**CSC INDIA**

**Asset Management Project**

Contents

[1. Introduction 3](#_Toc462412812)

[1.1 Purpose 3](#_Toc462412813)

[1.2 Abstract 3](#_Toc462412814)

[1.3 Scope 3](#_Toc462412815)

[2. Module Description 4](#_Toc462412816)

[2.1 Use case model 4](#_Toc462412817)

[2.1.1 Employee Module use case 4](#_Toc462412818)

[2.1.2 Manager Module use case 5](#_Toc462412819)

[2.1.3 Support Module use case 7](#_Toc462412820)

[2.1.4 Admin Module use case (Ignore Re-Activate User module in the below screen) 8](#_Toc462412821)

[3. Technical Specifications 9](#_Toc462412822)

[3.1 Technologies Used: 9](#_Toc462412823)

[3.2 Architecture Diagram 9](#_Toc462412824)

[3.3 Database Design 10](#_Toc462412825)

[Users: 10](#_Toc462412826)

[Request Table 10](#_Toc462412827)

[Asset Table 11](#_Toc462412828)

[TransferHistory Table 11](#_Toc462412829)

[4. Technical Detailed Description 11](#_Toc462412830)

[4.1 Presentation Layer 11](#_Toc462412831)

[4.1.1 Login page 11](#_Toc462412832)

[4.1.2 Home page 12](#_Toc462412833)

[4.2 Application/Business logic Layer 18](#_Toc462412834)

[4.2.1 New Asset Request 19](#_Toc462412835)

[4.2.2 Asset Transfer Request 20](#_Toc462412836)

[4.3 Data Layer 20](#_Toc462412837)

[5. Testing 21](#_Toc462412838)

[5.1 White-box testing 21](#_Toc462412839)

[5.2 Black-box testing 21](#_Toc462412840)

# 1. Introduction

## 1.1 Purpose

The purpose of this document is to define the requirements and design for an Asset management project that can be used for the purpose of managing the employee assets in an organization. It contains the description of all modules in detail and their functionalities.

## 1.2 Abstract

The Asset Management Project (AMP) is web based application developed using any Web Technologies. The application is aimed to manage the assets of employee in an organization. Based on the type of user it provides various required functionalities for Admin, Employees, Managers and Support team. It facilitates Request, Approval, Dispatch and Transfer of Assets among employees.

## 1.3 Scope

The AMP is a web application, with following functional requirements.

* All the users will be able to login through same login page.
* User should be able to Change Password
* Admin can Create, Modify, View and Delete the users.
* Employee should create a request for assets to support department which will be sent to their manager for approval.
* Employee should be able to track their request and its status.
* Employee can view their assets and their profile. Employee should also be able to transfer their assets to others employees if required.
* Manager can Approve/Reject/Send Back the pending asset request. Based on Manager’s action, request will be passed to the support team if approved or request will be sent to employee if rejected (no changes) or sent back for changes.
* Manager is also an employee hence he/she should be able to create, view and transfer the assets.
* Manager can track their team’s requests and view their assets and profile.
* The support team after viewing the approved asset request of an employee from his/her manager, will dispatch the asset details.
* Support team should be able to approve the new request from employee and should be able to view detailed reports.

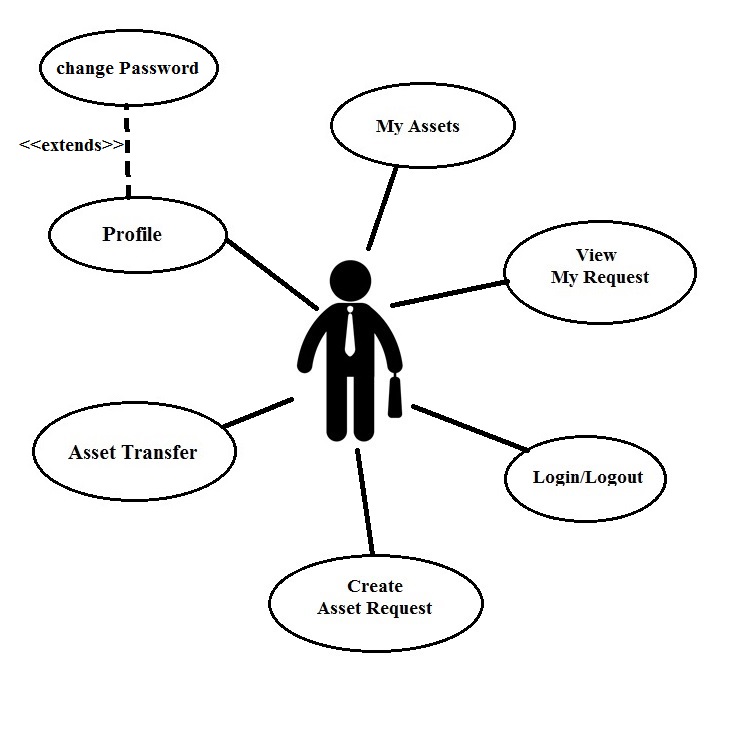
# 2. Module Description

## 2.1 Use case model

There are 4 roles in this Project. They are Employee, Manager, Support Team and Admin.

* Employee Module
* Manager Module
* Support Module
* Admin Module

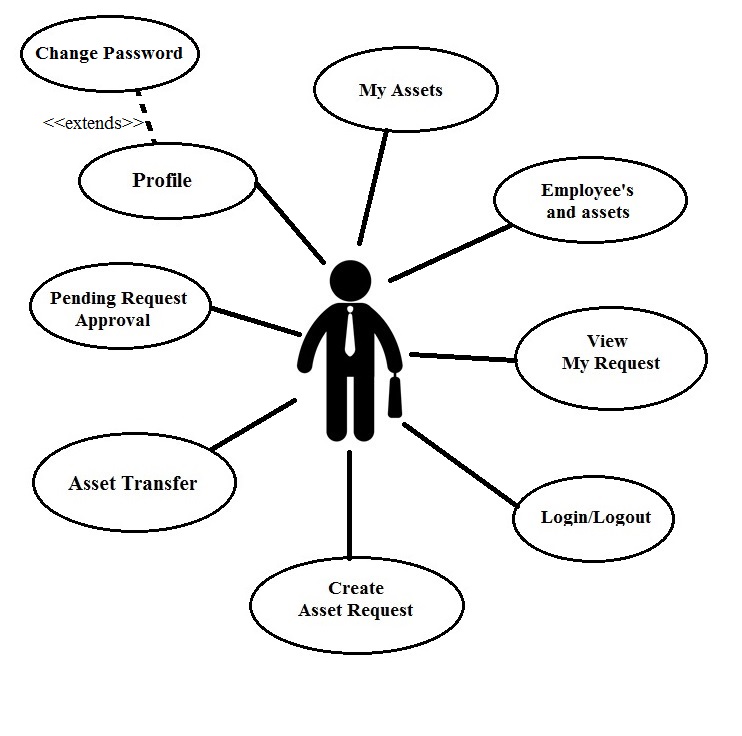
## 2.1.1 Employee Module use case



The use cases of employee are:

1. **Login/Logout**: The user will be able to login/logout from the application whenever he wants.
2. **Create Request:** The employee can raise a request for assets (Laptop, Mobile, SIM Card, Data Card).
3. **View My Request:** Here the user can track their pending asset requests.
4. **My Assets:** All the assets currently held by the user are to be displayed here.
5. **Asset transfer:** Here employee can transfer his assets to other employee.
6. **Profile:** All the employee details are displayed here. User can change his/her password from my profile options.

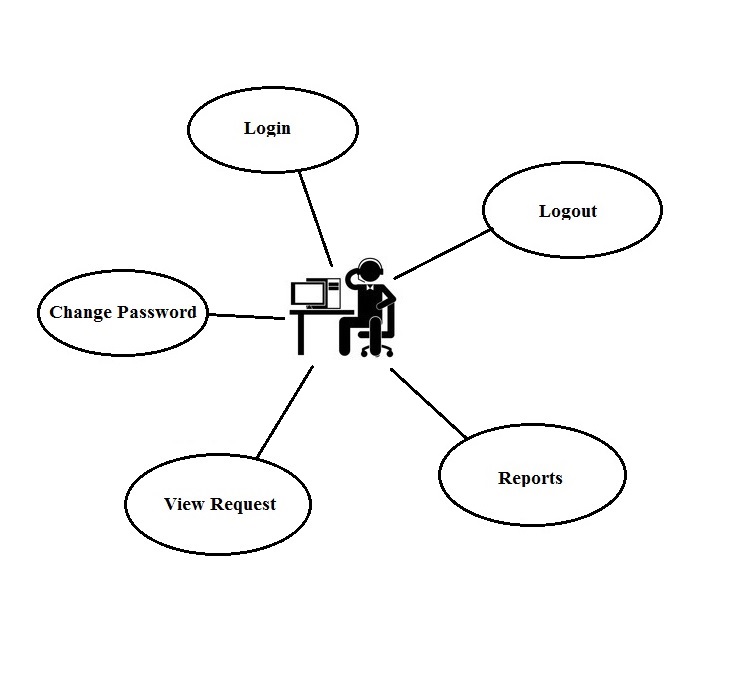
### 2.1.2 Manager Module use case



The use cases of manager are

1. **Login/Logout:** Manager will be able to login and logout from the application.
2. **Create Request:** The manager is also an employee so he/she too can raise a request.
3. **View My Request:** User should be able to track his pending asset requests here.
4. **My Assets**: All the assets which are currently held by manager will be displayed here.
5. **Asset Transfer:** Here User can transfer their assets to other employee upon Manager approval.
6. **Profile:** Employee details will be displayed here and he/she can change their password from the options available here.
7. **Pending Request Approval:** Manager should be able to view all the requests sent to him from the employees under him/her for approval.
8. **Employees and assets:** Manager can view all employees under him/her and the assets held by them.

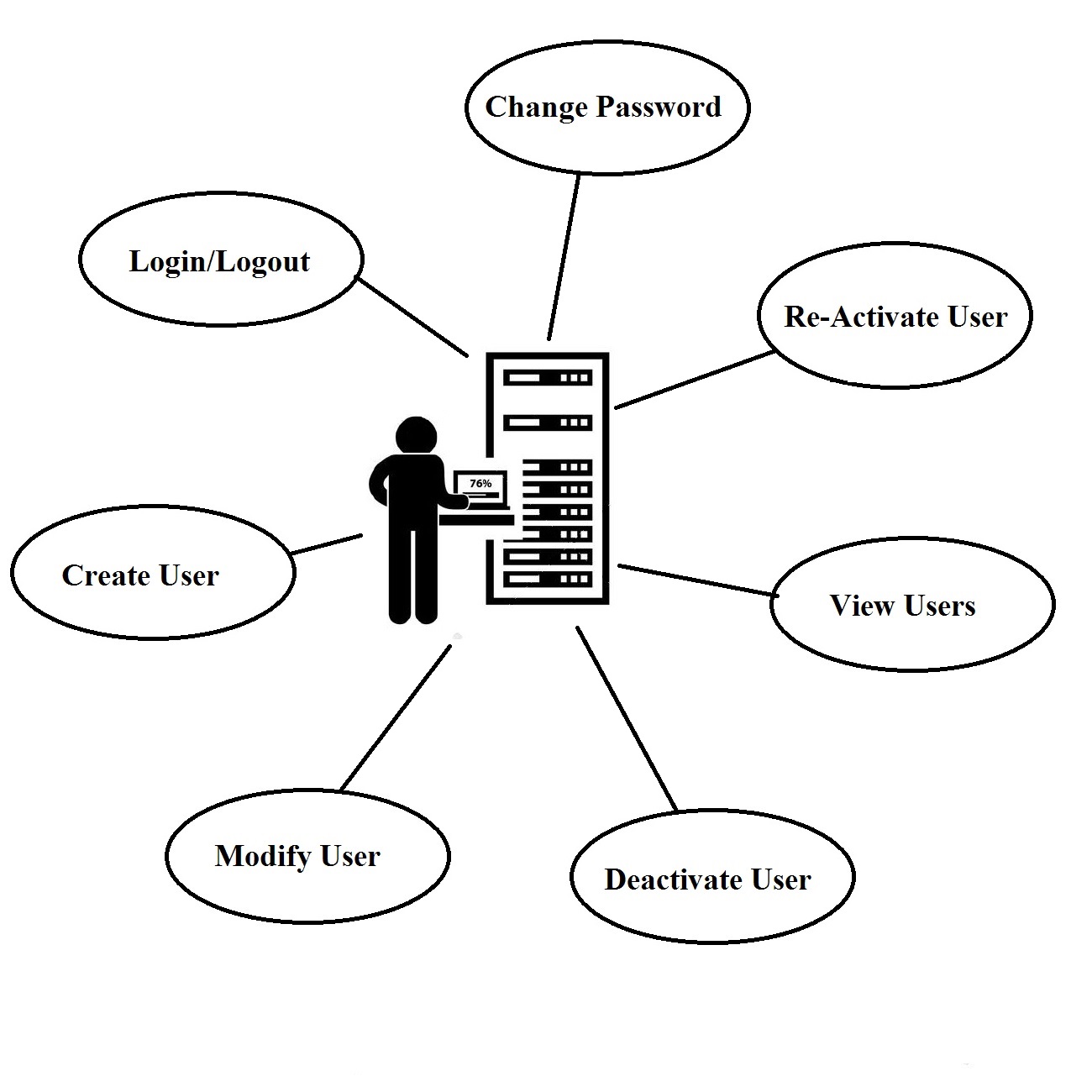
### 2.1.3 Support Module use case



The use cases of support team are

1. **Login/Logout:** User will be able to login and logout from the application.
2. **View Request:** Here the user can view approved requests of an employee and dispatches the asset by clicking “Dispatch” button which automatically notifies employee to collect his/her asset from support department. They update asset details into that request.
3. **Reports:** Can generate reports for the list of assets dispatched to various employees on particular date and for different criteria’s as required.
4. **Change Password:** User can change his password here.

### 2.1.4 Admin Module use case (Ignore Re-Activate User module in the below screen)



The use cases of admin are

1. **Login/Logout**: User will be able to login and logout from the application.
2. **Create User:** Creates a new user and defines a role for the created user with required details.
3. **Modify User:** Modify the details of employee/user.
4. **View User:** List of all users in the organization will be displayed here.
5. **Deactivate User:** User accounts will be deactivated when employee leaves the organization.
6. **Change Password:** Admin can change his password here.

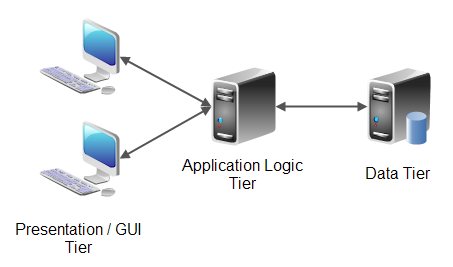
# 3. Technical Specifications

## 3.1 Technologies Used:

|  |  |
| --- | --- |
| Description | Software |
| Platform | .NET/Java |
| Database | MySQL/SQL Server |
| Server | IIS/Tomcat Server |

## 3.2 Architecture Diagram

The project is developed using 3-Tier Architecture. If we consider a 3-Tier Architectural WEB application, the browser becomes the client side application. The user communicates with the WEB/Application server via the browser. In the context of this project, the web pages represent the presentation layer. Here we can write code to design frontend and can perform validations. The data may be passed from this layer to business tier which has classes, methods and database connections. Now the queries are sent to Data Tier which gets data from MySQL/ SQL server and returns it to Business Tier.



**3-Tier Architecture**

## 3.3 Database Design

### Users:

*Note: Short\_Id or Email\_Id used as username*

|  |  |  |
| --- | --- | --- |
| Coloumn name | Type | Size |
| Short\_Id | Varchar | 25 |
| FirstName | Varchar | 25 |
| LastName | Varchar | 25 |
| Email\_ID | Varchar | 25 |
| Password | Varchar | 25 |
| Manager\_id | Int | 10 |
| Designation | varchar | 25 |
| Emp\_id | Int | 10 |
| Mobile | Int | 10 |
| DateOfJoining | Date |  |
| Active | Varchar | 1  A=Active  I=InActive |

### Request Table

*Note : IsApp – Approved/not (Flag to check)*

|  |  |  |
| --- | --- | --- |
| Coloumn name | Type | Size |
| Request\_id | Int(pk) | 10 |
| Emp\_id | Int(fk) | 10 |
| Short\_id | Varchar | 25 |
| Manager\_id | Int | 10 |
| ManagerApprovedDate | Date |  |
| AssetName | Varchar | 25 |
| RequestDate | Date | - |
| Status | Int | 1  1=Pending with Manager  2=Pending with Employee  3=Pending with Support  4=Rejected by Manager  5=Approved |
| Remarks/Comments | Varchar | 100 |

### Asset Table

|  |  |  |
| --- | --- | --- |
| Coloumn name | Type | Size |
| Emp\_id | Int | 10 |
| Asset\_id | Int | 10 |
| Request\_id | Int | 10 |
| IssuedDate | Date | - |
| AssetName | Varchar | 25 |

### TransferHistory Table

|  |  |  |
| --- | --- | --- |
| Coloumn name | Type | Size |
| Transfer\_id | Int | 10 |
| From\_Emp\_id | Int | 10 |
| Manager\_id | Int | 10 |
| Asset\_id | Int | 10 |
| AssetName | Varchar | 25 |
| To\_Emp\_id | Int | 10 |
| TransferDate | Timestamp | - |
| Transfer\_Status | Int | 1  1=Approved  2=Rejected |
| Comments/Remarks | Varchar | 100 |

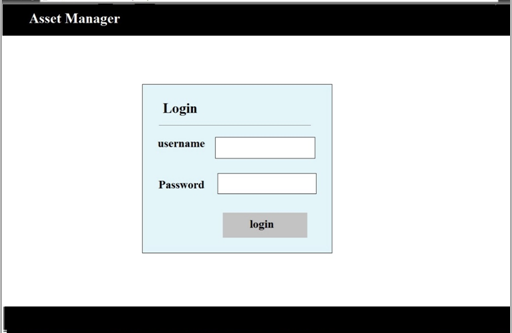
# 4. Technical Detailed Description

## 4.1 Presentation Layer

This is the view/GUI that is visible to user. This contains the design part.

### 4.1.1 Login page

The user should be presented with the home page/login page (when unauthenticated). A sample home/login page is as below.



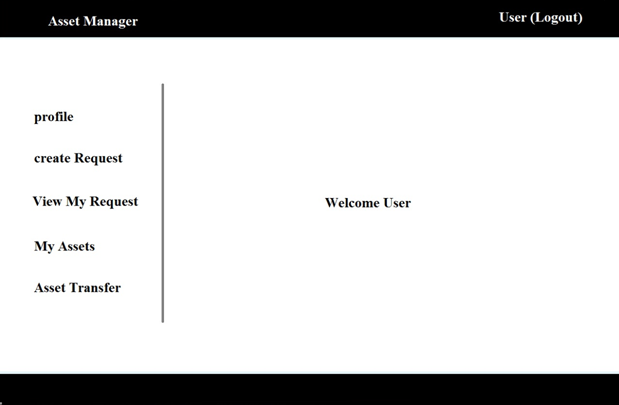
**Validations:**

> Username should be Short ID or Email ID.

### 4.1.2 Home page

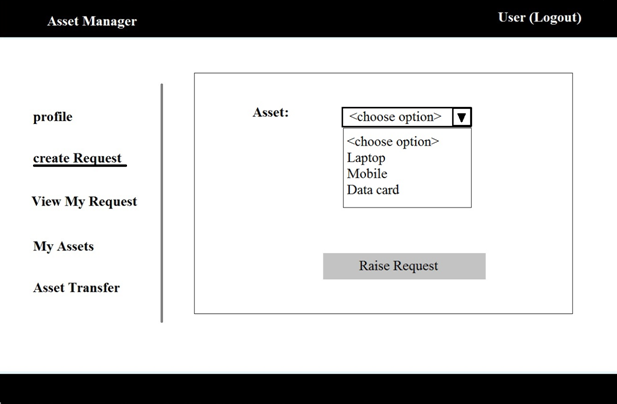
Users are provided with different home pages based on their roles.

#### 4.1.2.a. Employee Master



Once the employee logs into application, the above screen appears with options as discussed in use case. Now the user can explore the options available and navigate the pages.

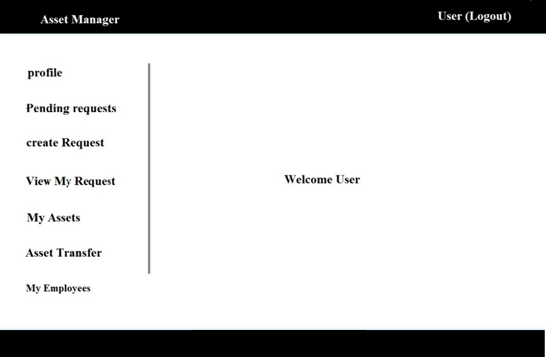
For instance, consider the user want to raise a request for asset, he clicks on the create request which opens following page.



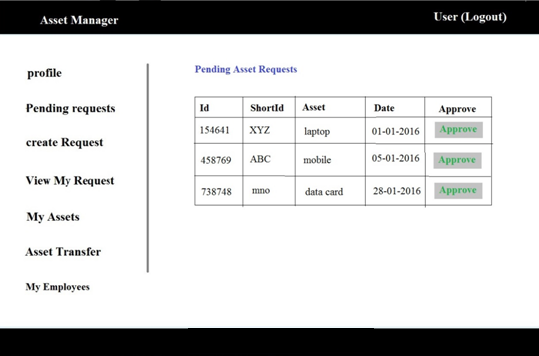
The request is simplified as above which asks for data from employee. Similarly, all other options are designed as discussed in the use case. Can be designed as per our way of representing the screen.

#### 4.1.2.b. Manager Home Screen

Manager home page is similar to that of employee. All the features of employee are available to manager. Additional to this, the manager has pending requests and will be able to view the details of the employees under him. His home looks like the below screen shot.



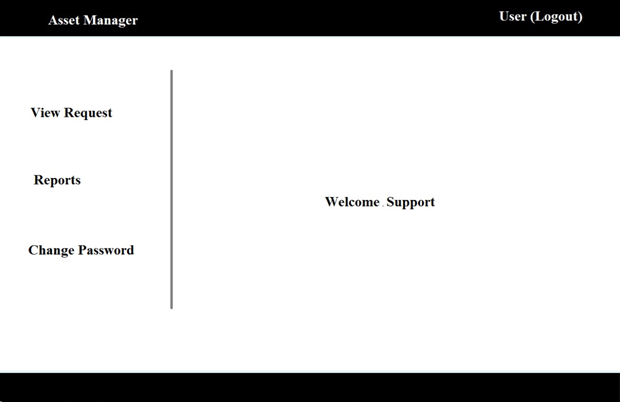
The manager can view the asset requests of his team only, it looks like below:



Manager should be given option to approve/reject or send back. This can be listed in a dropdown. Provide a button to submit the request stating whether it is approved or rejected or sent back.

#### 4.1.2.c. Support Master

The support team has the below functionalities which are similar in functionality of manager and employee. The home looks as below:

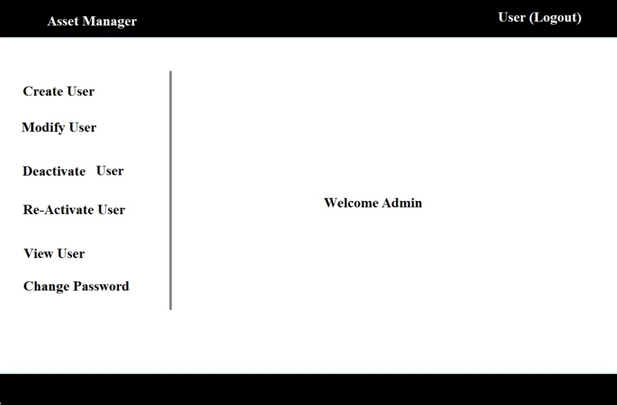


The support team should be able to view the approved requests from the manager and dispatches the asset accordingly by updating the asset details.

#### 4.1.2.d. Admin User

The admin is super user of all the users, he manages all the users, irrespective of their role. He can manage all the user accounts.

**(Ignore Re-Activate User module in the below screen)**



As mentioned above the admin can create, modify, deactivate and view user details.

## 4.2 Application/Business logic Layer

The key functionality of this application is the ability to process the request and dispatch of assets.

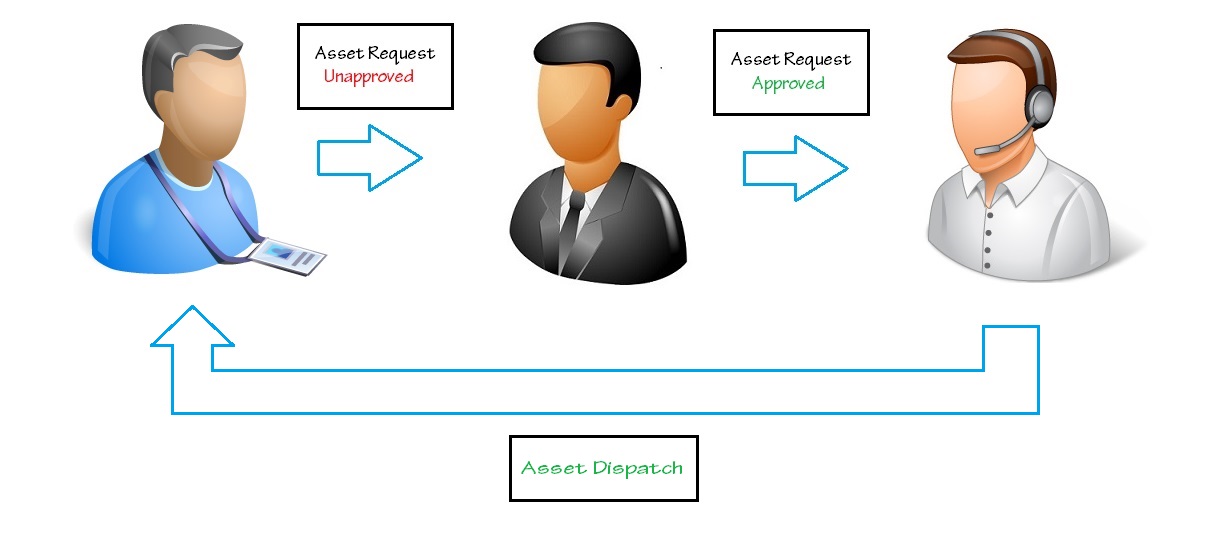
The asset request can be either for a new one or transfer the existing one.

### 4.2.1 New Asset Request

The new asset request flows in following levels

* First the employee creates request
* The manager has to approve/Reject/Send back the request
* Finally, the support views and dispatches the asset

The above is simply shown diagrammatically as below

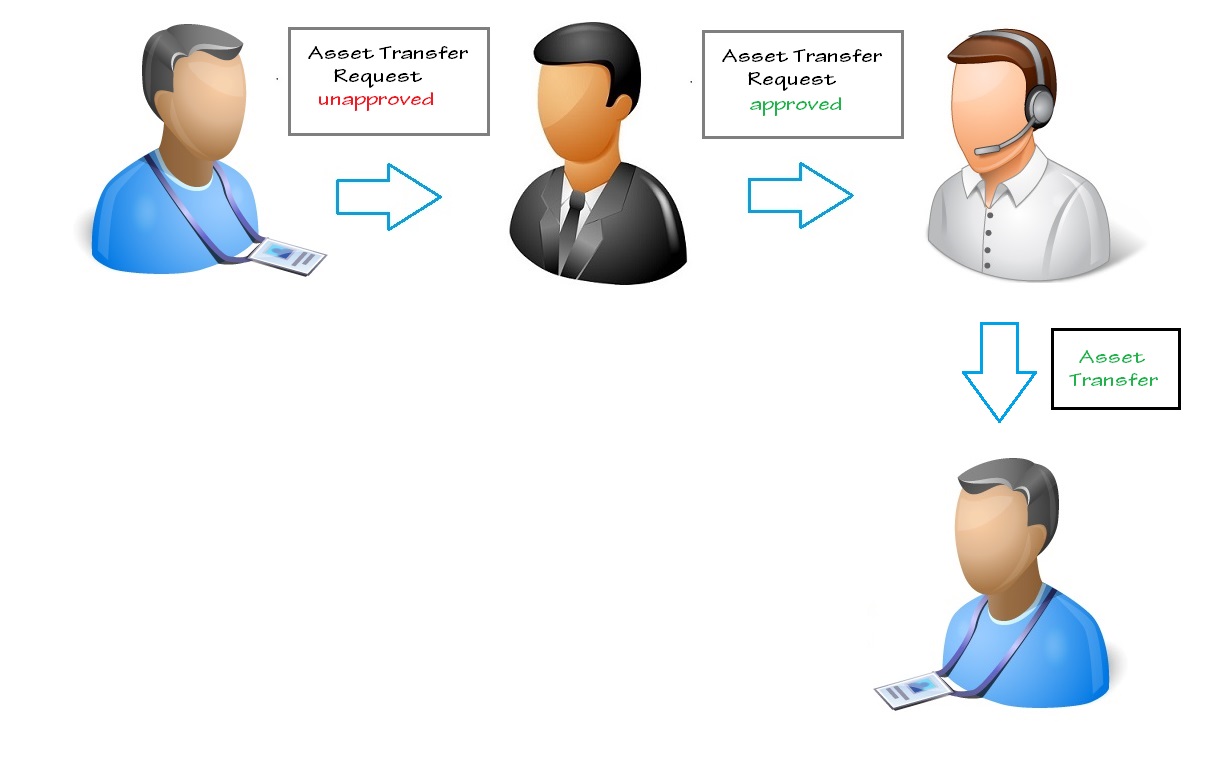


The request is initially pending for approval in the initial level, once the manager approves it is forwarded to the support team.

### 4.2.2 Asset Transfer Request

Here the asset will be transferred to other employee

* Employee has to create transfer request
* Now manager approves/Rejects/Sends back the request
* A notification will go to support team that an asset has been transferred from one employee to other employee. They can view the transfer asset details (no action from their side)



Here the request is raised along with the details of the employee to whom the asset need to be transferred. Then upon the manager’s approval, the support team gets a notification that a transfer has been done between the employees. Now the employee can handover the asset to other employee.

## 4.3 Data Layer

This is the layer in which the required values for the particular Functionality page is obtained. Based on the data acquired the computation is done and user is provided with particular view.

As described we have 4 tables in SQL database.

# 5. Testing

## 5.1 White-box testing

White-box testing (also known as clear box testing, glass box testing and structural testing, by seeing the source code) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user.

This is done by the developer itself in parallel to developing of above project. This involves appropriate structuring of query and logic together.

## 5.2 Black-box testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation, without seeing the source code.

Some example is:

* Login validations
* It should take correct Data Input Types
* It should redirect to correct pages
* No Data loss
* Proper alignment of web pages

Once the application is built the black box testing is done by creating some users and test with proper test case scenarios.