CHRONOSPHERE:

FRONT-END/BACK-END DESIGN

by: Ben Scholer

ABSTRACT

Nucor Buildings Group (NBG) currently uses a variety of desktop programs to manage employees, some of which are more than 10 years old. It is clear by looking at these programs that they are very outdated and difficult to use, which becomes a big problem with 9,000+ employees. Therefore, NBG has tasked Group Services with creating an easy-to-use web application.

Front-end design is arguably as important as the back-end design. While strong back-end design will make for a long-lasting program, strong front-end design will ensure that users enjoy the app. Although both types of design require creativity, front-end design should be viewed more like an art project for a programmer, whereas back-end development should be considered a more traditional programming project.

TABLE OF CONTENTS

Abstract 01

Database Design 02

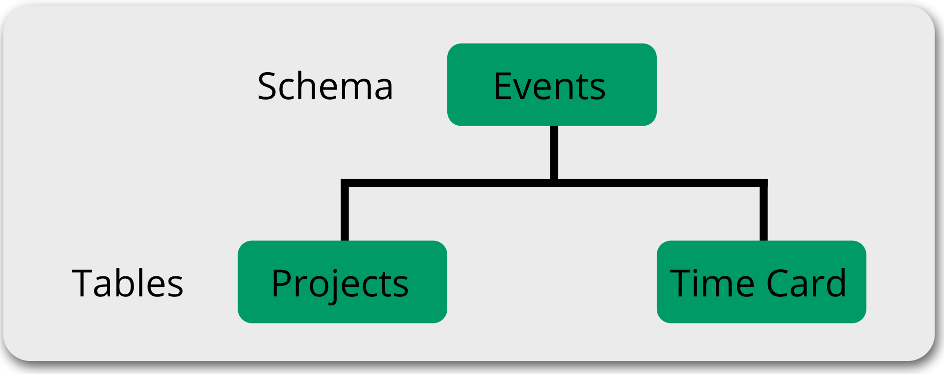
- Events.Projects 02
- Events.TimeCard 03

Front-End Design 04

- Day Cards 05
- Project Tracking 05
- Time Card 06

Works Cited 08

DATABASE DESIGN



Events.Projects

Entries in the projects table contain data about the projects employees work on throughout the day. Because of this use case, the projects table should be designed in a classic transactional CRUD pattern, where entries can simply be updated or deleted directly, without the need for additional entries. By using this pattern, drive space used by the table is kept to a minimum, which in the long run will make for faster SQL queries.

Fields

- EntryId 🔑
- CorpCode
- Date
- ActivityId 🔑
- Hours
- ProjectNumber

The EntryId field is simply an AutoNumber, a unique ID for each entry. The CorpCode field is the 5 letter designation, describing which NBG division the project belongs to. This is necessary due to overlapping project/job numbers among divisions. The date is simply the date that the entry's data was logged for. ActivityId is a foreign key, linked to a table of activities, which represents the activity the user worked on for the project. Hours is the amount of time the user worked on said activity, as a double, and ProjectNumber (aka JobNumber) is the name of the project the user worked on, for example, W18G2578A.

Events.TimeCard

Fields
Id
SubmitterEmployeeUPN
SubmitterDate
SubjectEmployeeUPN
SubjectDate
SubjectEventId
EventType

Entries in the TimeCard table contain data about when a person has worked each day. For the TimeCard, it is important that the design pattern of the database allows for keeping track of every time an employee's times are changed for the day. This is possible using an event stream design pattern. Using this pattern comes with disadvantages, primarily in space usage and query speeds. However, with the falling prices of storage mediums, this becomes a feasible option.

The main principle of the event stream is to never edit or delete any entries in the database, and instead, add new entries describing what should happen. This also allows for using the same database for applications that don't involve entering clock in/out times, such as a physical card swipe system, used currently by factory floor employees.

Another important design consideration is the possibility of an employee's manager editing his or her TimeCard. In this case, it becomes important to record not only whose times are changing, but also who changed them. It soon becomes helpful to think of each entry as two separate entities: the submitter data and the subject data.

The submitter fields include SubmitterEmployeeUPN, being the UPN of the employee making the change, and the SubmitterDate, being the DateTime describing when the entry was added to the database. The subject fields include SubjectEmployeeUPN, being the UPN of the employee whose TimeCard is being changed, the SubjectDate, describing the DateTime of when the clock in/out occurred, and the SubjectEventId, a nullable field which only has a value when representing a deleting entry. Besides the fields describing submitter and subject, each entry has an AutoNumber Id, and EventType, an integer describing if the entry is a clock in/clock out, or deletion of a previous entry.

As this design pattern can be difficult to grasp at first, here is an example scenario, with the corresponding entries in the diagram below:

- Bob arrives at work at 8:05 AM and clocks in, accidentally entering 9:05.
- Bob leaves work at 4:20 PM, correctly entering it into Chronosphere.
- At the end of the week, Bob's supervisor, Jill, while reviewing her reports' time cards, notices the error on Bob's card and fixes it.

For this example, we'll use an EventType of 0 for clocking in, 1 for clocking out, and 2 for deleting. Also, field names have been abbreviated.

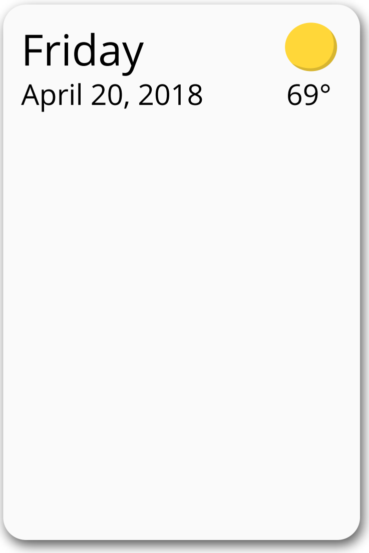
Id	SubmitUPN	SubmitDate	SubjUPN	SubjDate	SubjEventId	EventType
165	bob@nucor.com	2018-10-24 08:05	bob@nucor.com	2018-10-24 09:05		0
246	bob@nucor.com	2018-10-24 16:20	bob@nucor.com	2018-10-24 16:20		1
1158	jill@nucor.com	2018-10-28 14:16			165	2
246	jill@nucor.com	2018-10-28 14:16	bob@nucor.com	2018-10-24 08:05		1

FRONT-END DESIGN

Although there is no correct way to chose a front-end framework, some are better than others. Polymer, a tool originally used by Google to test Material Design, provides many benefits to web developers. It also comes bundled with a plethora of components, all built around Google's Material Design Guidelines. In recent years, Polymer has quickly become one of the primary forces driving the adoption of reusable web components. These web components provide developers with a way to bring Object-Oriented architecture to front-end web development.

The most obvious feature of Material Design is the use of card elements. These cards provide a convenient way for developers to organize their app into sections. While cards are not the right choice for every project, they conform very well to an app based around days of the week, such as Chronosphere. Creating individual cards for each day lets users look at their week as a calendar, instead of a list.

Day Cards

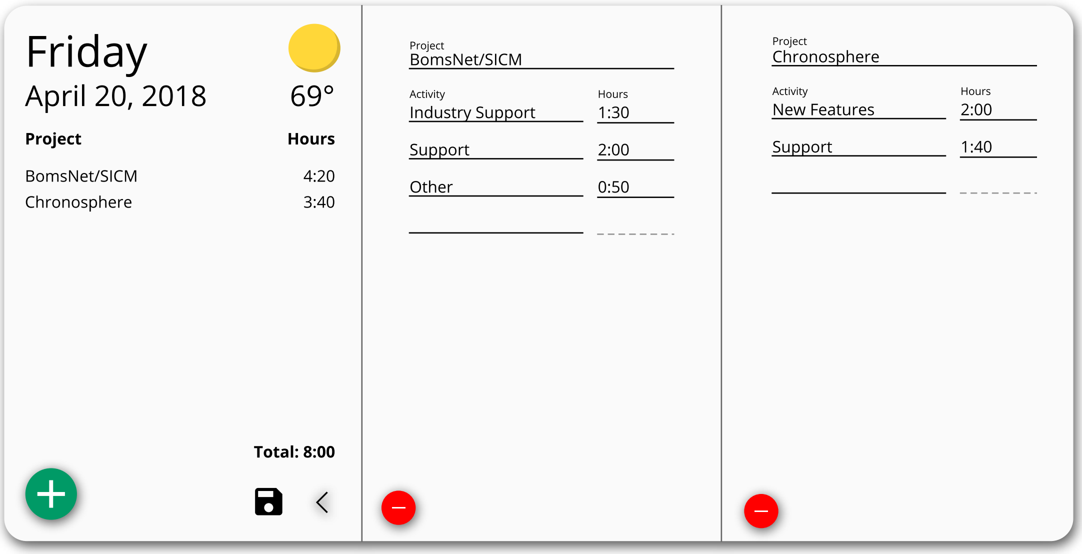


Using consistent design between facets of Chronosphere will be critical to the UX. As mentioned previously, using cards allows developers to separate information into digestible chunks. Since project tracking and the time card are both based on day-by-day activity, the general page layout should be very similar. Having a card that displays each day's information would be a simple way to accomplish these goals.

A fun way to improve the user experience could be the addition of weather data to the cards. This way, users can see at a glance what the weather looks like for the week. Caching weather data into the browser’s local storage would provide the users with historical data, as well as reducing the number of API calls necessary.

Project Tracking

Designing the project tracking card comes with a number of issues, due to the amount of information that must appear on the card. Users must have the ability to add and remove projects for a certain day. In addition, users must be able to edit the activities, as well as times for each activity within those projects. To implement this, it is logical to create summary cards that can expand and contract, so they don't take up too much room on the page.



Project	Hours
BomsNet/SICM	4:20
Chronosphere	3:40

Total: 8:00

Project: BomsNet/SICM

Activity	Hours
Industry Support	1:30
Support	2:00
Other	0:50

Project: Chronosphere

Activity	Hours
New Features	2:00
Support	1:40

The summary cards should show the user at a glance which projects he or she worked on for that day, and how much time was spent on each. Users can expand and contract the card with an arrow-like button, that when pressed, will rotate 180°, and open or close the card, revealing the projects.

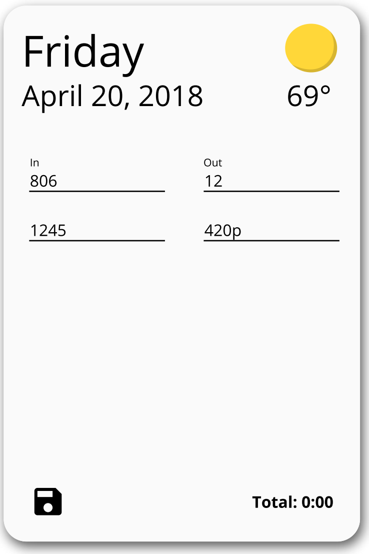
Another challenge with the project card is standardizing and validating user input. The key to accomplishing this is to keep the user feeling in charge. Instead of making the user scroll through a list of options for projects and activities, developers could create an autocomplete field, which lets the user enter queries. To further improve the experience, developers could implement fuzzy search. Fuzzy search is formally called approximate string matching and is the process of loosely matching strings. This allows the user to make small spelling mistakes and still find what he or she is looking for. In addition to fuzzy searching, developers can create a smart time input. A detailed description lies in the next section.

Time Card

Time cards have much less information to display, which makes design and development much simpler. The only components needed for the time card are inputs for clocking in and out. Developers could also add a clock in/out button that finds the appropriate input field and populates it with the current time. Of course, this button should only display for the current day and should not enter invalid information. However, standardizing user time input is important, and as with the fuzzy search, it is important to make the user feel like he or she is in charge.

To do this, developers can create a smart time input. These validate times, but also assist users with getting times into the HH:MM AA format. They can also add AM/PM dependent on the value of the last input. Examples of this behavior can be found on the next page. An important part of this, however, is waiting until the user has moved focus to another component before correcting the user's input.

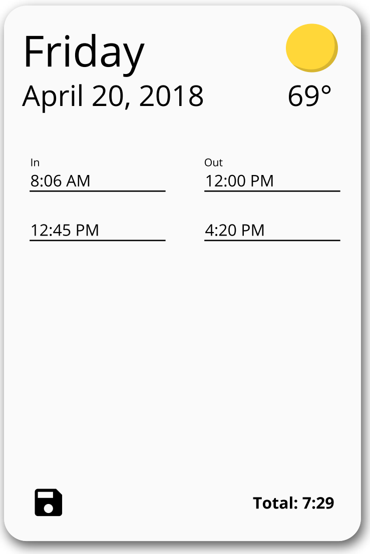
Raw Input


Friday
April 20, 2018
69°

In	Out
806	12
1245	420p

Total: 0:00

Corrected Input


Friday
April 20, 2018
69°

In	Out
8:06 AM	12:00 PM
12:45 PM	4:20 PM

Total: 7:29

WORKS CITED

About the Polymer Project. (n.d.). Retrieved October 23, 2018, from https://www.polymer-project.org/about

Burton Rodman Chronosphere Interview [Telephone interview]. (2018, October 22).

David Rettig Chronosphere Interview [Telephone interview]. (2018, October 22).

Eng, R. K. (2017, June 04). Chapter 3: What is Object-Oriented Programming? – Learn How To Program – Medium. Retrieved October 23, 2018, from https://medium.com/learn-how-to-program/chapter-3-what-is-object-oriented-programming-d0a6ec0

Adam Griswold Chronosphere Interview [Telephone interview]. (2018, October 22).

Material Design history. (2015, December 15). Retrieved October 23, 2018, from https://krify.co/tag/material-design-history/

Perera, S. (2018, April 04). A Gentle Introduction to Stream Processing – Stream Processing – Medium. Retrieved October 23, 2018, from https://medium.com/stream-processing/what-is-stream-processing-1eadfca11b97

Shiotsu, Y. (2018, May 30). An Introduction to Polymer. Retrieved October 23, 2018, from https://www.upwork.com/hiring/development/intro-to-polymer-javascript-library/

Tregoat, J. (2018, January 09). An Introduction to Fuzzy String Matching – Julien Tregoat – Medium. Retrieved October 23, 2018, from https://medium.com/@julientregoat/an-introduction-to-fuzzy-string-matching-178805cca2ab

Verraes, M. (2013, April 19). CRUD is an antipattern. Retrieved October 23, 2018, from http://verraes.net/2013/04/crud-is-an-anti-pattern/

Web Components | MDN. (n.d.). Retrieved October 23, 2018, from https://developer.mozilla.org/en-US/docs/Web/Web_Components