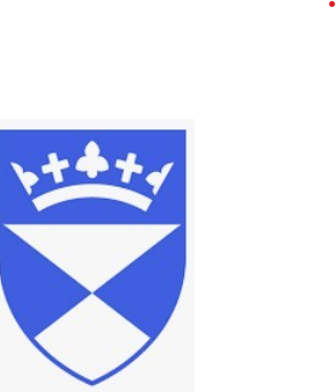
Databases as a Tool for Historical Simulation

George Tunde

Student ID: 2541357



A dissertation submitted for the award of

Master of Science in Data Science and Engineering

At the School of Science and Engineering,

University Of Dundee,

Dundee, Scotland

April 2024

Supervisor: Iain Murray

**Declaration**

I confirm that the work in this MSc project report was compiled solely by myself and has not been accepted in any previous application for a degree. All information sources were expressly acknowledged, and all verbatim extracts were distinguished by quotation marks.

**Signed by: Tunde George Date: 05/04/2024**

**Author Full Name: Tunde George Bamikole**

**Contents P age**

|  |  |
| --- | --- |
| [Abstract](file:///C:\Users\malnu\Downloads\RGU_Thesis_Template_2024%20(3).docx#_bookmark0) | 6 |
| [1 Introduction](file:///C:\Users\malnu\Downloads\RGU_Thesis_Template_2024%20(3).docx#_bookmark4) | 6 |
| [2 Background](file:///C:\Users\malnu\Downloads\RGU_Thesis_Template_2024%20(3).docx#_bookmark9) | 7 |
| [3 Problem](file:///C:\Users\malnu\Downloads\RGU_Thesis_Template_2024%20(3).docx#_bookmark19) Statement | 8 |
| [3.1 Objectives](file:///C:\Users\malnu\Downloads\RGU_Thesis_Template_2024%20(3).docx#_bookmark20) | 8 |
| [3.2 Development Process and Methodology](file:///C:\Users\malnu\Downloads\RGU_Thesis_Template_2024%20(3).docx#_bookmark21) | 8 |
| 3.2.1 Requirement Gathering | 8 |
| 3.2.2 Transfer of Gathered Data to MySQL Workbench | 9 |
| 3.2.2.1 Automatic Data Transfer | 9 |
| 3.2.2.2 Manual Transfer | 10 |
| 3.2.3 Design Phase | 10 |
| 3.3 Project Plan and Deliverables | 11 |
| 4.0 Design | 12 |
| 4.1 Database Selection | 12 |
| 4.2 Data Model Design | 13 |
| 4.3 User Interface Design | 13 |
| 4.4 Programming Language and Framework | 15 |
| 4.5 Design Trade-offs and Justifications | 15 |
| 4.6 Appraisal | 15 |
| 4.6.1 Reasons for the choice of technologies used. | 15 |
| 4.6.2 Why Google Forms for Data Input? | 15 |
| 4.6.3 Why Amazon RDS Database for Data Storage? | 15 |
| 4.6.4 Why MySQL Workbench for Database Management? | 16 |
| 4.6.5 Why choose Streamlit? | 16 |
| 4.6.6 Why choose Anaconda? | 16 |
| 4.6.7 Alternative technologies for creating the Historical Events visualisation | 17 |
| 4.7 Things to consider working on | 17 |
| 4.7.1 User Testing | 17 |
| 4.7.2 Data Analysis | 17 |
| 5 Implementation and Testing | 17 |
| 5.1 Overview | 17 |
| 5.2Data collection/gathering | 17 |
| 5.3 Database Setup | 18 |
| 5.4 Data Modelling | 18 |
| 5.5 App development | 18 |
| 5.6 Debugging and Optimisation | 19 |
| 5.7 Summary | 19 |
| Evaluation / Testing | 19 |
| Functionality Testing | 19 |
| Performance Testing | 19 |
| Scalability testing | 19 |
| Reliability Testing | 20 |
| Cost Implications Testing | 20 |
| Compatibility Testing | 20 |
| 7 Description of the final product | 20 |
| 8 Summary and Conclusions | 21 |
| 9 Future Work | 22 |
|  |  |

**Abstract**

This project focuses on creating a user-friendly experience using databases as a tool for historical simulation. The main goal is to enable users, irrespective of their technological expertise level, to visualise and interact effectively with historical events displayed on maps, improving their understanding of historical events while also visualising information about key events that happened in the past.

By leveraging modern database technologies, specifically Amazon RDS, and other frameworks for app development like Streamlit and Anaconda, an interactive map application for visualising historical data was developed, ensuring a user-friendly experience.

Users can input historical events through Google Forms. The stored Google Form responses are then transferred to a Google Sheet, from where they are imported to the AWS RDS instance through MySQL Workbench.

Leveraging the power of the Python programming language, these data were retrieved in JSON format and displayed through Streamlit by issuing a run command from the Anaconda command prompt. The resultant map application allows users to interact with and simulate preloaded historical events.

The project explores the importance of database management in historical research and simulation. It demonstrates the effectiveness of organising and analysing historical events’ data through the use of databases. Through careful design and implementation, this project clearly explains the potential of databases as a veritable tool for historical events simulation by giving visually pleasing and interactive insights into past events.

# Introduction

Historical simulation refers to the representation of historical event(s) or historical process(es) using a suitable information technology medium. Simulations can take visualisation a step further by creating a dynamic model that is responsive to user interaction in an immediate and tangible way.

It is noteworthy that simulations do not possess the ability to recreate history, as they are unable to tell what fully happened in full detail. However, simulations allow proper representation of complex historical events in a manner that is more dynamic and visually appealing than written text can ever be. Often, history enthusiasts are not too interested in “the whole thing that happened” as much as knowing the range of experiences, otherwise referred to as key events, that happened in time past.

To enable historians and researchers to use the technological tools available for historical simulation, they must accept the tool’s legitimacy and critically evaluate its usefulness to achieve meaningful simulations (Taylor, 2003).

Historical simulation is vital in fields of study such as urban and regional planning, traffic studies, finance, and military strategy planning. However, the efficacy of the results generated is heavily dependent on the accuracy of the data presentation and analysis process. The associated complexity of accessing a large historical data volume efficiently hugely limits historical simulation. To address this, the project is geared toward gathering data from historical event maps and literature, creating a database from the gathered information, and utilising it to simulate historical events. The project proposes using databases as a tool for carrying out historical simulations. Two different Historical events of high interest will be depicted using a mix of a data input interface, a database hosted in the cloud and a map that simulates the history associated with the chosen events.

It is worth mentioning that the result will be immensely helpful to history enthusiasts, military strategists, and the academic community at large.

Historical scholarship's primary goal is to explain complex relationships that affect changes in the past (Weinberg, 2018). Frequently occurring patterns of change are understood depending on geographic understandings—or variations over space. Historical events such as the Great Migration, the early development of the European economy, and the development of the American West, amongst others, are a few examples of historical and geographic events. Using the knowledge of both space and time gives a broader understanding of various events and transformations. This becomes more evident if discernment about characteristics of people and events which were part of the historical event is possible.

Historical simulations can be achieved using maps that give snapshots of historical realities. The advent of digital technologies has resulted in the development of maps that display comprehensive information alongside clear and explanatory visualisation obtained from clicking specially designed marker points. This makes it possible for historical realities at different points to be obtained easily at the click of a computer mouse.

From the database and map created, scholars and planners across different fields can easily inquire about the relationship between historical change and places and how this relates to several things in current planning patterns, quickly testing theories and questions. Questions such as: Was there a likely relationship between technological advancement and top cadre military personnel planning capabilities? Were there other relationships? What does this relationship connote for the development of military strength over the past few decades? Amongst other questions, historical simulations can easily and quickly answer them. Historical simulations can be used for assumption testing and building understanding that traditional research methods cannot easily offer.

# Background

The main goal of historical simulation is the recreation of past events and scenarios in a bid to understand the final outcome and various implications. The outcomes and implications plays a key role in future planning purpose. Prediction of trends, assessment of developmental impact on communities, military manoeuvres planning using historical wars as a yardstick amongst others can be addressed through relevant historical events simulations.

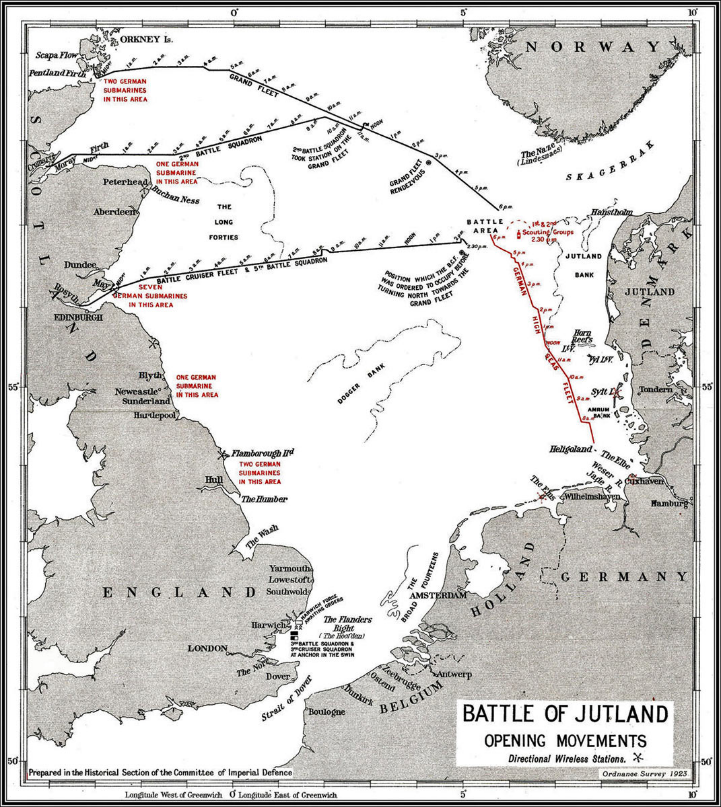
However, the use of historical events for decision making and achieving proper insights is usually hampered by the ease of access to information and the accuracy of historical information gathered. It is worth stating that traditional storage and retrieval of historical data using spreadsheets, hand written ship logs, spreadsheets or manual records are often incapable of handling the large volume of information needed for comprehensive historical simulations. Traditional historical simulation techniques often encounter limitations due to the complexities of managing and accessing large volumes of historical data efficiently.

In response to these challenges, this project proposes the utilisation of databases as a powerful tool for historical simulation.

Before this time (Wiedemann, 1999) presented an approach for creating a flexible event modelling and simulation environment using database technologies. His study concluded that the database structure used for simulation should allow for the flexibility of relationships among different objects in simulation models. Furthermore, he seriously discussed the use of relational or object-oriented databases. To implement his simulation, he opted for a relational database with database tables whose attributes are in two groups. The tables had a set of attributes for administration and others for the storage of the object's data. The resultant data structure was applied for every model entity. He also developed a relational language called SimSQL for model manipulation during design and run-time.

Historical simulation using maps has been carried out by a few researchers in recent times. Ellis (2020) was able to create a map that simulates the various locations of the 314 warships that were used by the United Kingdom during World War 1. Prior to the research, there was no available visualisation of the historical positions of all the 314 ships together, which could help properly visualise the large scale and complexity of British naval operations during the war which necessitated the research work.

Not only this, Bolt et al. (2016) studied the battle of Jutland and accessed its impact on the people of Portsmouth. A dynamic map that serves as a historical memorial of officers with their name, rank, service number, ship name, date of birth, place of birth, next of kin, next of kin address and location of death was created. This tool was valuable for the remembrance and celebration of the fallen hero and active participants in the battle.



**Fig 2.0 Map of the Battle of Jutland**

Advancements in technology have increased the drive for the creation of simulation software and tools for facilitating historical analysis. Software products such as Simul8 and AnyLogic provide simulation capabilities by utilising historical data to generate insights and optimise processes. These tools have vast functionalities but still depend on external data sources or databases. This has the effect of limiting their flexibility and adaptability.

Many technological developments have made historical simulation much easier. Big Data and Data Analytics, Machine Learning and Artificial intelligence, Virtual Reality (VR) and Augmented Reality (AR), Geographic Information Systems (GIS), Computational Modelling and Simulation, and cloud computing, amongst others, have made data storage and analysis much easier. Historians are empowered with tools and resources for past event simulation exploration of alternative historical scenarios and getting new insights into the complexities of human history.

Furthermore, in database management, some researchers have been able to investigate the database's role in supporting simulation and decision-making processes. Garcia-Mila et al. (2019) and Liang et al. (2021) explored the integration of databases with simulation models. Their research highlighted the potential benefits of using centralised data storage and efficient data retrieval.

Despite these advancements, a wide gap exists in integrating databases used for historical simulation purposes. The databases currently available may not be able to properly address the complexities of historical data management, including temporal dependencies, data versioning, and scalability. Also, database integration with simulation models needs a thorough consideration of data formats, query mechanisms, and synchronisation processes.

In summary, the background literature and existing products provide the required foundation for exploring the use of databases as a tool for historical simulation. By synthesising insights from relevant research, this project aims to address the limitations of traditional simulation methods and contribute to advancing historical analysis techniques.

**3.0 Problem Statement**

The major issue addressed by this work is the inefficiency and inadequacy of traditional methods applied while managing historical data utilised for simulation purposes. It is worth mentioning that spreadsheet-based systems lack scalability and can become cumbersome as data volume increases. Keeping records manually is prone to errors and inconsistencies. This limitation of manual data handling leads to inaccuracies in simulations. Furthermore, quick retrieval and seamless integration simulation models pose significant challenges.

Historical simulation, an important field in historical research and analysis, is faced with the inadequacies, inefficiencies, and limitations associated with traditional data management methods, which heavily depend on spreadsheet-based systems. Spreadsheet-based systems are not scalable and have limited data handling capability. As data volume increases, historical data increases as well, and traditional spreadsheet systems struggle to measure up to the challenges posed by ever-growing data due to an ever-increasing volume and complexity. This makes the data management and handling process error-prone and cumbersome.

Another common management practice, manual data management, faces the challenge of inaccuracies and inconsistencies. Human-generated data entry errors and manipulation can affect the reliability and validity of the insights generated. Furthermore, the unavailability of standardised procedures and version control in manual data management hinders the maintenance of data traceability and integrity over a long period.

Given these limitations, the need for an innovative approach to historical data management and simulation is essential. Leveraging the AWS RDS, a modern database technology or similar modern database technology offers a good solution to the aforementioned challenges as it provides a scalable, reliable, and efficient avenue for storing, organising and accessing historical data for simulation operations, providing a useful tool for historical simulation.

**3.1 Objectives**

The major objective of this project is developing and demonstrating the effectiveness of using databases as a tool for historical data simulation. Specifically, the main aim of the project is the achievement of the following:

* Easy visualisation of historical events: this involves the development of a user-friendly interface with a querying tool for enabling easy visualisation of events.
* To allow adequate user interaction with historical events visualisation: the application for simulation is designed in such a way that the user can effectively interact with it and ascertain what happened at specific times.
* Improvement of Simulation Accuracy: integration of the database with simulation models to produce accurate and timely data inputs, which leads to more realistic simulations.

Achievement of the aforementioned objectives will help revolutionise the way historical simulations are conducted, empowering users to gain better insights and interact effectively in historical visualisations.

**3.2 Development Process and Methodology**

The project development process follows an iterative and collaborative approach. It incorporates elements of agile methodology to ensure flexibility and responsiveness to evolving requirements. Methodology selection involves the stages listed below:

**3.2.1 Requirement Gathering**

This entails reviewing online publications, checking for available ship log records and reviewing historical maps. This was done to identify and prioritise key requirements for the database solution that will make it adaptable to many forms of historical events. Generic details that cut across different historical events such as Event Name, Route ID, Location Name, Latitude, Longitude, Key Event Notes, Time, Year, Month, Day, Country Enroute and Object ID, Object Name, Action and Key Effects of action were identified as exciting fields that cuts across all significant Historical events. These fields were used to create the Location Google Form and the ObjectAndEffect Google Forms, respectively.

The gathered information was transferred to Google Sheets, downloaded, and saved as .csv files before being imported to the MySQL Workbench for easy backup in the AWS RDS cloud service.

Entry requirements/validation options were included in the form designs. This was done to ensure data accuracy, completeness, and integrity and to improve user experience. A case in point is when it was specified that route ID should be specified as a number. Also, it was ensured that respondents using the system would give all necessary information before form submission could be affected to avoid incomplete responses. Lastly, data Integrity was adequately handled by preventing invalid or irrelevant responses. For example, constraints were set for the format of year, day, month, day, and time entry, thus preventing respondents from providing nonsensical values. User experience was seriously considered by providing concise guidance to prospective users on form completion. For example, the failure of the form to submit when entry requirements are not met was incorporated to guide users in correcting their responses.

**3.2.2 Transfer of Gathered Data to MySQL Workbench**

Data gathered was automatically transferred to the database.

**3.2.2.1 Automatic Data Transfer**

The Python code line used for automated data transfer is outlined in Fig 3.0.

import mysql.connector

from mysql.connector import Error

from googleapiclient.discovery import build

from google.oauth2.credentials import Credentials

# This function fetches Google Sheet Data

def fetch\_data\_from\_sheet(sheet\_id, range\_name):

    creds = Credentials.from\_authorized\_user\_file('credentials.json', SCOPES)

    service = build('sheets', 'v4', credentials=creds)

    sheet = service.spreadsheets()

    result = sheet.values().get(spreadsheetId=sheet\_id, range=range\_name).execute()

    return result.get('values', [])

# The function below inserts data into MySQL

def insert\_data\_into\_mysql(db\_config, insert\_stmt, data):

    try:

        connection = mysql.connector.connect(\*\*db\_config)

        cursor = connection.cursor()

        for row in data:

            cursor.execute(insert\_stmt, row)

        connection.commit()

    except Error as e:

        print(f"Error: {e}")

    finally:

        if connection and connection.is\_connected():

            cursor.close()

            connection.close()

# Setup of Google Sheets API

SCOPES = ['https://www.googleapis.com/auth/spreadsheets.readonly']

# MySQL connection configuration

mysql\_config = {

    'user': 'georgieoloye',

    'password': 'apakooloye',

    'host': historical-simulation-project1.cvu6w2msg3b3.us-east-1.rds.amazonaws.com,

    'database': 'EVENT\_SIMULATION',

    'raise\_on\_warnings': True

}

# This fetches data from Location Form

location\_form\_sheet\_id = '1a2b3c4d5e6f7g8h9i0j'

location\_form\_range = 'Sheet1!A2:K'

location\_data = fetch\_data\_from\_sheet(location\_form\_sheet\_id, location\_form\_range)

# This SQL INSERT statement is for the Location Table

location\_insert\_stmt = (

    "INSERT INTO location\_table (historical\_event, route\_id, location\_name, latitude, longitude, "

    "key\_event\_notes, time, year, month, day) "

    "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

)

# Inserts data into the Location table

insert\_data\_into\_mysql(mysql\_config, location\_insert\_stmt, location\_data)

# This Fetches data for the ObjectAndEffect form

object\_effect\_form\_sheet\_id = '0j9i8h7g6f5e4d3c2b1a'

object\_effect\_form\_range = 'Sheet1!A2:M'

object\_effect\_data = fetch\_data\_from\_sheet(object\_effect\_form\_sheet\_id, object\_effect\_form\_range)

# The SQL INSERT statement for the ObjectAndEffectTable

object\_effect\_insert\_stmt = (

    "INSERT INTO ObjectAndEffect\_table (event\_name, route\_id, object\_name, object\_id, "

    "latitude, longitude, time, action, key\_effect, countryEnroute) "

    "VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)"

)

# This inserts data into ObjectAndEffect Table

Insert\_data\_into\_mysql(mysql\_config, object\_effect\_insert\_stmt, object\_effect\_data)

**Fig 3.0 Code Line for data transfer from Google Form/Google Sheets to MySQL database**

**3.2.2.2 Manual Transfer**

Alternatively, to transfer data to MySQL Workbench, Google Sheets can be exported as CSV by Opening the Google Sheet to be exported. Subsequently, the process flow below is followed.

File > Download > Comma-separated values (.csv, current sheet).

This will help to download the sheet as a CSV file.

**Preparation of downloaded CSV File:** To ensure the downloaded CSV file was correctly formatted, each field needs to be compared to the fields in the tables created in the database.

Subsequently, MySQL Workbench was launched, and a connection was made using the cloud-based database endpoint (hostname), username, and password.

**Import CSV File into MySQL Workbench:** After establishing the connection to the database, the database where the CSV file will be imported will be navigated.

Right-click the database name. The "Table Data Import Wizard" will then be selected, and the table will be imported.

**Data verification:** After the import process, SQL queries must be run to confirm that the suitable tables were imported into MySQL Workbench.

**3.2.3 Design Phase**

An apt design specification was developed using requirements gathered to carry out the design. The database schema was outlined, and the data collection and storage strategy was properly reviewed. The stages identified were:

* Input: This involves inputting historical data by users using Google Forms and Google Sheets.
* Edit: In this stage, an AWS relational database is created for data storage in the cloud and connected to MySQL Workbench to ease access, querying and editing. The data generated at the input stage was then imported into the database through MySQL Workbench. The gathered data was thereafter downloaded as a .csv file after proper storage in the cloud-based MySQL database and converted to a JSON file format using the Python programming language. The output file, test.json, was subsequently uploaded into Visual Studio code for use in the development of the **streamlit\_app.py** app.

import csv

import json

import os

# This part of the code defines the directory path

desktop\_path = os.path.join(os.path.expanduser("~"), "Desktop")

# The code par below gives the path to the CSV files

csv\_files = [

    os.path.join(desktop\_path, 'Location Table.csv'),

    os.path.join(desktop\_path, 'Location Table2.csv')

]

# This gives the path to the JSON file

json\_file = os.path.join(desktop\_path, 'test.json')

# The code line below combines the CSV files

combined\_data = []

for csv\_file in csv\_files:

    with open(csv\_file, 'r') as f:

        csv\_reader = csv.DictReader(f)

        combined\_data.extend(csv\_reader)

# The code line below writes the combined data to a new JSON file

with open(json\_file, 'w') as f:

    json.dump(combined\_data, f, indent=4)

print("Combined JSON file created:", json\_file)

**Fig 3.1: Code line for combining the .csv file for the two historical events to be simulated and also converting the combination to a JSON file ‘test.json’.**

This code can be modified to take care of several historical events.

* Running: Once the Python code is completed in Visual Studio code for the creation of the Streamlit\_app web-based application, the required Python libraries will be installed into Visual Studio code through the terminal. The required libraries are: Streamlit, pandas, folium, streamlit\_folium, folium\_static and numpy

The command required for the installation is:

**pip install streamlit pandas folium streamlit\_folium numpy**

Thereafter, the command line:

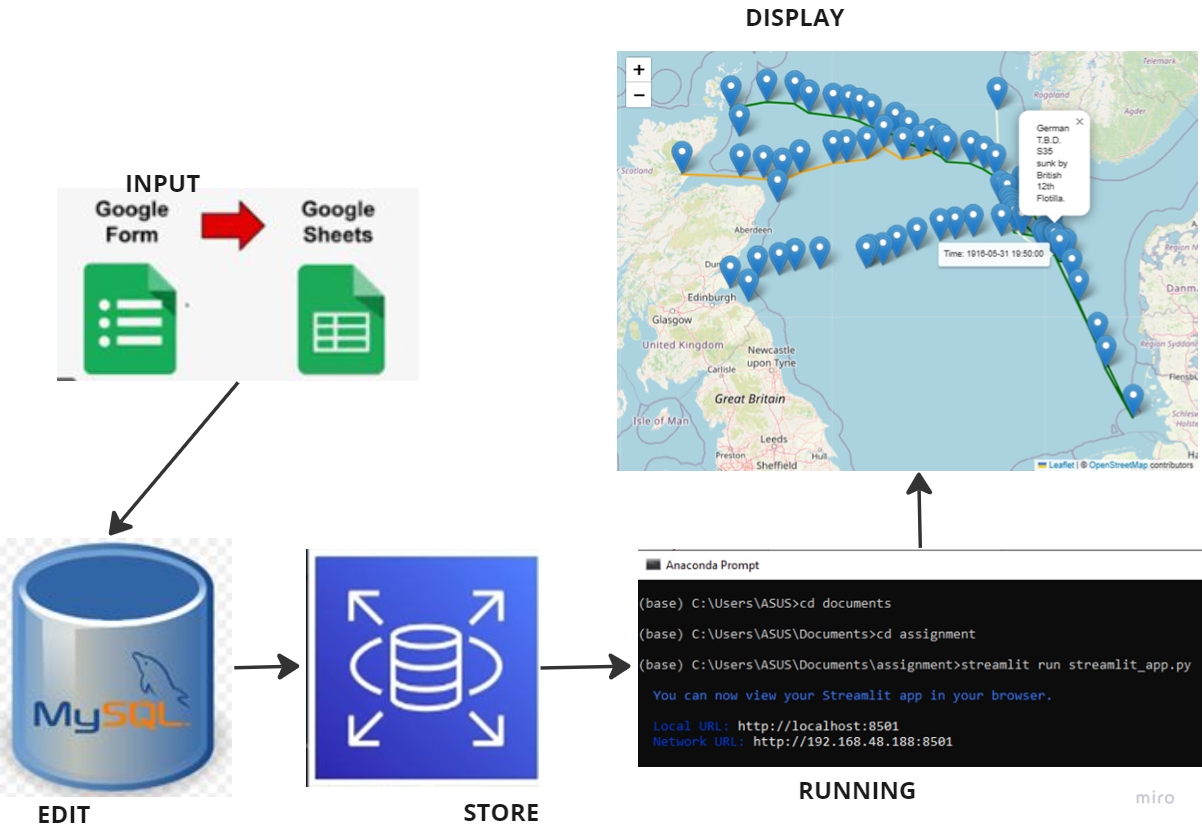
**streamlit run streamlit\_app.py**

is executed/run on the Anaconda prompt to display historical events.

The creation of the Streamlit app is extensively explained in the app development section of this project.

Once the web-based app Streamlit is running, a visual display will be seen as a web page. The visual display allows user interaction and gives the required visualisation.

The proposed visualisation software design is depicted with the system architecture in Fig 3.2.



**Fig 3.2: System Architecture**

Prototype Development: The initial prototype of the database solution was done taking into consideration key elements that usually occur across historical events. Only one historical event (The battle of Jutland) was first simulated before looking at how to scale up the Streamlit application.

Data ingestion method, applicable interface-based queries that can be issued by people having different computer use experiences and integration with the simulation, were carefully worked on.

**3.3 Project Plan and Deliverables**

The project plan encompasses the following deliverables and timescales:

**Requirement Specification and Background Reading (Week 1-2)**

During the first week, I had my first meeting with my supervisor. Extensive discussion on the project deliverables was done. In the second week, based on the previous week's description, the detailed requirement and turn-around time for identified individual tasks involved were used to create a Gantt chart. The Gantt chart prescribes the timeline for the project deliverables. Online research was suggested to have a feel of similar projects published by researchers and system developers and subsequent analysis of research findings to find any research similar to the project topic.

In addition, historical events with available online data, such as maps and ship log records, were reviewed.

**Design Specification (Week 3-4)**

Various findings were collated and submitted for review. Sequel to this, a comprehensive database design specification that considers the likely fields that cut across different historical events was brought up. The location table with fields such as event name, Route ID, Latitude, Longitude, KeyEventNotes, Time, Year, Month and Year and the Object and Effect Table with Object ID, latitude, and longitude is recommended.

The database schema and the loading process using the import function in MySQL Workbench for the Google worksheet were investigated for use.

**Prototype Development (Week 3)**

A comprehensive study of the Design requirements and a careful review of likely applications for project design were done.

An initial prototype of the database solution was done while strongly focusing on the prospective app’s core functionalities and required historical simulation.

**Progress Review (Week 4 - 5)**

The progress made was reviewed, and feedback was given on the progress made.

Furthermore, the Database, Tables Design and Google form Design were presented for review and correction.

Submission of a write-up on proposed events and key happenings, dates, maps, and literature was planned for review.

**Testing and Feedback (Week 6-7):**

This entails writing up deliverables and submission for verification of conformance to project goals.

Also, the drafting of the ethics required for app testing and the selection of the testing team to ensure the development of an app that will bring about users’ satisfaction were planned. This covered the execution of rigorous testing of the prototype app and gathering feedback to identify and include identified areas of improvement. Risk Assessment was also planned.

**Iteration and Refinement, further App testing and collation of responses from participants (Week 8)**

Based on feedback received from prospective users and the project supervisor, iterations are expected to be done on the prototype, incorporating enhancements, and addressing issues.

Adherence to this project plan and methodology was strictly adhered to, with the aim of delivering a robust and scalable database solution that satisfies the needs of historical simulation enthusiasts and contributes immensely to the advancement of simulation techniques. Fig 3.2 Shows the Gantt chart used to plan the achievement of the project objectives.

Appendix 1 shows the project management plan employed using the Gantt Chart.

**4.0 Design**

**4.1 Database Selection**

A critical choice at the beginning of this project was to select an appropriate database technology that would provide a solid foundation for a historical simulation tool.

Upon proper consideration and evaluation, a relational database management system (RDBMS) was chosen due to its widespread use, robustness, and flexibility.

MySQL Workbench, which uses MySQL database management system, a graphical user interface (GUI) tool for MySQL that allows management of databases for the performance of various tasks, was adopted.

MySQL Workbench was chosen as it allows seamless connection to the database created in the cloud. Not only this, new datasets provided by users can be easily uploaded into the AWS database in the cloud by using the database endpoint, username and password configured in the cloud. This makes it easy to store data without the fear of losing stored data and allows easy data access from anywhere in the World.

To create the location and ObjectAndEffect Tables in the database, the Create Tables command was executed in MySQL workbench. The command lines for the table creation are given below:

CREATE TABLE Location (

RouteID INT PRIMARY KEY,

EventName VARCHAR(240),

LocationName VARCHAR(240),

Latitude DECIMAL(12, 9),

Longitude DECIMAL(12, 8),

KeyEventNotes VARCHAR(240),

Time TIME,

Year INT,

Month INT,

Day INT,

CountryEnRoute VARCHAR(240)

);

CREATE TABLE ObjectAndEffect (

ObjectID INT PRIMARY KEY,

RouteID INT,

BattleName VARCHAR(240),

ObjectName VARCHAR(240),

Latitude DECIMAL(12, 9),

Longitude DECIMAL(12, 9),

Time TIME,

Year INT,

Month INT,

Day INT,

Action VARCHAR(240),

KeyEffectOfAction VARCHAR(240),

CountryEnRoute VARCHAR(240),

FOREIGN KEY (RouteID) REFERENCES Location(RouteID)

);

**4.2 Data Model Design**

A key aspect of the design process involved developing a comprehensive data model to represent historical data in a structured and efficient manner. Entity-Relationship (E-R) modelling techniques was used to conceptualise the relationships between different entities within the historical dataset.

The data model consists of entities such as events, time periods, locations, and actors, each with defined attributes and relationships. For example, the "Event" entity encapsulates information about specific historical events, including their date, location, participants, and outcomes. Relationships between entities, such as “Route ID”, "Action” and “Key Event Notes” were established using foreign key constraints to maintain data integrity. The entity relationship diagram is shown in Fig 4.0.

|  |
| --- |
| **Location Table** |
| Event Name |
| RouteID |
| Location Name |
| Latitude |
| Longitude |
| Key Event Notes |
| Time |
| Year |
| Month |
| Day |
| CountryEnRoute |

|  |
| --- |
| **ObjectAndEffect Table** |
| Event Name |
| Object Name |
| Object ID |
| Latitude |
| Longitude |
| Time |
| Year |
| Month |
| Day |
| Action |
| Key Effect of Action |
| CountryEnRoute |
| RouteID |

Fig 4.0: **Entity Relationship Diagram**

**4.3 User Interface Design**

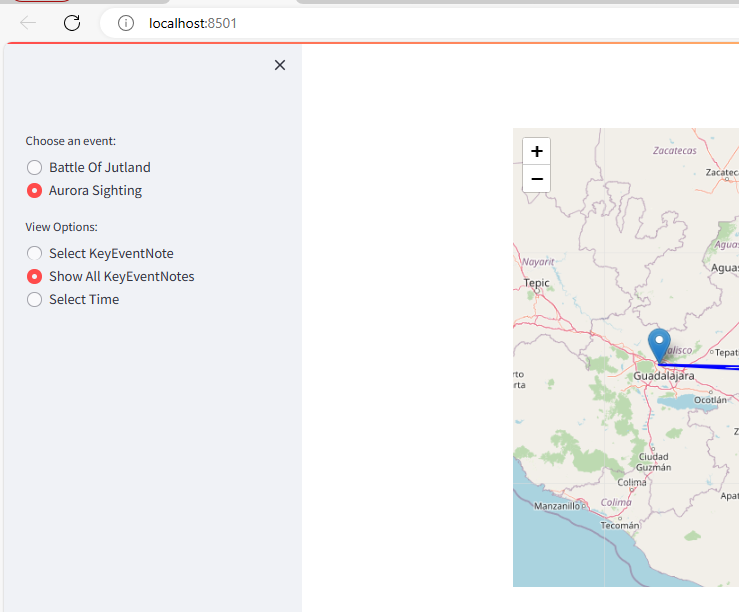
Usability and accessibility were primary considerations in designing the user interface for interacting with the historical database. A web-based interface was opted for to ensure platform independence and ease user access across different devices and operating systems.

The user interface features intuitive search and query functionalities, allowing users to retrieve historical data based on various criteria such as time period, location, and event type.

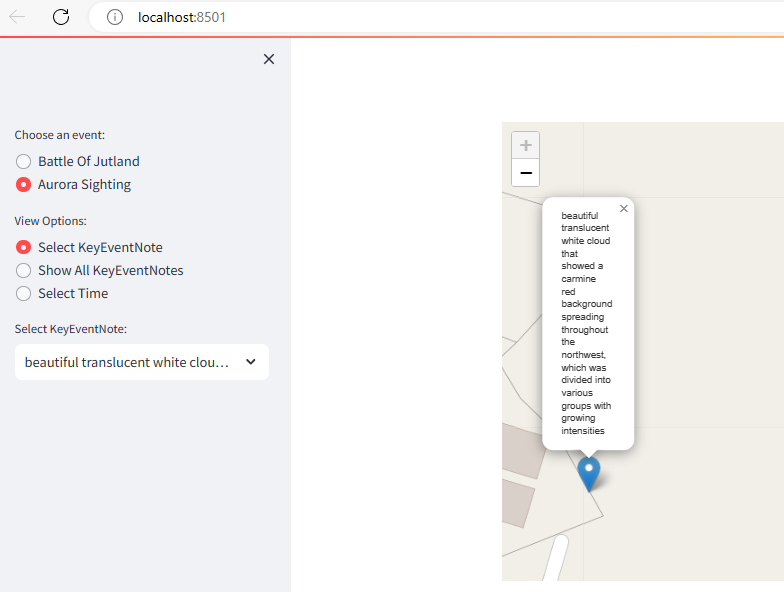
Additionally, interactive visualisation tools were incorporated to facilitate data exploration and analysis, enabling users to gain insights from the historical dataset visually.

Fig 4.1 shows radio buttons that allow users to choose the historical event they want to visualise and the view option. When show AlltheKeyEventNotes is selected, all the events that happened in the selected event will be displayed. To allow user interactivity, when the select time option is picked, users can move a time-based slider that allows them to know what happened at the selected time.

Also, a drop-down functionality is enabled when the select KeyEventNote is selected.



**Fig 4.1: Choosing an Event (Aurora sighting) as well as showing all events that happened.**



**Fig 4.2: Selecting KeyEventNote**

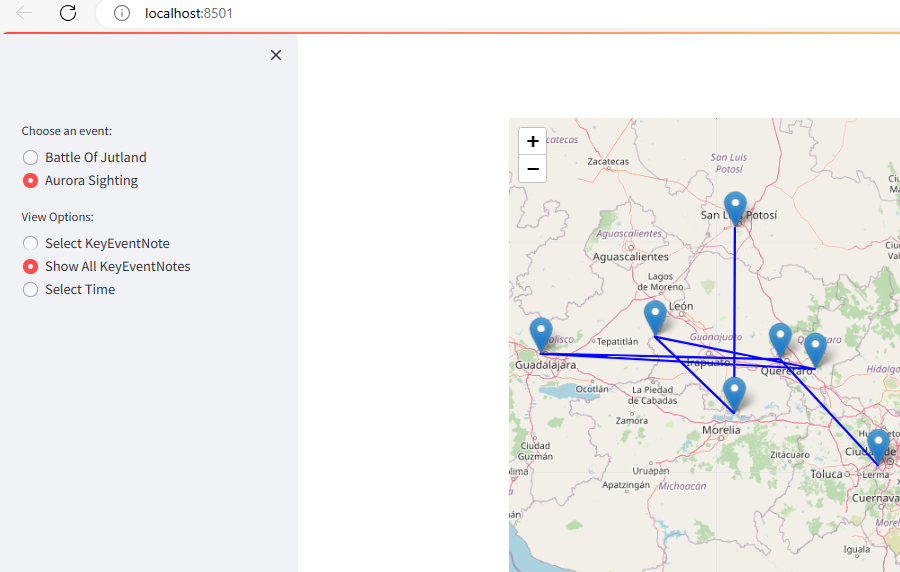
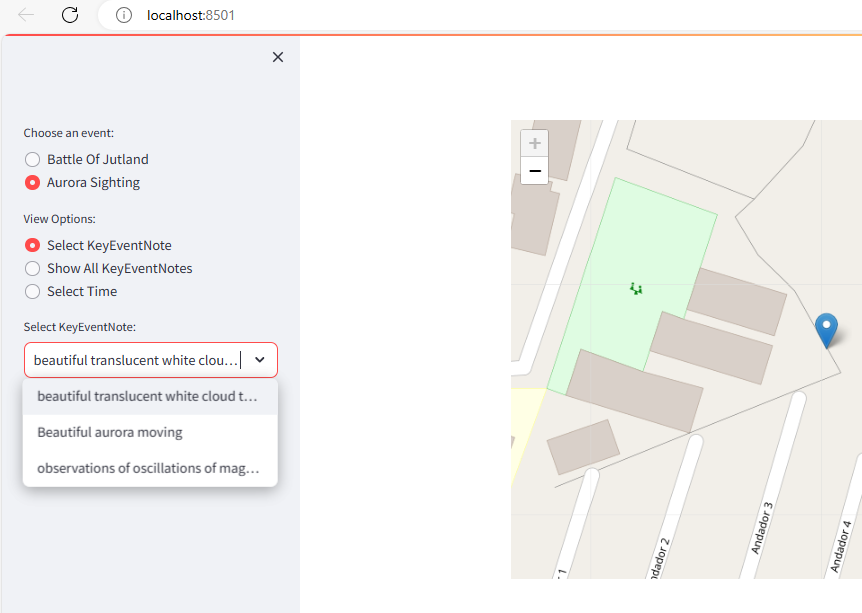
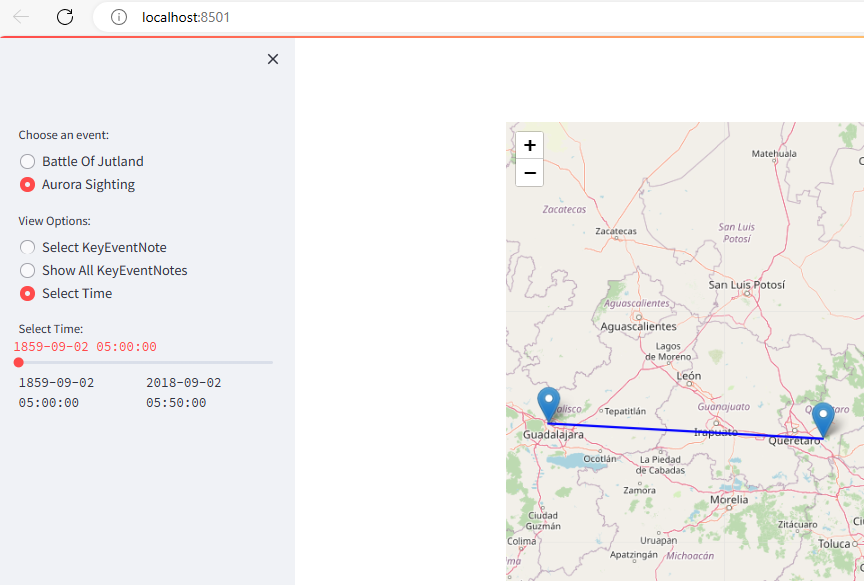


Fig 4.3 Displaying All KeyEventNotes



**Fig 4.4: Using the Select KeyEventNote with the drop down bar**



**Fig 4.5: Using the time based slider for user interaction**

**4.4 Programming Language and Framework**

For getting data in an acceptable format for visualisation Python programming language was used. This helped to achieve optimal performance, scalability, and maintainability.

For the app development, the Streamlit framework, developed using the Python programming language, was used. This helped to facilitate the development of dynamic web components and enhance user experience.

**4.5 Design Trade-offs and Justifications**

Throughout the design process, various trade-offs were considered to balance competing priorities such as performance, complexity, and development time. For example, while a NoSQL database may offer better scalability for unstructured data, an RDBMS, MySQL, was opted for due to its robustness.

Similarly, the decision to develop a web-based user interface over a desktop application was driven by considerations of accessibility and ease of deployment, despite potential trade-offs in performance for certain tasks.

Overall, the design choices were made after careful consideration of project requirements, feasibility, and long-term sustainability, aiming to deliver a database solution that effectively meets the needs of historical simulation practitioners.

# 4.6 Appraisal

The project provided a good simulation of historical events as users can actually query the visual display without the knowledge of any form of query language. Furthermore, users who need to visualise different historical events can provide data by filling out the designed Google form. This will eventually be used for the simulation of different historical events aside from the ones displayed in the map.

## 4.6.1 Reasons for the choice of technologies used.

Google Forms was used for data input, AWS MySQL RDS was used in conjunction with MySQL Workbench, and the command for running the Streamlit app was done in the Anaconda environment.

**4.6.2 Why Google Forms for Data Input?**

Ease of Use: it provides a user-friendly interface which facilitates the creation of custom forms that can be used for various inputs. Using Google forms makes it easy for users from all over the world to provide input of historical data without requiring technical expertise.

Accessibility: It is worth stating that Google Forms can be accessed from any device that has internet connection. Hence, allowing data submission at any time, from anywhere, thus, enhancing convenience and accessibility.

Data Validation: Google forms have functionality for supporting data validation using data validation rules. This allows the enforcement of constraints on input fields to ensure data accuracy and consistency.

Integration with Google Sheets: For Streamlit\_app.py app, Google form allows automatic storage in Google Sheets spreadsheet. This enhances easy data management, analysis, and sharing within teams.

Real-Time Collaboration: The use of Google Forms allows real-time collaboration. It allows multiple users to work on the same form at the same time, hence reducing data collection time for collaborative historical events simulation projects.

**4.6.3 Why Amazon RDS Database for Data Storage?**

Amazon RDS MySQL was selected as the database for simulation due to the following reasons:

Scalability: AWS RDS can support an ever-expanding dataset and constantly increasing number of users. This helps to achieve optimal performance and enhances the reliability of historical simulation apps.

High Availability: It offers automated backups, snapshots, and multi-AZ deployments, reducing data loss risk and enabling availability and data durability.

Security: It offers great security features, such as encryption at rest and in transit, IAM integration, and network isolation. This ensures the confidentiality and integrity of historical data.

**4.6.4 Why MySQL Workbench for Database Management?**

Easy data import feature: MySQL has an import feature that allows quick and easy import of data.

Query Development: it has a SQL editor that facilitates the testing of SQL queries required for data analysis or possible data manipulation.

Easy connection to the AWS RDS: MySQL Workbench allows easy connection to the AWS RDS MySQL using the endpoint, username and password used for creating the AWS RDS.

**4.6.5 Why choose Streamlit?**

Simple and Pythonic: Streamlit has a simple and Pythonic nature which enables software developers to produce nice and readable codes that perform optimally with minimal effort. Through the use of Python’s intuitive syntax and design principles, Streamlit helps to achieve the creation of tidy, concise and maintainable codes which improves codes readability, understanding and collaboration amongst the members of software developing teams.

Fast, Interactive Prototyping: Streamlit fast tracks the app prototyping process by giving a streamlined environment for creating interactive applications with little effort. The fact that it has an intuitive interface and reactive design allows developers to quickly iterate on ideas, and perform immediate visualisation changes and modifications in real-time, which supports creativity and experimentation by giving developers the opportunity to explore different design concepts and features without the overhead of manual code compilation or deployment.

Sequel to this, Streamlit allows software developers to prototype and modify their software with high speed and efficiency.

Rich Visualisation Capabilities: Streamlit has several built-in components and libraries which provides top notch capabilities for producing rich, interesting, user friendly and immersive visualisations. Ranging from graphs, scatter plots, charts, dashboards and interactive maps, it gives app developers the tools required to produce awe-inspiring and compelling historical data visualisation. The fact that it integrates seamlessly with visualisation libraries such as Matplotlib and Plotly, assists developers to leverage advanced features and customisation options, thus ensuring that visualisation outputs are informative as well as appealing visually.

Seamless Integration with Python Ecosystem: Streamlit is a Python-based framework. This makes it easy for it to adapt to the extensive Python libraries and tools, thus easing interoperability and code reusability. It allows the use of popular Python libraries like Pandas, NumPy, and Scikit-learn, which all seamlessly integrate into Streamlit applications.

Streamlined Development Workflow: Streamlit's declarative syntax streamlines its development workflow. This helps build highly functional applications without the worry of writing complex code structures.

**4.6.6 Why choose Anaconda?**

Environment Management: Anaconda simplifies Python environment management through the provision of tools like Conda, which enables users to manage, create, and switch seamlessly between various Python environments.

This is of immense benefit when carrying out several projects with several dependencies or when working in collaboration with other developers. Anaconda helps to create isolated environments for Streamlit apps, hence allowing proper management of dependencies while also minimising conflicts. This capability of Anaconda enhances reproducibility and streamlines the app development and deployment process.

**Package Management:** The package management capabilities of Anaconda streamline the process of installation and management of dependencies for Streamlit and other required packages. The Anaconda repository or external channels can be leveraged by using conda to install packages. This has the benefit of ensuring consistent and reliable package installations across different environments. Also, it provides a comprehensive package index, thus easing the search and installation of packages required for the development of Streamlit app. This capability of Anaconda simplifies the app setup process and minimises the possibility of experiencing compatibility issues or dependency conflicts.

**Compatibility:** Anaconda allows cross-platform compatibility, which enables Streamlit apps to run seamlessly across several operating systems, such as Windows, macOS, and Linux. This removes the need for making platform-specific adjustments or configurations, thus ensuring that Streamlit apps behave consistently regardless of the underlying operating system.

This capability allows a seamless development experience and improves the portability of Streamlit apps across various computing environments.

With the assistance offered through Streamlit and Anaconda's joint capabilities, interactive historical simulation apps can be created, deployed, and managed easily while still maintaining compatibility, reliability, and efficiency across the app development lifecycle.

## 4.6.7 Alternative technologies for creating the Historical Events visualisation

Other frameworks can be used to create visualisation. These include Leaflet.js, Mapbox GL JS, Google Maps JavaScript API, D3.js, and OpenLayers.

Leaflet.js: this is an open source JavaScript library that can be used for creating interactive maps. It is lightweight and gives a highly customisable means of showing historical events on maps

Mapbox GL JS: this is a powerful JavaScript library. It can be used for producing user friendly and customisable maps. It has several advanced features like vector tile rendering, data-driven styling, and real-time data updates. The real time data update feature will make it applicable for use in dynamic events visualisation

Google Maps JavaScript API: this offers a wide range of tools for embedding Google Maps into web based applications. It offers a good alternative to the Streamlit used in this project.

D3.js: this is a JavaScript library widely used for data visualisation. It provides functionalities for creating choropleth maps which can be used for historical events visualisation.

OpenLayers: This is an open-source JavaScript library used for creating web based maps in browsers.

Of these solutions, if I am able to carry out this project all over again, I will try to test the use of Leaflet.js as I would like to include a Play button that allows me to slowly watch the progression of happenings in historical event alongside the time-based slider that I used for this project.

## 4.7 Things to consider working on

Due to time constraints the following were not fully catered for. However, other researchers considering further work on this project could do the following:

**4.7.1 User Testing**

Aside myself that tested and my supervisor that observed the testing process, external users were not employed to test the app. This would have afforded other users the opportunity to evaluate the usability and effectiveness of the historical simulation tool. The resultant effect will be that specific visualisation tasks that will be performed by users based on a structured testing protocol will help to further assess the key functionalities and features of the system.

**4.7.2 Data Analysis**

Analysis of data collected from user testing sessions to identify common themes, patterns, and issues related to system usability and functionality was not carried out. There is need for further work on the development of quantitative metrics. This metrics includes but not limited to app response time, performance of the visualisation and interaction buttons, effect of poor internet bandwidth on the system, performance under heavy load request from a large number of users, overall users experience amongst others.

Analysis of such metrics will better help to assess the efficiency and effectiveness of the system.

Also, likely areas of improvement for the system can be easily identified through the user testing process.

Furthermore, ethical consideration was not carried out as extensive users testing was not done. Obtaining informed consents from participants in the testing process, ensuring confidentiality of provided participant data, prioritising the well-being of participants and seeking approval from the relevant authorising authority in the School of Data Science and Engineering needs to be sought to comply with ethical consideration.

**5 Implementation and Testing**

**5.1 Overview**

The implementation of the simulation involved several key stages, such as data collection/gathering, database modelling, Database setup in the AWS cloud and integration of the database to MySQL Workbench, conversion of data in MySQL Workbench to JSON datatype, writing and running the Python script for the Streamlit app, and testing. Throughout the implementation process, agile methodologies and iterative development practices were both employed to ensure flexibility and responsiveness to evolving requirements of different types of historical events.

**5.2Data collection/gathering**

Google forms were deployed for the collation of historical data from various sources. It also offers easy usage by future prospective software users.

**5.3 Database Setup**

An AWS MySQL RDS instance was set up in the AWS cloud and linked with MySQL Workbench. MySQL Workbench was installed on the local machine, and the correct endpoint, user name, and password generated from the created AWS RDS were used to access the database hosted in the AWS RDS cloud.

To create a DB instance, the following steps were carried out:

* Sign in to the AWS Management Console and open the Amazon RDS console at https://console.aws.amazon.com/rds/.
* In the upper-right corner of the Amazon RDS console, AWS region where the DB instance will be created was chosen.
* In the navigation pane, Databases was selected.
* Create database, then Standard create were subsequently selected. The database engine to use was also selected as MySQL.

Appendix 2 and Appendix 3 show the different database engines in AWS and how to connect AWS Database instance to MySQL Workbench.

**5.4 Data Modelling**

With the database environment in place, the next phase focused on data modelling using Entity-Relationship (E-R) diagrams. The data model was refined based on feedback from my supervisor while also ensuring that historical datasets are accurately represented.

**5.5 App development**

Python code was used for converting .csv files to JSON, Streamlit app was used for getting the required map visualisation. Key functionalities implemented in the visualisation included ease user interaction and prompt query processing.

Fig 5.2 and Fig5.3 shows code snippets used in the streamlit\_app.py.

# Using 'test.json' is loaded into 'df'

# Load data

df = pd.read\_json('test.json')

**Fig 5.2: Code snippet that loads the json file**

The code snippet in Fig 5.2 loads test.json for use in map creation.

For simulating historical events and meeting the core objectives of easy visualisation of historical events using a user-friendly interface that allows adequate interaction with historical events, the code snippet in Fig 5.3 was used.

# This part of the code handles different functionalities that allows good visualisation and user friendly interaction based on user selection

view\_option = st.sidebar.radio("View Options:", ["Select KeyEventNote", "Show All KeyEventNotes", "Select Time"], key='view\_option')

if view\_option == "Select KeyEventNote":

    key\_event\_note\_select = st.sidebar.selectbox("Select KeyEventNote:", filtered\_data['KeyEventNotes'].unique(), key='key\_event\_select')

    selected\_data = filtered\_data[filtered\_data['KeyEventNotes'] == key\_event\_note\_select]

    display\_map(selected\_data)

elif view\_option == "Show All KeyEventNotes":

    display\_map(filtered\_data)

elif view\_option == "Select Time":

    time\_options = filtered\_data['DateTime'].dt.strftime('%Y-%m-%d %H:%M:%S').sort\_values().unique()

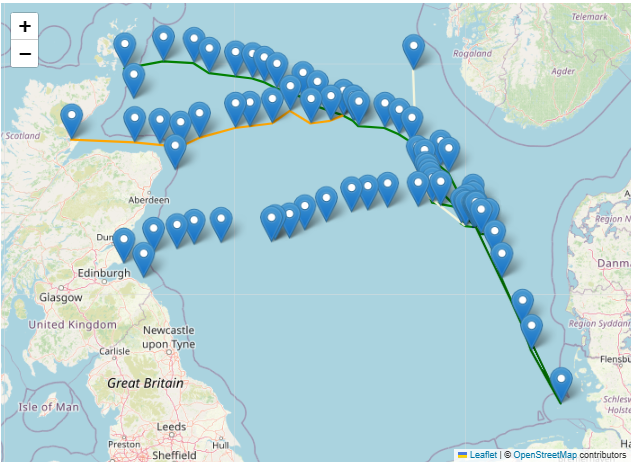
    selected\_time = st.sidebar.select\_slider('Select Time:', options=time\_options, key='time\_select')

    selected\_data = filtered\_data[filtered\_data['DateTime'].dt.strftime('%Y-%m-%d %H:%M:%S') == selected\_time]

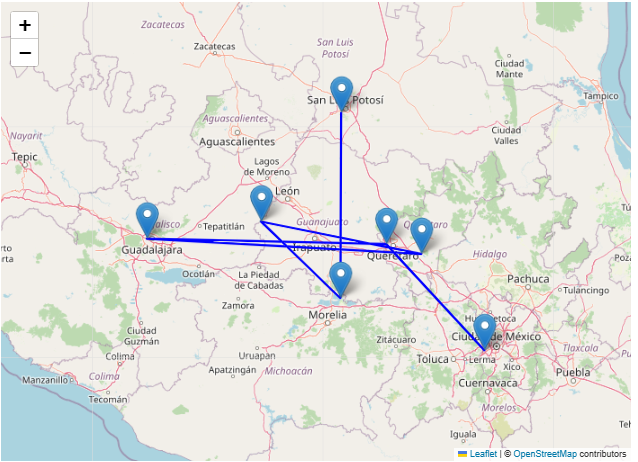
    display\_map(selected\_data)

**Fig 5.3: Streamlit\_app.py code snippet**

The code inserts radio button for selecting the event name from a list, creates a drop down for selecting Key events notes and also inserts a time based slider that allows the user to have a time based interaction with the displayed map. Not only this, by clicking on the marker points, details of latitude, longitude time and event that occurred can also be visually displayed for the user to see.



**Fig 5.4: Streamlit\_app.py showing the a historical event (Battle of Jutland)**



**Fig 5.5: Streamit\_app.py showing a historical event (Aurora sightings in Mexico)**

**5.6 Integration and Testing**

Integration testing was done to ensure easy communication

Between the database and the Streamlit app. Functional testing was performed to verify the performance of core functionalities such as data retrieval, visualisation, and user interaction. Not only this, two events were used in the app to assess the responsiveness and scalability of the system under various load conditions.

**5.6 Debugging and Optimisation**

During the implementation process, debugging and optimisation efforts were carried out to correct any issues or bottlenecks encountered. To optimise the system's performance and efficiency, the code was restructured without changing its original functionality (code refactoring) while also ensuring that the code could achieve query optimisation by the user during visual display.

**5.7 Summary**

In summary, the visualisation web app development, Streamlit\_app for displaying and simulating historical events involved the development of a robust backend database system and a highly interactive user-friendly interface that helps in better visualisation of historical events compared to existing systems. The design approach enabled the delivery of a high-quality system that meets the needs of historical simulation effectively.

# Evaluation / Testing

The streamlit app retrieves data from an AWS-hosted database. Due to a time constraint, user testing could not be carried out. However, to evaluate and test its performance, various aspects were considered and tested by myself in collaboration with my supervisor. These include functionality, performance, reliability, and cost implications.

**6.1 Functionality Testing**

Upon running the command line **streamlit run streamlit\_app.py** on the Anaconda command prompt, the check was carried out to ensure that the app accurately displays historical events on the map as contained in the cloud-based database. Checks were carried out to ensure that when key events are selected or the time-based slider is used, the map.

**6.2 Performance Testing**

The response time when retrieving data under high latency bandwidth was evaluated. The time it takes to load historical events onto the map was measured as a maximum of three seconds.

**6.3 Scalability testing**

Initially, only one event (Battle of Jutland) was used to simulate a historical event. However, to test the app's scalability, another event, an aurora sighting, was added to the simulation. The fact that the app was able to display the aurora sighting event by uploading the same data fields used for the first events and upgrading the Streamlit app code shows that the app is scalable and can take up more events to display.

**6.4 Reliability Testing**

At the initial stage of testing, the app, at times, refuses to load. However, after a few tweaking, the app started functioning reliably without crashing or experiencing unexpected errors.

However, if the internet network is poor, the app does not load. As the database is hosted in the cloud, the issue of database unavailability was never experienced.

**6.5 Cost Implications Testing:**

Since the database is hosted in the cloud, the only associated cost is the cost of a subscription to AWS Web services.

However, if this is compared with the associated possibility of the cost that will be incurred in data gathering all over again as a result of system failure or corrupted hard drives when stored locally, then storing the information in the cloud is more cost-effective and safer.

**6.6 Compatibility Testing:**

After running the Streamlit app command on the Anaconda prompt (streamlit run streamlit\_app.py), the local URL generated, <http://localhost:8501> was copied and pasted on the Mozilla Firefox browser, Chrome browser while still running on the edge browser.

The app was observed to be compatible with all three browsers and to respond well across them. Not only this, but the radio buttons also tagged ‘Choose an event’, ‘View Options’ and ‘Select Time’ were all tested and found to be fully functional when used on the Mozilla Firefox browser, Chrome browser and Edge browser.

# 7 Description of the final product

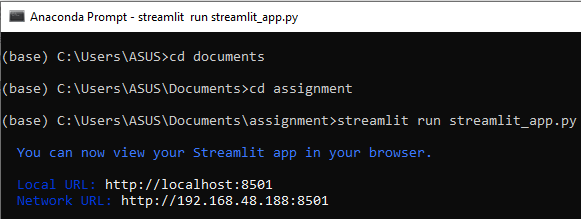
The final software application, streamlit\_app.py, provides a user-friendly interactive interface for displaying historical events on a map. Users can see a visual display of historical events, including the latitude, longitude, time, year, day, and month, and also get an exact description of their occurrence at selected event marker points on a map bounded by a box with zoom-in and zoom-out functionality.

**To run the streamlit\_app.py application, the following steps are followed:**

* Visual Studio code and Anaconda must be installed on the local machine.
* The completed code is placed in a location (e.g. Documents) on the local machine.
* The Anaconda command prompt is opened and the command:

**streamlit run streamlit\_app.py**

is executed.

******

**Fig 7.0: Running the streamlit\_app.py command on the Anaconda prompt**

Once done, the streamlit\_app.py software comes up. The software application has a left sidebar that has radio buttons tagged ‘choose an event’. Under this is displayed the events whose data has been loaded into the application.

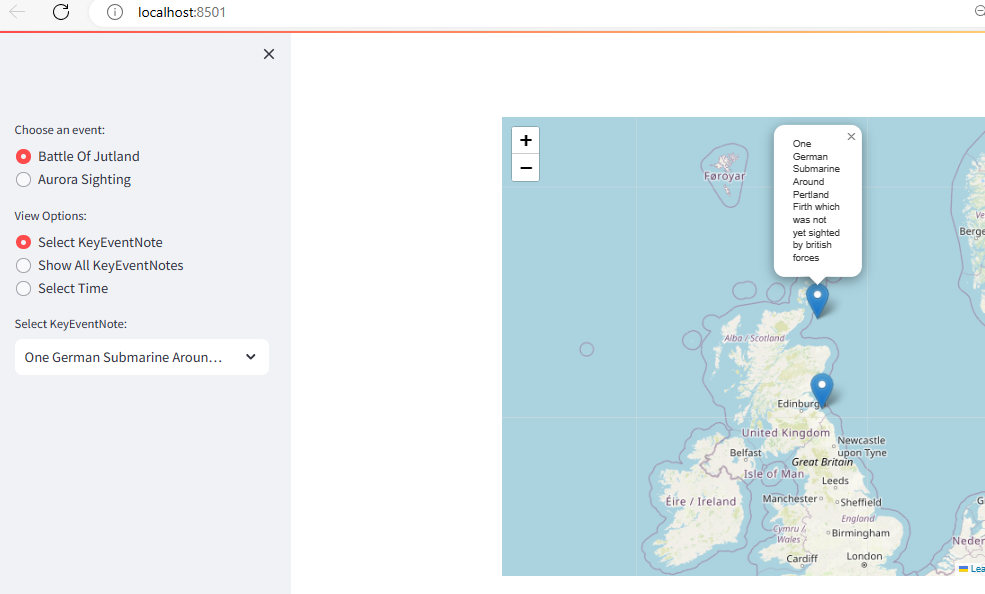
Also, there is radio button that is tagged ‘View options’. Under these there are radio buttons tagged ‘Select KeyEventNote’, ‘Show All KeyEventNotes’ and ‘Select Time’. There is also a drop-down bar that is tagged ‘Select KeyEventNote’.

By selecting any of the events, and either of the View Options buttons, the corresponding selection is displayed. The drop-down bar that shows key events in history can also be clicked to get a drop-down of any event based on users choice.

For interacting with the map, the app has a time-based slider that can be dragged from left to right and vice versa to get a simulation of different key events on the map described with marker point.

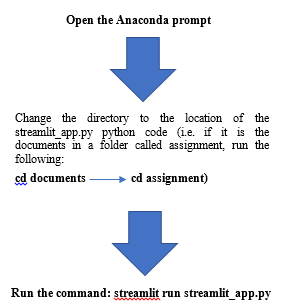
The time-based slider helps to interact with the different events loaded into the app.

Also, there is a drop-down menu that can be clicked to see various interesting events stored in historical events.



**Fig 7.1: Selection of an event with a radio button under view option**

To use the app, the flow chart in Fig 7.2 is applicable.

****

**Fig 7.2. Flowchart for running the streamlit\_app.py app**

# 8 Summary and Conclusions

This project served as a means of creating historical visualisations using database as a Tool. The following were provided by the project:

* A means of having a Historical Data Repository: This project serves as a reliable repository for historical data. It encompasses two different events, several timelines and geographical locations. The database system used helped to carefully catalogue and store historical event data in a database in a very structured format. This ensures easy access and retrieval of historical data from anywhere in the World.
* Interactive Simulation Interface: Through the use of the streamlit\_app.py app simulation interface, users can interact with historical events in a visually pleasing manner. This enables a dynamic exploration of historical events and scenarios.

The interactive maps generated from the user interface interactive features display key historical events, helping users interact visually with historical contexts.

* Historical Event Visualisation: The project provides visual representation of Historical events through the representation of events with markers representing significant locations and time periods on the map interface.

Users can click on the marker points to get detailed information about the latitude, longitude, time, and description of the event that occurred at selected points.

* Query and Analysis Tools: It is possible to query the visual display without any form of query language knowledge. By moving the time-based slider or selecting an event from the drop-down menu, query to get what happened based on users’ choice is displayed on the map. This gives a useful query and analysis tools, which empowers users to extract insights and patterns from a large volume of historical dataset.

Users can use the app to carry out custom queries to filter and analyse key occurrences in historical events using several criteria, such as time and description on the drop down button/bar.

If I had the opportunity to do this project over again, I would use both the Streamlit framework and Leaflet.js, comparing the benefits of each technology, as I believe I will be able to add the Play button to my simulation aside from the slider that I used.

In conclusion, the "Databases as a Tool for Historical Simulation" project offers sophisticated yet user-friendly software for historical exploration and simulation. Using AWS cloud-based database technology and a highly interactive user interface, the project assists users in exploring history, visualising historical events, and, most importantly, interacting with history, thereby gaining valuable insights from historical data.

# 9 Future Work

This project demonstrated the efficiency that can be achieved by using databases as a tool for historical simulation as it enables the creation of a user friendly and highly interactive platform for querying, visualising and analysing historical events. Despite these, several avenues still exist for future work. This includes:

Feature Expansion: Currently, the developed app provides functionality for inputting, storing, querying and visualising historical events. Future work could include advanced search capabilities and the authentication of users before been allowed to use the app to improve user experience and increase functionality.

Enhanced Testing and Evaluation: Performance of more extensive usability testing such as using a larger number of users at the same time will help to better refine the platform and identify areas for improvement.

Evaluation of Performance and Scalability: The need to get a larger data set and larger number of users at the same time to evaluate the app performance as data and users volume grows is important. Things that can be used to assess this includes: database performance, query execution times, and the ability of the platform to accommodate increased data volume and user load adequately.

To this end, I plan to continue working on this project to address the recommended areas of improvement. Not only this, I intend to collaborate with researchers and history enthusiasts to further improve the field of historical analysis and simulation as if this is not done as a matter of urgency, valuable historical data that are still kept on spreadsheets and in paper format may be lost over time.

## Acknowledgements

# I acknowledge the contributions of my supervisor, Iain, my classmates, my family members and, most importantly, God Almighty.

# References

Bolt, John, et al. “Battle of Jutland.” Porttowns.port.ac.uk, 24 June 2016, porttowns.port.ac.uk/jutland-map/index.html. Accessed 30 Jan. 2024.

Ellis, Peter. “Animated Map of World War I UK Ship Positions.” Free Range Statistics, 5 Dec. 2020, freerangestats.info/blog/2020/12/05/wwi-ships. Accessed 30 Jan. 2024.

Ellis, Peter. “UK Royal Navy Ship Locations in World War I.” Www.youtube.com, 5 Dec. 2020, www.youtube.com/watch?v=MKuMbWAZ1cA. Accessed 30 Jan. 2024.

González-Esparza, J. A., and M. C. Cuevas-Cardona. “Observations of Low-Latitude Red Aurora in Mexico during the 1859 Carrington Geomagnetic Storm.” Space Weather, vol. 16, no. 6, June 2018, pp. 593–600, <https://doi.org/10.1029/2017sw001789>.

Imperial War Museums (2018). Battle of Jutland Timeline. [online] Imperial War Museums. Available at: <https://www.iwm.org.uk/history/battle-of-jutland-timeline>.

Q, Liang, et al. Centralised Data Storage and Efficient Retrieval: Enhancing Simulation Models through Database Integration. 2021.

R., Garcia-Mila, et al. Integration of Databases with Simulation Models: A Review of Current Practices and Future Directions. 2019.

Taylor, T. (2003). Historical Simulations and the Future of the Historical Narrative. [online] Available at: https://quod.lib.umich.edu/j/jahc/3310410.0006.203/--historical-simulations-and-the-future-of-the-historical?rgn=main [Accessed 10 May 2023]

Wiedmann T. 1999. Database oriented modelling with simulation micro functions. In proceedings of the 1999 winter simulation conference, ed. P.A. Farrington, H. B. Nembhard, D.T. Sturrock, and G.W. Evans, 586-590: Society of computer simulation.

Weinberg, E. (2018). Using Geospatial Data to Inform Historical Research in R. Programming Historian. [online] Available at: https://programminghistorian.org/en/lessons/geospatial-data-analysis [Accessed 30 Jan. 2024].