Project opal

David Bel, Gary Graham, Cody Rozic

# organization

The customer of Project Opal is JKS Enterprises, an organization consisting of Ace Automotive and Diamond Lacey’s Taxi. The payroll & shift management system is under review. The current solution involves manually recording clock-in and clock-out times of various taxi operators on any medium- timesheets, regular A4 or even sticky notes. The dispatchers (who are to be recording the current data) don’t have total control over the data since drivers can easily modify the documents with little effort & no oversight. The payroll department at JKS has the responsibility of compiling the data and writing the paycheques. Due to the lack of paper trail, JKS has to deal with internal friction between management and employees due to disagreements of who came in and when. JKS has modernized a few sectors of its business using a variety of technologies, such as Microsoft Access. The users of the current system includes cab drivers, dispatchers, mechanics, managers, bookkeepers, repairmen, and owners. Its inputs are time cards, and hourly rates. It outputs paycheques and historical data.

The system itself will be hosted on two machines- a kiosk stationed in the parking garage where drivers can easily clock in and clock out from and a secondary server for any web aspects. The database file will have to be located off-site for easy backup and retrieval.

JKS Enterprises has shared a few goals they want to achieve with Project Opal time management system. The most important is improving the reliability and integrity of historical shift data. The previous system caused a multitude of issues with not knowing who had original recorded shift data or who had modified it along the way before being received by payroll. This caused inconsistency between the dispatchers & accountants, as well as disagreements between employees and general management.

# requirements

## Functional

### High Priority

* System should manage shifting starting and ending (clocking in and clocking out)
* System should provide review data for employees
* System should allow managers to modify shifts

### medium priority

* System should show managers an overview of hours worked by all employees
* System should generate graphs & analytics based on historical data
* System should be able to automatically print cheques

### low priority

* System should show messages from administrators on the screen

## Usability

### high priority

* System should allow a user to clock-in or clock-out in at most 3 clicks
* System should have a simple interface with a minimum of technical jargon

### medium priority

* System should include help dialogs
* System should have multiple types of cues

## Reliability

### high priority

* System should remain active 99.15% of the time (6 hours of downtime a month due to critical failures)
* System should have adequate failure recovery protocols
* System should maintain data integrity as well as minimize data loss (barring electrical failure or other critical computer hardware malfunction)

### medium priority

* System should be separated so as to allow easy backup of the database and other data-critical components of the application

### low priority

* System should have multiple points of failure, rather than rely on a single machine

## Performance

### high priority

* System should be able to handle up to 100 active shifts
* System should be able to process more than 10,000 historical shifts
* System should be able to generate at least 5 reports daily

### medium priority

* System should be able to generate 3 large reports (>1 hour processing time) bi-weekly

### low priority

* System should be able to handle 10 active logins

## Security

### high priority

* System should have unique user accounts that are protected by a password
* System should encrypt all passwords upon entry and not store any passwords as plaintext
* System should not divulge passwords to administrators or technical staff
* System should prohibit access to particular controls by unprivileged users

### medium priority

* System should use multiple points of failure to minimize compromised machines.

## Platform

### medium priority

* System should be able to run on any machine that contains at least 512 MB of RAM and 1 GB of available storage
* System should have an off-terminal location for database backup
* System should consume low levels of bandwidth (100 – 500 kbps)

### low priority

* System should use a standard database manager to ensure data integrity

# Application Description

The application (Project Opal) seeks mainly to improve the reliability of recording shifts in JKS Enterprises. By using modern data recording techniques such as employing SQL databases and strict, data-cleansing forms, JKS Enterprises can regain confidence in the historical data of their shifts and be able to compensate their workers correctly and fairly. Project Opal also seeks to reduce human time involved in the process of managing employee scheduled time and assist human resources in creating accurate documentation for accounting purposes and generating a backlog of data for employees and management to review for strategic planning.

Project Opal overhauls an arduous and error-prone system by automating each step in the process and implementing safeguards and internal tests along the way. Project Opal also lets the regular users of the system (drivers) record their hours without confusion and provides information about their recent activities. The system is designed to be simple and easy-to-use for anyone, with no amount of technical knowledge required to get benefit.

The new system will be used by a variety of users within the organization- cab drivers, dispatchers and mechanics will be included in the regular users. Managers, bookkeepers and owners will be counted among administrative users.

Version 1 of Project Opal has a bevy of immediately useful features to JKS Enterprises. The user view of the system can record shifts, record vehicle numbers, give incremental or comprehensive review data. The user view also protects all user logins with passwords that are encrypted upon entry. Clocking in and clocking out automatically log out the user in order to speed use and reduce accidentally forgetting to log out. The data that is recorded is stored in a database for transportation and use.

The administrative view lets administrative users choose users and date ranges for automatic analysis. The administrative view shows previous shifts, a bar graph of shifts worked by month and a pie chart of day/night shift distribution. The administrative view is run as a web service that allows it to be accessible from a number of administrative machines and is completely separated from the user view- minimizing risk of compromised data.

## Use Case Diagram

Regular User

Administra-tor

## Review of Requirements

## Functional

### High Priority

* System should manage shifting starting and ending (clocking in and clocking out)
  + System supports as of version 1
* System should provide review data for employees
  + System supports as of version 1
* System should allow managers to modify shifts
  + System does not allow modification due to complexity of being SOX-compliant

### medium priority

* System should show managers an overview of hours worked by all employees
  + System shows more granular results. Not completed due to priority
* System should generate graphs & analytics based on historical data
  + System supports as of version 1
* System should be able to automatically print cheques
  + Feature rejected due to scope issues

### low priority

* System should show messages from administrators on the screen
  + Not implemented, deemed little use and too confusing

## Usability

### high priority

* System should allow a user to clock-in or clock-out in at most 3 clicks
  + System allows user to clock-in/clock-out in 2 clicks
* System should have a simple interface with a minimum of technical jargon
  + System uses visual indicators for relevant actions

### medium priority

* System should include help dialogs
  + Not enough time in cycle- priority for version 2
* System should have multiple types of cues
  + System supports as of version 1

## Reliability

### high priority

* System should remain active 99.15% of the time (6 hours of downtime a month due to critical failures)
  + System has not failed during testing, however duration testing not feasible
* System should have adequate failure recovery protocols
  + System atomically stores records without database locking
* System should maintain data integrity as well as minimize data loss (barring electrical failure or other critical computer hardware malfunction)
  + System can store database file in remote location

### medium priority

* System should be separated so as to allow easy backup of the database and other data-critical components of the application
  + System supports as of version 1

### low priority

* System should have multiple points of failure, rather than rely on a single machine
  + System separates user view & administrative view

## Performance

### high priority

* System should be able to handle up to 100 active shifts
  + System tested with over 100 active shifts
* System should be able to process more than 10,000 historical shifts
  + System tested with over 10,000 historical shifts
* System should be able to generate at least 5 reports daily
  + System can generate historical data almost instantaneously

### medium priority

* System should be able to generate 3 large reports (>1 hour processing time) bi-weekly
  + Assumed, however untested

### low priority

* System should be able to handle 10 active logins
  + System supports as of version 1

## Security

### high priority

* System should have unique user accounts that are protected by a password
  + System supports as of version 1
* System should encrypt all passwords upon entry and not store any passwords as plaintext
  + System supports as of version 1
* System should not divulge passwords to administrators or technical staff
  + System supports as of version 1
* System should prohibit access to particular controls by unprivileged users
  + System supports as of version 1

### medium priority

* System should use multiple points of failure to minimize compromised machines.
  + System is separated between user view and administrator view

## Platform

### medium priority

* System should be able to run on any machine that contains at least 512 MB of RAM and 1 GB of available storage
  + System supports as of version 1
* System should have an off-terminal location for database backup
  + System has separate flat file database
* System should consume low levels of bandwidth (100 – 500 kbps)
  + System consumes low levels of bandwidth by heavily utilizing Javascript

### low priority

* System should use a standard database manager to ensure data integrity
  + System uses SQLite, an industry accepted standard for SQL databases

# technical details

#### *User view*

* C#
* .NET Framework 4.5
* Windows Forms GUI toolkit

#### Administrator view

* Python 3.2.1
* Bottle web server
* Front-end work done using HTML, CSS & jQuery
* Chart.js used for graph generation

The user view and the application view share the same flat file database that is handled by SQLite, allowing them to be differing in languages and platforms but operate on the same data. This provides advantages to either view (user view not requiring internet access and having a standard GUI toolkit and the administrator view being accessible from any machine and being located separate from a client machine for data integrity).

## Concepts of note

### Secure.cs / secure.py

Both applications have portable code that is used for hashing passwords when used in the system. Since storing passwords in plaintext is a risk should the database file be leaked, all passwords are encrypted on entry into the system and are stored encrypted. The passwords that are to be verified are checked against their encrypted passwords- this allows no one to see someone’s secret password.

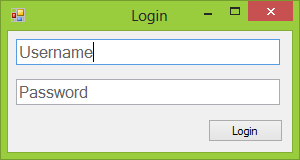
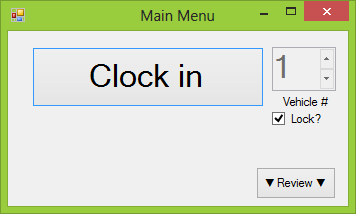
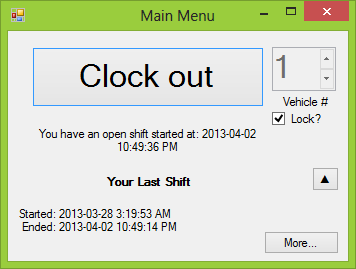
It is worthwhile to note that such an encryption can be defeated through a rainbow table attack, however since such an attack takes a good amount of time, it is deemed that the database isn’t visible enough to be susceptible.

### databaseconnection.cs / dbinterface.py

Due to record locking issues, communication with the SQLite database was abstracted into wrapper files that allowed for atomic (complete the entire operation or complete none of it) operations on rows. This moved some responsibility off of programmers to open and close their own connections and ensured data was transferred. This improvement greatly sped up development times.

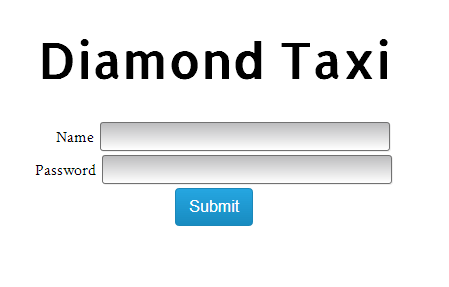
## Screenshots

### user view

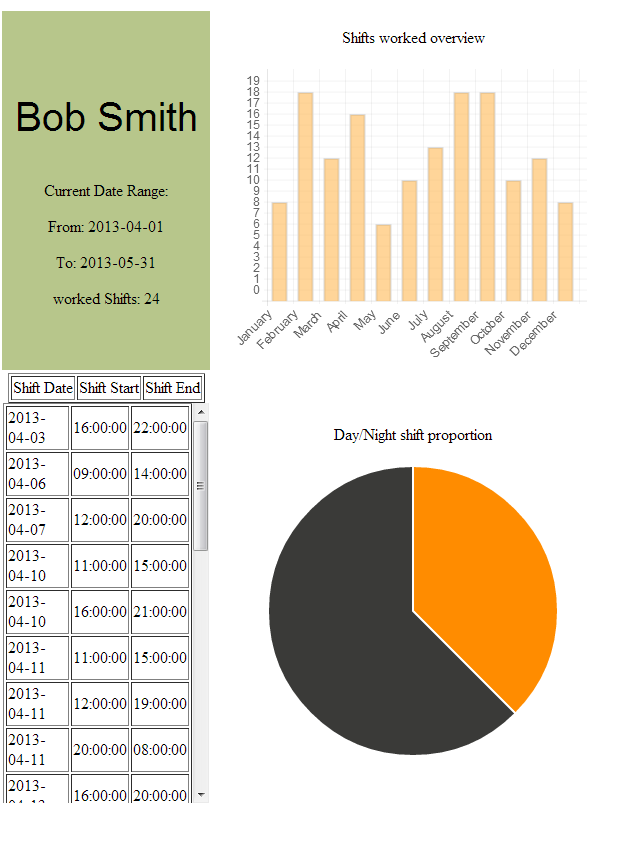
  

### 

### administrator view







## Application architecture diagram

### user view

### administrator view

Secure.cs

DatabaseConnection.cs

Logger.cs

Data Layer (C#/SQL)

Program.cs

Shift.cs

User.cs

Business Layer (C#)

Presentation Layer (Windows Form GUI Toolkit)

MainMenu\_Form.cs

ReviewShifts\_Form.cs

MessageBox\_Form.cs

Login\_Form.cs

## Class diagram

