‘2018

**Employee Evaluation and Rating Calculation using Sentiment Analysis**

**Summer Training .NET Project**

**Aryaman Maheshwari**

**Deep Jethloja**

**Abstract**

The project provides the client a service for the admin-manager-developer model to communicate efficiently and also provides an automatic rating for every developer taking into consideration the feedbacks of his manager and all the other developer.

The project used ADO.NET entity framework based models and is structured upon as a standard MVC-structure written in C# .

**Data Flow**

Upon hitting the Login Page, the UserName and Password determines the type of user using the application. It can be an Admin, a manager, or a developer. It is assumed that there is only one Admin, and that a manager can get as many developers in his team, but a developer has only one manager.

The Admin once verified with the username and password can proceed upon to manage managers (CRUD) , allocate them a task (A Task is defined with a mangerID, it is to be allocated to, a deadline, and a summary about that task), View the progress of currently-going on, incomplete tasks and View the history of previously allocates task and see Ratings of all the developers.

The Manager can in a similar manner manage his developers (CRUD), allocate them sub-tasks and give them feedbacks, view their progress, view his own history and can see the ratings of all the developers he is in-charge of.

The developer can see his ratings, provide feedbacks to his co-workers and see his previously completed sub-tasks.

**Back-end Logics**

The applications has tables on admin, manager and developers, separately, it has a table of the feedbacks of all the employees under all the managers and it has tables for storing sub-tasks and tasks with the relevant column for determining if the task/sub-task is active or not. There is a trigger in the feedback table which handles the insert queries; it updates the rating of the relevant developer according to the newest feedback, taking the average of all of the feedbacks in the history about the developer.

We have used the ParallelDots API for determining the positivity in the feedback string for the developer. It is a decimal number between 0-1 and is saved with every feedback using sentiment analysis. Rating of the developer is the average value of positivity\*100, giving the developer a rating out of hundred.

A separate controller for the login page redirects to the right user login. Every three type of user have their distinct controller and each CRUD model has its distinct controller. The application has one edmx model updated from database which is auto-generated by the visual studio itself.

The application uses Stored procedures in the database and not inline, for doing various features like verifying login credentials and determining what kind of users wants to login, determining which task/sub-task is currently active or not, making sure that other a manager cannot know about another managers developers etc.

The application uses iframe for showing all the views for a user with a sidebar navigation.

**Applications**

The application enables an organization or a firm with huge number of employees to function efficiently and facilitates smooth communication within the employees and between the employees and the manager.

Also the rating system for the developers helps the managers to analyse the performance of the developers working under him and also encourages the employees to perform better to improve their own ratings.