

Time Clock Web Application

James Dawson & Mikkel Kringelbach

<http://timeclock.edisoncode.com/>

1. 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, receive management messages, and view time card data/history. 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 and are actually the primary focus of UI development. The majority of the employees will access the web application through one of many iPads that are available for use. 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

- A. 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.
- B. 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.
- C. 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.

- D. 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.
- E. 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.
- F. 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.
- G. 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.
- H. Messages are described by a message id, manager id, and the message body. The message id is unique.
- I. 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.
- J. Holidays are described by an id, date, and whether or not it repeats. The holiday id is unique.
- K. 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.

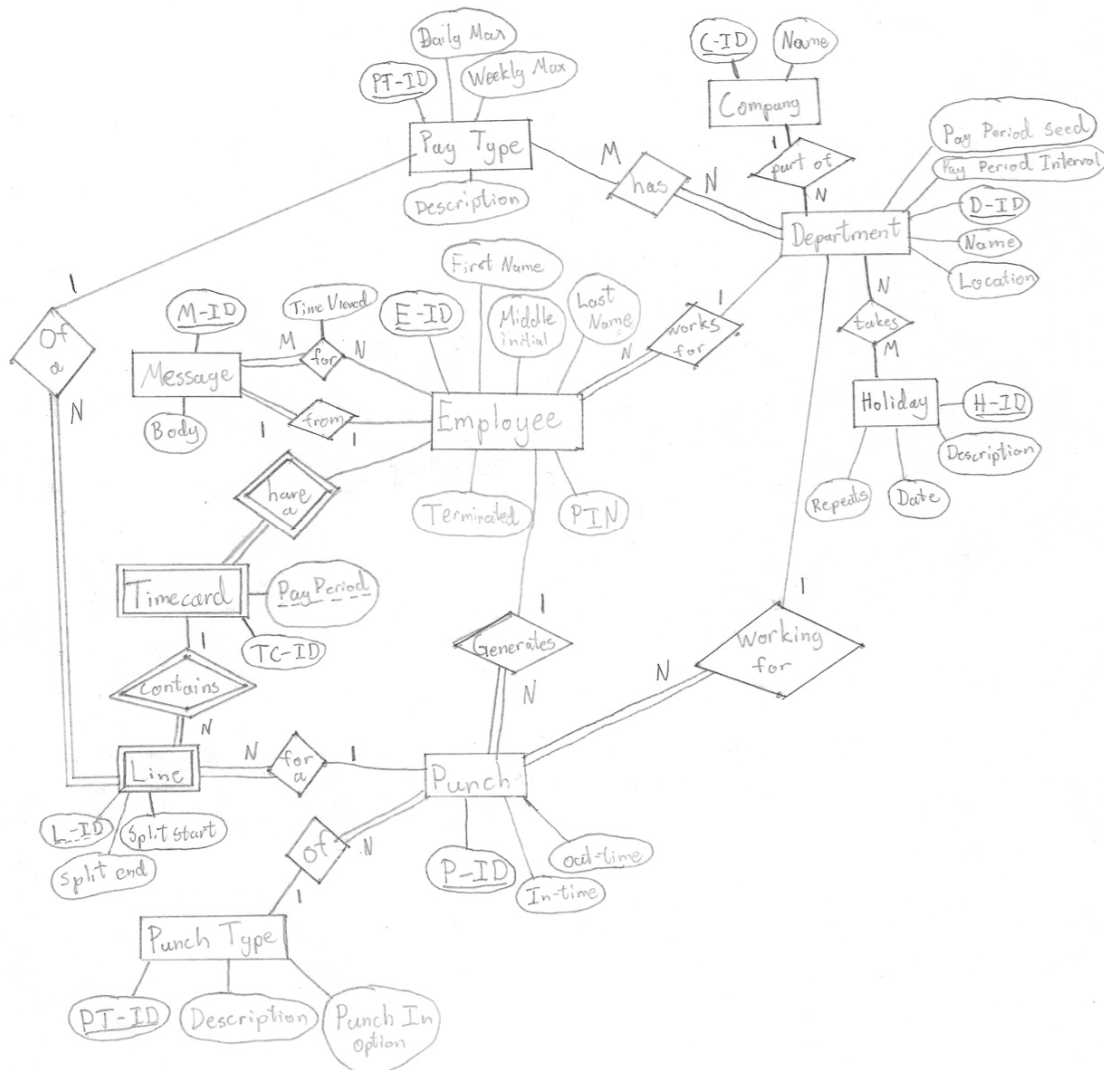
We are working in Visual Studio 11 Beta and compiling the application for .NET 4.5 Beta. We believe that by using the latest available technologies we will be better prepared to take advantage of these tools in the workforce. Most of the SQL communication will take place using Entity Framework 4.3.1 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.

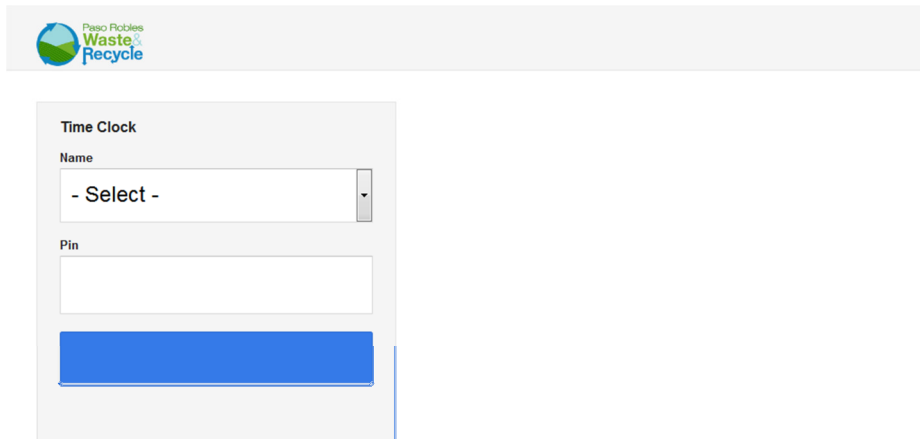
Files needed for the website such as our CSS files, JavaScript files, and images will be stored on the file system of the web server. Compression and caching will be configured to aid in performance.

ER-Diagram



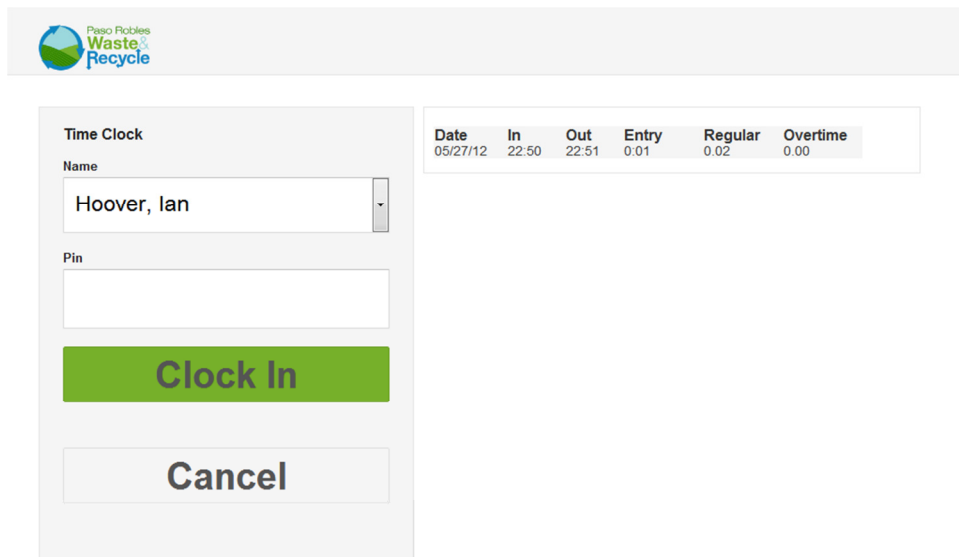
Screen Shots

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.



The screenshot shows the initial screen of the Time Clock application. At the top is a header bar with the logo "Paso Robles Waste Recycle". Below the header is a form titled "Time Clock". The form contains a "Name" field with a dropdown menu showing "- Select -" and a "Pin" field with a text input. A large blue button is positioned below the Pin field.

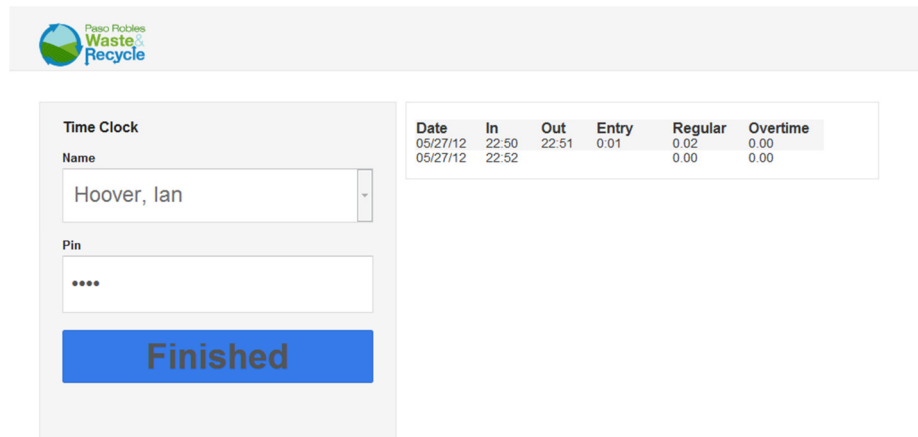
Initial screen of the application. Employee needs to select their name from the list. As soon as they selected their name the application will issue an HTTP GET AJAX request to the server to retrieve their current status, which determines whether they are clocking in or out, and pull up their current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



The screenshot shows the Time Clock application after an employee has been selected. The "Name" dropdown menu now displays "Hoover, Ian". Below the Pin field, there are two buttons: a green "Clock In" button and a grey "Cancel" button. To the right of the form is a table displaying the employee's current status.

Date	In	Out	Entry	Regular	Overtime
05/27/12	22:50	22:51	0:01	0:02	0:00

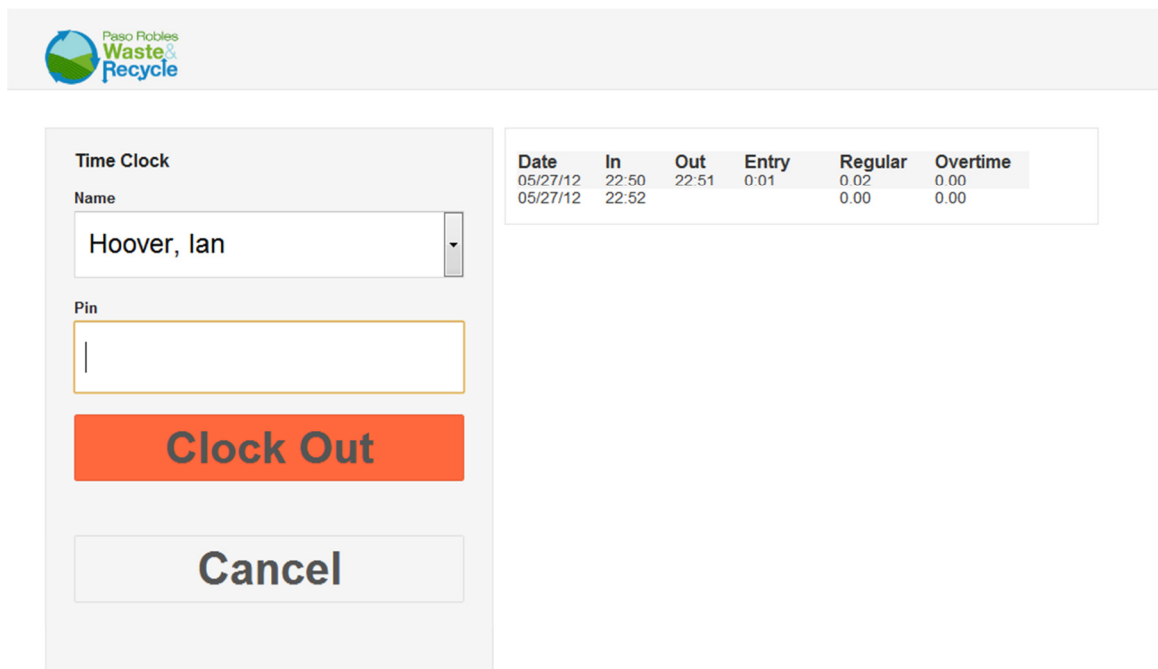
This shows an employee selected from the list. The application has determined they do not have an open punch and thus must be clocking in.



The screenshot shows the 'Time Clock' interface. At the top is the 'Paso Robles Waste Recycle' logo. Below it, the 'Time Clock' section contains a 'Name' dropdown menu with 'Hoover, Ian' selected, a 'Pin' input field with four dots, and a large blue 'Finished' button. To the right is a table showing punch data.

Date	In	Out	Entry	Regular	Overtime
05/27/12	22:50	22:51	0:01	0:02	0:00
05/27/12	22:52			0:00	0:00

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.



The screenshot shows the 'Time Clock' interface after a punch is triggered. The 'Name' dropdown still shows 'Hoover, Ian', but the 'Pin' input field is now empty. Below the pin field are two buttons: a large orange 'Clock Out' button and a smaller grey 'Cancel' button. The table on the right is identical to the previous screenshot.

Date	In	Out	Entry	Regular	Overtime
05/27/12	22:50	22:51	0:01	0:02	0:00
05/27/12	22:52			0:00	0:00

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

Time Clock Web App

Log in

HOME / MANAGE / EMPLOYEES

Employees

Punches

Companies

Departments

Messages

Create New

|< << Page 1 of 1 >> >| [Refresh Grid]

Items Per Page: 20

EmployeeID ↑ ↓	FirstName ↑ ↓	MiddleInitial ↑ ↓	LastName ↑ ↓	Terminated ↑ ↓	Pin ↑ ↓	DepartmentID ↑ ↓	ManagerID ↑ ↓	
10-Jed	James		Dawson	<input type="checkbox"/>	1234	1	10-Mikkel	Edit Delete
10-Mikkel	Mikkel		Kringelbach	<input type="checkbox"/>	4321	1	10-Jed	Edit Delete

|< << Page 1 of 1 >> >| [Refresh Grid]

Items Per Page: 20

© 2012 - James Dawson & Mikkel Kringelbach

Version: 1.0.0.0

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.

Time Clock Web App

Log in

HOME / MANAGE / COMPANIES

Employees

Punches

Companies

Departments

Messages

Create New

|< << Page 1 of 1 >> >| [Refresh Grid]

Items Per Page: 20

Name ↑ ↓	
Paso Robles Waste & Recycling	Edit Delete
Paso Robles Country Disposal	Edit Delete
Paso Robles Roll-Off	Edit Delete

|< << Page 1 of 1 >> >| [Refresh Grid]

Items Per Page: 20

© 2012 - James Dawson & Mikkel Kringelbach

Version: 1.0.0.0

Create and Edit Companies.

Time Clock Web App

Log in

HOME / MANAGE / DEPARTMENTS

Employees

Punches

Companies

Departments

Messages

Create New

|< << Page 1 of 1 >> >| [Refresh Grid]

Items Per Page: 20

Name ↑ ↓	Location ↑ ↓	PayPeriodSeed ↑ ↓	PayPeriodInterval ↑ ↓	
Yard	2951 Wallace Drive Paso Robles, CA	1/7/2012 12:00:00 AM	14	Edit Delete
Shop	2951 Wallace Drive Paso Robles, CA	1/7/2012 12:00:00 AM	14	Edit Delete
Managers	2951 Wallace Drive Paso Robles, CA	1/7/2012 12:00:00 AM	14	Edit Delete
Office	2951 Wallace Drive Paso Robles, CA	1/7/2012 12:00:00 AM	14	Edit Delete
Drivers	2951 Wallace Drive Paso Robles, CA	1/7/2012 12:00:00 AM	14	Edit Delete

|< << Page 1 of 1 >> >| [Refresh Grid]

Items Per Page: 20

© 2012 - James Dawson & Mikkel Kringelbach

Version: 1.0.0.0

Create and Edit Departments.



Time Clock Web App

Manage Status Users Log in

Log in.

Enter your user name and password below.

User name

Password

Log in

© 2012 - James Dawson & Mikkel Kringelbach

Version: 1.0.0.0

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.

All screenshots were from a beta version or old version of the application and we recommend that you view the final version to get a feel for how it works now.

4. Implementation Details

Our application is entirely based on recording and reporting data. Without a DBMS our client would have to return to the way they were tracking employee time before moving to a DBMS based system. Specifically, recording precise in and out times for specific employees requires some sort of relational model in order to correctly associate all of the data.

We choose to use a JavaScript framework to help manage the browser UI. After evaluating several frameworks we decided to use knockout.js. Knockout uses a Model-View-View-Model pattern for managing the UI and proved to be highly flexible and extremely simple to use. We were able to bind our html fields to knockout objects which were then populated with data returned from AJAX calls.

The queries that trigger in the background when the AJAX API calls are made to the application are written using Entity First Framework. We felt that it was appropriate to leverage the power of the framework since Jed had already written the more complex queries as SQL stored procedures. Instead of redoing those procedures in SQL for the new schema the data is queried with LINQ calls.

Mikkel wrote some incredible queries, which are called recursively, to populate the timecard lines table after an employee's punch is complete or changed. The reason we choose to store this information in a table (as opposed to recalculating it as required) is that it more often than not does not change after the punch has been finished. This will lead to some performance gains when generating reports.

5. Evaluation

We have exhaustively tested the accuracy of the calculations that the application is making. Since we are dealing with data that impacts the amount employees get paid and any mistakes could lead to fines for the employer we took the accuracy very seriously. Fortunately, we had an existing system with nearly a full years' worth of data to simulate our tests with. We wrote some scripts to populate the new schema with old data and then generated reports from each system to compare results. This analysis lead to us discovering numerous rounding and other minor mathematical issues with the new application rather quickly.

Also, we have “clicked” through the application in every direction/path we could think of and did not discover any issues.

6. Future Work

Our client would like us to continue developing the application even after the class has been completed. We were unable to finalize the payroll export functionality as well as some of the lower level management tools. It is our intention to finish these features for the client over the summer.

One major feature missing from this project is security. We plan to use the built in ASP.Net Membership components to add in management authentication before the application is made live at the client site.

Also, we will be working with the client to properly facilitate sick and vacation processing.

7. Lessons Learned

Going through the design phase of the project was very helpful. We were able to identify many issues before every writing any code for the application or beginning to create the SQL structure. Once our design was finished we have a really nice framework to work with. This made the coding progress nicely.

Time tracking is difficult. This project took a lot of effort when it came to correctly making all of the calculations with time.

Also, we definitely found some weaknesses in Entity Framework along the way. Some of the more complex queries we wanted to do were simply not possible in Entity Framework as a single statement. We know that it would have been possible to accomplish the queries in SQL with multiple layers of nesting and toyed with the idea of using a mix of LINQ and stored procedures. We settled on writing the queries in LINQ by breaking the statements apart to accomplish them.