Requirements Document

2/1/2012

Rev 1.0

Shea Orr

Mark Brownell

Adam Richards

Scott Seward

Table of Contents

1. **Introduction**
2. **Project Description**
3. **Requirements and Functional Specifications**
   - 3.1. Secure website to access timeline software
   - 3.2. Create historical timelines based on different events / lifetimes
   - 3.3. Modify generated timelines to specific display, including images, colors
   - 3.4. Assign permissions to edit / view / create timelines
   - 3.5. Export timelines into image (JPEG) or PDF format
   - 3.6. Embed interactive timelines into web pages
   - 3.7. Save all timelines and event information in database for future retrieval
   - 3.8. Optional Functionality
4. **Constraints and Feasibility Issues**
   - 4.1. Hardware Constraints
   - 4.2. Software Constraints
   - 4.3. Environmental Issues
   - 4.4. Compatability Constraints
5. **Execution Plan**
   - 5.1. Hardware Setup and Survey of Languages and Techniques
   - 5.2. Coordination of Efforts and Design of Implementation Details.
   - 5.3. Testing and Goal Parallelism
   - 5.4. Dynamic Cooperative Development
   - 5.5. Extensive User Testing and Prototyping
   - 5.6. Final Bug Fixes and Optimizations
   - 5.7. Functional Demo, Ready for Presentation and Delivery
   - 5.8. Unscheduled Diversions From Calendar
6. **Appendices**
   - 6.1. HTML5
   - 6.2. Javascript

1. Introduction

Timelines are used for many things from business to designing reports to display a schedule of an item in development to education and showing important invents in history. While business has had various software system designed with their specific goals in mind; education has not received the same level of professional software either being the business timeline software just repackaged towards education or under-designed simplified timeline that does not meet the needs for the education area. This has left instructors and students alike in a situation of using subpar software, or painstakingly creates a timeline from image editing software to get a decent amount of a detail necessary to show visually a set of events. History, unlike business, uses timelines more fluidly by adding more events as they happen or become discovered. It should then be easier to just modify an already created timeline or event rather than producing a new one. Businesses usually need simple timelines that can either be recreated fairly easy, or do not need to live past one presentation. The educational community can benefit from a more dynamic general solution than that. Since anyone will be able to use the Chronoview software, anyone can upload a timeline or event or create a new timeline on the page. This means that timelines can be created from anywhere, to suit any departmental or field related needs and then shared with anyone else who could benefit from it. This flexibility will let this information span across many new domains from elementary education to the most prestigious Universities and beyond.

By developing this web application we aim to facilitate and support the sharing of historical data in a clear, easy to use, highly portable and interactive way so that this information becomes easier to share, educate, and inspire.

This Historical Timeline Module will be able to save our sponsor and anyone who uses it, time and money. Instead of creating a timeline in software that has been tailored for a specific domain, like business, this module will be able to accomplish any timeline related task. It will also provide for further time savings through the ability to modify existing timelines so that the user can tailor one to fit their specific needs. Furthermore, through the use of a central database that keeps record of previously entered events, users can save more time by using existing events to populate their personal timelines without entering redundant information. All of this functionality will be provided to the user in a very friendly and usable environment.

2. Project Description

Cole Mitchell is an instructor in the Philosophy Department at Northern Arizona University. While teaching his classes he would like to use timeline to display information about important events in the world and for specific people and their personal accomplishments. Right now Instructor Mitchell has to choose between spending a lot of time to create these timelines by hand or trying to use an expensive and inefficient timeline program that has been designed more for creating timelines for businesses and business projects and not for historical or educational purposes. Also there is an issue of having to re-add all the information even after using it in another timeline, which can be a very lengthy and tedious task to have to continually add the same information just because you are creating a different timeline that displays some of the same information.

The current way to create timelines is inefficient and time consuming having to use an ill designed program for creating timelines, having to build them painstakingly in paint program/Microsoft Word or doing it by hand without a computer requires much more time than should be necessary for this job. But there is also a possible need for expanding to more than just one single person using the timelines created or the events used. By creating a Timeline web module that will allow multiple people to create timelines and events that can then be shared with others so they do not have to go through the process of adding all these events every time on their own, this will save time for everyone that needs to create timelines. This will benefit the education community as a whole since everyone will be able to share and use the events and timelines produced. Everything is moving more towards the internet and the sharing of data this is a section that has been largely overlooked since the only need to keep timelines to use over again is education.

There are a few products out there that market themselves to create timelines for education, Timeline Maker and SmartDraw are two but they lack the detail for use in education and their layouts are more specifically linked towards business models and projects. Also these two applications lack the ability to share information between people or set up past the timelines themselves and you would need the specific program to view or edit the timelines there. Websites that offer the ability to make timelines on the web are simplistic and in many cases cost money to use, they lack the level of detail to display a date and label on a timeline. Our system will allow for more information that you can view when you select specific events. Also being able to export these timelines for use in PowerPoint and in class presentations will provide a great way to display the data other than using the website. Our solution should be able to:

- Users must be able to upload or add events to the database.
- Create Timelines from these events
- Modify move around and display events clearly on timelines
- Save timelines on the website for viewing later
- To save the timeline as an image that can then be easily used in a presentation
- Bind specific events to time intervals like someone's life span

There are a few timeline generation software packages currently available, but mostly for business and project related purposes, and generally offering limited time spans. The goal of this project is to create a web-based graphical timeline creation and viewing software that is capable of spanning hundreds or thousands of years into the past. These timelines will visually showcase important historical events and figures and their roles throughout the years.

3. Requirements and Functional Specifications

3.1. Secure website to access timeline software

3.1.1. Users will be able to log-in to the website using a username and password. The first set of usernames and passwords will be provided by the ChronoView team, upon which others users can be created via the website.

3.1.2. Visitors will also be able to use a guest account to access the timeline website

3.1.3. Users already in the database, with permission to do so, will be able to create other users and assign access privileges

3.1.4. PERFORMANCE: Users should be able to log in to the website within 6 seconds, normally would be faster but hosting issues might compromise this.

3.1.5. FUNCTIONAL SPECS: The website access will be controlled by a server-side login script, and database table containing user information, passwords and permissions.

3.2. Create historical timelines based on different events / lifetimes

3.2.1. Upload Excel CSV, tab-delimited file containing event information. Users will be able to select the file type they are uploading. Historical files should at minimum contain the following fields: (start_date, end_date, short_label, full_label, description, category, links, priority)

3.2.2. Assign categories to events or lifetimes, included in upload file or afterward

3.2.3. Automatically assign different colors to each category, colors for each category may also be changed at any time. The category color affects events or lifetimes in that category displaying on the visual timeline.

3.2.4. Add lifetimes or events individually to specific timelines, with or without previously uploading file with historical event information

3.2.5. Auto-generate visual timeline based on uploaded events, in a track-based visual format. The number of time increments in the timeline will be constant and chosen at a later date. Based on the number of increments, the date range of each increment will be calculated depending on the total length of time to be displayed by the timeline. Individual tracks (*aligned horizontally*) will then be used to display various historical events and lifetimes, with display elements colored by default or according to category. The number of tracks will depend on the number of events in each timeline.

3.2.6. PERFORMANCE: Users should see and automatically generated visual timeline after uploading a file with 20 events, within 20 seconds.

3.2.7. FUNCTIONAL SPECS: Uploading event files will use server-side scripts (*most likely PHP*). Auto-generating timelines is estimated to be a combination of PHP, CSS and JavaScript.

3.3. Modify generated timelines to specific display, including images, colors

3.3.1. Once timeline is auto-generated, the user can visually manipulate the timeline display elements. This includes being able to reposition the events between tracks and adjusting the size of elements.

3.3.2. The user may also bind historical events to lifetime ranges they belong to, and when moving the range, all corresponding events move along with it.

3.3.3. Users can add images or files to events individually, to be displayed only when viewing more details on the interactive version of the timeline. Images can also be statically added to the non-interactive versions of the timeline for exportation purposes.

3.3.4. Interactive timelines have option to display only certain categories, checkboxes will be present for each category within a timeline to allow the visitor to narrow the display selection by category.

3.3.5. PERFORMANCE: Users should be able to view results of visual manipulation commands within 5 seconds of issuing them, 10 seconds for image or file uploads (*in most cases*)

3.3.6. FUNCTIONAL SPECS: Most interactive functionality will be accomplished using advanced JavaScript functions, uploading images or files and attaching to events will use PHP scripts.

3.4. Assign permissions to edit / view / create timelines

3.4.1. Timelines can be assigned permissions, including which users are allowed to access.

3.4.2. Timelines can also be simply password protected, requiring visitors to enter the correct password before viewing the timeline.

3.4.3. PERFORMANCE: Users should be able to assign permissions to a timeline in 1 minute or less.

3.5. Export timelines into image (*JPEG*) or PDF format

3.5.1. Once timeline is visually appealing, user is able to export to JPG or PDF format to include in various presentations, documents etc.

3.5.2. PERFORMANCE: Timelines should export to JPEG or PDF format within 5 minutes, potentially offering to e-mail a copy to the user to avoid waiting.

3.5.3. FUNCTIONAL SPECS: Programmatically generating JPEG or PDF file versions of a visual timeline can be very difficult. The hope is to rely on JavaScript functionality to capture screen shots and generate the image of the timeline automatically. If PDF generation is desired, potentially relying on the user having the ability to print iFrame windows (*containing the manipulated visual timeline*) to Acrobat Distiller or directly to PDF.

3.6. Embed interactive timelines into web pages

3.6.1. Simple code generation tool will allow custom size frames to be created, housing certain interactive timelines that have been created to be embedded in web pages, for which code is automatically generated (*based on a user form, specifying frame dimensions, and specific timeline to display*)

3.6.2. PERFORMANCE: Users should be able to set the iFrame width, height, and select a timeline to display and have the generated code for use in web pages within 2 minutes.

3.6.3. FUNCTIONAL SPECS: PHP scripts will be used to generate the code for iFrames to allow timelines to be embedded in web pages (*for now only on the NAU network*)

3.7. Save all timelines and event information in database for future retrieval

3.7.1. All timeline details, event / lifetime information will be stored in the database

3.7.2. All user records and permissions will also be stored in the database

3.7.3. All generated web inclusion code will be stored in database

3.7.4. FUNCTIONAL SPECS: All data will be stored in a relational MySQL database on the hosting server, specific table schema to be developed later.

3.8. Optional Functionality

3.8.1. Keyboard controls will allow users to move timeline display elements using the arrow keys on their keyboard

3.8.2. Shift snapping events to timeline grid, will make maneuvering elements in the visual timeline easier

3.8.3. Dynamic zoom in and out feature will display events if within a zoom level capable of doing so. Otherwise, users will have to click on a lifetime to view events contained within.

3.8.4. Embedding timelines within timelines. Recursion. This would allow other (*already created*) timelines to be added as ranges in other (*higher view*) timelines.

4. Constraints and Feasibility Issues

4.1 - Hardware Constraints

Our software will be designed such that it will operate independently of hardware. Our goal is to have a web based solution that only requires a server to run basic Apache, PHP, and MySQL services. We expect these services to be supported by all popular operating systems and their associated hardware.

4.2 - Software Constraints

We expect to use modern web development strategies, languages, and markup languages like Javascript, AJAX, PHP, and HTML5 for example. These will require the users to have a modern browser that is capable of handling these techniques.

4.3 - Environmental Issues

The application we will rely mostly on client side computation done through an internet browser of choice. The user's computer will handle the processing of Javascript and HTML5, which will be the majority of the functionality of the system. Some PHP will be handled by the server and so will MySQL database queries, but these will be minimal compared to client side computations.

4.4 - Compatibility Constraints

We expect to have a large degree of compatibility with our software because it will be browser based. Any operating system capable of running a modern browser, equipped with HTML5 and Javascript processing abilities, will be able to use our system.

5. Project Execution Plan

5.1 - Hardware Setup and Survey of Languages and Techniques (February 1st - February 10th)

Setup web server that will host the required services. Team members explore various techniques and methods of potential solutions. (i.e. pros/cons and feasibility of HTML5, Javascript, Java applets, Flash etc.)

5.2 - Coordination of Efforts and Design of Implementation Details. (February 10th – February 20th)

After a summary of findings from milestone 1 team members should agree on the details and methods for implementation. Overall structure of the software should be developed and broken into key components that can be distributed among team members for work.

5.3 - Testing and Goal Parallelism (February 20th - March 15th)

This is the core development stage where the major functionality is created. The software needs to be shown to be progressing towards the intended goal. Regular in-house testing should occur to ensure that features are functional before moving onwards to further feature implementation. Several graphical interface models should exist at this time and should be tested with intended users to provide for interface optimizations.

5.4 - Dynamic Cooperative Development (March 15th - April 1st)

Most components should be functional and nearly complete. Team members should take time to redistribute work load to balance completion amongst components. Some near complete user testing may be possible at this point.

5.5 - Extensive User Testing and Prototyping (April 1st - April 13th)

Development should be focused on real world use scenarios at this time. Test users should be able to be given the software, and with little instruction, generate a useable output from the software without much difficulty.

5.6 - Final Bug Fixes and Optimizations (April 13th - April 20th)

Any remaining and known bugs should be fixed at this point. Any known performance enhancements should be made as well. Code should be cleaned up for better readability and future maintenance and updating. The product should be fully functional and nearly error free.

5.7 - Functional Demo, Ready for Presentation and Delivery (April 20th - April 27th)

The software should embody all of the functional requirements and be free of major bugs and errors. The software should be able to be used by anyone to generate a useful output during a life demonstration. The software should then be ready to be delivered to the client for use in the field.

5.8 - Unscheduled Diversions From Calendar

During the testing and goal parallelism phase we should be able to implement all of the major and basic features of the software product. If these goals are not completed then we have allotted time in the Dynamic Cooperative Development stage to better target the work force towards areas that may be lacking in functionality and have not kept up to their proposed level of completion. The Extensive User Testing and Prototyping is another area of time where changes can be made to the software if needed because of poor user experiences or unforeseen bugs and flaws in the software. This section can also be started early if the Dynamic Cooperative Development is successful in less time than planned.

6. Appendices

6.1 - HTML5

HTML5 is the 5th revision of the Hyper Text Markup Language developed by the World Wide Web Consortium. It is regarded as the successor to HTML4, however the HTML5 specification has not yet been finalized. Thus, the language is still not standardized across all web browsers and frameworks. Currently, HTML5 is supported by the following browsers: Google Chrome 3.0+, Mozilla Firefox v4.0+, Microsoft Internet Explorer 9+, Safari 3.1+, and Opera v11+.

6.2 - JavaScript

The JavaScript scripting language has been around since the mid 1990's and is considered a staple tool of web programmers over the years. Because of its early introduction to the web, JavaScript has grown and evolved into a robust and secure language; solidifying its support across all web browsers and platforms.