Attendance System for Multi-Shift Manufacturing Industries

BY

Mohammad Ali Azam
ID: 012-141-016

This Project Report Presented in Partial Fulfillment of the Requirements for the Degree of Master of Science in Computer Science & Engineering

UNDER THE SUPERVISION OF

Dr. Mohammad Nurul Huda
Professor & MSCSE Coordinator,
Department of Computer Science and Engineering
United International University

UNITED INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

DECEMBER 2017

APPROVAL

This Project titled **"Attendance System for Multi-Shift Manufacturing Industries",** submitted by Mohammad Ali Azam to the Department of Computer Science and Engineering, United International University, has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of MSc. in CSE and approved as to its style and contents. The presentation has been held on December 04, 2017

BOARD OF EXAMINERS

Dr. Mohammad Nurul Huda — **Supervisor**
Professor and MSCSE Coordinator
Department of Computer Science and Engineering
United International University, Dhaka

Mr. Mamun Elahi — **Examiner**
Asst. Professor and CCNA LMC
Department of Computer Science and Engineering
United International University, Dhaka

Dr. Mohammad Nurul Huda — **Ex-Officio**
Professor and MSCSE Coordinator
Department of Computer Science and Engineering
United International University, Dhaka

ACKNOWLEDGEMENT

We would like to express my heartiest thanks and gratefulness to almighty Allah for His divine blessing makes us possible to complete this project successfully.

We are grateful to and wish our profound our indebtedness to **Dr. Mohammad Nurul Huda**, **Professor and MSCSE Coordinator**, Department of Computer Science and Engineering, United International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of human resource management automation influenced us to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior draft and correcting them at all stages have made it possible to complete this project.

We would like to express our heartiest gratitude to Dr Mohammad Nurul Huda, Associate Professor and MSCSE Coordinator**,** Department of CSE, for his kind help to finish our project and to other faculty member and the staff of CSE department of United International University.

We would like to thank our entire course mate in United International University, who took part in this discussion while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our family members.

Abstract

HR departments of many manufacturing industries in Bangladesh are reluctant to prepare a roaster for daily laborers. Usually a manufacturing industry runs multiple shifts for workers. Workers' attendances are captured through the biometric devices such as fingerprint devices into a database which needs to be mapped worker to appropriate shift automatically. 'Automated Multi-Shift Attendance System' can map workers' shift automatically by analyzing attendance data without having a roaster. Furthermore, it can successfully analyze multiple shift on a single on which an individual worker works and produce intuitive report rather than just displaying First-In/Last-Out report.

Contents

LIST OF FIGURES

CHAPTER ONE

1.1 Introduction

In traditional attendance management system, application only produces attendance report based on first-in/last-out basis. Thus, even when a worker in a manufacturing industry works for more than 1 shift, say for example, in a morning shift and in night shift – traditional attendance management system fails to report correct output.

Furthermore, in Bangladesh, many manufacturing industries manage workers' shift verbally. That is, they are reluctant to deal with a roaster. Workers do not work in shifts in any defined manner.

1.2 Motivation

Usually manufacturing industries run multiple shifts a day; some shifts starts in a day and ends in another. For example, a shift may start at 10:00 PM on January 20, 2017 and end at 6:00 AM January 21, 2016. Thus, everything complicates the process of recording attendance for the HR managers. 'Automated Multi-Shift Attendance System' addresses all these issues and ease the task of recording attendance information, analyze attendance data, maps workers' shifts appropriately, and finally manages over time and leaves

1.3 Layout

The project has two modules, one is user and settings management and the other is reporting. There are 2 types of user in the system who can use the web application after logging into the system and their access privileges are managed through role based security. Following are the 2 types of users:

a) Super User
b) Admin User

CHAPTER TWO

LITERATURE OVERVIE

3.1 Introduction to Multi-Shift Attendance System

Most of the attendance system can cope up with a scenario where their employees work in a single shift. In manufacturing industries, in many cases, workers work in more than one shift (though not more than 2 shifts in a day). Moreover, a shift may start in a day and may end on another day. In this instance of multi-shift attendance shift, where there is a proximity device to capture attendance data, most of the software found now-a-day only calculate FIRST ENTRY IN ATTENDANCE DEVICE as entry and LAST ENTRY IN ATTENDANCE DEVICE as exit.

Thus, when a shift starts in a day and ends on another day, most of the software cannot calculate the attendance properly, especially in manufacturing industries. In other word, most of the software application that captures and calculates attendance are not suitable for manufacturing industries, but for corporate offices.

3.2 Problems

As already stated in introduction, here are couple of problems those are addressed by the proposed 'Automated Multi-Shift Attendance System':

- Workers' shifts are not fixed, and HR managers are unable to set shift while registering a new employee

- Industries have at least 3 shifts. They may have multiple shifts with varying length.

- Workers' shift assignment does not follow any static structure or guidelines. A worker may choose to work in a morning shift in a week and in evening shift in the next week. A worker may change his/her own shift even in the middle of the week.

- As workers' shifts assignment does not follow any guidelines, HR managers are reluctant to maintain a roaster. HR managers only log attendance information: workers' check in

and check out time, overtime, leave etc. based on their daily activity/inactivity

- Displaying daily reports and summary sometimes become complicated because a worker may start his/her day in each date and finish the shift in completely different date. For example, a worker may start his/her shift at 10:00 PM January 20, 2017 and check out at 6:00 AM January 2017. However, reports need to display this attendance as attendance of January 20, 2017

- A worker may work for 2 consecutive shifts. Time for the second shift should be considered as over time.

- Some employees are working in a fixed shift (e.g., 9:00 AM - 6:00 PM)

At a glance, above problems may seem to be simple. However, when capturing employee attendance logs who have no roaster, for a system, it is hard to distinguish their shifts.

3.3 Solutions

To solve the above-mentioned problems, a system needs to be developed that:

- Helps to setup dynamic shifts. A dynamic shift should have entry/exit time along with early entry and late exit boundary. Early and late exit boundary should not collide with other shifts

- Helps to set up fixed shifts for regular employees

- Helps to set up holidays, and leaves

- Helps registering employee with dynamic or fixed shift. When registering, leaving the shift options empty will indicate the system that the employee can freely choose to work in any of the dynamic shifts

- Can be integrated with another system that import the attendance log from biometric devices and export to the attendance system

- Should be able to detect duplicates while importing attendance logs

- Should be able to classify shifts based on shift configuration

- Should be able to detect overtime

- Should be able to detect employees in leave

Should be able to detect a worker shift that start in a date and ends in a different date (i.e., system should not try to find an exit for the same day and if not found, it should not log an 'exception', rather it should wait for the exit to be exhibited in the next day and act upon)

2.3.1 Admin User Feature

a) **Manage Employee:** Admin user can manage employees using the following features:
   i. **Add Employee**: He/she can add employee with basic information such as name, joining date, ID, department, designation, salary, weekend etc.
   ii. **Edit Employee**: He/she can later edit the employee to modify existing information
   iii. **Set Web Access**: He/she can enable/disable web access for the new employee. When super user enable web access, he/she provide the employee a username/password and also assign a user role (such as 'Admin')

b) **Manage Leave:** He/she can approve employee leave

c) **Settings:** Under settings, Admin user can setup the following:
   i. **Departmen**t: He/she can add or edit departments
   ii. **Branch**: He/she can add or edit branches
   iii. **Weekend**: He/she can define weekends
   iv. **Leave**: He/she can define leave type and default allotment
   v. **Holiday**: He/she can define holidays

d) **Report:** Under report, Admin user can view the following reports:
   i. Daily Attendance Report

ii. Monthly Attendance Summary
iii. Salary Report

2.3.2 Super User Feature

A super user can avail all the features that an Admin user has, and additionally the following feature:

a) **Add Admin User:** A super user can add admin user into the system

3.4 Summary

This chapter introduces the problem behind managing multi-shift attendance for same employee, especially in manufacturing industries and proposes a solution. Later, feature of 2 different types of users are listed.

CHAPTER THREE
REQUIREMENT SPECIFICATION

3.1 Use case Model

Use case model depicts each use of a system to which users or sub-systems interact. Thus from the user case point of view it is easier to understand the goal of a user or another sub-system on how they wish to interact with the system.

A use case diagram graphically represents its actors (user or another sub-system) and application features with which they communicate. Use cases can be divided into multiple use case diagrams to graphically represent different parts of the system. As such, same model may be seen in different use case diagrams provided that each instance of the model is consistent (if, for example, a model name is changed elsewhere, same name must be used to make it consistent across the diagrams).

Use case models may have packages to group related system features so that it can ease analysis, communication, development, navigation, planning and maintenance. Use case contains textual information to describe specification on how events flow within the use cases.

Use case serve the purpose as a primary specification of the overall system's funcational requirements. Based on this functional requirement, a use case should be worthy enough so that developer can derive appropriate design before development. It also helps to design test cases, user documentation and even planning for development iterations.

Use case model contains the following basic elements:

Actor
The user or sub-system that uses a business use case of the system being developed/proposed.

Use Case
It depicts the use case of the system from the business requirement perspective. It contains name, possibly with a formal description as specification.

Association
This notation it to describe the relationship among actors and use cases on how they communicate.

Generalizations
Generalization is used to reflect relationship between actors so that common properties can be re-used.

Dependencies
<<**extend**>> and <<**include**>> notations are used to depict dependencies between use case models. To depict a mode as an optional behavior of another model, we use <<**extend**>> dependency. To re-use a common behavior, we use <<**include**>> notation.

Use case diagram of client module are given bellow:

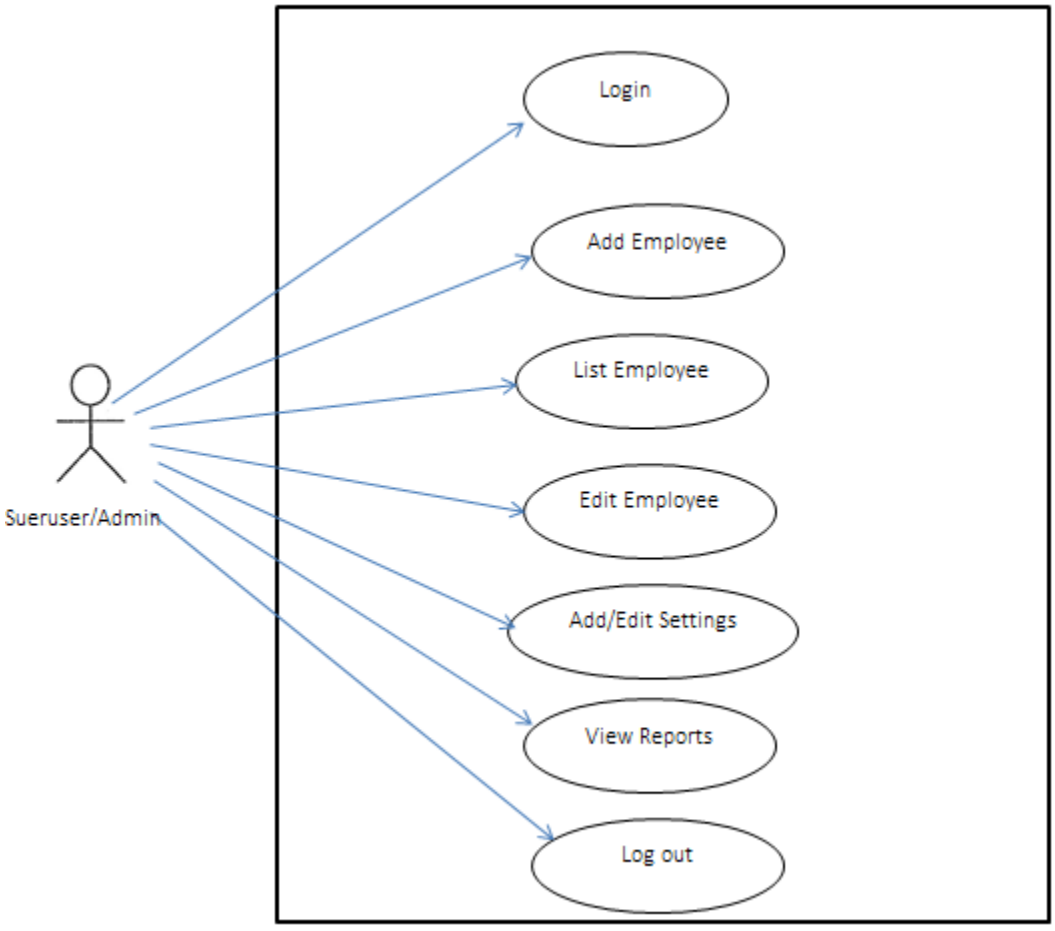


Figure 3. 1 Use Case Diagrams (Web Application)

3.2 Use Case Descriptions

MSAS-01: Add Employee

Action Name	Add Employee
Actor	1. Super User 2. Admin User
Pre-condition	1. User is logged in 2. Required settings such as Department, Designation, and Shift are created
Post Condition	1. Employee information is persisted
Action Flow	1. User open the form for adding new employee 2. Enters all required information such as: a) First name b) Last name c) Date of birth d) Joining date e) Employee Number 3. Clicks 'Save' to save entered information
Exception	1. In case of exception, a friendly error message should be shown to the end user 2. Error should be saved in server's error log file

MSAS-02: List Employee

Action Name	List Employee
Actor	1. Super User 2. Admin User
Pre-condition	1. User is logged in 2. There should be at least one user to be listed in the grid of employees
Post Condition	None
Action Flow	1. User goes to the grid of employees from User Management

	2. A grid with list of employees is displayed
Exception	1. In case of exception, a friendly error message should be shown to the end user 2. Error should be saved in server's error log file

MSAS-03: Edit Employee

Action Name	Edit Employee
Actor	1. Super User 2. Admin User
Pre-condition	1. User is logged in 2. Required settings such as Department, Designation, and Shift are created
Post Condition	1. Employee information is persisted
Action Flow	1. User goes to the grid of employee and select an employee to edit 2. Modifies any or all of the following information: f) First name g) Last name h) Date of birth i) Joining date j) Employee Number 3. Clicks 'Save' to save entered information
Exception	1. In case of validation error, proper validation message should be displayed beside the input fields 2. In case of exception, a friendly error message should be shown to the end user 3. Error should be saved in server's error log file

MSAS-04: Import Attendance from Device

Action Name	Import Attendance from Device
Actor	1. Attendance Importer Sub-system
Pre-condition	1. Device should be connected via network
Post Condition	1. Attendance log should be imported into the application database
Action Flow	1. Importer sub-system connects to the proximity device 2. Request attendance logs for a particular date 3. Reads the attendance record fetched from the device and re-arrange in a format compatible as the application expects and sort date/time wise 4. Saves the attendance log file in a pre-defined log file location 5. Invoke the application's web URL with a parameter mentioning the log file location to import the log 6. Application reads the log file and stored in the database
Exception	4. In case of application failure, error should be properly loged

MSAS-05: View Daily Attendance Report

Action Name	View Daily Attendance Report
Actor	1. Admin User
Pre-condition	1. User is logged in 2. Current or past date is selected to view daily report
Post Condition	2. Daily report for the particular date is displayed
Action Flow	1. User selects a current or past date from the interface 2. User optionally selects a department to view attendance report of the employee from that department 3. User clicks submit button to produce the report
Exception	1. In case of validation error, proper validation message should be displayed beside the input fields 2. In case of exception, a friendly error message should be shown to the end user 3. Error should be saved in server's error log file

CHAPTER FOUR
DESIGN

4.1 Design of Data Model

Data modeling is an essential and unavoidable step to translate conceptual data model expressed through the user requirements into physical data model. Physical data models involve interpreting data elements to group them in database tables. Stakeholders of a project may express what data needed to be stored, manipulated, queried (from different sources), and it forms a conceptual data model. Engineers then transform those data models into database tables, and relate tables using relationship to efficiently manage application data.

Essential modeled data is used to:

- Store application data as resource
- Help integrate other system, and
- Sometimes represent data imported from other integrated system

In our system, data has been modeled to store both application data and data from integrated system, especially data fetched from proximity devices.

Eventually our data has modeled in a relational database management (RDBMS), thus in a later section we will discuss database tables' structure and relationship among those tables.

4.2 Flow Chart

Flowcharts help to clearly depict the flow of data with input sources and output destinations. Flowchart also helps to form brainstorming ideas to defined strategies during the planning stage of an application. As flowchart is a graphical tool, stakeholders can easily visualize flow of the data within the application and can validate.

Definition

A flowchart graphically represents the sequence of operations or step-by-step progression of a programming or business model, using connecting lines and conventional symbols.

Function

Flowcharts can be used to determine the key points of a business or program model. It can also be used to connect those key points and establish relationships between processes.

Types

There are three types of flowcharts: high-level, detailed and matrix. While the high-level (or top-down) flowchart only gives a birds-eye view of the key points, the detailed and matrix flowcharts break down the processes and give more key points.

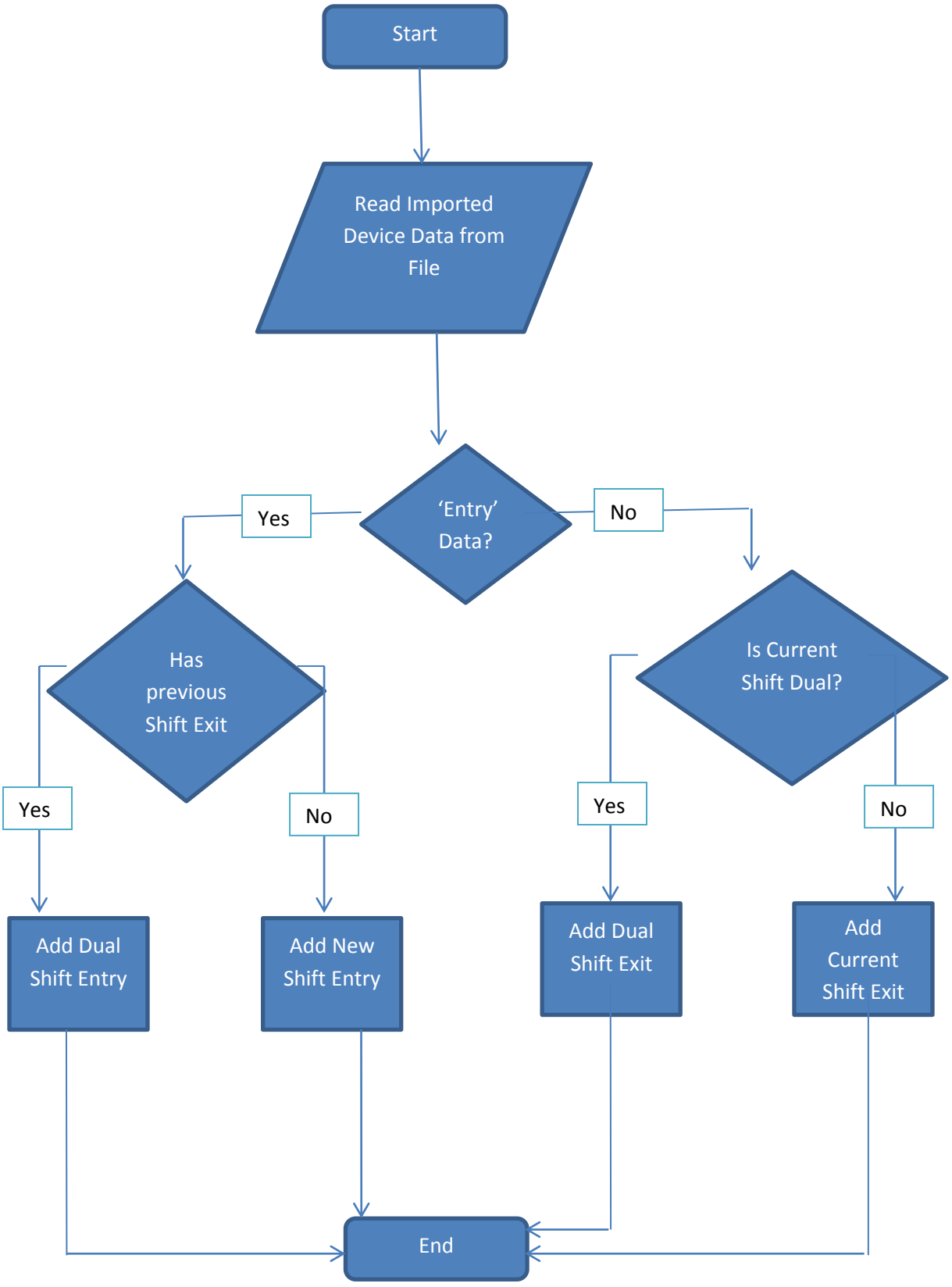
Uses

For a presentation where only the key points are needed, use a high-level flowchart; this is usually the case with business models. The detailed or matrix flowchart should be used when more information about the key points and their relationship to each other is needed.

Shapes

Shapes used in flowcharts include boxes, circles, diamonds and triangles. Each shape represents a certain action or conclusion of action.

Flowchart of Import Attendance



4.3 Database Design

Database design is the most important factor for a software design. We have used MySQL as relational database management system (RDBMS) to make the system very portable. Database is used to store the actual data. The diagram of a physical database is given below (use migrated MySQL database into MS Access Database to draw the below relationship).

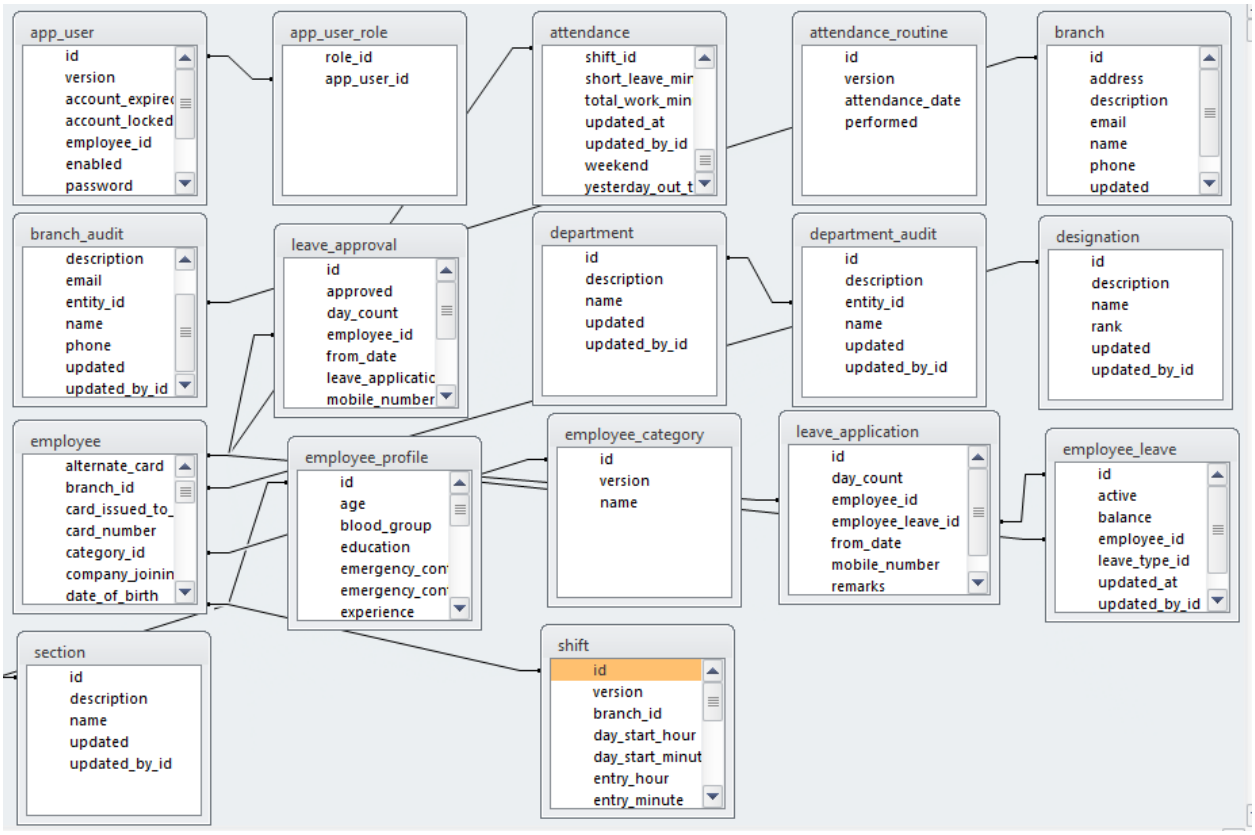


Fig 4.5: Database diagram

4.4 Summary

This chapter describes detail design of the application. The design of the data model, flowcharts of different procedure, and database design are described in this chapter.

CHAPTER FIVE
IMPLEMENTATION AND TESTING

5.1 Implementation Overview:

To develop the system, popular MVC framework has been followed. Groovy on Grails platform, which is basically build on top of Java technology and Spring framework has been chosen to build a robust and secure web application.

5.1.1 Methodology

Object Oriented Programming (OOP): To develop this centralized system, OOP concept is used to ensure the encapsulation, hiding, re-use and abstraction.

Model View Controller Architecture (MVC): To ensure data hiding, and implementing the secure access to database, MVC is used in this project.

Persistent Database Connection: To ensure, maximum database connection performance for load balancing and increasing performance, hibernate data access/persistency technology is utilized along with JDBC driver.

5.1.2 Technology

MySQL: To make the application simple and light weighted, we have used My SQL Access database.

Groovy On Grails: We have used groovy as a language (on top of Java) and Grails as a framework (build on top of popular Spring framework)

Jasper Report: We have used Jasper report to produced downloadable Excel report

After completing the Software Requirement Specification (SRS), Work Flow Diagram, Entity Relationship Diagram (ERD), the Graphical User Interface (GUI) is designed.

This software's GUI is designed according to the Constantine and Lockwood described collection of principles for improving the quality of user interface design. These principles are

- **The structure principle**. Design should organize the user interface purposefully, in meaningful and useful ways based on clear, consistent models that are apparent and recognizable to users, putting related things together and separating unrelated things, differentiating dissimilar things and making similar things resemble one another. The structure principle is concerned with overall user interface architecture.
- **The simplicity principle**. Design should make simple, common tasks simple to do, communicating clearly and simply in the user's own language, and providing good shortcuts that are meaningfully related to longer procedures.
- **The visibility principle**. Design should keep all needed options and materials for a given task visible without distracting the user with extraneous or redundant information. Good designs don't overwhelm users with too many alternatives or confuse them with unneeded information.
- **The feedback principle**. Design should keep users informed of actions or interpretations, changes of state or condition, and errors or exceptions that are relevant and of interest to the user through clear, concise, and unambiguous language familiar to users.
- **The tolerance principle**. Design should be flexible and tolerant, reducing the cost of mistakes and misuse by allowing undoing and redoing, while also preventing errors wherever possible by tolerating varied inputs and sequences and by interpreting all reasonable actions reasonable.
- **The reuse principle**. Design should reuse internal and external components and behaviors, maintaining consistency with purpose rather than merely arbitrary consistency, thus reducing the need for users to rethink and remember.

After completing all the above activities, the coding and implementation started.

Web Application

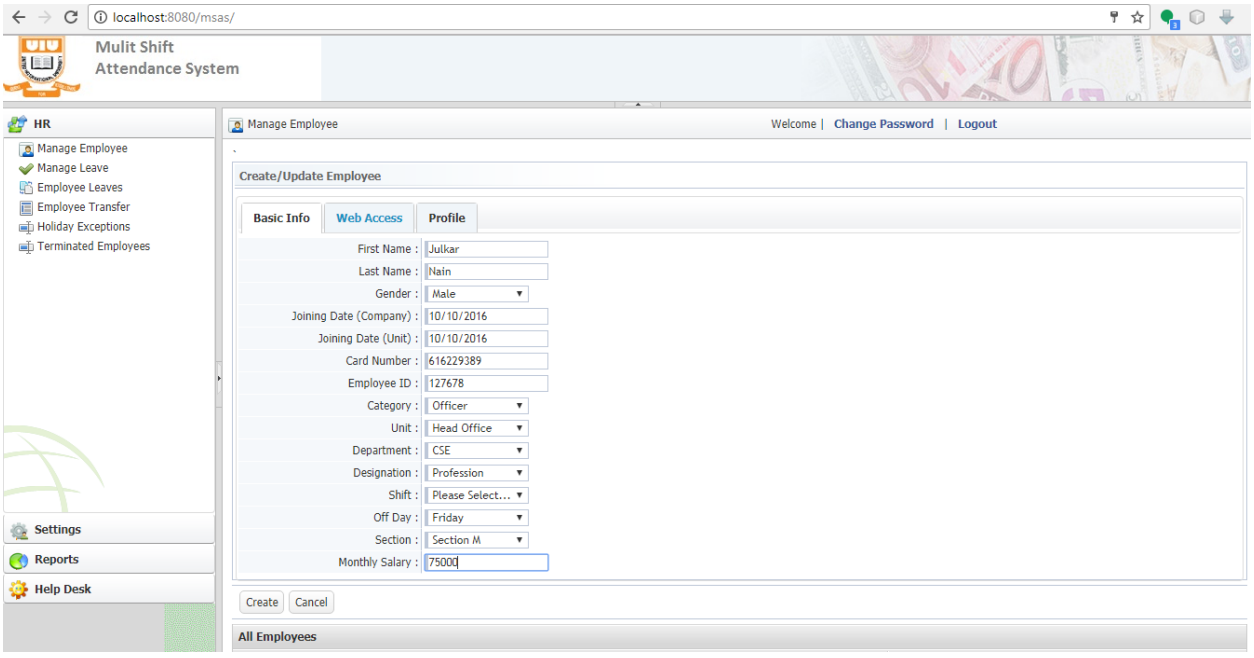
Web application offers mainly 3 parts:

1. To setup various configuration such as organization structure (branch, department, designation), shifts, holidays etc.
2. To register employees
3. To view reports

Following sections highlights some important parts of the web application:

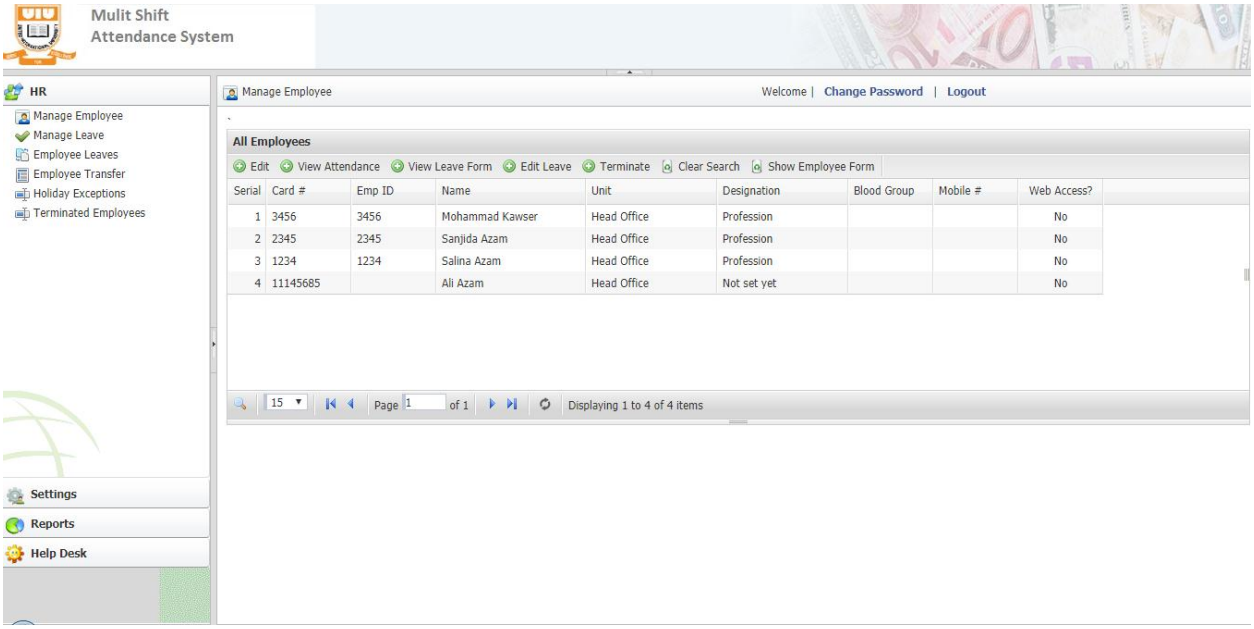
Add/Edit Employee

Following screen shot shows an interface using which a user can add or edit new employee into the system:



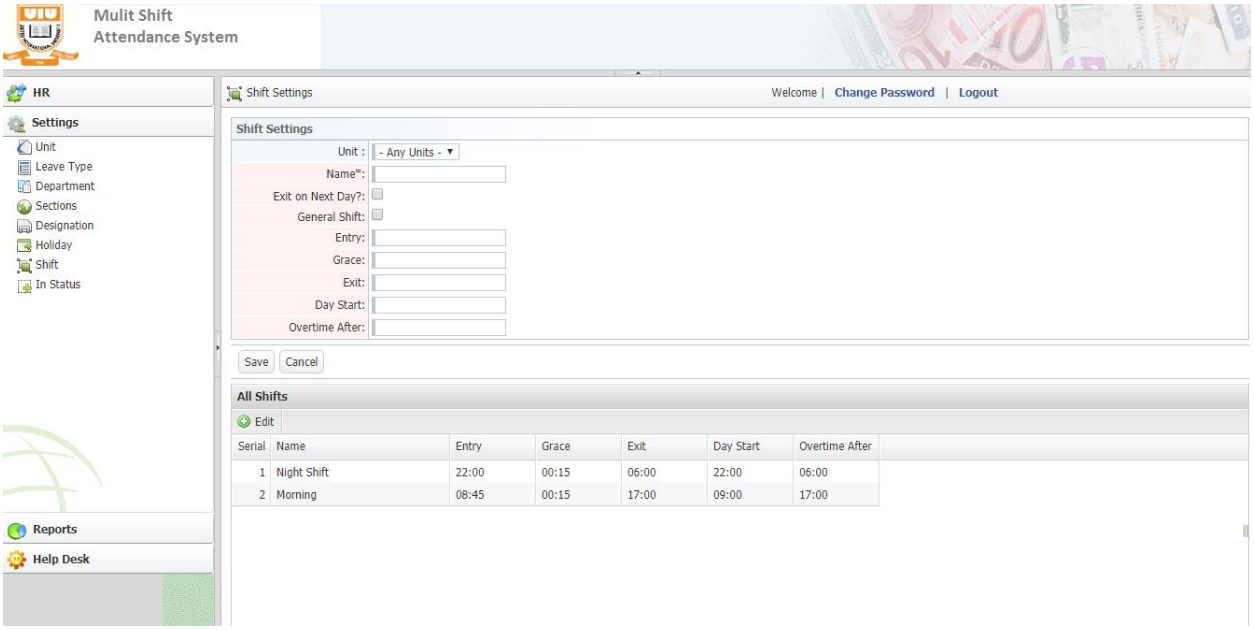
View List of Employees

Following screen shows an interface that shows a list of all employees already added into the system:



Shift Settings

An Admin user can setup shifts for the organization using the following interface:



View Attendance Report

User of the application can view attendance report for a particular date and branch using the following interface:

United International University
Multi Shift Attendance System

Report Name: Attendance Report
Unit: Head Office
Date: Sun, 16 July, 2017

Print this report

SL.	Emp. Id	Name	Department	Designation	Section	Category	In Time	Out Time	2nd IN	2nd OUT	Status
1	Not assigned	Ali Azam	Not set yet	Not set yet	Not set yet	Not set yet	NA	NA	NA	NA	ABSENT
2	1234	Salina Azam	CSE	Profession	Section M	Officer	NA	NA	NA	NA	ABSENT
3	2345	Sanjida Azam	CSE	Profession	Section M	Officer	NA	NA	NA	NA	ABSENT
4	3456	Mohammad Kawser	CSE	Profession	Section M	Officer	NA	NA	NA	NA	ABSENT

Total: 4			
PRESENT	0	LATE	0
ABSENT	4	OFF DAY	0
LEAVE	0		

Prepared By: Checked By: Head of Department:

4.2 Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Through the software testing business can asses the risks of the implemented system. Software testing is conducted to find defects in software implementation.

Software testing can be stated as the process of validating and verifying that a software program/application/product:

1. meets the requirements that guided its design and development;
2. works as expected;
3. can be implemented with the same characteristics.
4. satisfies the needs of stakeholders

Software testing, depending on the testing method employed, can be implemented at any time in the development process. Traditionally most of the test effort occurs after the requirements have been defined and the coding process has been completed, but in the Agile approaches most of the

test effort is on-going. As such, the methodology of the test is governed by the chosen software development methodology.

4.2.2 The Box Approach

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

White-Box testing: In a white box testing methodology, tester known in advance regarding the internal working and implementation details of the application under test (AUT). As such, the tester should have adequate knowledge in programming. Who designs a white box test case models design the input for the tests and also determine the outcome after executing each of the test cases

Techniques used in white-box testing include:

- API testing
- Code coverage
- Fault injection methods
- Mutation testing methods
- Static testing methods

Black-box testing

Black-box testing

In a black-box testing technique, tester only knows the business requirements without knowing the internal design or implementation details. That is, while using this technique, tester treats the

application under test as a Black-box. There are various black-box testing technique including but not limited to:

1. Equivalence partitioning
2. Boundary value analysis
3. All-pairs testing
4. Decision table
5. Use-case testing

Visual testing:

Object of visual testing is to represent the test result and evidence visually to the developers for ease of understanding. It greatly increase the clarity of the failure visually so that developer can understanding what was happening.

CHAPTER FIVE
IMPLEMENTATION AND TESTING

6.1 Conclusion

The purpose of the web application to integrate with proximity device to capture attendance log in a rather complex shift configuration of an organization where an employee may work for more than 1 shift in a day, and yet produce the correct attendance report.

By producing correct attendance report, it means, even when a shift overlaps a day, the web application can seamlessly calculate the attendance and report thereby.

6.2 Contribution

- Making the system web based so that many user can access the system simultaneously
- Making the system secured with login and authentication, security is role based
- Making the system to capable of producing real time report

6.3 Future Work

Following may be added to **smart cheque writer** in future:

1. Support Employee Self Service (ESS)
2. Accessible from mobile app, even when employee is on the go
3. Checking location when the employee is on leave or out for office work using GPS
4. More reports on the basis of user requirements.

References

1. Chitresh Saraswat and Amit Kumar, 2010. An Efficient Automatic Attendance System using Fingerprint Verification Technique - Vol. 02, No. 02, 2010, 264-269
2. O. Shoewu, Ph.D. and O.A. Idowu, B.Sc., Number 1. May 2012. Development of Attendance Management System using Biometrics. Volume 13.
3. Seema Rao and Prof.K.J.Satoa - April 2013. An Attendance Monitoring System Using Biometrics Authentication - Volume 3, Issue 4
4. Mateusz Mrozewski, 2009. Introduction to Grails