Employee Portal – Solution Approach

Business Story

Our client needs a software system to aid his employees. We've been hired to implement the solution. The client wants the system to be usable only by authenticated users. Also, he wants some extra functionality to be accessible for some users which he calls admin.

The key features that the client has in mind are:

1. A way to send public notices to all users.
2. A way to manage issues. Users should be able to raise issues, which the admin users can then take up and work on.
3. A way to manage employee details. Admin users should be able to update employee details, add new employees etc. All users should be able to search other employees by various criteria.

DAC View

Find below the suggested implementation of various DAC modules:

1. Login –
    a. UserDAC
        i. IUserDTO UserDAC.Login(username, password) – Checks if a user exists with specified username and password and termination date is set to null (username is EmailId of the employee)
            1. Yes – return UserDTO for the username with it's EmployeeDTO property set to corresponding employee.
            2. No – return null.

2. My Profile –
    a. UserDAC
        i. IUserDTO UserDAC.UpdateProfile(IUserDTO) – Update Employee Table and User Table accordingly.

3. Notices –
    a. NoticeDAC
        i. Non – Admin Users
            1. IList<INoticeDTO> NoticeDAC.GetCurrentNotices() – Get all the notices for which today's date lies between start and expiration date of the notice and IsActive is true.
        ii. Admin Users
            1. IList<INoticeDTO> NoticeDAC.GetActiveNotices() – Get all the notices with IsActive set to true.

2. bool NoticeDAC.DeleteNotice(noticeId) – Set IsActive false for the notice of the given noticeId and return true if IsActive was set to false successfully otherwise return false.
3. INoticeDTO NoticeDAC.UpdateNotice(INoticeDTO) – Update the notice table accordingly, Is Updated successfully
    a. Yes – return the updated NoticeDTO
    b. No – return Null
4. INoticeDTO NoticeDAC.CreateNotice(INoticeDTO) – Create a new entry in the notice table , if inserted successfully
    a. Yes – return the inserted NoticeDTO
    b. No – return null

4. Issues –
    a. IssueDAC
        i. Non - Admin Users –
            1. IList<IIssueDTO> IssueDAC.GetAllIssuesByEmployeeId(EmployeeId) – Return all the issues, get information from the issue table and issue history table (get assigned to and status from the latest issue history entry of the corresponding issue), which are active and are posted by the given EmployeeId.
            2. bool IssueDAC.DeleteIssue(IssueId) – Set IsActive false for the issue of the given IssueId and return true if IsActive was set to false successfully otherwise return false.
            3. IIssueDTO IssueDAC.UpdateIssue(IIssueDTO) - Update the issue table (changes only in Issue table , no changes in Issue history table)
                a. Yes – return the updated IssueDTO
                b. No – return Null
            4. IIssueDTO IssueDAC.CreateIssue(IIssueDTO) – Create a new entry in issue and issue history table
            5. IIssueDTO IssueDAC.GetIssue(int issueId) – Return IssueDTO corresponding to the passed issue id. The IssueDTO should also contain nested inside it list of IIssueHistoryDTOs. Refer mockups in business story document for reference.
        ii. Admin Users
            1. IList<IIssueDTO> IssueDAC.GetAllActiveIssues() –
            Return all the active issues, get information from the issue table and issue history table (get assigned to and status from the latest issue history entry of the corresponding issue).
            2. IIssueHistoryDTO IssueDAC.UpdateIssueByAdmin(IssueHistoryDTO) – This should make an insertion in the IssueHistory table against an entry in the Issue table. If successfully done, return the relevant IssueHistoryDTO else return null.

3. IIssueDTO IssueDAC.GetIssue(int issueId) – Same as of non-admin Users. No Need to make a new one.

5. Employees
    a. Non-admin Users
        i. IList<IEmployeeDTO> IUserDAC.SearchEmployees(ISearchEmployeeDTO, bool checkTerminationDate)
        Returns a list of employees matching the search criteria specified in properties of SearchEmployeeDTO. The following rules will be used for search –
            1. All the properties having non-null values will be used for search.
            2. If no search criteria is defined, no results are returned.
            3. If multiple search criteria are used, results will be "OR" of all results
            4. Admin will be able to search users with termination date set to non-null, while other users will not be able to.
    b. Admin Users
        i. IList<IEmployeeDTO> IUserDAC.SearchEmployees(ISearchEmployeeDTO, bool checkTerminationDate) – Same as in non-admin users, but checkTerminationDate variable will be passed false.
        ii. IUserDTO IUserDAC.UpdateUser(IUserDTO) – Updates the user information and related employee information as defined in requirements and returns the updated information for successful update. In case of failure null is returned.
        iii. IUserDTO IUserDAC.CreateUser(IUserDTO) – Makes an insert in the User and Employee table and return corresponding IUserDTO for successful operation. In case of failure returns null.
6. Miscellaneous
    a. IList<IDepartmentDTOs> IDepartmentDAC.GetAll() – Used wherever a list of departments is required
    b. IDepartmentDTO IDepartmentDAC.GetDepartment(int departmentId) – Used wherever a specific department is required.
    c. IUserDTO IUserDAC.GetUserByEmailId(string emailId) – This will be used while creating or editing a user to make sure that every user have unique emailIds.