# Time Clock Web Application

### James Dawson & Mikkel Kringelbach

[**http://timeclock.edisoncode.com/**](http://timeclock.edisoncode.com/)

### Introduction

The goal of this application is to record and report employee time. Jed’s family owns a small business in California and up until the summer of 2011 their employees tracked their time using a manual punch card system. The data then had to be transcribed into the payroll software before employees could be paid. Not being satisfied with the options available at the time, Jed wrote a simple time tracking application that the business still uses. The major benefit of the application is exporting time data directly to the payroll software. Our goal for this project is to greatly improve on the application, add missing functionality, and improve the user experience.

The application will have two interfaces: one that is dedicated to allowing employees to clock in and out and the other that is dedicated to managing the system.

The main interface will be used by employees to clock in and out, go on break, and receive management messages. This interface will be the default starting location for the application and will not require any authentication other than the employee’s pin. Alternative mobile views will be rendered when mobile devices access the site. The interface should be as simple and intuitive as possible.

The administrative interface will be used to monitor employee status, create messages, perform CRUD (create, read, update, delete) actions on departments, holidays, punches, breaks, punch types, pay types, and pay rules. Also, the data will be able to be exported to payroll software (Sage MAS90).

### 2. Detailed Application Requirements

1. Each department is described by the name, location, number, a seed for the pay periods and the interval of each pay period. The department number is unique.
2. A pay type is described by an id, a description, the weekly max, the daily max, and the next pay type (once thresholds are met). The pay type id is unique.
3. A punch type is described by an id, description, and an attribute determining if it is an option to be checked in with. The id is unique.
4. A punch is described by a punch id, employee id, the in and out times, punch type, and the department. The punch id is unique, the out time is allowed to be null, if the employee is currently working.
5. A timecard represents a collection of Lines for a given pay period and is described by the employee ID, the pay period, and timecard id. The timecard id is unique.
6. The timecard lines represent a single line on a time card. They are described by a line number, timecard id , punch id, pay type id, split start time, and split end time. The combination of line number and timecard are unique. Split start and split end are only allowed to be within the time of the punch the line refers to.
7. An employee is described by first and last name, middle initial, manager id, pin, id (alpha numeric to match payroll software), department, and employment status. The employee id is unique, and the middle initial is allowed be null.
8. Messages are described by a message id, manager id, and the message body. The message id is unique.
9. Messages for indicate which employees will receive a given message. It is described by a message id, employee id, and the time it was viewed. The combination of message id and employee id are unique. The viewed time set to null if the message is pending.
10. Holidays are described by an id, date, and whether or not it repeats. The holiday id is unique.
11. Each company is described by an id and a name. The id is unique for each company.

### 3. Design

ASP.NET

Due to client specifications our application is being written using ASP.NET. This environment allows us to rapidly provision a high quality, stable, and scalable web application that takes advantage of Microsoft technologies. The client operates several MS Server 2008 servers on their local network and wishes to run the application on IIS 7.5. While IIS does support PHP, the environment that it is most suited for is ASP.NET. Additionally, the client has licensing for and actively uses MS SQL Server 2008.

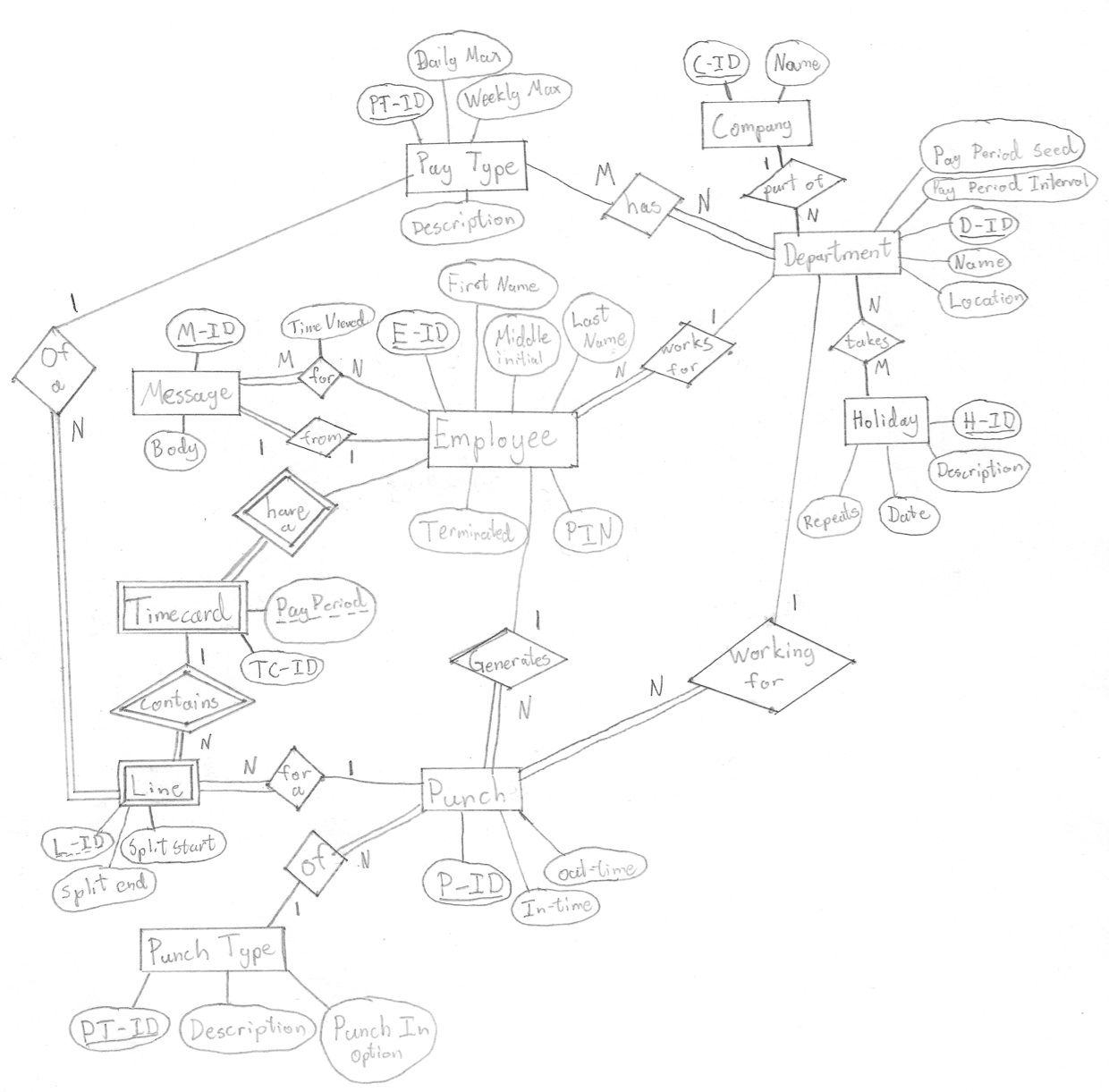
We are working in Visual Studio 11 Beta and compiling the application for .NET 4.5 Beta. We believe that by using the latest available technologies we will be better prepared to take advantage of these tools in the workforce. Most of the SQL communication will take place using Entity Framework 4.3.1 Code First principles. This essentially removes SQL code from the application as it is all generated by Entity Framework. We will however furnish the raw SQL statements that we would estimate Entity Framework is generating for the more important features of the application.

Storage

All data for the application will be stored in MS SQL Server 2008. Our schema allows for the storage of all relevant information.

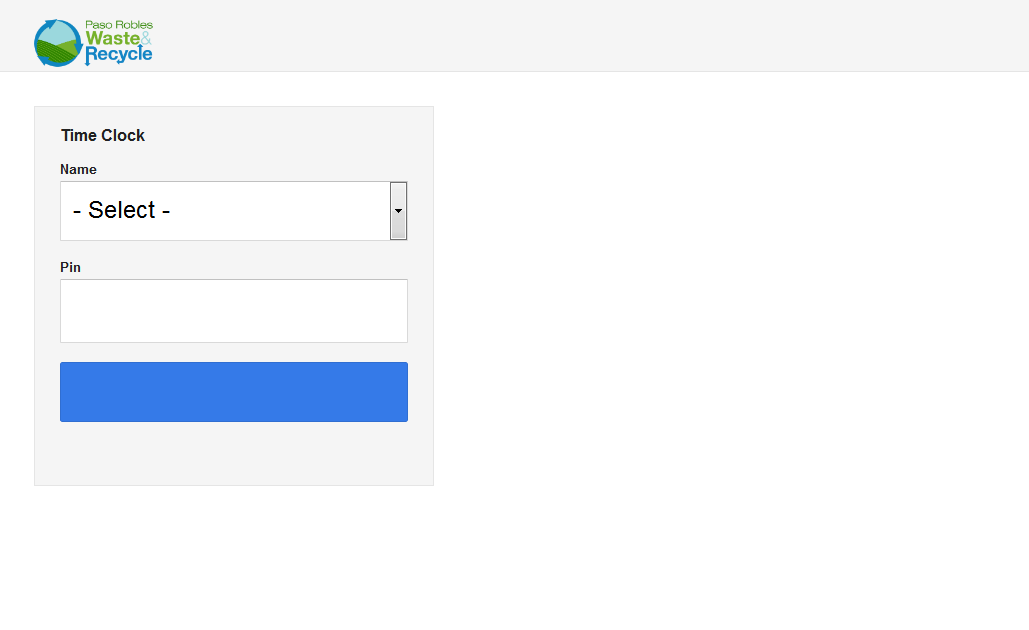
Files needed for the website such as our CSS files, JavaScript files, and images will be stored on the file system of the web server. Compression and caching will be configured to aid in performance.

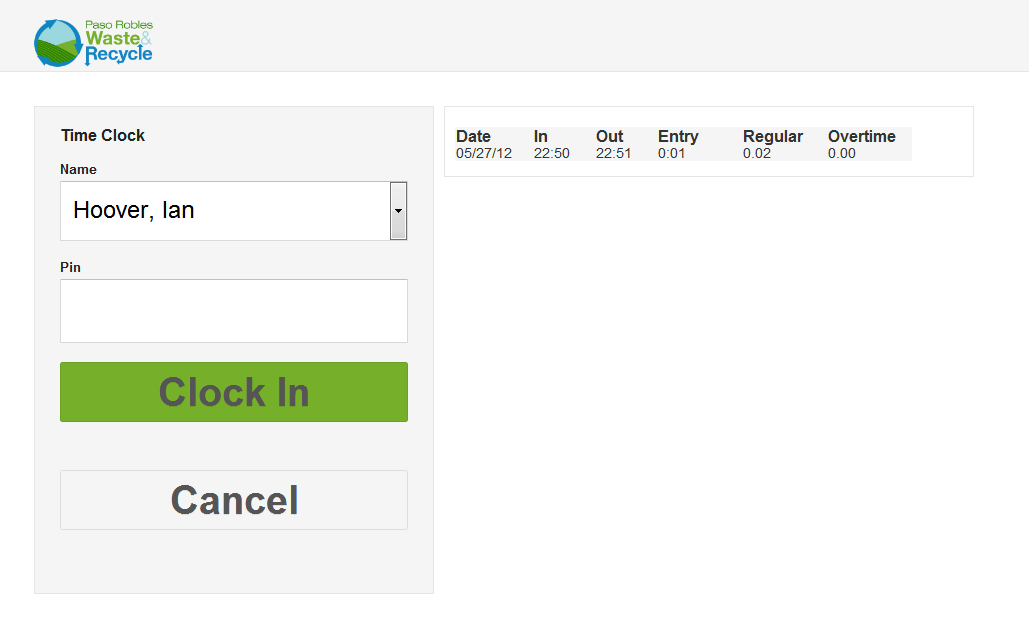
ER-Diagram



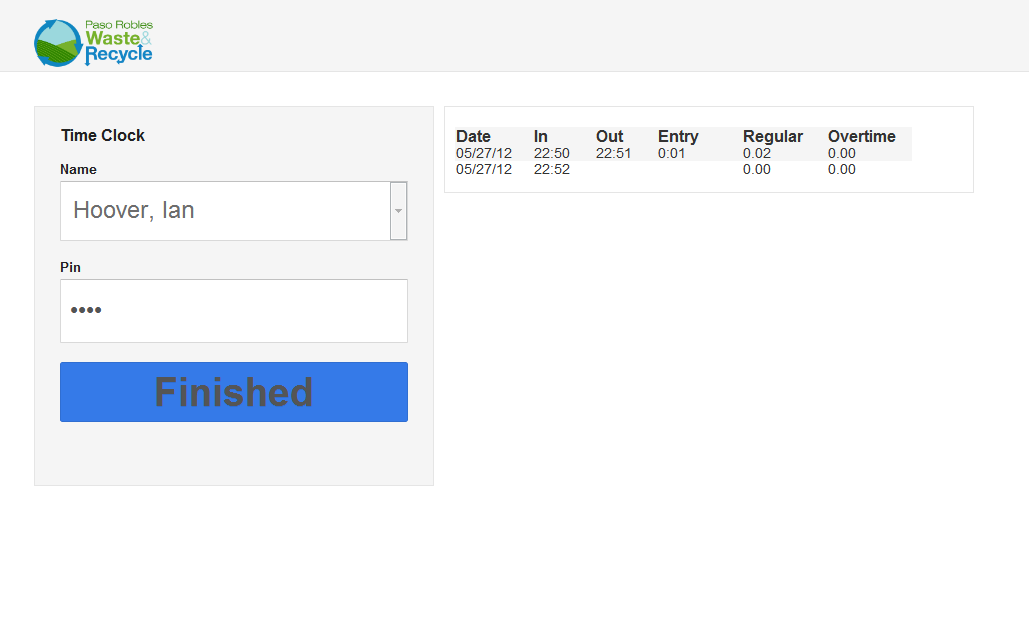
Screen Shots

Some of these screen shots are taken from the existing application that Jed created. They will serve as the building blocks for the new application. Since the employees of the company are already familiar with the interface that they use daily we do not intend to make any major changes to the look and “feel” of the main application. Our efforts will be focused on the management side as well as adding new features.

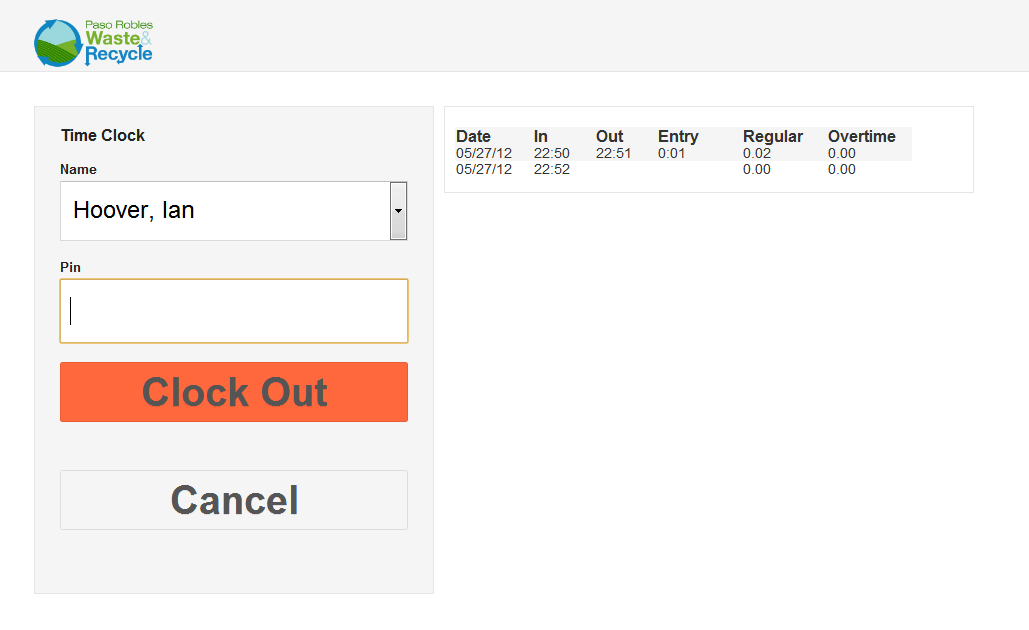
  
Initial screen of the application. Employee needs to select their name from the list. As soon as they selected their name the application will issue an HTTP GET AJAX request to the server to retrieve their current status, which determines whether they are clocking in our out, and pull up their current time card.



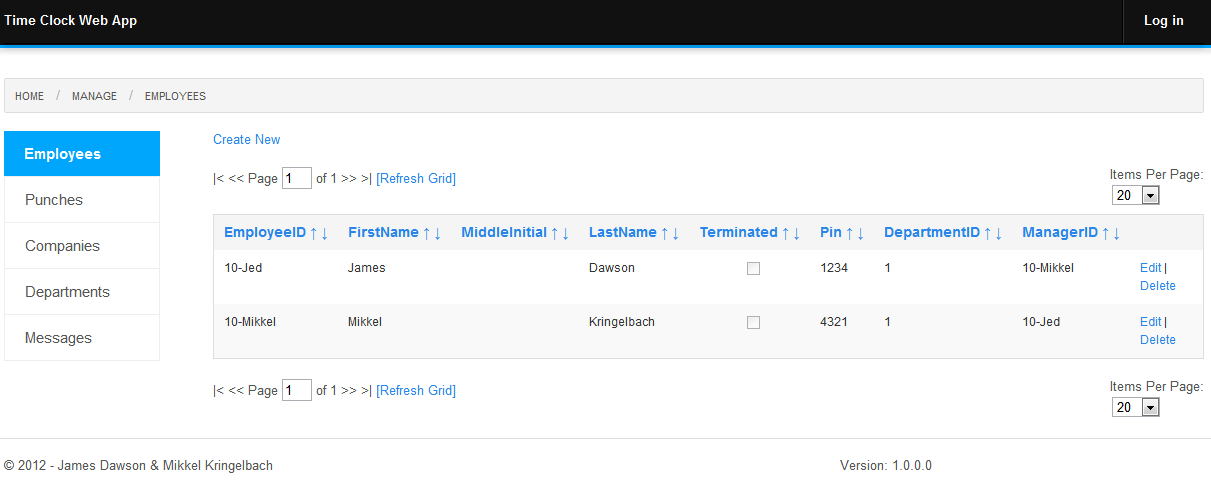
This shows an employee selected from the list. The application has determined they do not have an open punch and thus must be clocking in.



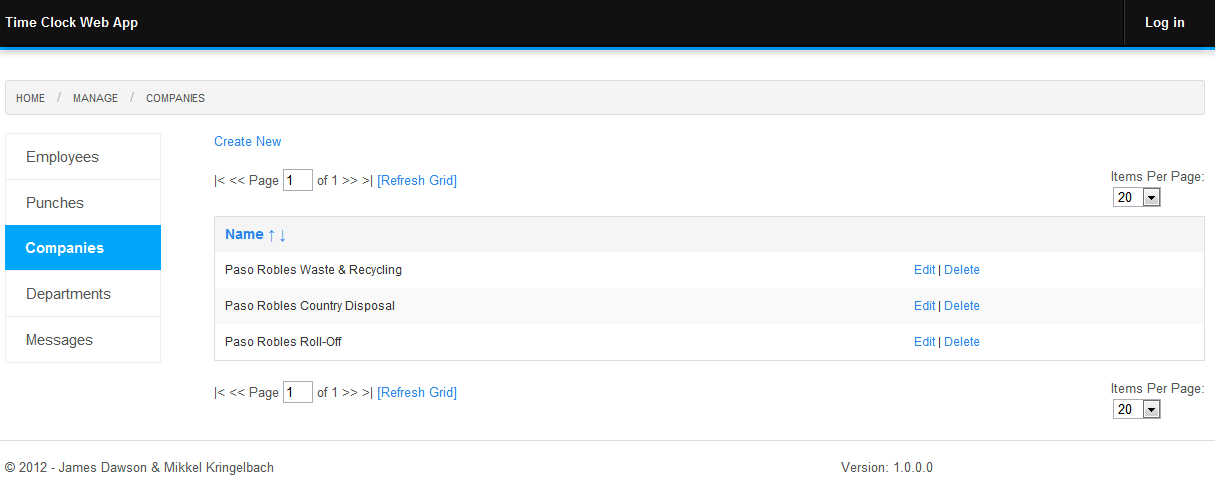
After the employee has triggered a punch they are shown the open punch along with a finished button. When the employee touches the finished button (or lets the application idle for 45 seconds) the application resets to the initial view.



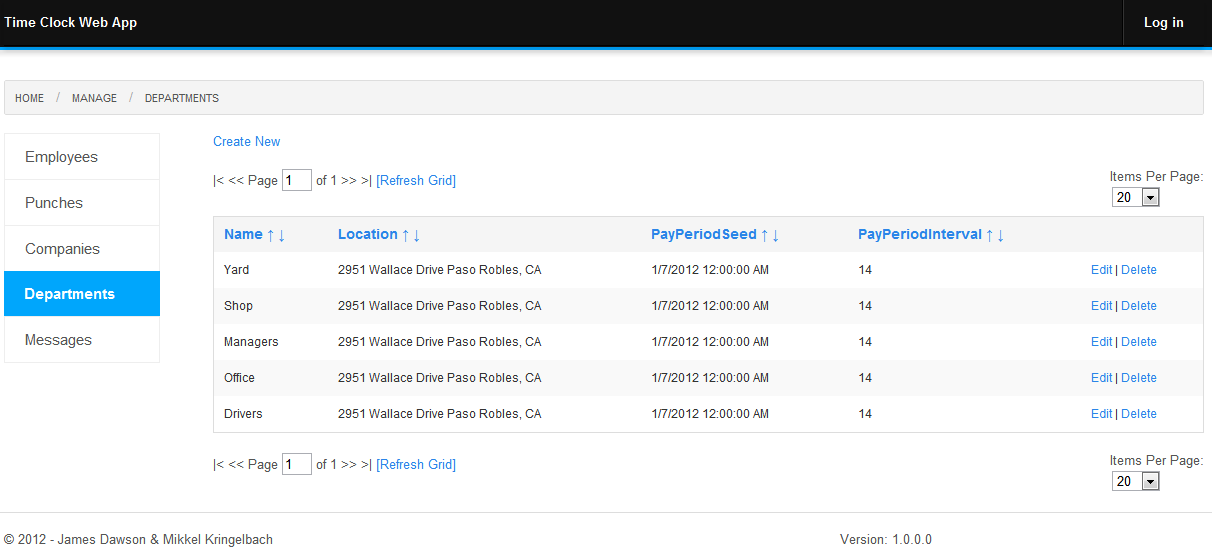
A view of the application when the employee has an open punch.



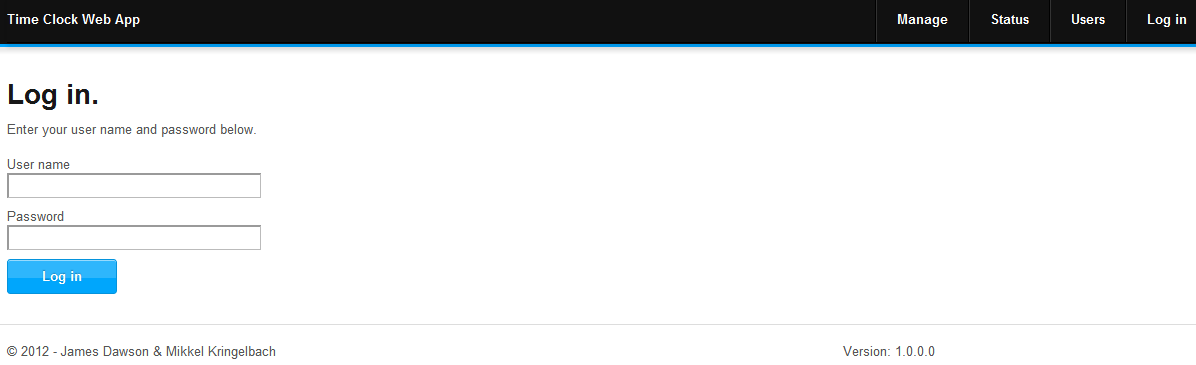
This is the view of the management side of the application. From this view it will be possible to add and edit employees. Note: Authentication has not yet been implemented so that is why the “Log In” button is still visible.



Create and Edit Companies.



Create and Edit Departments.



This is the log in page for the application. We plan to use ASP.NET membership to handle authentication for our application. This will allow us to have a fully tested and highly secure application.

### 4. Implementation Details

Give a basic (brief) overview of why your application needs to make use of a DBMS. Highlight which of **your application requirements** the DBMS provides.

Also, describe what happens when a button is clicked or the content of a filter is changed? In particular, how is the query formed based on user’s set parameters? Are any indexes used in order to speed up the query execution? …

### 5. Evaluation

Explain the steps that you have taken to test the correctness, effectiveness, and intuitiveness of your system.

### 6. Future Work

What additional functionalities you plan to implement beyond this class?

### 7. Lessons Learned

What challenges have you run into during the design, implementation, and testing of your system? How did you address these issues? What would you have done differently next time?