Integration Project

Team Name

List all team members and their CBU IDs here

Date:

Version:

# Table of Contents

Contents

[Table of Contents 2](#_Toc162256810)

[2 Introduction 4](#_Toc162256811)

[2.1 Project Summary 4](#_Toc162256812)

[2.2 Background 4](#_Toc162256813)

[2.3 Project Scope 4](#_Toc162256814)

[2.4 Assumptions 4](#_Toc162256815)

[2.4.1 General Assumptions 4](#_Toc162256816)

[2.4.2 Data Assumptions 4](#_Toc162256817)

[2.4.3 Security Assumptions 4](#_Toc162256818)

[2.4.4 User Interface Assumptions 4](#_Toc162256819)

[3 Functional Objectives 4](#_Toc162256820)

[3.1 High Priority 4](#_Toc162256821)

[3.2 Medium Priority 5](#_Toc162256822)

[3.3 Low Priority 5](#_Toc162256823)

[4 Analysis 5](#_Toc162256824)

[4.1 System Use Case Diagram 5](#_Toc162256825)

[4.2 Use Case Personas 5](#_Toc162256826)

[4.2.1 {Persona 1} 5](#_Toc162256827)

[4.2.2 {Persona 2} 6](#_Toc162256828)

[4.2.3 {Persona 3} 6](#_Toc162256829)

[4.3 Use Case Descriptions (for selected cases) 6](#_Toc162256830)

[Add Employee 6](#_Toc162256831)

[4.4 Information Flow Diagram (IFD) 7](#_Toc162256832)

[5 Conceptual Design 7](#_Toc162256833)

[5.1 Conceptual Model (EER) 7](#_Toc162256834)

[5.2 Tasks 7](#_Toc162256835)

[5.2.1 Task 1 7](#_Toc162256836)

[5.2.2 Task 2 7](#_Toc162256837)

[6 Logical Design 7](#_Toc162256838)

[6.1 Logical Data Model (ERD) 7](#_Toc162256839)

[6.2 Abstract Code (w/ SQL) 8](#_Toc162256840)

[6.2.1 Task 1 8](#_Toc162256841)

[6.2.2 Task 2 8](#_Toc162256842)

[7 Physical Implementation 8](#_Toc162256843)

[8 Appendix 8](#_Toc162256844)

[8.1 Supporting Information for the Conceptual Data Model (EER) 8](#_Toc162256845)

[8.1.1 Sample Data 8](#_Toc162256846)

[8.1.2 Functional Dependencies 8](#_Toc162256847)

[8.1.3 Data Dictionary 9](#_Toc162256848)

[8.2 Database Creation Script 9](#_Toc162256849)

[8.3 Sample Data Insertion Script 9](#_Toc162256850)

[8.4 Functional Demonstration Script(s) 9](#_Toc162256851)

[8.5 Demonstration Video Script 9](#_Toc162256852)

[Change History 10](#_Toc162256853)

# Introduction

## Project Summary

## Background

## Project Scope

## Assumptions

### General Assumptions

{place general assumptions for the overall system here and change the MS Word style to Normal; if there are none then state that to confirm that there was thought put into potential assumptions and that none were identified}

### Data Assumptions

#### Health Plan Data

The table containing available health plans and annual costs for adults and children will be assumed to be provided. It will be placed in the data system for use in the payroll calculation. The addition of a system to add/edit available health care plans is a future development opportunity.

### Security Assumptions

This system is for demonstration purposes only and will not employ any security features that would be designed to prevent malicious activity. This system should not be deployed into a production environment without additional security measures in place.

### User Interface Assumptions

This system is being designed as the data layer and, as such, will have no user interface provided. There will be input documents and output documents that may come from or feed systems that make use of user interfaces but the development of those user interfaces is beyond the scope of this system.

{place other assumption catagories here (i.e., Functional Assumptions, Performance Assumptions, etc.). Delete this placeholder if no other assumptions are included}

# Functional Objectives

List the specific functions that the final data system should be able to achieve. Organize functional objectives in terms of priority.

## High Priority

## Medium Priority

## Low Priority

# Analysis

## System Use Case Diagram



See <https://www.lucidchart.com/pages/uml-use-case-diagram>

## Use Case Personas

{Personas should be generic “types” that describe actors that provide data to or consume data from the system. Each persona (actor) that appears in the system use case diagram should be defined here.}

### {Persona 1}

Describe this persona and what makes it unique from the other personas defined for the system. P

### {Persona 2}

Describe this persona and what makes it unique from the other personas defined for the system. P

### {Persona 3}

Describe this persona and what makes it unique from the other personas defined for the system. P

## Use Case Descriptions (for selected cases)

Notes:

* For all use cases, the system will assume that the authentication and authorization are handled by the interface collecting data (not part of this system).

### Add Employee

|  |  |
| --- | --- |
| Use Case Name: | Add Employee |
| Use Case Persona: | Human Resources Administrator |
| Summary: | This use case allows a human resources administrator to add a new employee to the system. |
| Basic Flow: | 1. Administrator opens form to add new employee (form is EXTERNAL to this system)    1. Enter Employee Name (Title, First, Middle, Last, Suffix)    2. Enter Employee SSN    3. Enter Employee Type (Salaried or Hourly)    4. Enter Employee Start Date    5. Enter Employee Annual Salary or Hourly Rate (depends upon employee type)    6. Add Employee Dependents       1. Dependent name       2. Dependent SSN       3. Dependent type (spouse or child)    7. Enter Employee Health Plan Choice 2. Submit all data from populated form 3. Validate data    1. All required fields defined    2. Confirm employee is NEW (i.e., SSN does not already exist in system)    3. Confirm that start date is > current date 4. Insert Employee (Employee ID assigned) |
| Alternative Flows: | Step 3b:   * If employee is NOT new then administrator is informed that employee already exists and existing employee ID is provided. * Administrator shifts to the Edit Employee use case. |
| Extension Points: | New Employee Data Entry Form |
| Preconditions: | * Employee type and salary/hourly rate has been defined. * Employee has provided all personal information (including a list of all dependents and health plan choice). |
| Postconditions: | Employee can clock in (if hourly) and will be included in payroll reports issued after start date. |
| Business Rules: | * All employees and dependents are fundamentally identified using their social security number (SSN) to avoid entering the same employee (or dependent) into the system more than one time. * Employees must choose from available health plans. * Salaried employees have an annual salary. * Hourly employees have an hourly rate. * Employees cannot be both hourly and salaried at the same time. * Employees can be converted from hourly to salaried using the Edit Employee use case.   + Effective date of conversion MUST be the first day of the month (new pay period) |

## Information Flow Diagram (IFD)

A diagram showing information (DATA) flow between all documents / clients of the system and the data system.

A diagram of a database

Description automatically generated

# Conceptual Design

## Conceptual Model (EER)

Sample data, functional dependencies for each entity, and data dictionary defining all terms contained within the EER show below are all provided as part of Appendix 8.1.

A screenshot of a computer screen

Description automatically generated

## Tasks

A description of tasks that will need to be programmed to enforce business rules that cannot be enforced using a data model.

### ConfirmTimeCardApprovalData

The time card approval process must produce data that is consistent with the type of employee (i.e., Regular and Overtime Hours for hourly and Regular and Bonus Pay for Salaried). Note: it is possible for an employee’s status to be changed at any time. The output of the time card approval status must indicate if the numbers being provided are hours or pay AND must the data provided must match what is expected based on the current employee job status.

# Logical Design

## Logical Data Model (ERD)

The logical entity relationship model is the final model created by mapping the Conceptual Model (EER) provided in section 5.1 to a set of relations with all relationships defined using foreign keys. This data model defines how the conceptual model will be implemented within a DBMS. The logical model should not assume any particular dbms. The logical model should define the following database components:

* Relations in 3rd Normal Form (3NF)
* Primary keys for all relaions
* Foreign keys for all relationships
* Constraints needed to enforce business rules

## Abstract Code (w/ SQL)

The abstract code defininig logic to implement tasks defined in section 5.2.

### Task 1

Abstract code implementing logic for task 1.

### Task 2

Abstract code implementing logic for task 2.

# Physical Implementation

Summarize the assumptions used to generate the database creation script provided in Appendix 7.3. Include reasoning used to design/implement the following database components:

* Constraints (other than referential integrity constraints)
* Indexes

# Appendix

## Supporting Information for the Conceptual Data Model (EER)

The conceptual data model defines entities that make up the proposed data system and the relationships connecting those entities. Unique identifiers should be specified for each entity.

### Sample Data

#### TimeCardApprovalData

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| EmployeeSSN | PayPeriodID | DataType | Regular | BonusOvertime |
| 122472347 | 20240101 | Hours | 80.0 | 0.0 |
|  |  |  |  |  |

#### Raw Output Data

#### Entity Data

### Functional Dependencies

#### Table 1

#### Table 2

### Data Dictionary

#### Employee

|  |  |  |
| --- | --- | --- |
| Attribute type | SQL data type | Description |
| First Name | CHAR(N) | Variable-length name of up to 30 characters representing employee’s legal first name. Ex: John, Jonathan, etc. REQUIRED |
| Middle Name | CHAR(N) | Variable-length name of up to 30 characters representing employee’s legal middle name (may be NULL or initial only). |
| Last Name | CHAR(N) | Variable-length name of up to 30 characters representing employee’s legal last (or family) name. REQUIRED |
| Suffix | CHAR(N) | Variable-length suffix of up to 10 characters representing employee’s legal suffix (may be NULL or initial only). Ex: Jr., Sr., III, etc. |
| Title | CHAR(N) | Variable-title of up to 10 characters representing employee’s preferred title (may be NULL or initial only). Ex: Mr., Mrs., Ms., Dr., etc. |
| SSN | Integer | Social Security Number. Store as integer (no hyphens). REQUIRED |
| StartDate | Date | Year, month, and day (time not required) that employee started working for the company. REQUIRED |
| Annual Salary | Decimal(11,2) | Annual salary for salaried employees. REQUIRED for salaried employees. |
| Hourly Rate | Decimal(11,2) | Hourly rate for hourly employees. REQUIRED for hourly employees. |

## Database Creation Script

A series of SQL commands that will create a physical implementation of the database on a specific DBMS (i.e., MySQL). It is acceptable to provide this as a separate SQL file with the name of the file specified here.

## Sample Data Insertion Script

A series of SQL commands that will successfully (i.e., without error) popullate the database with sample data that can be used to demonstrate all functionality. It is acceptable to provide this as a separate SQL file with the name of the file specified here.

## Functional Demonstration Script(s)

A series of SQL commands that successfully demonstrate each use case defined in this specification. Each script should clearly identify the before and after state of the data as part of the script. It is acceptable to provide this as a separate SQL file with the name of the file specified here.

## Demonstration Video Script

Provide a “movie script” summary of what will be demonstrated in the final video.

Change History

|  |  |  |
| --- | --- | --- |
| Version | Date | Description |
| 1.00 | 2/26/2024 | * Copied over from previous year * Removed sections that will no longer be required * Cleaned up all sections to allow for development from scratch |
| 1.01 | 3/25/2024 | * Added section 2.4 (and subsections) to contain any assumptions. * Added MS Word style for template prompts. * Added Change History * Updated TOC options to include Change History |