**Software Sharks (Group 8)**

**Schedule Shark**

**Requirements Analysis V2**

**Authors:**

Kaitlin Anderson

Jeremy Warden

Josh Lewis

Han Chen

03/16/2016

**Version: 0.02**

**Table of Contents**

Overview……………….……………………………………….3

The Problem……………………………………………….3

Requirements…………………………………….......................4

User Requirements………………………………………...4

System Requirements……………………………………...5

Functional…………………………………………………6

Non-Functional Requirements…………………….………7

System Design………………………………………………….8

Use Case Diagram of Whole System..………..…………...8

Activity Diagram of Schedule Interaction..……….……....9

ERD of Database…………………………………….…...10

Change Log…..………………………………………………..11

Glossary…..…………………………………………….……..12

OVERVIEW

**-- PROBLEM --**

During the last few years, multiple members of our group have experienced poor scheduling techniques used for our various places of employment. It is a rather daunting task to balance everything that pertains to each employee such as availability and requesting off in order to formulate an accurate schedule. Additionally, to add an extra headache to the scheduling manager, they also have to create multiple schedules for various positions that are held. Managers already have enough work to attend to with daily problems that occur at the workplace, so we believe we can alleviate some of that stress with the implementation of Schedule Shark!

REQUIREMENTS

**-- USER REQUIREMENTS --**

**Primary: Kaitlin Anderson**

**Secondary: Jeremy Warden**

* **Employees**
  + Server
  + Bartender
  + Busser
  + Food-Runner
  + Cashier
  + Hostess/Host
  + Supervisor
* **Managers**
  + Floor Manager
  + Branch Manager
* **Manager Requirements**
  + Edit Schedule
    - Swap shifts
    - Remove employees
    - Add employees
    - Change times of shifts
  + Generate Registration Code for New Employees
  + Accept Request Off from Employees
  + Accept Availability Changes from Employees
  + Contact All Employees
* **Employee Requirements**
  + Give Availability
    - Day of availability
    - Time of availability
  + Request Time Off
  + Edit Availability
  + Contact employees within the same job title
* **Manager and Employee Requirements**
  + Registration
  + User Login
  + View Schedule
  + General Search of Employees

**-- SYSTEM REQUIREMENTS --**

**Primary: Jeremy Warden**

**Secondary: Kaitlin Anderson**

* **Cloud Storage**
  + Azure
* **LAMP Stack (stored on azure)**
  + Linux
    - Virtual Machine is powered by Linux, creating a safe environment for us to utilize the resources necessary to run our application
  + Apache
    - Web server where we will be hosting our web application
  + MySQL Database
    - Our web based application will be database driven, using user data in order to function properly
  + Python/PHP
    - We will be communicating between our controller and our model with a server side scripting language

**-- FUNCTIONAL REQUIREMENTS --**

**Primary: Han Chen**

**Secondary: Josh Lewis**

* **User Login**
  + On correct input, advances user to site.
  + On incorrect input, allows user to try again or change password.
* **Give Availability**
  + Store Employees availability
  + Use information for generating Schedule
  + Edit availability
* **Request Time off**
  + Send request off dates to manager in order for approval
  + Store date on approval
  + Use information for generating schedule
* **View Schedule**
  + Each type of employee will have ability to view the corresponding schedule
* **Edit Schedule**
  + Managers should be able to make changes to the schedule
* **Contact Other Employees**
  + Managers should be able to mass e-mail employees.
  + Employees should be able to contact similar employees, as well as their manager.
* **Adding New Employees**
  + Generate random code for employee to enter and gain access to system

**-- NON-FUNCTIONAL REQUIREMENTS --**

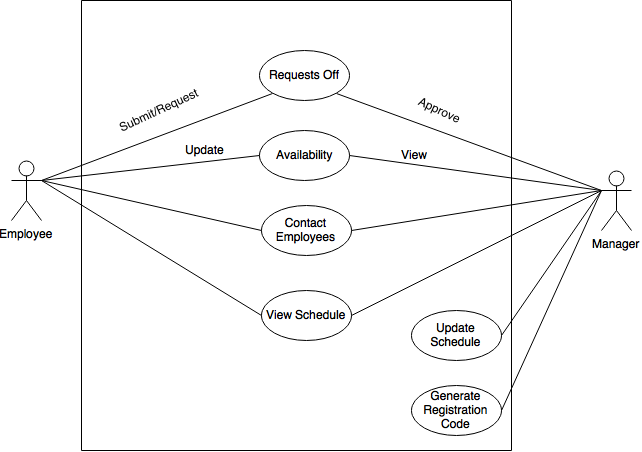
**Primary: Josh Lewis**

**Secondary: Han Chen**

* **Stability**
  + The program should be stable; it should have a one percent failure rate.
* **Efficiency**
  + The program should be fast and efficient, responding in under a minute to queries and requests.
* **Recoverability**
  + The program should recover gracefully from incorrect inputs and from system outages.
* **Database**
  + Database should be able to handle large amount of data and simultaneous requests.
* **Security**
  + System should be secure, not just anybody can register for an account in the system, and employees must register with an access code generated by a manager.
* **Platforms**
  + Should work on multiple web platforms including, IOS and Android web browsers.

SYSTEM DESIGN

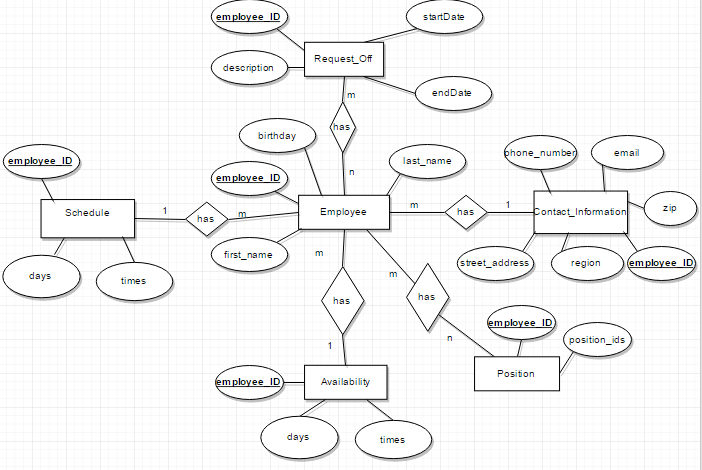
**-- USE CASE DIAGRAM OF WHOLE SYSTEM --**



**-- ACTIVITY DIAGRAM OF SCHEDULE INTERACTION --**

Macintosh HD:Users:kaitlinanderson:Library:Containers:com.apple.mail:Data:Library:Mail Downloads:74DE98FE-871F-4794-A042-D62B1234E862:Activity_DiagramV2.png

**-- ERD OF DATABASE --**



CHANGE LOG

|  |  |  |  |
| --- | --- | --- | --- |
| # | Date | By | Description |
| 01 | 03/10/2016 | All | Sprint one meeting: decide how tasks are divvied up |
| 02 | 03/15/2016 | Kaitlin Anderson &  Jeremy Warden | Create user requirements |
| 03 | 03/15/2016 | Jeremy Warden &  Kaitlin Anderson | Create system requirements |
| 04 | 03/15/2016 | Josh Lewis &  Han Chen | Create functional requirements |
| 05 | 03/15/2016 | Han Chen &  Josh Lewis | Create non-functional requirements |
| 06 | 03/15/2016 | Jeremy Warden | Cerate DDL, User Case |
| 07 | 03/15/2016 | Kaitlin Anderson | Create ERD |
| 08 | 03/17/2016 | Josh Lewis | Integrate the documents and diagrams, create table of contents |
| 09 | 03/17/2016 | Han Chen | Create change log and glossary |
| 10 | 04/10/2016 | Kaitlin Anderson | Update Requirements Analysis Document |

GLOSSARY

**Schedule**

A list of employees, and associated information, for example, position, working time, responsibilities for a given time period.

**User Requirements**

What the users expect the software to be able to do. The user requirements can be used as a guide to planning cost, timetables, milestones, testing, etc.

**System Requirements**

In order to work efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as system requirements and are often used as a guideline as opposed to an absolute rule.

**Functional Requirements**

It essentially specifies what the system should do. It specifies a behavior or function, for example, display the name, available time and edit the employees’ information, etc.

**Non-functional Requirements**

It essentially specifies how the system should behave and that it is a constraint upon the systems behavior. One could also think of non-functional requirements as quality attributes for of a system.

**Entity Relationship Diagram (ERD)**

Dealing with scheduling, involves a lot of data. We use this ERD diagram in order to give a pictorial representation of how our data will be stored. ERD’s use relationships between the data in order to store it more accurately and more clean.

**Data Definition Language (DDL)**

We take the ERD diagram and create a series of CREATE TABLE statements in SQL that allow our design to come to life on our Linux Virtual Machine.

**User Interface (UI)**

It is essential to the process that we show a rough draft of interface that our user will visually see. This allows us to get your approval on the scheme and also discuss within our group what layout will be the most effective for our application.