# Table of Contents

Table of Contents 1

Table of Figures 2

Executive Summary 4

1. Introduction 5
   1.1 Objective 5
   1.2 Client 5
   1.3 Constraints 5
   1.4 Roles 6
   1.5 Scope 6

2. Requirements 7

3. Design 9
   3.1 Features from Current Works 9
      3.1.1 Histopedia 9
      3.1.2 Histography 10
      3.1.3 Infoplease 11
   3.2 Past Front-End Designs 13
   3.3 Early Design 14
   3.4 Final Design 21

4. Implementation 25
   4.1 Initial Approach 25
      4.1.1 Front End 25

5. Testing/Evaluation/Assessment 33
   5.1 Testing / Evaluation Approach 33
   5.2 Instructions for User Testing 34
      5.2.1 Guided Instructions 34
      5.2.2 Unguided Instructions 40
   5.3 Questions / Feedback 40

6. Users' Manual 41
   6.1 Home Page 41
   6.2 Index Page 41
   6.3 Creating a New Timeline 42
   6.4 Adding Events to a Timeline 42
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.5 Navigating a Timeline</td>
<td>43</td>
</tr>
<tr>
<td>6.6 Editing a Timeline</td>
<td>44</td>
</tr>
<tr>
<td>6.7 Comparing, Merging and Contrasting Timelines</td>
<td>44</td>
</tr>
<tr>
<td>7. Developer's Manual</td>
<td>46</td>
</tr>
<tr>
<td>7.1 Database</td>
<td>46</td>
</tr>
<tr>
<td>7.2 Languages and Frameworks</td>
<td>46</td>
</tr>
<tr>
<td>7.3 Modifying the database</td>
<td>47</td>
</tr>
<tr>
<td>7.4 Adding a new page</td>
<td>47</td>
</tr>
<tr>
<td>7.5 Migrating to MySQL</td>
<td>47</td>
</tr>
<tr>
<td>8. Lessons Learned</td>
<td>48</td>
</tr>
<tr>
<td>Acknowledgements</td>
<td>50</td>
</tr>
<tr>
<td>References</td>
<td>51</td>
</tr>
</tbody>
</table>
### Table of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1.1.a</td>
<td>Histropedia Timeline Display</td>
<td>10</td>
</tr>
<tr>
<td>3.1.1.b</td>
<td>Histropedia Expanded Event</td>
<td>11</td>
</tr>
<tr>
<td>3.1.2.a</td>
<td>Histography Main Timeline Display</td>
<td>12</td>
</tr>
<tr>
<td>3.1.3.a</td>
<td>Infoplease Main Timeline Display</td>
<td>13</td>
</tr>
<tr>
<td>3.2.a</td>
<td>Wireframe for Timeline</td>
<td>14</td>
</tr>
<tr>
<td>3.2.b</td>
<td>Wireframe for Add Event</td>
<td>15</td>
</tr>
<tr>
<td>3.3.a</td>
<td>Home Page Concept</td>
<td>16</td>
</tr>
<tr>
<td>3.3.b</td>
<td>Autofill Searching</td>
<td>17</td>
</tr>
<tr>
<td>3.3.c</td>
<td>Timeline Concept Page</td>
<td>18</td>
</tr>
<tr>
<td>3.3.d</td>
<td>Mobile Timeline Concept</td>
<td>19</td>
</tr>
<tr>
<td>3.3.e</td>
<td>Timeline Merge Concept</td>
<td>20</td>
</tr>
<tr>
<td>3.3.f</td>
<td>Login Page Concept</td>
<td>21</td>
</tr>
<tr>
<td>3.3.g</td>
<td>Add Event Concept</td>
<td>22</td>
</tr>
<tr>
<td>3.4.a</td>
<td>ToE Home Page</td>
<td>23</td>
</tr>
<tr>
<td>3.4.b</td>
<td>Login/Register Modal</td>
<td>23</td>
</tr>
<tr>
<td>3.4.c</td>
<td>Index Page</td>
<td>24</td>
</tr>
<tr>
<td>3.4.d</td>
<td>Timeline Page</td>
<td>24</td>
</tr>
<tr>
<td>3.4.e</td>
<td>Timeline Tile</td>
<td>25</td>
</tr>
<tr>
<td>4.1.1.a</td>
<td>Rough draft timeline overview page</td>
<td>26</td>
</tr>
<tr>
<td>4.1.1.b</td>
<td>Rough draft new timeline</td>
<td>27</td>
</tr>
<tr>
<td>4.1.1.c</td>
<td>Rough draft timeline display</td>
<td>28</td>
</tr>
<tr>
<td>4.1.1.d</td>
<td>JSON data format example</td>
<td>29</td>
</tr>
<tr>
<td>4.1.1.e</td>
<td>JSON Format Properties 1</td>
<td>30</td>
</tr>
<tr>
<td>4.1.1.f</td>
<td>JSON Format Properties 2</td>
<td>30</td>
</tr>
<tr>
<td>4.1.1.g</td>
<td>Date object properties</td>
<td>31</td>
</tr>
<tr>
<td>4.1.1.h</td>
<td>Text object properties</td>
<td>31</td>
</tr>
<tr>
<td>4.1.1.i</td>
<td>Media object properties</td>
<td>32</td>
</tr>
<tr>
<td>4.1.1.j</td>
<td>SQL to JSON Conversion</td>
<td>33</td>
</tr>
<tr>
<td>5.2.1.a</td>
<td>Home Page</td>
<td>35</td>
</tr>
<tr>
<td>5.2.1.b</td>
<td>Account Register Page</td>
<td>36</td>
</tr>
<tr>
<td>5.2.1.c</td>
<td>New Timeline Page</td>
<td>36</td>
</tr>
<tr>
<td>5.2.1.d</td>
<td>Empty Timeline Page</td>
<td>37</td>
</tr>
<tr>
<td>5.2.1.e</td>
<td>Populated Timeline Page</td>
<td>37</td>
</tr>
<tr>
<td>5.2.1.f</td>
<td>New Event Page</td>
<td>38</td>
</tr>
<tr>
<td>5.2.1.g</td>
<td>Timeline Edit Page</td>
<td>38</td>
</tr>
<tr>
<td>5.2.1.h</td>
<td>Selecting Timeline to Merge Page</td>
<td>38</td>
</tr>
<tr>
<td>5.2.1.i</td>
<td>Merged Timelines Display</td>
<td>39</td>
</tr>
<tr>
<td>5.2.1.j</td>
<td>Comparison of Timelines Page</td>
<td>40</td>
</tr>
<tr>
<td>5.2.1.k</td>
<td>Contrast of Timelines Page</td>
<td>40</td>
</tr>
<tr>
<td>6.1.a</td>
<td>Home Page</td>
<td>42</td>
</tr>
</tbody>
</table>
Figure 6.2.a: The Index Page 43
Figure 6.3.a: Making a New Timeline 43
Figure 6.4.a: Adding Events to a Timeline 44
Figure 6.4.b: New Event Form 44
Figure 6.5.a: Timeline Display Page 45
Figure 6.6.a: Editing a Timeline 45
Figure 7.1.a: Database Relational Diagram 47
Figure 7.2.a: Flask Framework Example 48
Figure 7.2.b: Knightlab Timeline Base 49
Executive Summary

The goal of The Timeline of Everything is to provide a proof of concept for an alternative way for intuitively and quickly viewing history. Using the idea of “walking through history”, a timeline is used where the user can move step by step through the events of key points in history. In addition, users can create their own timelines utilizing already created events or creating their own new events. Each event in a timeline can have a multimedia element associated with it alongside text. Since there are multiple viewpoints of events, three ways to compare timelines were created. First, a simple merging of the two timelines where all events from both would be displayed; second, a contrast of the timelines showing the differences between the two; and lastly, a comparison showing the similarities. Upon doing one of these, a new timeline is generated showing the events, thus allowing an easy way to modify this new timeline.

Because this proof of concept might be used to show potential investors, a visually appealing site was the priority for this project. In the future, high traffic optimization and security need to be higher priority goals. The database is currently utilizing SQLite due to its ease of use, however for future implementation, MySQL or similar database should be used. Due to their similarities migrating from one to the other should be fairly easy. Another future goal is utilizing multiple date/time systems; by using a hh:mm:ss or frame system a movie maker could use this system to storyboard and arrange scenes. Using custom time systems, book, tv, and movie lovers could create timelines for their favorite stories.

The site is currently hosted on http://timeline.cs.vt.edu/. And should be available until March of 2020 unless it is extended.
1. Introduction

1.1 Objective
The goal of this project is to provide a platform where users can display data from history and then analyse the data through comparing, contrasting, and “side by side”. This project is serving more as a proof of concept so the work can continue to be worked upon either by the current members of the team, or any teams to follow up in the future. That being said, the intended finishing point for our group is to provide a platform that can crowdsource information from events throughout history so timelines can be generated and analysed by users. The team itself will only be providing a few example timelines in order to display the functionality of the site and gather interest from the public over what is possible with the Timeline of Everything. One possible future addition would be allowing the use of fictional data (on a separate site / subsite) as well as fictional calendars as seen in popular movies, tv series, games, and books.

1.2 Client
Our client is Dr. Eric Van De Velde. He has mainly provided the idea for the project as a means of displaying information from various sources to compare and contrast their differences in order to get a better understanding of how events in history are related and the scope of time between them. Once providing the base idea for the project, Eric has mainly left the implementation of said project in our hands, while providing suggestions for ideas to inspire and “supercharge” the system. Listening to what inspired the project in his own words, we decided upon a dynamic timeline that allows users to add in their own events will be the best way of presenting the information in a clean and legible manner.

1.3 Constraints
The constraints of the website are to provide a website that anyone can intuitively pick up and create a timeline. The timelines must be accessible to all users, along with the events that make up those timelines. Users will have the option to search for events that have already been created by other users. If that event satisfies the user’s desire for detail and credible sources, the user can utilize that event in their own timeline. Otherwise, the user is free to make their own events with their own sources to add to their timeline. The team is also intending to provide base code that is legible and well documented so any team in the future that works on the project and continues the work will have an easy time...
picking up where we left off. With the possibility of another team taking over the project in the future, we also wanted to make sure the transition was as easy as possible. For this reason, the site will be hosted on the Virginia Tech website, cs.vt.edu, so there are no hosting fees or accounts to be transferred over in the future.

1.4 Roles

The roles of our group members are fairly flexible such that any member who is available will assist another member with a task. With that in mind, the general focus members have taken is shown in the table below.

<table>
<thead>
<tr>
<th>Group Member</th>
<th>Roles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matthew Jehnke</td>
<td>Database creation and management, data population for demos.</td>
</tr>
<tr>
<td>Jon Conlen</td>
<td>Project management, graphic work.</td>
</tr>
<tr>
<td>Sourabh Shetty</td>
<td>Front end implementation.</td>
</tr>
<tr>
<td>Yifan Zhou</td>
<td>Front end implementation.</td>
</tr>
<tr>
<td>Huanbo Fu</td>
<td>Back end implementation</td>
</tr>
</tbody>
</table>

1.5 Scope

The Timeline of Everything is meant as a proof of concept to encourage future work and participation of the public in the idea of the product. In the same way that wikipedia has become such a major influence in society off of crowdsourcing information, the client Eric shared that same desire to get people involved in our project. For this reason, there is no intention to scrub existing sites for information to populate the database, but instead provide a functional, intuitive, and visually appealing site that will encourage people to participate in the growing platform.
2. Requirements

The main requirement of this project was to create a proof of concept that would serve as a way to attract users to the idea of a way to document and compare history in an easy and visually appealing way. We did not have much of a specified list of requirements for the final product beyond that so we began with the idea of using a crowdsourced timeline and went on from there. Without the clear goal line from the beginning, ambitions grew as the project progressed and left us with more ideas than we had time to implement.

As it stands, we wanted to produce a site that would allow users to intuitively go to, make an account and begin viewing, creating and manipulating timelines and have it feel natural. The site was to be made in the style of sites like Wikipedia that are basic in nature but allow for great expansion and potential based on the interest of the community surrounding it. The original intended audience around the timeline was thought to be the historian community whether it be professional or casual. After getting into the design of the site, we realized it could be much more with allowing users free reign over the content they supply the timelines. Users can document literature, wars, movies, video games, music, and whatever else they could think of that would benefit from visualizes the data over a given time and have the ability to compare the data to other timelines for reference of the crossover between events.

With this base ability in place, our goal now leans towards populating the database with interesting timelines that will get the attention of the general public and get feedback to improve the site overall and grow its user base. Some of the aspects moving forward we would like to see moving forward is the ability to allow for public and private timelines. Currently only the user who originally created a timeline is able to edit that timeline in any way. Other users are allowed to see it but it does not quite have the full potential of crowdsourcing information in a way such that wikipedia does where anyone can edit any page. We would like to leave it optional to make a private page in case a user would like to utilize the site for a personal project or study they have interest in and don’t want the information tampered with, but we would also like to see timelines being opened up the public and gather the full knowledge of the user base into the timelines to grow them to sizes unreasonable for one person to be expected to implement and maintain.

We would also like to see advancements in security. None of the members of the current team have extensive prior experience with security which makes it difficult to
know all of the ways that the current program can be exploited. We also have the ability for a user to include their sources for events within the description of their events, but we would like to see a more dedicated way of saving the source material for the more technical or academic timelines added into the site. This would provide an easy way for people to reference the material quickly to decide if the information is scholarly and reliable.
3. Design

3.1 Features from Current Works

One of the key tools we used in determining how we want to design a timeline were other timeline sites. There is a large selection of timelines to examine, even ones that attempt to serve a similar purpose to our project. Each of them have benefits and drawbacks that we can use as a learning experience and figure out what we think will make the best experience for our site.

3.1.1 Histropedia

One of the most valuable tools in designing our ideal timeline site was Histropedia. The site contains many of the features we find to be most useful for a dynamic timeline, but also some that can be refined in some way.

![Histropedia Timeline Display](image)

Figure 3.1.1.a Histropedia Timeline Display

To start off with, we will discuss the features that we plan to implement from Histropedia. One of the most important details is that the timeline is dynamic. The user is able to zoom in and out to show varying spans of time and get more detail of the events you are examining. Each event can be selected to show more information as shown in Figure 2 below. Histropedia also features autofill searching, so when a user is trying to look up a timeline, the search bar will autofill suggestions that may pertain to what the user is looking for. This includes both events
and timelines and offers the user the ability to add events to an existing timeline or create their own. The user can also merge or replace entire timelines if they would like to compare events between the two.

With the features included in Histropedia that we feel are necessary in an ideal timeline creation tool, we would like to refine the overall design by including our own touches. By default, Histropedia overlaps events which is typically necessary because images are included on the event on the timeline. We would like to avoid overlapping events in order to make the timeline appear less cluttered and easier to digest. In addition, the events with overlapping ranges are displayed using small bars and force the user to traceback to the event bubble to figure out what they are looking at. The expanded information also appears on the sidebar blocking a portion of the timeline and exclusively links to wikipedia, Twitter, Books, Tv and Film, Music and Youtube. While it will be an option of the user on the Timeline of Everything to use these sources and mediums as a source, we don’t want it to be the only option. Instead we are looking to include the timeline on the bottom 25% of the screen with the expanded information and picture included on the top of the screen. This will allow the user to visualize the timeline uninterrupted while also being able to read up on specific events they would like to know more about and half full freedom of the source of that information.

3.1.2 Histography

A more visually appealing and modern approach to a timeline can be seen through Histography. With first impressions, the timeline looks impressive and certainly gets the users attention.
Some of the features that we found useful about Histography include the options to filter the timeline by various categories as seen on the left of figure 3 above. The site is also colorful and creative in the transition it makes between shifting timeline and time periods for the current timeline. Refining features such as these that make the site feel more alive and modern are the current stretch goals depending on progress towards the end of the term.

The main issues we encountered when using Histography include the difficulty of scanning through individual events shown on the timeline. Each event is given a very small dot which you can hover over to expand. Through personal use and user testing with a diverse background of participants, there was an overall consensus that the site was more visually appealing than practical in this approach and it would not be an ideal implementation for our own work. This site in particular maps out the events from all of the entries in wikipedia which displays more like a heat map of activity over the years of history. This does not align with the goals of our project since the user cannot compare and contrast individual events easily or create their own timelines.

3.1.3 Infoplease

The main draw of Infoplease is the simplicity of its design. The page is not cluttered, there are no unnecessary elements, it simply displays the information it is trying to get across. While there is value in this, the site also benefits from having static information and imagery to display. Not knowing beforehand what the gaps in time between events will be and how much space will be able to be afforded to a given event makes it difficult to take an approach such as the one in
Infoplease. More consistency in design for a dynamic approach can be useful by displaying less information of the event on the timeline, while moving that information to another section the user can access easily. In the case of Infoplease, this could have been achieved through the sides of the page since they utilized a slim vertical timeline. The space was mainly empty instead or used for site navigation.

Figure 3.1.3.a: Infoplease Main Timeline Display
3.2 Past Front-End Designs

Instead of simply drawing a static image, a basic hard coded HTML timeline was created to get an idea of how the user should interact the timeline. To do this, a dynamic, browser based visualization library, vis.js, was used to generate the timeline and modal buttons were added for comparing timelines and adding events. The use of modal was chosen here since it allows the user to stay on the same page with no need of reloading the timeline. Upon clicking one of the buttons, for example the “Add Event” button, the modal would open allowing the user to search for and add an event.
The main issue with this design was that the vis.js library was a huge, almost 60,000 line, script, and documentation on visually customizing was sparse. In order to produce a viable product in a timely matter, another way to visualize the timeline was needed or a sacrifice to the visual appeal was needed. The clear and obvious choice was to change the system used to visualize the timeline.

3.3 Early Design

With the requirement given by our client, we came up with the design as following:

1. Home page
   We decided to go with a Wikipedia style home page due to the clean simply design. Due to the large amounts of data that will be stored, using an autocomplete search function allows an easier method of access for the user. If user accounts are created with private timelines, a user homepage would have a "my timelines" page listing out all of the timelines that user created. This would not be efficient if this grows to the size of Wikipedia's 5.8 million content pages. (at the time of writing according to the statistics page on Wikipedia)
2. Timeline page
The header on the timeline pages includes a search bar, when user types the name of the timeline, the search bar will return possible results. If the user can’t find the timeline, then they can create a new one (if they have an account) by clicking “Create a New Timeline” in the drop down menu.
In addition a register / login section is displayed in the far right corner. Upon logging in this would change to the users name and would redirect the user to their homepage. A horizontal timeline was chosen as it allows for more details to be displayed as well as being better able to display overlapping events. A vertical timeline was considered since it would work better on mobile devices, however as events overlap, less horizontal room is available for the details of the event.

Use of the timeline needs to be simple and intuitive. Clicking on an event, or using the arrows to the left and right of the details of the event to move to the next / previous event, will present the details of said event, this can include text, pictures, sources, as well as an embedded YouTube video. Sources to the event will link the user to a new tab/window.
For useability on smaller devices, a mobile version was created where the timeline filled a majority of the page and the details of the event are more vertically aligned. Consideration for the analysis interface will need to be considered and might need to be changed depending on user feedback. The current idea for mobile is to keep them in the menu drop-down.
3. Timeline Merge
The initial approach is that the user can choose which timeline they want to merge to the current one by searching. The search bar will return some possible timeline, the user can only choose the timeline in the drop down menu. By clicking it, the data of the two timelines will be shown on the current page, but it will not create a new timeline in our database. The user can merge up to 5 timelines.
4. User account
The user account is only used to ensure that the user has access to edit or create a timeline, and the website only allows the administrators to manage the timeline. So we don’t need a user registration page, instead we create the user account in our database, and only need a login page.

Figure 3.3.e: Timeline Merge Concept
5. Edit Timeline
By clicking the “Edit” button on the timeline page, it will direct the user to the “Edit Page” (if the user has the access):
The events will be displayed in the order of start date, and the user can edit or add events in this page.

3.4 Final Design

The overall structure of the site stayed fairly consistent over the progression of the project. The main changes came with functionality and aesthetic appeal of the pages. We decided to keep
the simple home page in the style of Wikipedia for a streamlined look as seen below.

We changed the user account register and login pages over to modals so that they would neatly drop down and disappear giving the site a more fluid feel rather than redirecting to a quite empty page.

With the addition of the plain home page that allows for the user to search through all existing timelines, we also made an Index page which allows the user to see individual card views of some of the available timelines along with the search bar at the top if the user is looking for a specific timeline. This page also features the button for creating new timelines. Each card view of a timeline features the timeline name, description, and background picture for that timeline. It also has a pencil icon in the lower right hand side of the card view to allow the user to edit a timeline assuming they are the original creator of said timeline.
With the timeline itself being the main focus of the project, that page in particular has remained fairly consistent throughout the continuation of the project since it was the first feature we invested time into having working as we intended. The main change for the timeline page itself is the inclusion of buttons on the top left allowing the original creator of a timeline to add events, compare, contrast and merge timelines with one another to compare their data.

Shown in more detail in section 5.2.1 of this report, these buttons will allow you to manipulate timelines as listed above. The Merge button will allow the user to combine every event listed in each of the two timelines they are combining into a single timeline for further analysis of those events and how they line up. This will exclude duplicate events in order to not clutter a timeline.
with duplicated if they feature similar topics or moments in history. The compare feature will allow the user to compare two timelines, resulting in a timeline that contains the overlap in events between the two timelines. This can help to quickly cut down on irrelevant events that might clutter your view when you only want to see the overlap. In a similar manner, the contrast button will allow a user to create a new timeline consisting of the events that differ between two timelines in case they want to be able to see where two similar timelines might divert and find inconsistencies in documentation. In all three of these cases, the user who called the merge, compare or contrast on the timelines will be given edit permissions on the new timeline created so they can further manipulate data within said timeline.

One of the newer features to help a user search for a more broad topic of timelines rather than a specifically named one, there are now tags. Tags are implemented using the description of the timeline where the '#' character is parsed to create a tag of the word proceeding it. On the index page, these tags can be searched for or directly clicked on one of the card views to direct the user to other timelines containing that tag.

![List of Timelines](image)

Figure 3.4.e: Timeline Tile
4. Implementation

4.1 Initial Approach

4.1.1 Front End

Initially, we chose to use React JS as our front end framework. We chose that simply because two group members had some experience on it. However, when we started to implement the timeline design, we realized that it will be horrendously difficult to implement a timeline by ourselves. Therefore, we looked for some timeline frameworks that are close to our design and we found Timeline JS. Unfortunately, this framework doesn't work with React. At that point, we have two choices, either find a way to convert Timeline JS into something compatible with React or give up the progress we already made on React and choose a different framework that will work with Timeline JS.

Then, we split our front end group into two. One group tries to figure out how to convert Timeline JS into a React component, and other group tries to migrate what we already down into Python. It turns out that even we tried to contact with some professionals, the first group still don't know how to convert thousands of lines of code in Timeline JS into a React component. Eventually, we decided to work on the Python.

Within Python, we chose the Flask framework, since it is the best one to get off the ground quickly. It's also a good choice for a small team of developers to quickly prototype an application, which was essential since we only had approximately three months to complete this project.

![Figure 4.1.1.a: Rough draft timeline overview page](image-url)
This is our current home view. Here we can see a list of all the timelines that are currently available to view. Once the search functionality is implemented, the homepage will be replaced with dashboard of the user’s own timelines.

![Timeline of Everything](image)

**Figure 4.1.1.b: Rough draft new timeline**

This form lets the user create a new timeline, providing a title and a summary of the timeline being created. This creates a master timeline, one that events can be added to, and eventually be merged with other timelines as well.
This is the view that is displayed when a timeline is successfully created. We provide the data to timeline.js in a JSON format, which it uses to render the timeline shown in the picture. The specifics of the JSON are provided in the documentation of timeline.js, and are explained below.
This is a sample of the JSON format as defined by the documentation on the timeline.js website.
The format is defined as follows:
As of now we have no plans of using the “eras” or the “scale” parameters.

The “slide” object used for the “title” and the “events” parameters itself has a lot of parameters, but the only ones relevant to our project are the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>events</td>
<td>Yes</td>
<td>A JSON list of &quot;slide&quot; objects (see below)</td>
</tr>
<tr>
<td>title</td>
<td>No</td>
<td>A &quot;slide&quot; object (see below)</td>
</tr>
<tr>
<td>eras</td>
<td>No</td>
<td>A JSON list of &quot;era&quot; objects (see below)</td>
</tr>
<tr>
<td>scale</td>
<td>No</td>
<td>Either human or cosmological. If no scale is specified, the default is human. The cosmological scale is required to handle dates in the very distant past or future. (Before Tuesday, April 20th, 271,821 BCE after Saturday, September 13 275,760 CE) For the cosmological scale, only the year is considered, but it’s OK to have a cosmological timeline with years between 271,821 BCE and 275,760 CE.</td>
</tr>
</tbody>
</table>

Figure 4.1.1.e: JSON Format Properties 1

As of now we have no plans of using the “eras” or the “scale” parameters.

The “slide” object used for the “title” and the “events” parameters itself has a lot of parameters, but the only ones relevant to our project are the following:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_date</td>
<td>Yes, except for title slides</td>
<td>A &quot;date&quot; object (see below)</td>
</tr>
<tr>
<td>end_date</td>
<td>No</td>
<td>A &quot;date&quot; object (see below)</td>
</tr>
<tr>
<td>text</td>
<td>No, but recommended</td>
<td>A &quot;text&quot; object (see below)</td>
</tr>
<tr>
<td>media</td>
<td>No</td>
<td>A &quot;media&quot; object (see below)</td>
</tr>
</tbody>
</table>

Figure 4.1.1.f: JSON Format Properties 2
For dates, since we are using a historical context, to start with we’re only dealing with individual days. For now, we’re not going to be dealing with individual hours, minutes or seconds of the day on which the event happened.

Hence, the relevant properties of the “date” object is as follows:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>year</td>
<td>Yes</td>
<td>A number. Don’t use commas. BCE years should be negative numbers. Don’t use the letters “BC”, “BCE” or any others.</td>
</tr>
<tr>
<td>month</td>
<td>No</td>
<td>A number from 1-12 (Javascript experts don’t outsmart yourselves: Timeline corrects for Javascript’s strange use of “0” for “January”, etc.)</td>
</tr>
<tr>
<td>day</td>
<td>No</td>
<td>A number</td>
</tr>
</tbody>
</table>

Figure 4.1.1.g: Date object properties

Text for any of the event or title slides are described as “text” objects, defined in the following format:

<table>
<thead>
<tr>
<th>Name</th>
<th>Required?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>headline</td>
<td>No</td>
<td>Any text. HTML markup is OK. Blank is also OK.</td>
</tr>
<tr>
<td>text</td>
<td>No</td>
<td>Any text. HTML markup is OK. Blank is OK. Not used for era objects.</td>
</tr>
</tbody>
</table>

Figure 4.1.1.h: Text object properties
“Media” objects are used to define a picture for any slide, with additional information such as alternate text in case of broken images, or for accessibility purposes, titles, credits, and even links to websites, which can be used to add sources for any historical information.

<table>
<thead>
<tr>
<th>Name</th>
<th>Required?</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Yes</td>
<td>In most cases, a URL (see <a href="#">media type documentation</a> for complete details).</td>
</tr>
<tr>
<td>caption</td>
<td>No</td>
<td>Any text. HTML markup is OK.</td>
</tr>
<tr>
<td>credit</td>
<td>No</td>
<td>Any text. HTML markup is OK.</td>
</tr>
<tr>
<td>thumbnail</td>
<td>No</td>
<td>A URL for an image to use in the timenav marker for this event. If omitted, Timeline will use an icon based on the type of media. Not relevant for title slides, because they do not have a marker.</td>
</tr>
<tr>
<td>alt</td>
<td>No</td>
<td>An alt tag for your image. If none is provided, the caption, if any, will be used.</td>
</tr>
<tr>
<td>title</td>
<td>No</td>
<td>A title for your image. If none is provided, the caption, if any, will be used.</td>
</tr>
<tr>
<td>link</td>
<td>No</td>
<td>Optional URL to use as the href for wrapping the media with an &lt;a&gt; tag.</td>
</tr>
<tr>
<td>link_target</td>
<td>No</td>
<td>Optional target to be associated with link if used.</td>
</tr>
</tbody>
</table>

Figure 4.1.1.i: Media object properties

Since our data is stored as SQL, one of our challenges is to convert this data into the required format. For that we currently use JavaScript to format it according to the requirements:
Our plan is to subsequently do this conversion before rendering the page to the user.
5. Testing/Evaluation/Assessment

5.1 Testing / Evaluation Approach

Testing and evaluation of the current design of the site and its functionality will be done in two tiers. First, a team member will take a design either created by themselves or another member and implement it into the site. Once they have the function operational, a team member that was not involved in the direct creation of that feature will test it. This will consist of stress testing the functionality to see if an edge case can be found to pull out bugs in the operation of the feature. This allows for an unbiased evaluation of what the application is actually doing compared to what you expect it to do since the person testing was not directly involved in the creation of the feature.

Once the current version of the site has been stress tested for bugs by the developing team, the site will move into user testing. A group of users with a wide variety of experience will be chosen. This will range from classmates in the Multimedia/Hypertext course, underclassmen in Computer Science, and friends and family of varying age groups with no development experience whatsoever. This will provide the team with a range of opinions and approaches to the uses of the site that can give a good impression of the general market that will be accessing the site. Users will be taken through the process of creating two separate timelines. One will be with a direct set of step by step instructions for creating a timeline. The second will be a topic for a timeline and have the user self populate the timeline in order to see how intuitive and accessible the application is. After the hands on experience, the test groups will be asked to answer a few questions and give any feedback they have for the application.
5.2 Instructions for User Testing

5.2.1 Guided Instructions

1. Go to timeline.cs.vt.edu and click register on the top right corner.

Figure 5.2.1.a: Home Page
2. Enter a username and password and log into your account.

3. You will be redirected back to the home page, where you can see the available timelines. Make a new timeline with the “New” button on the right side of the page.

4. Make the title “[your username]’s Guided”.

5. Set the summary to “This is a guided experience for testing and evaluation.”

6. You can also set a background image for the timeline by pasting an image link in the “Background Image” section. This is most easily done by google searching for something like a flower, and opening the image in its own tab. Copy the link into the image section.
7. Save the timeline. You will be redirected to your timeline page with no events currently in it.

![Image: Empty Timeline Page](image1.png)

**Figure 5.2.1.d: Empty Timeline Page**

8. Click the “Add Event” button on the left side of the screen, and search for the “Siege of Yorktown” event. Select the event and add it to your timeline.

![Image: Populated Timeline Page](image2.png)

**Figure 5.2.1.e: Populated Timeline Page**

9. Now add another event with the same button as step 8, but this time make a new event using the button at the bottom labeled “Create a new Event”. Set the title to “Virginia Tech Founded”. Set the summary to “The date that Virginia Tech was founded.” Set the start and end dates to 06/20/1872. Set the image to be the HokieBird. You can leave the source blank, this is just to give credit to the site you retrieved the image from. Click
“Save” and you should now have a second event on your timeline.

Figure 5.2.1.f: New Event Page

10. From the timeline page you can traverse events with the arrows on the side of the screen or directly clicking the event on the timeline.
11. At any time you can click the edit button on the right side of the screen to edit the timeline itself or the events within it.

Figure 5.2.1.g: Timeline Edit Page

12. Next to the Add Event button on the top left, you will find the Compare/Merge button. Click that and search for the “War of Independence” timeline.

Figure 5.2.1.h: Selecting Timeline to Merge Page
13. Select it and merge with your timeline. You will see the resulting timeline consists of both the events from both your timeline and the War of Independence timeline.

![Merged Timelines Display Page](image)

**Figure 5.2.1.i: Merged Timelines Display Page**

14. Return to the home page by clicking the site logo on the top left corner.
15. Select your guided timeline and click the Compare/Merge button again.
16. This time, search for the War of Independence timeline again but click the Compare button.
17. This will result in a timeline of all the events that overlap between the two timelines. In this case it will only be the Siege of Yorktown.

Figure 5.2.1.j: Comparison of Timelines Page

18. Repeat the process, but now use the Contrast button.
19. The resulting timeline will contain all the unique events between the two timelines. In this case it will have the Declaration of Independence and Virginia Tech Founded.

Figure 5.2.1.k: Contrast of Timelines Page
5.2.2 Unguided Instructions

1. Create a timeline and make the title “[your username]’s Unguided”.
2. Add the Virginia Tech Founded event from your Guided timeline and make two new events. The first about Virginia Tech’s football team’s first season and the second about the Foundation of Virginia Tech’s College of Engineering.
3. Compare and contrast this timeline with your guided timeline.

5.3 Questions / Feedback

1. Did you find the site easy to navigate?
2. Did you have any trouble finding events or timelines you were looking for?
3. Did you have any trouble with the timeline compare, contrast and merge tools offered?
4. Did you experience any bugs while using the application?
5. Do you have any feedback regarding functionality or features that should be offered?
6. Other suggestions and comments?
6. Users' Manual

For a step-by-step set of instructions, see the Guided Walkthrough in section 5.2.1 on page 33.

6.1 Home Page

From the home page you will have access to a search bar for all the timelines available along with an index page, register drop down and log in drop down. If it is your first time on the site feel free to register so you can begin making timelines. Be sure to log in afterwards.

Figure 6.1.a: Home Page

6.2 Index Page

Once logged in, you can access the index page. This page presents the timelines in a card view where you can see the title, description and picture of the timelines. You will also have a New button in the top right where you can add a new timeline.
6.3 Creating a New Timeline

Clicking on the New timeline button will open up a page where you can enter the timelines title, summary and a link to the background image. Once you click save, an empty timeline will be made where you can populate it with events.

The site also features tags for timelines. When making the timeline description initially or by editing the timeline description with the edit button on the top left of the timeline page, you can enter any “#tag” in the description and it will automatically be set by the site. Tags are used to better navigate timelines you are searching for to narrow the search to a topic rather than a specific timeline name.

6.4 Adding Events to a Timeline

When on a timelines page, you have an edit option on the top right, and an add event, compare/merge button on the top left. Use the add event button to add new events to your timeline that will be accessible to others when searching for that event.
An existing event made by another user can be used by searching for its name and clicking the add button at the bottom. You can also scroll to the bottom and add a new event. Fill out the form as the sections are labeled when making a new event and click save.

6.5 Navigating a Timeline

The events added will now be a part of your timeline. The timeline can be navigated by either using the arrow keys if on pc, swiping if on mobile, the arrows on the right and left sides of the screen, or directly clicking on the events on the timeline bar at the bottom of the page. The timeline can be zoomed in or out with the plus and minus icons on the left side of the timeline.
6.6 Editing a Timeline

The edit button can be used to edit both the timeline itself and the events that are contained within it. The timeline and events can also be deleted entirely if you wish to do so.

6.7 Comparing, Merging and Contrasting Timelines

The Compare/Merge button on the timeline page can be used to have your timeline interact with other timelines. Clicking the Compare/Merge button will bring up a search for other timelines. Find the timeline you want to interact with your current timeline, select it, and choose either Merge, Compare, or Contrast at the bottom.
Merge will combine the events of your timeline and the selected timeline and put them in a new timeline.
Compare will show only events that the two timelines have in common.
Contrast will only display the events that are different between the two timelines.

7.1 Database

A relational diagram was used to better represent the flow and interconnections between our database, seen above in 7.1.a.

To walk through it, a user table holds the users unique id, their username and their password. The user id is used to keep track of the original creator of a timeline and the only person who has edit permissions for that timeline.

In the timeline table, the id for the timeline, the id of the user who created it, when it was created, its title, summary and a background image link are all stored. A timeline consists of events which have a link of the timeline id and the event id in a relation table called timeline_has. This will be used to keep track of any events that are a part of a timeline.

An event table stores an events id, title, summary, startDate, endDate, image and credit. The image and credit are optional to the user if they decide to provide an image for the event and if they would like to credit a source for said image. The dates are stored as date objects which are discussed more in section 4.1.1.

Along with events, timelines can also have tags. These tags are pulled from the timeline summary where any '#' character is included with a string proceeding it. The tags have an id and the name of the tag itself stored in the tags table. The relation between a tag and a timeline is stored in the timeline_tags table to know which tags are associated with a timeline. These tags are used to filter through timelines based on more broad categories if users are trying to look through more specific timelines without having a direct name, such as a history tag or a war tag.
7.2 Languages and Frameworks

We chose Python as the language to use for the backend of the project. We used the Flask framework for Python since it makes it easier to quickly get started with a project. For instance, the following code creates a perfectly valid web application:

```
from flask import Flask
app = Flask(__name__)

@app.route('/
  def hello_world():
    return 'Hello, World!'
```

![Figure 7.2.a: Flask Framework Example](image)

Their documentation page ([http://flask.pocoo.org/docs/1.0/quickstart/](http://flask.pocoo.org/docs/1.0/quickstart/)) also provided helpful tips for every aspect of development in short, concise ways that made it easier to create the application.

On the front end, we used HTML, CSS and JavaScript. To make the styling cleaner, we also used the Tailwind CSS framework on top of our own CSS. For more information see their website at [https://tailwindcss.com/](https://tailwindcss.com/). While we used our own custom CSS alongside Tailwind, Tailwind can be used without writing a single line of CSS. Another advantage of using Tailwind is that the page can be made responsive by just using different variants of the Tailwind classes. Flask internally uses the Jinja template library to render templates so we made use of that to have a common header and so that we could reuse elements like pop-up modals.

To render the timeline, knightlab’s open-source library [TimelineJS](https://www.knightlab.com/timelinejs/) was used. The default configuration for TimelineJS involves the use of a Google Spreadsheet, but in order to have finer control and make the page more dynamic, we used the more complex JSON option instead. Doing this the timeline data can be stored in our own database, and some pre-processing can be performed before it is passed to the TimelineJS library.
7.3 Modifying the database

The schema.sql is used to initialize the database. This file has all of the SQL commands for creating all the necessary tables as well as inserting all the sample data we want to see at startup. Using this file tables can be edited or new tables can be created. If a table is created or modified, blog.py will then need to be accordingly edited to include any new select, insert, etc statements to account for the changes. Any inserts in schema.sql will need to be updated if editing an existing table. Upon changing this data, the runAndReset.bat should be used to ensure the server doesn’t have any old tables in it. Unfortunately this will delete all timelines that users have added. In the future a convert option may be implemented to fix this issue by editing the instance.sqlite file to the new data.

7.4 Adding a new page

To create a new page, first, create the HTML page. Then to redirect to that page, create a route inside blog.py using one of the other pages as a template. To link to the other page from one of the already created html use href="{{ url_for('name') }}" or similar to redirect to the page for ‘name’. For a more in depth explanation, follow the quickstart guide on the Flask website.

7.5 Migrating to MySQL

Because SQLite and MySQL are very similar switching over to MySQL should be a fairly straightforward process. There are converters to convert between the two, however due to the smaller scale of the project at the current time, manually changing the data is a viable option. Instead of writing out a multi-page how-to here, it will be easier (and more well written) to look
up a how-to on google/stackoverflow/github. Any access to the database in the python files (*.py) will need to be updated with the new MySQL code.
8. Lessons Learned

When working with a team of people who have never met, with varied experience and knowledge of the aspects of building an application. We approached this by first finding out if any members of the team had a preference to any aspect of the application, whether it be the front-end, back-end, design, testing, database or a more well rounded role. It turned out that few people were specialized in any individual task so we ended up splitting the work evenly and agreeing to help each other with any complications that come up over time.

Some of the hurdles that had to be overcome during the development of the application included where we would host the finished product. Considering the opportunity of future teams carrying on the work that this team began, we wanted to have a clean cut way of transferring information without needing to disclose any personal information. The original route of hosting on AWS would have run into this issue since the account hosting the server would need a personal credit card attached to the account. This lead us to look towards hosting our application on cs.vt.edu where we could easily give the information to access the site to a future team and there would be no maintenance costs for the current team to host the project.

One of the first technical problems we as a team encountered was completing autofill suggestions when searching for timelines and events. This essentially came down to a lack of experience with the subject among team members that required further research to get more direction. Overall with a few discussions between team members and a couple of attempts, the autofill ended up getting complete and prepared the team for any instances of implementing features we had no prior experience with into our application.

The team also began with a rough interpretation of what we wanted out of the site in order to get a functional prototype that we would refine later. Much of this came down to having unnecessary, blank and unappealing pages for features such as account login and creation. The team later decided to clean up the site by turning some of these empty pages into smaller modals that could easily drop down and disappear when the user no longer needed to access them. This gave the site a more modern looking site with better flow for the user. With this switch over to modals, new issues had come up with respect to the transfer of data and the updating of the page to recognizing the new user account login. This would change access to editing and creating timelines and change the overall appearance of the user interface. Again, this came down to a lack of experience and after some research and inspection of existing designs, we were able to overcome the difficulties and provide the site we intended to from the beginning.

Overall the team learned a lot about the dynamics of splitting up the work of a full scale project from design phase, to developing, testing and refining. Along with this
came the realization of how important it is to make clear lines as to what everyone is responsible because it is very easy to fall behind on a schedule put in place. Having a constant flow of communication between team members about their progress and intentions for deadlines leaves few last minute surprises and an overall happier and better cooperating team. One thing that truly helps when it comes to catching all the team members up with the current progress on the project was having group meetings in person. This was a more effective means of communicating for the macro progress of the project, while discussing refinements and particular questions was more helpful in a messaging format.
Acknowledgements

We would like to acknowledge Dr. Eric F. Van de Velde for giving us this project and allowing us a lot more free reign than other projects would have allowed. This allowed us to actually think about some fun and interesting use cases and design the project around that.

eric.f.vandevelde@gmail.com

We would also like to acknowledge Dr. Edward A. Fox for teaching the class that gave us the opportunity to work on this project, giving us ideas and directions, as well as setting up the virtual machine used to host the timeline.

fox@vt.edu or edfox0714@gmail.com
References

“The Timeline of Everything.” Histropedia, histropedia.com/.

“Timeline of History.” History, histography.io/.


“Vis.js.” Vis.js - A Dynamic, Browser Based Visualization Library., visjs.org/.

“Quickstart - Flask 1.0.2 Documentation.” Flask, http://flask.pocoo.org/docs/1.0/quickstart/