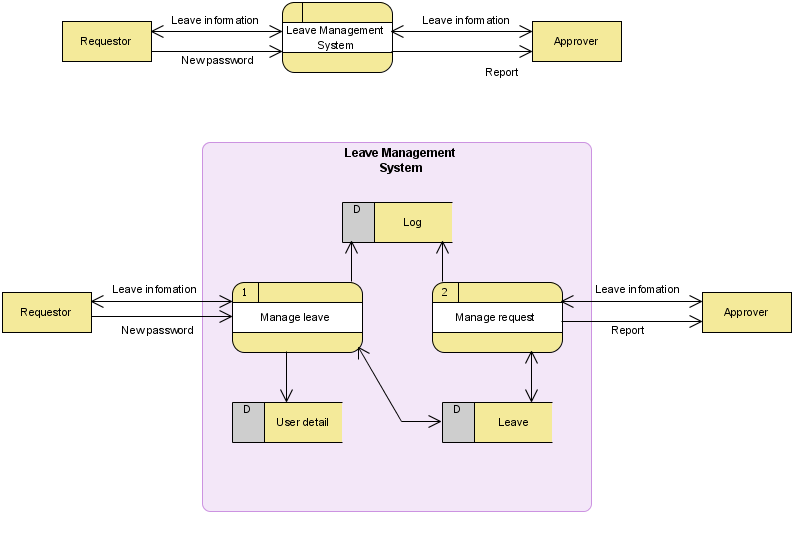
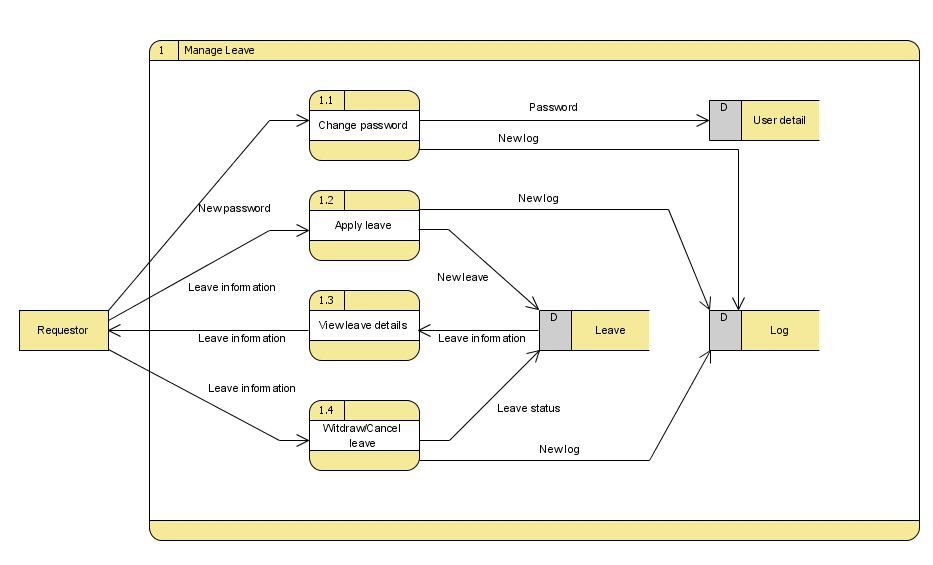
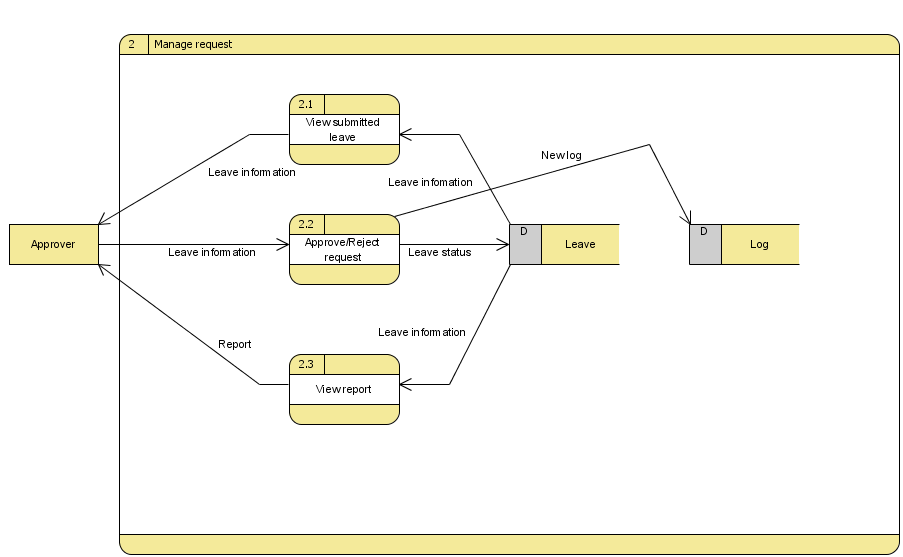
**DFD**





**ERD**

**Database Design**

Database: LeaveManagementSystem

Tables Summary: Leave, User, Log, Position

Table Details:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Leave | | | | |
| Name | DataType | Constraints | Null | Comment |
| LeaveID | integer(10) | PKUnique | No | Auto increment |
| UserID | integer(5) | FK ([User.UserID](#nCVBPJSGAqACBDN.)) | No | Leave’s owner |
| DateStart | date |  | No |  |
| DateEnd | date |  | No |  |
| State | integer(2) |  | No | State of leave |
| Subject | varchar(30) |  | No |  |
| Reason | varchar(100) |  | No |  |
| Communication | varchar(50) |  | No | How to contact |
| CurrentLe-aveDays | integer(2) |  | No | Leave policy may change in future so this field is used to review leave in past |
| Date | date |  | No | Day when applying leave |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| User | | | | |
| Name | DataType | Constraints | Null | Comment |
| UserID | integer(5) | PKUnique | No | Auto increment |
| Username | varchar(30) | Unique | No |  |
| Password | varchar(50) |  | No |  |
| Fullname | varchar(30) |  | No |  |
| SuperiorID | integer(5) | FK ([User.UserID](#nCVBPJSGAqACBDN.)) | No |  |
| PositionID | integer(10) | FK ([Position.PositionID](#zrDBPJSGAqACBDPV)) | No |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Log | | | | |
| Name | DataType | Constraints | Null | Comment |
| LogID | integer(10) | PKUnique | No | Auto increment |
| UserID | integer(5) | FK ([User.UserID](#nCVBPJSGAqACBDN.)) | No |  |
| Time | time(7) | Unique | No |  |
| Action | varchar(30) |  | No |  |
| LeaveID | integer(10) | FK ([Leave.LeaveID](#nCVBPJSGAqACBDN.)) | Yes | For leave-related action |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Position | | | | |
| Name | DataType | Constraints | Null | Comment |
| PositionID | integer(10) | PKUnique | No |  |
| PositionName | varchar(15) |  | No |  |
| LeaveDays | integer(2) |  | No |  |

**System Architecture**

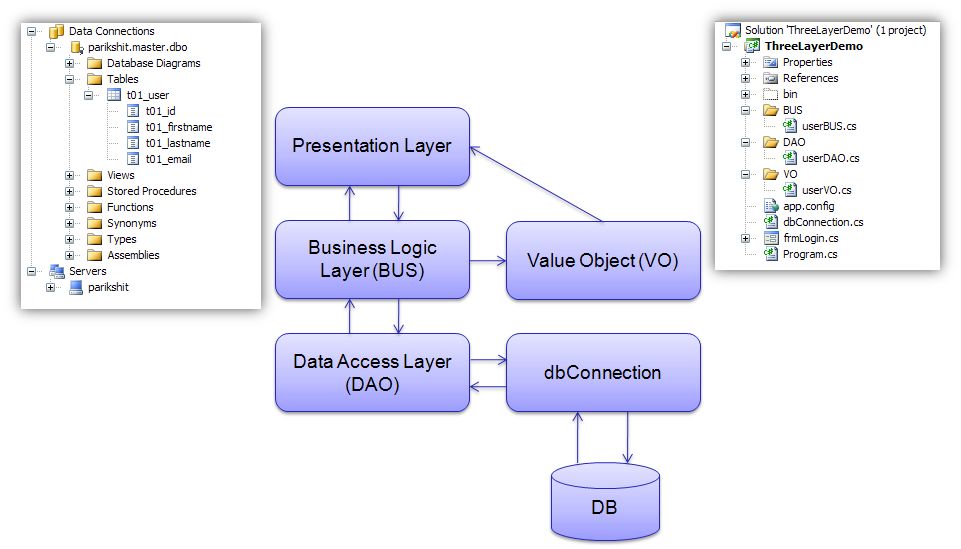
System architecture of our project is: 2-tiers and 3-layers

***2-tiers Architecture***



In the two-tier model, a Java application talks directly to the data source. This requires a JDBC driver that can communicate with the particular data source being accessed. A user's commands are delivered to the database or other data source, and the results of those statements are sent back to the user. The data source may be located on another machine to which the user is connected via a network. This is referred to as a client/server configuration, with the user's machine as the client, and the machine housing the data source as the server. The network can be an intranet, which, for example, connects employees within a corporation, or it can be the Internet.

***3-layers Architecture***



3 main layers are: Presentation, Business and Data. For example in figure:

**dbConnection**

This class is mainly used to do the database activity like Select, Update and Delete query to database. It also checks if the database connection is open or not. If database connection is not open, then it opens the connection and performs the database query. The database results are to be received and being passing in Data Table in this class.

This class takes the database setting from the app.config file so it’s really flexible to manage the database settings.

**Database Access Layer**

Database Access Layer (DAO) builds the query based on received parameters from the Business Logic Layer and passes it the dbConnection class for execution. And simple return results from the dbConnection class to Business Logic Layer.

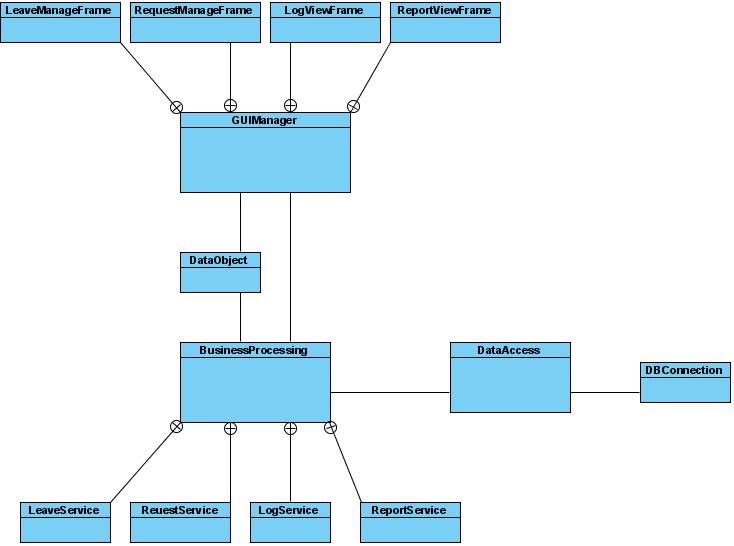
**Business Logic Layer**

Business Logic Layer (BUS) works as a bridge between Presentation Layer and DAO. All the user values received from the presentation layer are being passed to BUS. The results received from the DAO are in row data in Data Table format but in BUS it’s converting into Value Objects (VO). Business Logic Layer (BUS) is the most important class in the whole architecture because it mainly contains all the business logic of the program. Whenever a user wants to update the business logic of the program only need to update this class.

**Presentation Layer**

Presentation Layer is the only layer which is directly connected with the user. So in this matter, it’s also a really important layer for marketing purposes. Presentation Layer is mainly used for getting user data and then passing it to Business Logic Layer for further procedure, and when data is received in Value Object then it’s responsible to represent value object in the appropriate form which user can understand.

**Class Diagram**

****

**Class Diagram Explanation**

* 3 main Layers are:

+ GUIManager (Presentation)

+ BusinessProcessing (Business logic)

+ DataAccess (Data access)

* The GUIManager will manage 4 frame using to provide to user with 4 feature: mange leave, manage request, view log and view report.
* Data display in presentation layer is DataObject, get from BusinessProcessing by 4 services matched with 4 features. BusinessProcessing and GUIManager can talk directly with each other for more operations.
* And about operations related to database, they are executed by DataAccess class. It recieves orders from BusinessProcessing class and retrieve/modify database. DBConnection is a bridge between DataAccess and specific database server.