbc.NavbarSimple(
    brand="Homicide Media Tracker",
    color="primary",
    dark=True,
    children=[
        dbc.NavItem(dbc.NavLink("Data Entry", href="/")),
        dbc.NavItem(dbc.NavLink("Data Import", href="/import")),
        dbc.NavItem(dbc.NavLink("Data Display", href="/display")),
        dbc.NavItem(dbc.NavLink("Delete Data", href = "/delete_table")),
        dbc.NavItem(dbc.NavLink("Duplicate Data", href = "/duplicate_data")),
        dbc.NavItem(dbc.NavLink("Data Visualization", href="/visualization")),
        dbc.NavItem(dbc.NavLink("Custom Data Visualization",
href="/custom_visualization"))


    ]
)

# Footer
footer = html.Footer(
    "Homicide Media Tracker © 2024",
    style={'text-align': 'center', 'padding': '20px', 'background': '#f1f1f1'}
)

# Data Entry Layout
data_entry_layout = dbc.Container([
    dbc.Card([
        dbc.CardHeader("Homicide Data Entry"),
        dbc.CardBody([
            dbc.Row([dbc.Col([dbc.Label("News Report URL"), dbc.Input(id='url-
input', type='text', placeholder="Enter news report URL")], width=6)]),
            dbc.Row([dbc.Col([dbc.Label("News Outlet"), dbc.Input(id='outlet-
input', type='text', placeholder="Enter news outlet")], width=6),
                     dbc.Col([dbc.Label("Date of Publication"),
dbc.Input(id='publication-date-input', type='date')], width=6)], style={'margin-
bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Author"), dbc.Input(id='author-input',
type='text', placeholder="Enter author name")], width=6),
                     dbc.Col([dbc.Label("Headline"), dbc.Input(id='headline-input',
type='text', placeholder="Enter headline")], width=6)], style={'margin-bottom':
'15px'}),
            dbc.Row([dbc.Col([dbc.Label("Number of Subs"), dbc.Input(id='subs-
input', type='text', placeholder="Enter Number of Subs")], width=6),
                     dbc.Col([dbc.Label("Wire Service"), dbc.Input(id='wire-input',
type='text', placeholder="Enter Wire Service")], width=6)], style={'margin-bottom':
'15px'}),
            dbc.Row([dbc.Col([dbc.Label("Victim Name"), dbc.Input(id='victim-name-
input', type='text', placeholder="Enter victim name")], width=6),
                     dbc.Col([dbc.Label("Date of Death"), dbc.Input(id='death-date-
input', type='date')], width=6)], style={'margin-bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Age of Victim"), dbc.Input(id='victim-age-
input', type='number', placeholder="Enter victim age")], width=6),
                     dbc.Col([dbc.Label("Race of Victim"), dcc.Dropdown(id='race-
dropdown', options=race_options, placeholder="Select race")], width=6)],
style={'margin-bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Type of Location"),
dbc.Input(id='location-type-input', type='text', placeholder="Enter type of
location")], width=6),
                     dbc.Col([dbc.Label("Province"), dcc.Dropdown(id='province-
dropdown', options=[{'label': k, 'value': k} for k in provinces.keys()],
placeholder="Select a province")], width=6)], style={'margin-bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Town"), dcc.Dropdown(id='town-dropdown',
placeholder="Select a town")], width=6),
                     dbc.Col([dbc.Label("Sexual Assault"), dcc.Dropdown(id='sexual-
assault-dropdown', options=bool_options, placeholder="Select option")], width=6)],
style={'margin-bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Mode of Death"), dbc.Input(id='mode-of-
death-input', type='text', placeholder="Enter mode of death")], width=6),
                     dbc.Col([dbc.Label("Robbery"), dcc.Dropdown(id='robbery-
dropdown', options=bool_options, placeholder="Select option")], width=6)],
style={'margin-bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Suspect Arrested"),
dcc.Dropdown(id='suspect-arrested-dropdown', options=bool_options,
placeholder="Select option")], width=6),
                     dbc.Col([dbc.Label("Suspect Convicted"),
dcc.Dropdown(id='suspect-convicted-dropdown', options=bool_options,
placeholder="Select option")], width=6)], style={'margin-bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Perpetrator Name"), dbc.Input(id='perp-
name-input', type='text', placeholder="Enter perpetrator name")], width=6),
                     dbc.Col([dbc.Label("Perp Relationship"),
dcc.Dropdown(id='relationship-dropdown', options=relationship_options,
placeholder="Select relationship")], width=6)], style={'margin-bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Multiple Murders"),
dcc.Dropdown(id='multi-murder-dropdown', options=bool_options, placeholder="Select
option")], width=6),
                     dbc.Col([dbc.Label("Extreme Violence"),
dcc.Dropdown(id='extreme-violence-dropdown', options=bool_options,
placeholder="Select option")], width=6)], style={'margin-bottom': '15px'}),
            dbc.Row([dbc.Col([dbc.Label("Intimate Femicide"),
dcc.Dropdown(id='intimate-femicide-dropdown', options=bool_options,
placeholder="Select option")], width=6),
                     dbc.Col([dbc.Label("Notes"), dbc.Textarea(id='notes-input',
placeholder="Enter additional notes")], width=6)], style={'margin-bottom':
'15px'}),
            dbc.Button("Submit", id="submit-button", color="success",
className="mt-3"),
            html.Div(id="output-message", className="mt-3"),
            dbc.Button("Export to CSV", id="export-button", color="secondary",
className="mt-3"),
            dcc.Download(id="download-dataframe-csv"),
            html.Hr(),
        ]),
    ], className="mb-4")
])
#Data display layout
data_display_layout = dbc.Container([
    dbc.Card([
        dbc.CardHeader("Homicide Data Display"),
        dbc.CardBody([
            dbc.Row([
                dbc.Col([
                    dcc.Checklist(
                        id='column-checklist-1',
                        options=[{'label': col, 'value': col} for col in
['article_id', 'news_report_url', 'news_report_platform', 'date_of_publication',
'author', 'news_report_headline', 'wire_service', 'no_of_subs']],
                        value=['article_id', 'news_report_url',
'news_report_platform', 'date_of_publication', 'author', 'news_report_headline',
'wire_service', 'no_of_subs'],
                        labelStyle={'display': 'block'}
                    ),
                ], width=4),

                dbc.Col([
                    dcc.Checklist(
                        id='column-checklist-2',
                        options=[{'label': col, 'value': col} for col in
['victim_name', 'date_of_death', 'age_of_victim', 'race_of_victim',
'type_of_location', 'place_of_death_town', 'place_of_death_province']],
                        value=['victim_name', 'date_of_death', 'age_of_victim',
'race_of_victim', 'type_of_location', 'place_of_death_town',
'place_of_death_province'],
                        labelStyle={'display': 'block'}
                    ),
                ], width=4),

                dbc.Col([
                    dcc.Checklist(
                        id='column-checklist-3',
                        options=[{'label': col, 'value': col} for col in
['sexual_assault', 'mode_of_death_specific', 'robbery_y_n_u', 'suspect_arrested',
'suspect_convicted', 'perpetrator_name', 'perpetrator_relationship_to_victim',
'multiple_murder', 'extreme_violence_y_n_m_u', 'intimate_femicide_y_n_u',
'notes']],
                        value=['sexual_assault', 'mode_of_death_specific',
'robbery_y_n_u', 'suspect_arrested', 'suspect_convicted', 'perpetrator_name',
'perpetrator_relationship_to_victim', 'multiple_murder',
'extreme_violence_y_n_m_u', 'intimate_femicide_y_n_u', 'notes'],
                        labelStyle={'display': 'block'}
                    ),
                ], width=4)
            ], className="mb-3"),


            dbc.Button("Display Table", id = 'display-button', color = "success",
className = "mt-3" ),
            html.Div(id='table-container',  className="mt-3")
        ]),
    ], className="mb-4")
])
app.layout = dbc.Container([
    data_display_layout
])
```

```
data_import_layout = dbc.Container([
    dbc.Card([
        dbc.CardHeader("Homicide Data Import"),
        dbc.CardBody([
            html.H3("Upload CSV to Import Data in to the Current Table"),
            dcc.Upload(id='upload-data-1', children=html.Div(['Drag and Drop or ',
html.A('Select a CSV File')]), style={'width': '100%', 'height': '60px',
'lineHeight': '60px', 'borderWidth': '1px', 'borderStyle': 'dashed',
'borderRadius': '5px', 'textAlign': 'center', 'margin': '10px'}, multiple=False,
accept=".csv"),
            html.Div(id='upload-output-1'),
            html.Hr(),
            html.H3("Upload CSV to Import Data in to a New Table"),
            dcc.Upload(id='upload-data-2', children=html.Div(['Drag and Drop or ',
html.A('Select a CSV File')]), style={'width': '100%', 'height': '60px',
'lineHeight': '60px', 'borderWidth': '1px', 'borderStyle': 'dashed',
'borderRadius': '5px', 'textAlign': 'center', 'margin': '10px'}, multiple=False,
accept=".csv"),
            html.Div(id='upload-output-2')
        ])
    ])

])
app.layout = dbc.Container([
    data_import_layout
])
# Data Visualization Layout
data_visualization_layout = dbc.Container([
    dbc.Card([
        dbc.CardHeader("Homicide Data Visualization"),
        dbc.CardBody([
            dbc.Row([
                dbc.Col([
                    dbc.Label("Select Plot Category"),
                    dcc.Dropdown(
                        id='plot-category-dropdown',
                        options=[{'label': 'Homicides Over Time', 'value':
'homicides_over_time'}, {'label': 'Geographical Distribution', 'value':
'geographical_distribution'}, {'label': 'Demographic Insights', 'value':
'demographic_insights'}, {'label': 'Victim Perpetrator Relationship', 'value':
'victim_perpetrator_relationship'}, {'label' : 'Multivariate Comparisons', 'value'
: 'multivariate_comparisons'}],
                        placeholder="Select a plot category"
                    )
                ], width=6),
                dbc.Col([
                    dbc.Label("Select Plot Type"),
                    dcc.Dropdown(
                        id='plot-type-dropdown',
                        options=[],
                        placeholder="Select a plot type"
                    )
                ], width=6),
            ]),
            html.Div(id='plot-container', style={'textAlign': 'center', 'margin':
'20px'})
        ]),
    ], className="mb-4")
])

# Layout for Custom Visualizations (with just bar graph for now)
custom_visualization_layout = dbc.Container([
    dbc.Card([
        dbc.CardHeader("Custom Data Visualization"),
        dbc.CardBody([
            dbc.Row([
                dbc.Col([
                    dbc.Label("Select X-Axis"),
                    dcc.Dropdown(
                        id='x-axis-dropdown',
                        options=[{'label': 'Victim Age', 'value': 'age'},
                                 {'label': 'Province', 'value': 'province'},
                                 {'label': 'Race', 'value': 'race'},
                                 {'label': 'Perpetrator Relationship', 'value':
'VIC SUSP RELATIONSHIP'}],
                        placeholder="Select X-Axis"
                    )
                ], width=6),
                dbc.Col([
                    dbc.Label("Select Y-Axis"),
                    dcc.Dropdown(
                        id='y-axis-dropdown',
                        options=[{'label': 'Count', 'value': 'count'}],  # For a
bar graph, we only allow 'count' for now.
                        placeholder="Select Y-Axis",
                        value='count'  # Default to 'count'
                    )
                ], width=6),
            ]),
            dbc.Row([
                dbc.Col([
                    dbc.Button("Generate Bar Graph", id='generate-bar-graph-
button', color="primary", className="mt-3")
```

```
                ], width=12)
            ]),
            dcc.Graph(id='custom-bar-graph', style={'textAlign': 'center', 
'margin': '20px'})
        ]),
    ], className="mb-4")
])

duplicates_table_layout = dbc.Container([
    dbc.Card([
        dbc.CardHeader("Duplicate Data"),
        dbc.CardBody([
            dbc.Row([
                dbc.Col([
                    dcc.Input(
                        id='duplicate-column-input',
                        type='text',
                        placeholder='Enter column names separated by commas',
                        style={'width': '100%'}
                    ),
                ], width=6),
                dbc.Col([
                    dbc.Button("Check for Duplicates", id='check-duplicates-
button', n_clicks=0, color="warning", className="me-2"),
                    dbc.Button("Delete duplicates", id='delete-duplicates-button', 
n_clicks=0, color="danger"),
                ], width=6, className="d-flex justify-content-end")
            ], className="mb-3"),

            html.Div(id='duplicates-message', className="mb-3")
            ]),
        dbc.Col([
            dbc.Button("Display Duplicate Deleted Data", id = 'display-duplicate-
button', color = "success", className="mt-3"),
            html.Div(id='duplicate-table-container', className = "mt-3")
        ]),
        ],className = "mb-4"),
    ])
app.layout = dbc.Container([
    duplicates_table_layout
])

delete_table_layout = dbc.Container([
    dbc.Card([
        dbc.CardHeader("Delete Data"),
        dbc.CardBody([
            dbc.Row([
```

```python
    elif pathname == '/duplicate_data':
        return duplicates_table_layout
    else:
        return data_entry_layout


# Province-Town Callback
@app.callback(
    Output('town-dropdown', 'options'),
    Input('province-dropdown', 'value')
)
def update_town_dropdown(province_value):
    if province_value:
        return [{'label': town, 'value': town} for town in
provinces[province_value]]
    return []

# Handle Data Submission
@app.callback(
    Output('output-message', 'children'),
    Input('submit-button', 'n_clicks'),
    State('url-input', 'value'),
    State('outlet-input', 'value'),
    State('publication-date-input', 'value'),
    State('author-input', 'value'),
    State('headline-input', 'value'),
    State('subs-input', 'value'),
    State('wire-input', 'value'),
    State('victim-name-input', 'value'),
    State('death-date-input', 'value'),
    State('victim-age-input', 'value'),
    State('race-dropdown', 'value'),
    State('location-type-input', 'value'),
    State('province-dropdown', 'value'),
    State('town-dropdown', 'value'),
    State('sexual-assault-dropdown', 'value'),
    State('mode-of-death-input', 'value'),
    State('robbery-dropdown', 'value'),
    State('suspect-arrested-dropdown', 'value'),
    State('suspect-convicted-dropdown', 'value'),
    State('perp-name-input', 'value'),
    State('relationship-dropdown', 'value'),
    State('multi-murder-dropdown', 'value'),
    State('extreme-violence-dropdown', 'value'),
    State('intimate-femicide-dropdown', 'value'),
    State('notes-input', 'value')
)
```

```
def submit_form(n_clicks, url, outlet, pub_date, author, headline, subs, wire,
victim_name, death_date,
                victim_age, race, location_type, town, province, sexual_assault,
mode_of_death,
                robbery, suspect_arrested, suspect_convicted, perp_name,
relationship,
                multi_murder, extreme_violence, femicide, notes):
    if n_clicks is None:
        return ""

    cur = conn.cursor()

        # Prepare the SQL insert statement
    insert_query = '''INSERT INTO homicide_news
            (news_report_url, news_report_platform, date_of_publication, author,
news_report_headline, no_of_subs,
            wire_service, victim_name, date_of_death, age_of_victim,
race_of_victim, type_of_location,
            place_of_death_town, place_of_death_province, sexual_assault,
mode_of_death_specific, robbery_y_n_u,
            suspect_arrested, suspect_convicted, perpetrator_name,
perpetrator_relationship_to_victim,
            multiple_murder, extreme_violence_y_n_m_u, intimate_femicide_y_n_u,
notes)
            VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s)'''


    values = (url, outlet, pub_date, author, headline, subs, wire, victim_name,
death_date,
            victim_age, race, location_type, province, town, sexual_assault,
mode_of_death,
            robbery, suspect_arrested, suspect_convicted, perp_name, relationship,
            multi_murder, extreme_violence, femicide, notes)


        # Execute the insertion
    cur.execute(insert_query, values)
    conn.commit()  # Ensure the transaction is committed
    cur.close()  # Close the cursor
    conn.close()

    return "Data successfully inserted!"

# Handle CSV Export
@app.callback(
    Output("download-dataframe-csv", "data"),
    Input("export-button", "n_clicks"),
    prevent_initial_call=True
)
def export_csv(n_clicks):
    if n_clicks:
        with psycopg2.connect(
            host="localhost", port="5432", database="homicide_main",
            user="postgres", password="deadlocked611"
        ) as conn:
            query = "SELECT * FROM homicide_news"
            df = pd.read_sql(query, conn)
            return dcc.send_data_frame(df.to_csv, "homicide_news.csv")



# Handle CSV Upload
@app.callback(
    Output('upload-output-1', 'children'),
    Input('upload-data-1', 'contents'),
    prevent_initial_call=True
)
def upload_csv(contents):
    if contents:
        # Split the contents into metadata and base64-encoded data
        content_type, content_string = contents.split(',')

        # Decode the base64-encoded string
        decoded = base64.b64decode(content_string)

        try:
            # Specify semicolon as the delimiter and handle bad lines
            df = pd.read_csv(io.StringIO(decoded.decode('utf-8')), sep=';',
on_bad_lines='skip')
            print(df.head())  # Print first few rows for debugging

            # Try appending the data to the database
            df.to_sql('homicide_news', engine, if_exists='append', index=False)
            return "CSV data appended successfully."

        except pd.errors.ParserError as e:
            return f"Parsing error: {e}"

        except UnicodeDecodeError as e:
            return f"Decoding error: {e}"

        except Exception as e:
            return f"An error occurred: {e}"
    return "No contents provided."


@app.callback(
    Output('upload-output-2', 'children'),
    Input('upload-data-2', 'contents'),
    prevent_initial_call= True
)
def upload_csv_to_new_table(contents):
    if contents:
        content_type, content_string = contents.split(',')
        decoded = base64.b64decode(content_string)
        try:
            df = pd.read_csv(io.StringIO(decoded.decode('utf-8')), sep = ';',
on_bad_lines='skip')
            df.to_sql('homicide_complete', engine, if_exists='append', index =
False)
            return "CSV data appended to a table homicide_complete successfully."
        except pd.errors.ParserError as e:
            return f"Parsing error: {e}"
        except UnicodeDecodeError as e:
            return f"An error occured: {e}"

        except Exception as e:
            return f"An error occurred: {e}"
    return "No contents provided"

@app.callback(
    Output('table-container', 'children'),
    [Input('display-button', 'n_clicks'),
     Input('column-checklist-1', 'value'),
     Input('column-checklist-2', 'value'),
     Input('column-checklist-3', 'value')]
)

def display_selected_columns(n_clicks, selected_columns_1, selected_columns_2,
selected_columns_3):
    if n_clicks is None or n_clicks == 0:
        return "Please click the 'Display Table' button to show data", None
    selected_columns = selected_columns_1 + selected_columns_2 + selected_columns_3

    if not selected_columns:
        return "No columns selected. Please select at least one column", None

    try:
        with psycopg2.connect(
            host="localhost", port="5432", database="homicide_main",
            user="postgres", password="deadlocked611"
        ) as conn:
            query = f"SELECT {', '.join(selected_columns)} FROM homicide_news"
            df = pd.read_sql_query(query, conn)

        if df.empty:
            return "No data found for the selected columns.", None

        table = dash_table.DataTable(
            columns=[{"name": col, "id": col} for col in df.columns],
            data=df.to_dict('records'),
            page_size=50,  # Show 20 rows per page
            style_table={'overflowX': 'auto'},
            style_cell={'textAlign': 'left'}
        )

        return "", table
    except Exception as e:
        return f"Error:{str(e)}",None

#This is the duplicates tab, it will do all the procesisng for the duplicates data

@app.callback(
    [Output('duplicates-message', 'children'),
     Output('duplicate-table-container', 'children')],
    [Input('check-duplicates-button', 'n_clicks'),
     Input('delete-duplicates-button', 'n_clicks'),
     Input('display-duplicate-button', 'n_clicks')],
    [State('duplicate-column-input', 'value')]
 )
def update_duplicates_display(check_clicks, delete_duplicates_clicks,
display_duplicates_clicks, duplicate_columns):
    ctx = dash.callback_context
    if not ctx.triggered:
        print("No input was triggered.")
        return dash.no_update, dash.no_update

    triggered_id = ctx.triggered[0]['prop_id'].split('.')[0]
    print(f"Triggered by: {triggered_id}")

    # selected_columns = selected_columns_1 + selected_columns_2 +
selected_columns_3

    if triggered_id == 'check-duplicates-button':
        message = check_duplicates(check_clicks, duplicate_columns)
        return message, dash.no_update
    elif triggered_id == 'delete-duplicates-button':
        message, table = delete_duplicates(delete_duplicates_clicks,
duplicate_columns)
        return message, table
    # elif triggered_id == 'delete-record-button':
    #     print(f"Attempting to delete record with ID: {delete_record_id}")
    #     message, table = delete_record(delete_clicks_data, delete_record_id,
selected_columns)
    #     print(f"Delete operation result: {message}")
    #     return message, table
    elif triggered_id == 'display-duplicate-button':
        message, table = display_duplicates_table(display_duplicates_clicks)
        print(f"display duplicate columns returned: message='{message}',
table={'not None' if table is not None else 'None'}")
        return message, table

    print("No condition was met.")
    return dash.no_update, dash.no_update

def check_duplicates(n_clicks, columns):
    if n_clicks is None or n_clicks == 0:
        return ""

    if not columns:
        return "Please enter one or more columns to check for duplicates."

    column_list = [col.strip() for col in columns.split(',')]

    try:

        with psycopg2.connect(
            host="localhost", port="5432", database="homicide_main",
            user="postgres", password="deadlocked611"
        ) as conn:
            query = f"""
                SELECT {', '.join(column_list)}, COUNT(*)
                FROM homicide_news
                GROUP BY {', '.join(column_list)}
                HAVING COUNT(*) > 1
            """
            df = pd.read_sql(query, conn)
        print("Hello there_check dup")
        if df.empty:
            return "No duplicate records found based on the selected columns."
        else:
            return dbc.Table.from_dataframe(df, striped=True, bordered=True,
hover=True)
```

```python
    except Exception as e:
        return f"An error ocurred : {str(e)}"

def delete_duplicates(n_clicks, column_name):
    if n_clicks == 0 or not column_name:
        return '', dash.no_update

    try:
        with psycopg2.connect(
            host="localhost", port="5432", database="homicide_main",
            user="postgres", password=
                        FROM homicide_news
                        GROUP BY {column_name}
                        HAVING COUNT(*) > 1
                    ) as subquery
                """)
                duplicate_count = cursor.fetchone()[0]

                # Delete duplicates from the main table
                cursor.execute(f"""
                    DELETE FROM homicide_news
                    WHERE ctid NOT IN (
                        SELECT MIN(ctid)
                        FROM homicide_news
                        GROUP BY {column_name}
                    )
                """)


                # Fetch the cleaned data
                # cursor.execute('''SELECT * FROM homicide_news_v1''')
                # cleaned_data = cursor.fetchall()

                conn.commit()

            # Create DataFrame from cleaned data
            # df_cleaned = pd.DataFrame(cleaned_data)
            # table = dbc.Table.from_dataframe(df_cleaned, striped=True,
bordered=True, hover=True)

            return f"{duplicate_count} duplicate groups found. Duplicates removed
from main table and saved to 'duplicates' table.", table
    except Exception as e:
        print(f"Error in delete_duplicates: {str(e)}")  # Log the error
        return f"An error occurred: {str(e)}", dash.no_update

def display_duplicates_table(n_clicks):
    if n_clicks is None or n_clicks == 0:
        return "Please click the 'Display Duplicate Table' button to show data"

    try:
        with psycopg2.connect(
            host="localhost", port="5432", database="homicide_main",
            user="postgres", password="deadlocked611"
        ) as conn:
            query = '''SELECT * FROM duplicates'''
            print(f"Executing query: {query}")
            df = pd.read_sql_query(query, conn)
        print(f"Query executed successfully. Dataframe shape: {df.shape}")
        if df.empty:
            print("The resulting dataframe is empty.")
            return "No data found.",None

        table = dash_table.DataTable(
            columns=[{"name": col, "id": col} for col in df.columns],
            data=df.to_dict('records'),
            page_size=50,  # Show 20 rows per page
            style_table={'overflowX': 'auto'},
            style_cell={'textAlign': 'left'}
        )


        #table = dbc.Table.from_dataframe(df, striped=True, bordered=True,
hover=True)
        print("Table created successfully.")
        return "", table  # Return only the table, no message needed
    except Exception as e:
        error_msg = f"Error in display_duplicates_table:
{str(e)}\n{traceback.format_exc()}"
        print(error_msg)
        return f"An error occurred: {error_msg}", None

##################################################################################
#####################################
#This is the delete tab, it has all the delete stuff in it
@app.callback(
    Output('delete-table-container', 'children'),
    [Input('delete-record-button', 'n_clicks'),
     Input('display-delete-button', 'n_clicks')],
    [State('delete-record-input', 'value')]
)
def update_delete_data_display(delete_clicks_data, display_delete_clicks,
delete_record_id):
    ctx = dash.callback_context
    if not ctx.triggered:
        print("No input was triggered.")
        return dash.no_update  # No update if no interaction

    triggered_id = ctx.triggered[0]['prop_id'].split('.')[0]
    print(f"Triggered by: {triggered_id}")

    if triggered_id == 'delete-record-button':
        # Handle deletion of the record
        print(f"Attempting to delete record with ID: {delete_record_id}")
        message, table = delete_record(delete_clicks_data, delete_record_id)
        print(f"Delete operation result: {message}")
        return [html.Div(message), table] if table else [html.Div(message)]

    elif triggered_id == 'display-delete-button':
        # Handle displaying the deleted records
        table = display_delete_table(display_delete_clicks)
        if table:
            return [table]
        return [html.Div("No data found.")]

    print("No condition was met.")
    return dash.no_update

def delete_record(n_clicks, article_id):
    if not article_id or n_clicks == 0:
        return '', dash.no_update

    if not article_id:
        return "Please enter an article_id to delete the data entry.",
dash.no_update

    try:
        # Convert article_id to integer
        article_id = int(article_id)

        with psycopg2.connect(
            host="localhost", port="5432", database="homicide_main",
            user="postgres", password="deadlocked611"
        ) as conn:
            with conn.cursor() as cursor:
                #Fetch the record to be deleted
                cursor.execute("SELECT * FROM homicide_news WHERE article_id = %s",
(article_id,))
                record = cursor.fetchone()
                print(record)

                if record is None:
                    return f"No record found with the article_id {article_id}.",
dash.no_update


                cursor.execute("""
                    CREATE TABLE IF NOT EXISTS delete_dash (
                        LIKE homicide_news INCLUDING ALL
                    )
                """)
                # Insert the fetched record into the delete table
                cursor.execute("""
                    INSERT INTO delete_dash (article_id, news_report_url,
news_report_headline, news_report_platform,
                    date_of_publication, author,wire_service, no_of_subs,
victim_name, date_of_death,race_of_victim,
                    age_of_victim,place_of_death_province, place_of_death_town,
type_of_location, sexual_assault,
                    mode_of_death_specific, robbery_y_n_u,perpetrator_name,
perpetrator_relationship_to_victim,
                    suspect_arrested, suspect_convicted, multiple_murder,
intimate_femicide_y_n_u, extreme_violence_y_n_m_u, notes)
                    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s,
%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
                """, record)
                #Insert the fetched record in to the delete table
                cursor.execute("DELETE FROM homicide_news WHERE article_id = %s",
(article_id,))
                delete_count = cursor.rowcount

            # if delete_count == 0:
            #     return f"No record found with article_id {article_id}.",
dash.no_update

            # Fetch the updated data
            query = f"SELECT * FROM homicide_news"
            df = pd.read_sql_query(query, conn)

            conn.commit()

        table = dbc.Table.from_dataframe(df, striped=True, bordered=True,
hover=True)
        return f"Record with article_id {article_id} has been deleted.", table

    except ValueError:
        return "Invalid article_id. Please enter a valid integer.", dash.no_update
    except Exception as e:
        print(f"Error in delete_record: {str(e)}")
        return f"An error occurred: {str(e)}", dash.no_update

def display_delete_table(n_clicks):
    if n_clicks is None or n_clicks == 0:
        return html.Div("Please click the 'Display Duplicates Table' button to show
data")

    try:
        with psycopg2.connect(
            host="localhost", port="5432", database="homicide_main",
            user="postgres", password="deadlocked611"
        ) as conn:
            query = '''SELECT * FROM delete_dash'''
            print(f"Executing query: {query}")
            df = pd.read_sql_query(query, conn)

        print(f"Query executed successfully. Dataframe shape: {df.shape}")
        if df.empty:
            print("The resulting dataframe is empty.")
            return html.Div("No data found.")

        table = dbc.Table.from_dataframe(df, striped=True, bordered=True,
hover=True)
        # #table = dash_table.DataTable(
        #     columns=[{"name": col, "id": col} for col in df.columns],
        #     data=df.to_dict('records'),
        #     page_size=50,  # Show 20 rows per page
        #     style_table={'overflowX': 'auto'},
        #     style_cell={'textAlign': 'left'}
        # )

        print("Table created successfully.")
        return table  # Return only the table, no message needed
    except Exception as e:
        error_msg = f"Error in duplicates table:
{str(e)}\n{traceback.format_exc()}"
        print(error_msg)
        return html.Div(f"An error occurred: {error_msg}")


@app.callback(
    Output('plot-type-dropdown', 'options'),
    Input('plot-category-dropdown', 'value'),
)
def update_plot_type_dropdown(category_value):
    if category_value == 'homicides_over_time':
        return [{'label': 'Line Plot', 'value': 'line_plot'}, {'label': 'Bar
Chart', 'value': 'bar_chart'}]
    elif category_value == 'geographical_distribution':
        return [{'label': 'Choropleth Map', 'value': 'choropleth_map'}, {'label':
'Heat Map', 'value': 'heat_map'}]
    elif category_value == 'demographic_insights':
        return [
            {'label': 'Bar Chart (Race Breakdown)', 'value': 'race_bar_chart'},
            {'label': 'Age Distribution Histogram', 'value': 'age_histogram'},
            {'label': 'Gender Comparison Plot', 'value': 'gender_comparison'}
        ]
    elif category_value == 'victim_perpetrator_relationship':
        return [{'label': 'Relationship Bar Chart', 'value':
'relationship_bar_chart'}, {'label': 'Heatmap', 'value': 'relationship_heatmap'}]
    elif category_value == 'multivariate_comparisons':
        return [{'label': 'Scatter Plot', 'value': 'scatter_plot'}, {'label':
'Bubble Plot', 'value': 'bubble_plot'}]
    return []

# Render Plot Based on Selected Category and Plot Type
@app.callback(
    Output('plot-container', 'children'),
    Input('plot-category-dropdown', 'value'),
    Input('plot-type-dropdown', 'value'),
)
def render_plot(category_value, plot_type_value):
    # start_time = time.time()  # Start timing

    #fig = None  # Initialize fig to None

    if not category_value or not plot_type_value:
        return "Please select a plot type."
    try:
        fig = None
    # Homicides Over Time
        if category_value == 'homicides_over_time':
            query = """
                SELECT "VICTIM NAME", YEAR
                FROM open_day_homicide_data
                GROUP BY "VICTIM NAME", YEAR
            """
            df = pd.read_sql(query, conn)
            #df['date_of_death'] = pd.to_datetime(df['date_of_death'])
            #df['year_of_death'] = df['date_of_death'].dt.year
            data = df.groupby('year').size().reset_index(name='count')

            if plot_type_value == 'line_plot':
                fig = px.line(data, x='year', y='count', title='Homicides Over
Time')
            elif plot_type_value == 'bar_chart':
                fig = px.bar(data, x='year', y='count', title='Homicides Over
Time')

# Geographical Distribution
        elif category_value == 'geographical_distribution':
            query = """
```

```
                SELECT province, COUNT(DISTINCT "VICTIM NAME" || ' ' ||
MONTH::text) as count
                FROM open_day_homicide_data
                GROUP BY province
            """
            df = pd.read_sql(query, conn)

            # if plot_type_value == 'choropleth_map':
            #     fig = px.choropleth(df,
            #                         geojson=geojson_data,
            #                         locations='province',  # Use lowercase
'province'
            #                         featureidkey="properties.name",
            #                         color='count',
            #                         color_continuous_scale="Reds",
            #                         title='Homicides by Province',
            #                         scope='africa')

            #     fig.update_geos(fitbounds="locations")  # Focus on South Africa

            if plot_type_value == 'choropleth_map':
                fig = px.choropleth(df,
                        geojson=geojson_data,
                        locations='province',  # Use lowercase 'province'
                        featureidkey="properties.name",
                        color='count',
                        color_continuous_scale="Reds",
                        title='Homicides by Province',
                        scope='africa')

    # Focus on South Africa
                fig.update_geos(fitbounds="locations", visible = False)

    # Update layout: background color, centered title, and margins
                fig.update_layout(
                    paper_bgcolor="white",  # Change background around the map to
blue
                    title={
                        'text': 'Homicides by Province',
                        'x': 0.5,  # Center the title
                        'xanchor': 'center',
                        'yanchor': 'top'
                    },
                    margin={"r": 0, "t": 50, "l": 0, "b": 0}  # Adjust margins for
spacing
                )
```

```
        # Optional: Adjust the colorbar (count) position if needed
        # fig.update_coloraxes(colorbar=dict(
        #    orientation='v',  # Set colorbar to horizontal
        #    y=-0.5,  # Position it below the map (adjust 'y' as
needed)
        #    x=0.5,  # Center the colorbar horizontally
        #    xanchor='center',
        #    yanchor='bottom',
  #))


    # Demographic Insights
    elif category_value == 'demographic_insights':
      if plot_type_value == 'race_bar_chart':
        query = """
          SELECT race, COUNT(DISTINCT "VICTIM NAME" || ' ' ||
MONTH::text) as count
          FROM open_day_homicide_data
          GROUP BY race
        """
        df = pd.read_sql(query, conn)
        fig = px.bar(df, x='race', y='count', title='Race Breakdown of
Victims', color_discrete_sequence=['red'])

      elif plot_type_value == 'age_histogram':
        query = """
          SELECT DISTINCT ON ("VICTIM NAME", MONTH) age
          FROM open_day_homicide_data
          WHERE age != -1
        """
        df = pd.read_sql(query, conn)

        if df.empty:
          return "No valid age data available."
```

df.emp6ET EMC /P MCID 562>> BDC q0.0000088 0 595.32 841.92 re* n92 re340.0824 0.0824 rg0.639 0.0

```
                fig = px.bar(df, x="SUSPECT GENDER", y='count', title='Gender
Comparison of Perpetrators')

        # Category: Victim-Perpetrator Relationship
        elif category_value == 'victim_perpetrator_relationship':
            if plot_type_value == 'relationship_bar_chart':
                query = """
                    SELECT "VIC SUSP RELATIONSHIP", COUNT(DISTINCT "VICTIM NAME" ||
' ' || MONTH::text) as count
                    FROM open_day_homicide_data
                    WHERE "VIC SUSP RELATIONSHIP"IS NOT NULL
                    GROUP BY "VIC SUSP RELATIONSHIP"
                """
                df = pd.read_sql(query, conn)

                fig = px.bar(df,
                            x="VIC SUSP RELATIONSHIP",
                            y='count',
                            title='Homicides by Victim-Perpetrator Relationship')

            elif plot_type_value == 'relationship_heatmap':
                query = """
                    SELECT "VIC SUSP RELATIONSHIP", "MODE OF DEATH", COUNT(DISTINCT
"VICTIM NAME" || ' ' || MONTH::text) as count
                    FROM open_day_homicide_data
                    WHERE "VIC SUSP RELATIONSHIP" IS NOT NULL AND "MODE OF DEATH"
IS NOT NULL
                    GROUP BY "VIC SUSP RELATIONSHIP", "MODE OF DEATH"
                """
                df = pd.read_sql(query, conn)

                fig = px.density_heatmap(df,
                                        x="VIC SUSP RELATIONSHIP",
                                        y="MODE OF DEATH",
                                        z='count',
                                        title='Relationship vs Mode of Death
Heatmap')

        elif category_value == 'multivariate_comparisons':
            if plot_type_value == 'scatter_plot':
                query = """
                    SELECT "LOCATION (HOME/PUBLIC/WORK/UNKNOWN)", COUNT(DISTINCT
"VICTIM NAME"|| ' ' || MONTH::text) as homicide_count
                    FROM open_day_homicide_data
                    WHERE "LOCATION (HOME/PUBLIC/WORK/UNKNOWN)" IS NOT NULL
                    GROUP BY "LOCATION (HOME/PUBLIC/WORK/UNKNOWN)"
                """
                df = pd.read_sql(query, conn)

                # Scatter plot of location type vs homicide count
                fig = px.scatter(
                    df,
                    x="LOCATION (HOME/PUBLIC/WORK/UNKNOWN)",
                    y='homicide_count',
                    title='Location Type vs Homicide Count',
                    labels={
                        "LOCATION (HOME/PUBLIC/WORK/UNKNOWN)": 'Location Type',
                        'homicide_count': 'Homicide Count'
                    },
                    size='homicide_count',  # Size of points by homicide count
                    color='homicide_count'  # Color points by homicide count
                )
                fig.update_traces(marker_size=10)

            elif plot_type_value == 'bubble_plot':
                query = """
                    SELECT "MODE OF DEATH", "SUSPECT CONVICTED", COUNT(DISTINCT
"VICTIM NAME" || ' ' || MONTH::text) as count
                    FROM open_day_homicide_data
                    WHERE "MODE OF DEATH" IS NOT NULL AND "SUSPECT CONVICTED" IS
NOT NULL
                    GROUP BY "MODE OF DEATH", "SUSPECT CONVICTED"
                """
                df = pd.read_sql(query, conn)

                fig = px.scatter(df,
                                x="MODE OF DEATH",
                                y="SUSPECT CONVICTED",
                                size='count',
                                color= "SUSPECT CONVICTED",
                                title='Mode of Death vs Conviction Rates with
Frequency Bubble Size')

    # if fig:
    #     elapsed_time = time.time() - start_time  # Calculate elapsed time
    #     return dcc.Graph(figure=fig), f"Plot rendered in {elapsed_time:.2f}
seconds."  # Return time taken
        if fig:
            return dcc.Graph(figure=fig)
        else:
            return html.Div("Unable to create plot. Please try a different
selection.")
    except Exception as e:
        return html.Div(f"An erro occured: {str(e)}")
        #return "Please select a plot type."

@app.callback(
    Output('custom-bar-graph', 'figure'),
    [Input('generate-bar-graph-button', 'n_clicks')],
    [State('x-axis-dropdown', 'value'), State('y-axis-dropdown', 'value')]
)


def update_custom_bar_graph(n_clicks, x_axis, y_axis):
    if n_clicks is None or x_axis is None:
        # If no button click or no x-axis selected, return an empty figure
        return {}

    # Quote the x_axis for the SQL query if necessary
    if ' ' in x_axis or '-' in x_axis or x_axis.isupper():
        query_x_axis = f'"{x_axis}"'
    else:
        query_x_axis = x_axis

    # Construct the query to count unique murders, grouping by x_axis
    query = f"""
    SELECT {query_x_axis}, COUNT(DISTINCT "VICTIM NAME") as count
    FROM open_day_homicide_data
    GROUP BY {query_x_axis}
    """

    # Fetch data from the database
    try:
        df = pd.read_sql(query, con=engine)
        print(df)  # For debugging: print the data frame to check if it contains
data
    except Exception as e:
        print(f"Error in executing query: {e}")
        return {}

    if df.empty:
        print(f"No data returned for query: {query}")
        return {}  # Return an empty figure if the query returned no data

    # Ensure the column name for Plotly matches the DataFrame column
    x_axis_label = x_axis.strip('"')

    # Create a bar graph
    fig = px.bar(df, x=x_axis_label, y='count', title=f'Bar Graph of {x_axis_label}
vs {y_axis}')
    fig.update_layout(xaxis_title=x_axis_label, yaxis_title='Count')
    return fig



if __name__ == '__main__':
    app.run_server(debug=True)
```

Appendix E

Python code for Random Dataset Generation

```
import csv
import random
import uuid
from faker import Faker
from datetime import timedelta, datetime

# Initialize Faker
fake = Faker()
Faker.seed(0)

# Lists for generating random data
provinces = ['Gauteng', 'Western Cape', 'Eastern Cape', 'KwaZulu-Natal',
'Mpumalanga', 'Limpopo', 'Free State', 'Northern Cape', 'North West']
towns = ['Johannesburg', 'Cape Town', 'Durban', 'Pretoria', 'Port Elizabeth',
'Bloemfontein', 'Kimberley', 'Polokwane', 'Nelspruit']
races = ['Black', 'White', 'Coloured', 'Indian/Asian']
relationships = ['Spouse', 'Friend', 'Family', 'Stranger', 'Neighbor', 'Unknown']
platforms = ['News24', 'IOL', 'TimesLive', 'Daily Maverick', 'eNCA']
modes_of_death = ['Shooting', 'Stabbing', 'Strangulation', 'Blunt force trauma',
'Poisoning', 'Burning']
yes_no_unknown = ['Yes', 'No', 'Unknown']
locations = ['Urban', 'Rural', 'Township', 'Farm', 'Informal Settlement']

# Generate random date within the last 20 years
def random_date():
    start_date = datetime.now() - timedelta(days=365 * 20)
    random_days = random.randint(0, 365 * 20)
    return start_date + timedelta(days=random_days)

# Create CSV file with 10,000 entries
def generate_csv(file_name, num_entries):
    with open(file_name, mode='w', newline='', encoding='utf-8') as file:
        writer = csv.writer(file, delimiter=';')
        # Write header
        writer.writerow([
            'news_report_url', 'news_report_headline', 'news_report_platform',
'date_of_publication',
            'author', 'wire_service', 'no_of_subs', 'victim_name', 'date_of_death',
'race_of_victim',
            'age_of_victim', 'place_of_death_province', 'place_of_death_town',
'type_of_location',
            'sexual_assault', 'mode_of_death_specific', 'robbery_y_n_u',
'perpetrator_name',
            'perpetrator_relationship_to_victim', 'suspect_arrested',
'suspect_convicted', 'multiple_murder',
            'intimate_femicide_y_n_u', 'extreme_violence_y_n_m_u', 'notes'
        ])

        # Generate random data rows
        for _ in range(num_entries):
            writer.writerow([
                fake.url(),  # news_report_url
                fake.sentence(nb_words=6),  # news_report_headline
                random.choice(platforms),  # news_report_platform
                random_date().strftime('%Y-%m-%d'),  # date_of_publication
                fake.name(),  # author
                random.choice(yes_no_unknown),  # wire_service
                random.randint(1, 1000),  # no_of_subs
                fake.name(),  # victim_name
                random_date().strftime('%Y-%m-%d'),  # date_of_death
                random.choice(races),  # race_of_victim
                random.randint(18, 80),  # age_of_victim
                random.choice(provinces),  # place_of_death_province
                random.choice(towns),  # place_of_death_town
                random.choice(locations),  # type_of_location
                random.choice(yes_no_unknown),  # sexual_assault
                random.choice(modes_of_death),  # mode_of_death_specific
                random.choice(yes_no_unknown),  # robbery_y_n_u
                fake.name(),  # perpetrator_name
                random.choice(relationships),  # perpetrator_relationship_to_victim
                random.choice(yes_no_unknown),  # suspect_arrested
                random.choice(yes_no_unknown),  # suspect_convicted
                random.choice(yes_no_unknown),  # multiple_murder
                random.choice(yes_no_unknown),  # intimate_femicide_y_n_u
                random.choice(yes_no_unknown),  # extreme_violence_y_n_m_u
                fake.sentence(),  # notes
            ])

    print(f"{num_entries} entries generated successfully in {file_name}")

# Generate CSV file with 10,000 entries
generate_csv('homicide_news_100000-3.csv', 100000)
```