Mass Shooting Digital Library

CS 4624

Virginia Tech, Blacksburg VA

11/30/2023

Authors: Tai Nunez, Harris Naseh, Arjit Singh, Bhavya Patel, Dong Xiao
# TABLE OF CONTENTS

Mass Shooting Digital Library........................................................................................................... 1
CS 4624................................................................................................................................................. 1
Abstract.................................................................................................................................................. 3
**Introduction**....................................................................................................................................... 4
  Problem Statement................................................................................................................................. 4
  Motivation............................................................................................................................................... 4
  General Approach................................................................................................................................. 5
**Requirements**..................................................................................................................................... 6
**Design**.................................................................................................................................................. 9
**Implementation**.................................................................................................................................... 14
**User Manual**....................................................................................................................................... 22
**Developer Manual**............................................................................................................................ 29
**Lessons Learned**................................................................................................................................ 37
  Timeline................................................................................................................................................. 37
  Problems............................................................................................................................................... 39
  Solutions............................................................................................................................................... 40
  Future Work For Subsequent Semesters............................................................................................... 40
**Who Did What**.................................................................................................................................... 42
**Acknowledgments**.............................................................................................................................. 45
**References**......................................................................................................................................... 46
Abstract

In light of the escalating prevalence of mass shootings in the U.S., there is an urgent need for a structured digital repository to centralize, categorize, and offer detailed analyses of these events. This project aims to develop a comprehensive website functioning as a digital library. This library will house mass shooting objects where each object symbolizes a specific mass shooting event, elaborating on who, what, when, where, why, and how. The website's central features will include the ability to visualize and compare various mass shooting incidents, facilitating a broader understanding of trends, patterns, and anomalies. Users will be able to explore the data via geographic visualizations, timelines, and more, providing an immersive and informative experience. Underpinning the platform, our backend system will utilize Python, Flask, and MongoDB, ensuring robust data collection and management. This data includes information fields, URL sources associated with each event, and more. On the front end, technologies like NextJS, React, and Javascript will drive the user interface, supported by essential libraries such as React Chrono and Leaflet.js for advanced visualization. Deployment will be executed via Firebase or AWS for the frontend and Heroku for the backend. Two primary user categories have been identified: general users, who can view the data, and administrators, who can modify the contents. Ensuring the integrity of the data input, admin access will be safeguarded by authentication processes. In summary, this digital library emerges as a timely and crucial initiative in response to the rising tide of mass shootings in the U.S. This project aims to provide comprehensive insights into the tragic events that have marked the nation. Beyond its functional capabilities, the digital library strives to improve understanding, awareness, and ultimately, change in the narrative surrounding mass shootings.
Introduction

Problem Statement

In the past few decades, the United States has witnessed an alarming escalation in the occurrence of mass shootings, creating an urgent societal issue that commands incessant attention and action. Navigating through these tragic incidents, a diverse audience ranging from scholars and policymakers to social workers and concerned citizens seeks comprehensive, organized, and easily accessible information. With countless resources and news articles dispersed across the web, piecing together comprehensive and accurate narratives of these incidents becomes a challenging endeavor, hindering effective analysis, discussion, and policy-making in addressing mass shootings. Currently, users do not have the ability to search and compare different mass shooting events in an aesthetically pleasing and easy to use format.

Motivation

The need to comprehend, document, and strategize responses to mass shootings has culminated in a demand for a centralized repository of data and insights pertaining to such events. Understanding the Who, What, When, Where, Why, and How (WWWWH) of each incident not only serves to memorialize the events and victims but also equips various stakeholders with pivotal knowledge that may guide future research, policymaking, and preventive strategies. A singular, robust digital platform that amalgamates details, insights, and visual representations
related to mass shootings can thereby act as a linchpin in fostering a coherent and collective approach toward mitigating this devastating societal problem.

**General Approach**

In an endeavor to bridge this informational gap and facilitate multifaceted analyses of mass shooting events, our project, the "Mass Shootings Webpages Digital Library," is conceived. This digital library, orchestrated and managed by a proficient team under the guidance of Mohamed Farag (mmagdy@vt.edu), aims to serve as a centralized, comprehensive repository of detailed information about mass shootings in the United States. The platform will not only store and organize data regarding each event's WWWWWWH but also furnish users with visualization tools to compare and analyze different incidents. The team will create a website which users can peruse to lookup and compare different shooting events based on a variety of different fields such as geographic location, number of victims, date of the event, and more. This website will interface with our database of shooting events and urls via an application programming interface (API) working with our backend code.
Requirements

The digital library should function as a comprehensive website that provides extensive data about mass shootings in the US. A distinct object, termed as a 'Shooting Object', will symbolize each mass shooting event. The core fields for each Shooting Object will include answers to the Who, What, When, Where, Why, and How of the event. Additional attributes may encompass details like the weapon used, a list of victims, etc. Every Shooting Object will be stored within the MongoDB collection named events.

Each shooting event should be associated with relevant URLs. These URLs will be stored in a separate MongoDB collection, ensuring a one-to-many relational linkage between a Shooting Object and its corresponding web resources. The system should provide APIs to facilitate data collection from web pages as input and to streamline the process of object creation, retrieval, and manipulation within MongoDB.

There should be a clear distinction between 'users' and 'admins.' While users possess viewing rights, admins have the added capabilities of modifying and adding content to the digital library. An authentication mechanism should be established for admin access, using a login and log out method authenticating with an email and password.

Upon accessing the digital library, users should be greeted with a homepage listing the array of mass shooting events, along with a map view of all the shooting events as data points. The platform should facilitate event searches based on various criteria like date, location, number of casualties, etc. A significant feature to be implemented is the capability for users to visually
compare different shooting events. The visualizations could encompass geographical representations, timelines, and graphical comparisons of event attributes. Admins require a dedicated portal or page wherein they can add new events and delete existing ones.

The digital library should offer intuitive mechanisms to visually represent shooting events. This employs a map (indicating the locations and dates of the shootings) and timeline (detailing the chronology of events). An efficient system should be in place to ensure smooth navigation, easy comparison, and detailed exploration of individual events. Tools like React Chrono and Leaflet.js could be leveraged to enhance the user experience.

The integration between the backend (comprising Python, Flask, and MongoDB) and the frontend (utilizing NextJS, React, and other tools) must be seamless. This ensures that the digital library is both dynamic and responsive. For deployment purposes, we should deploy the finalized product on a Linux server.

Clear documentation, encompassing both user and developer manuals, should accompany the final deliverable. Reporting mechanisms should be integrated into the system, enabling periodic performance evaluation, bug tracking, and feature enhancement suggestions. Roles and responsibilities, as outlined for the participating members, should be adhered to ensure smooth project progression. Adherence to the established timeline, which specifies clear milestones at least twice a month, is essential to ensure timely project completion.
In conclusion, the Mass Shootings Webpages Digital Library project aims to offer a comprehensive platform detailing mass shootings in the US. The resultant digital library, with a dynamic backend, interactive frontend, and immersive visualization capabilities, is anticipated to help understand and evaluate mass shooting events both now and in the future.
Design

The homepage of the application is shown below in Figure 1. There is a welcome section with a map visualization where all the events in the database are shown as data points and clickable for more information, along with a search feature where the user can search for events and also filter based on further information. There is also a section for recent shootings that gives general information about them, along with a map preview. At the top, there is a navigation bar to log in for admins, go to visualizations or timeline, and go back to the homepage.

Figure 1. Homepage

The admin submit data page is shown below in Figure 2. This is where an authenticated admin can contribute data to our MongoDB database with more shooting events. There are fields for every crucial part of the shooting event that needs to be known, such as who did it, what happened, when did it happen, where did it happen, why did it happen, and how did it happen. There is also a section for the admin to provide the relevant URLs to support each event. The
user can also add additional fields if they deem necessary information such as, weapon used, casualties, etc.

![Submit Information Form]

**Figure 2. Admin submit form prototype**

The visualizations page is shown below in **Figure 3**. This is where a user can view the data and statistics visualized in an easy to understand manner. There are interactive graphs that compare two events and the statistics between them such as casualties. The user can also search the events to compare. This is accompanied with a general overview of the two events below.
The timeline page is shown below in **Figure 4**. The timeline includes all events in the database in a chronological format. For each event, there is a brief description of the incident, including the date, location, number of casualties, and in some cases, the identity of the perpetrator and their actions. The event points on the timeline are also clickable, allowing the user to navigate to a separate page for more information.
Figure 4. Timeline of Events Page

The login page is shown below in Figure 5. This login page is used to authenticate a user and become an admin.

Figure 5. Login Page
The single event page is shown below in **Figure 6**. This page shows full details about the shooting event, along with the additional fields and the urls associated with the shooting.

**Figure 6. Single Event Page**
Implementation

Website Overview:
The website is conceived to systematically track mass shooting events in the United States. By combining functionalities that cater to both data input and retrieval, the platform offers a centralized source of verified and comprehensive details about such incidents.

Technology Stack:
Backend: Python with Flask and MongoDB.
Frontend: Next.js.
Implementation Details:

1. Data Collection & Association:
Admin Submission:
A dedicated 'Submit Page' enables admins to enter comprehensive details about a mass shooting event. These details typically encompass variables such as date, location, casualties, among others. Alongside the event details, admins can also input URLs associated with the event. These URLs can serve as references, news articles, or other relevant sources that provide additional context or verification. A validation process ensures the accuracy and completeness of the submitted data before its storage. Once the user is ready to submit the frontend will send an Axios POST request to the backend.
I. Data Storage & Linking:

The backend processes the submitted details and stores them in MongoDB. Each shooting event in the database is linked to its associated URLs. This linking creates a one-to-many relationship between the event and the URLs, where one event can have multiple associated URLs. This data structure allows efficient storage and retrieval of both events and their related URLs. For the overall structure and flow of our system, see **Figure 7**.

The steps are as follows:

1. User selects events that they want to view.
2. The frontend sends an Axios request to the events API
3. Use the search function in the python backend to search MongoDB
4. The backend will send a JSON with all the events that match the search.
5. The frontend will show event using JSON

![Figure 7. System design flow](image-url)
2. User Roles and Authentication:

**User Categories**

The system classifies users into two categories: Regular Users and Admins.

Regular Users can view and retrieve shooting event details and their associated URLs.

Admins have enhanced privileges, allowing them to input new data or delete existing records, and add associated URLs.

**Admin Authentication**

Admins need to sign in to access their elevated privileges, which is done through Google Firebase.

A robust authentication process ensures that only verified admins can input or modify data.

The design flow diagram of the authentication feature is shown in **Figure 8**. The steps are as follows:

1. User navigates to the login page and enters the password.

2. The password is then verified through our authentication process, and if successful, the user is now an admin and is redirected to the admin page.

3. On this page, the admin can make changes such as adding new events or modifying them.

   This is done through API requests such as POST, PUT, DELETE.

4. The frontend then calls the API through our EventsAPI.

5. API interfaces with our backend, which handles all the logic of what needs to be done.

6. Performs queries on our database to perform relevant CRUD operations.

7. Database returns the updated data back through the backend
8. Backend communicates this through our API

9. Frontend receives API response with updated information to show to user/admin

Figure 8. Authentication system design flow

3. Data Retrieval & Presentation:

The platform provides functionalities for users to view details of each shooting event and its associated URLs. Next.js, the frontend framework, ensures that the presentation is intuitive, user-friendly, and responsive. Users can click on an event to view its details and see a list of linked URLs that provide further information or context. The comparison feature and how a user can access events based on certain parameters is shown below in Figure 9.

The steps are as follows:

1. User navigates to the visualizations page
2. User enters in the search criteria for the first event to be compared
3. The frontend then calls the API through our EventsAPI, passing in the user’s parameters as url parameters.
4. API interfaces with our backend, which interfaces with MongoDB via the `pymongo` library.

5. Database returns the data through the backend which responds to our HTTP request.

6. Backend communicates this through our API.

7. Frontend receives API response with updated information to show to show the page.

8. User repeats the process for the second event to compare.

9. Visualization uses the information in the events returned from the API to provide an aesthetically pleasing graph.

![Diagram showing the flow of information from backend to frontend through API and database]

**Figure 9. Compare feature design flow**

4. **Visualization**

   The visualization page serves as the interactive representation of mass shooting events, allowing users to grasp the extent and distribution of these incidents both over time and across geographical locations. By harnessing the capabilities of advanced JavaScript libraries, the page...
offers a multi-faceted view into the data, emphasizing casualties, timelines, and geographical contexts.

I. Chart Visualization (Chart.js):

Purpose: The primary goal of using Chart.js is to illustrate the casualties associated with each shooting event.

Implementation: Chart.js, a popular and responsive charting library, was integrated to create dynamic and interactive charts.

Data from the MongoDB backend, specifically the number of casualties from each event, is fed into Chart.js to generate these visualizations. Various chart types (like bar, line, or pie charts) can be used, depending on the specific representation desired. For example, a bar chart might display the number of casualties per event, while a line chart could show trends over time.

II. Timeline Visualization (VerticalTime.js):

Purpose: To showcase the chronology of shooting events, emphasizing the sequence and frequency over time.

Implementation: VerticalTime, a modern timeline component for React, is employed to create a chronological display of shooting incidents. Each entry on the timeline corresponds to a specific shooting event, sourced directly from the MongoDB backend. Users can navigate through time, accessing details of each event as they progress along the timeline.

The process for a user navigating through the visualizations and viewing the graphs and timelines is shown below in Figure 10.
The steps are as follows:

1. The user initiates the process by submitting a GET request to access the digital library.

2. Once inside the digital library, the user can click on the "Visualization" tab.

3. This action triggers the backend to fetch data through the "Events API". This data will likely be related to various events (in this context, possibly mass shootings) stored in the

4. After retrieving the data from the MongoDB database, the backend processes it and sends it back to the frontend as an API response. The frontend then displays this data to the user, likely in the form of a timeline or some visual representation.

5. Users can then interact with this displayed data. They can modify the order of shooting events to observe changes or trends over time.
Figure 10. Visualization design flow

III Geographical Visualization (Leaflet.js):

Purpose: To spatially represent the locations of shooting incidents, providing geographical context and emphasizing the spread or concentration of events.

Implementation:
Leaflet.js, a leading open-source JavaScript library for interactive maps, is incorporated to render detailed maps. Each shooting event from the MongoDB backend is geotagged and plotted on the map using markers or other visual cues. Users can zoom, pan, and interact with the map to explore different regions, with each marker offering additional details upon being clicked, such as the date, number of casualties, and any other pertinent information. The process of a user viewing a map with the shooting is shown in Figure 11.

Figure 11. Interactive map data flow
User Manual

Our target users are those who want to browse the mass shooting data that happened in the United States. We’ve built an online digital library for users to access. By now, our digital library is able to realize functions such as: search, interactive map, events visualization, events comparison, timeline, upload, delete, authentication.

**User environment requirements:**

- Basic computer manipulation
- Network connection
- A web browser
- Reading
- Writing

We have put the majority of computation jobs on our backend server so that the user doesn’t need any tech stack environments that the servers have such as Node.js, MongoDB, React, etc. All they need is a network connection for the data exchange between the client and the server, and a web browser that supports HTTP, html, javascript, CSS to display data correctly.

Users may also need the ability to read and manipulate their computer to interact with our digital library.

**Tasks supported:**

- Search mass shooting by keywords/date/location.
- Interact with the map in our homepage.
● Visualize and compare the mass shootings in the histogram.
● View all shootings chronologically in a timeline.
● Log in to our library as an administrator to manage (submit, delete) shooting events.

**Tutorial on use:**

Firstly, the user needs to type the URL of our digital library in the URL box of the browser to get to our digital library. For now, we don’t have a public domain name so we use a local proxy “localhost:3000”

![Image](localhost:3000)

**Figure 12. Type the URL in the URL box**

The user will be greeted with a map visualization with all the mass shooting events stored in the database. Each of these red dots are clickable and will send the user to the event information page for said event. The user can also use the navigation bar to navigate to different parts of the website such as log in, visualizations, timeline, etc.
For searching the mass shootings, users need to input the keywords/time/location into specific boxes, then click on the search button to do a search.
Explore important statistics and information about mass shooting events.

Search...

Advanced Search

**Advanced Search**

mm/dd/yyyy

Location

Search  Reset

**Figure 15. Make a search in home page**

For the visualization, users can click on the Visualization button on the home page, then they will be redirected to our visualization page. Over there, they can see the histogram of our stored mass shooting objects.

**Figure 16. Step 1 for visualization**

The user can also compare two events side by side by selecting any two events.
Figure 17. Step 2, visualization page

We also have a timeline page which shows all the mass shooting events in chronological order. Each event is accompanied by its summary. Each event is clickable and will take the user to the event details as shown in Figure 18.

Figure 18. Timeline page
We also provide a page for administrators to log in to access some privilege functionality. Administrators can access this page by clicking the “Login” button on the navigation bar.

![Login interface](image)

**Figure 19. Login interface.**

For administrators, we allow them to delete a specific mass shooting object on the page of that event by clicking the “Delete event” button on the top left corner.

![Delete event](image)

**Figure 20. Delete event**
For administrators, we allow them to submit the mass shooting object via the submit page.

Figure 21. Submit page
Developer Manual

Getting Started Tutorial:

To ramp up for this project, you will need to make sure you have the required executables installed on your local machine.

Recommended: Install a GUI code editor such as VSCODE which we use and recommend, you can download here: https://code.visualstudio.com/download.

Install node version 18.8.X or later, which you can download here:
https://nodejs.org/en/download

Install python version > 3.6.X, which you can download here:
https://www.python.org/downloads/

Make sure you have python installed. You can check by opening your terminal and running

```
python -v or python3 -v for mac.
```

You will need to navigate to the repository link:

https://git.cs.vt.edu/tainux/capstonemassshootingwebpage

and clone it onto your machine. To do this, follow the instructions listed below.

If you’re planning on using your local machine, open a terminal of your choice such as git-bash or zsh and run:

```
git clone https://git.cs.vt.edu/tainux/capstonemassshootingwebpage.git
```
Or if you’re planning on developing remotely using the secure shell (SSH) run:

```bash
git clone git@git.cs.vt.edu:tainunez/capstonemassshootingwebpage.git
```

![GitHub Repository](image)

**Figure 22. GitHub Repository**

Now that you have cloned the repository, you need to install the required dependencies for both the backend and frontend.

For the backend, it is helpful to create a python virtual environment to keep track of the versions required for all dependencies needed in order for the backend to function.

**Virtual Environment (VENV) Setup:**

1. From the root directory of the repository, navigate to the server directory, by running:

   ```bash
cd server
   ```
2. Create venv by running: `python -m venv venv` or `python3 -m venv venv` (if using mac).

3. To activate your virtual environment, open your terminal and run:

   `source venv/Scripts/activate` or `source venv/bin/activate` (if using mac).

4. Also make sure to be in the venv python interpreter by clicking your version in the bottom right corner if using VSCODE and making sure the path is set to venv/Scripts/python.

You should now be in venv, to install all backend dependencies, run:

`pip install -r requirements.txt`, or `pip3 install -r requirements.txt` if using mac.

Make sure to update requirements.txt if installing new dependencies to the project by running:

`pip freeze > requirements.txt`, or run: `pip3 freeze > requirements.txt` if using mac.

**NodeJs Installation:**

You’ll need to have NodeJS installed so you can use npm commands. Follow the install instruction from the driver and you should not have to do anything else with Node directly.

**Installing Dependencies:**
For the frontend, navigate to the frontend directory by running: \texttt{cd client/frontend} from the root directory of the repository. In this directory, you should see a file titled “package.json” which contains all the required dependencies needed for the frontend to compile and function.

- Install all of these dependencies by running: \texttt{npm i} from the client/frontend directory.

Check that your frontend works by running: \texttt{npm run dev} from the client/frontend directory. This will start a local proxy on your localhost that you can use for frontend development.

Check that your backend works by starting the api, navigate to the ‘server’ directory and run: \texttt{python app.py} or \texttt{python3 app.py} if using mac OS. This will start the API which the frontend can then hit to make HTTP requests.
Once all the files have been cloned, you will see the following within an IDE Explorer:

- The project is broken up by two overhead files: `client` and `server`. Client contains all the files for the frontend and server contains files related to the backend.

- Starting off with the `client` folder you will see: `.next` and `node_modules`. These are directories related to a Next.js project, a popular React framework. `.next` typically contains compiled and optimized production files, while `node_modules` houses dependencies.

- The majority of the code for the frontend is contained in `src/app`. Here you will find a file containing all the individual React components shown.

![IDE File Explorer](image)

**Figure 23: IDE File Explorer**

`src/app` is the main source directory of the project. Within this folder you will see:

- components: Houses several JavaScript and TypeScript components, including ones named logo, MassShootingEvent, Navbar, NavItem, and Clicked Event. It also has subdirectories named images, submit, and visual.
• There are also some styling (CSS) files and a TypeScript file (layout.tsx).

Additional you will see some configuration and meta files:

• .eslintrc.json is for configuring ESLint, a tool for identifying and reporting on patterns in JavaScript.
• next-env.d.ts, next.config.js, and tsconfig.json are configuration files for TypeScript and Next.js.
• package.json and package-lock.json are related to npm packages and their versions.

Server: backend portion of the project, written in Python with app.py and mongoInterface.py:

• __pycache__ is a directory Python uses for storing compiled bytecode files.
• venv is a virtual environment that is being used for Python dependencies.
• .env is used for storing environment variables.
• requirements.txt will list the Python dependencies required.
• README.md files: These Markdown files are present in both the main directory and the server directory, used to provide documentation on the project.

The mongoInterface.py file within the server folder sends an API request, and the JSON request is as follow:
Figure 24. JSON API Response

In this JSON file, we can see that the API returns mass shooting events, indexed by ID. It shows casualties, date, event name, location, motive, perpetrator, and summary. All of these values are then used for Mass Shooting Event components as seen in the frontend side.

Authentication:

We use Google Firebase Authentication. This is located in the src/app/firebase.tsx file. This file holds all the configuration information for setting up the authentication. The files that use firebase are login.tsx, logout.tsx, and submit.tsx. We use react-firebase-hooks as the framework for dealing with authentication features such as logging in, logging out, and checking credentials. On the backend, we use firebase_admin package to check the token and credentials for a user trying to perform admin actions. We use email/password authentication currently, which is held
in the console. The username for the admin account is mshootings@gmail.com and the password is Mshootingslibrary23.

Troubleshooting:

NodeJS: If you have an older version of NodeJS make sure you are on v18.17.x or greater.

Mac issues: If you are on mac make sure that when you installed the python liberties you used pip3 and not pip like for windows.

Python Version: Ensure you have python 3.7.x or higher installed so it can work with flask properly.
Lessons Learned

Timeline

9/14: Repository Setup, System Design, and Wireframe Drafting

1. Set up a dedicated repository on GitLab for version control and collaborative development.
2. Establish initial system design, outlining the architecture of the frontend, backend, and database.
3. Draft wireframes to visualize the basic layout and user interface of the website.

Deliverables:

1. GitLab repository with initial commit, share repository with all team members
2. System design document including data flow diagrams for each major feature.
3. Wireframe sketches or digital drafts.

9/28: Prototype Development and Database Setup

1. Develop a prototype utilizing pymongo and other relevant python libraries to interact with MongoDB.
2. Set up MongoDB cluster, designing the schema for shooting events and urls.
Deliverables:

1. Functional prototype demonstrating basic data interaction with MongoDB.
2. Database schema and MongoDB cluster setup.

10/15: Frontend Development

1. Develop the frontend of the webpage with hardcoded event objects to showcase the basic layout and functionalities.
2. Establish the React component structure, and initiate the design for visualization and comparison features.

Deliverables:

1. Webpage frontend with hardcoded event objects showcasing basic functionalities.
2. React component hierarchy diagram.

10/30: Backend-Frontend Integration and Visualization

Deliverables:
1. Integrated backend and frontend with basic visualization features such as map, comparison, and timeline.

11/14: Production Deployment, Final Presentation, and Report Submission

Deliverables:

1. Live website deployed to production.
2. Final presentation slides and speaking roles assignment.
3. Final project report submission.

Problems

1. During the initial phase of our project, setting up a unified development environment emerged as a notable challenge. Our team comprised members utilizing different operating systems (OS) which included varieties of Windows, and macOS. Furthermore, discrepancies in the versions of essential executables like Python and Node.js added layers of complexity to the setup process. This scenario was not anticipated and thus, it consumed more time than initially allocated in our project timeline. The different versions and OS environments introduced inconsistencies that could potentially lead to varied behavior of the application during the development and testing phases, thereby risking the quality and timeline of the project.
Solutions

1. To counter the issue, we decided to unify our development environment to ensure consistency across all setups. We chose versions of Python and Node.js that were well-supported and widely used, to minimize potential discrepancies. We held regular sync-up meetings to address any setup issues promptly. During these sessions, team members shared their experiences, problems faced, and solutions found which fostered a collaborative environment for problem-solving. We also made extensive use of version control systems, specifically Git, to track changes and ensure that everyone was working with the same code base and environment configurations.

Future Work For Subsequent Semesters

**Crowdsourced Data Collection:** Create a portal for eyewitnesses or individuals close to the incident to contribute information, photos, or videos which can be vetted and included in the database to provide a more comprehensive view of each event.

**Public Forums and Discussions:** Introduce a discussion forum or community section where users can discuss, share experiences, and provide support to one another.

**Educational Resources:** Include a section dedicated to educational resources, expert analyses, and articles discussing the prevention and impact of mass shootings.

**Predictive Analysis:** Incorporate predictive analytics tools to analyze trends and potentially predict the likelihood of future incidents based on historical data and other relevant factors.
Public API: Develop a public API that allows external developers and organizations to access and utilize the data for their own analysis and applications.

More admin control: Create a page for the admin to manage events. This page would list all the events and the admin can choose to edit or delete an event from this page.
Who Did What

Tai Nunez:

- **Lead:** Backend Python development using the pymongo library to interface with MongoDB
  - For each shooting event build MongoDB Document containing: Name of event, Geographic Location, Name of Place, Number of Fatalities, Date of Incident, Shooter Name, as well as supporting additional user-defined fields.
  - Two MongoDB collections: “Events” and “Urls”
  - Add capability to retrieve individual shooting events, or to retrieve all at once for comparison purposes.
  - Add filtering so users can retrieve events based on date, location, number of fatalities.

- **Help:**
  - Frontend Development: Create dynamically rendering details page and a full stack delete feature so admins can manage events.
  - Debug teammates issues along the way

Bhavya Patel:

- **Lead:** Python Flask API endpoint development, Lead on Reports and VTechWorks submission
  - Create API to collect and handle user/admin data
  - Lead on authentication system using Google Firebase
Works on backend and frontend, forms for logging in/logging out, route protection and token checks in endpoints, conditional rendering

- Organize report deadlines with the group

**Help**

- Frontend (snackbars and toast notifications for error and success messages), debugging for other errors in development

**Arjit Singh:**

- **Lead:** Home page layout, map visualization, clicked event details, presentation
  - Formatted overall layout of Home Page
  - Created MassShootingEvent components to display on homepage
  - Made ClickedEvent components to display event attributes as well as optional additional fields added in by users.
  - Used Leaflet to create the main map visualization with all MassShootingEvents as well as mini maps for each event shown below the main map.
  - For the main map visualization, made each red dot clickable to route users to the ClickedEvent details page.

- **Help:**
  - Requirements, Design

**Harris Naseh:**

- **Lead:** Visualization page, timeline page and taking notes during meetings
○ Develop the visualization page to visualize the number of casualties for shootings.
○ Developed comparing feature on visualization page
○ Created page to show shooting events in chronological order
  ■ Find robust and nice looking react library to display the information
○ Take note during meetings to keep track of work completed and work remaining

• Help:
  ○ teammates debug issues that came up

Dong Xiao:

Lead: Submit page, Login page, authentication

• Developed a submit page for administrators use
  ○ Support input of different fields of events.
  ○ Support url verification.
• Develop a login page for the basic authentication functionality.
• Develop a basic authentication function for security.
  ○ Very basic authentication that redirect users to another page

Help:

• Help with the debugging of the map.
• Help with the route protection on submit page.
Acknowledgments

Client: Dr. Mohamed Farag

- Email: mmagdy@vt.edu
References


