Low Level Design: User Management

**Low Level Design**

**For**

**User Management**

Prepared by: UMT Dev Team Date: 14-08-2023

Accepted by: Date:

**Revision History**

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 14-08-2023 | 1.0 | First Release | UMT Dev Team |
| 11-Dec-2023 | 1.1 | Fine tuning with changes after development | UMT Dev Team |

**Table of Contents**

[1. Introduction 4](#_heading=h.17dp8vu)

[1.1. Purpose 4](#_heading=h.35nkun2)

[2. Entities and Relations 4](#_heading=h.1ksv4uv)

[2.1. Entity Relationship Diagram 4](#_heading=h.44sinio)

[3. LLD: User Management 4](#_heading=h.3rdcrjn)

[3.1. Create User 4](#_heading=h.8tmfygo3t2lt)

[3.2. Register User 5](#_heading=h.26in1rg)

[3.3. EDIT User 6](#_heading=h.lnxbz9)

[3.4. Suspend User 8](#_heading=h.4i7ojhp)

[3.5. Resume User 8](#_heading=h.2xcytpi)

[3.6. Delete User 9](#_heading=h.1ci93xb)

[3.7. User Authentication - Local Users 10](#_heading=h.3j2qqm3)

[3.8. User Authentication - Module Users 11](#_heading=h.1y810tw)

[4. LLD: Module Management 12](#_heading=h.3whwml4)

[4.1. View Module and Module Functions 12](#_heading=h.tyjcwt)

[4.2. Search Functions 13](#_heading=h.dpphvrnbsje2)

[5. LLD: Role Management 13](#_heading=h.1t3h5sf)

[5.1. Create Role 14](#_heading=h.4d34og8)

[5.2. Edit Role 14](#_heading=h.2s8eyo1)

[5.3. Assign / De-assign Module Function 15](#_heading=h.2jxsxqh)

[6. LLD: User-Role Mapping 16](#_heading=h.z337ya)

[6.1. Search User 16](#_heading=h.qsh70q)

[6.2. Assign Role(s) 17](#_heading=h.3as4poj)

[7. LLD: Change Password 18](#_heading=h.s85pvnweuedc)

[7.1. CHANGE PASSWORD 18](#_heading=h.257xu9v0ii20)

[8. Password Policy 19](#_heading=h.2bn6wsx)

[8.1. Policy Configuration 19](#_heading=h.5kfqrlrmt0e1)

# Introduction

## Purpose

Developers refer to this document while developing the SATS user management functionalities.

# Entities and Relations

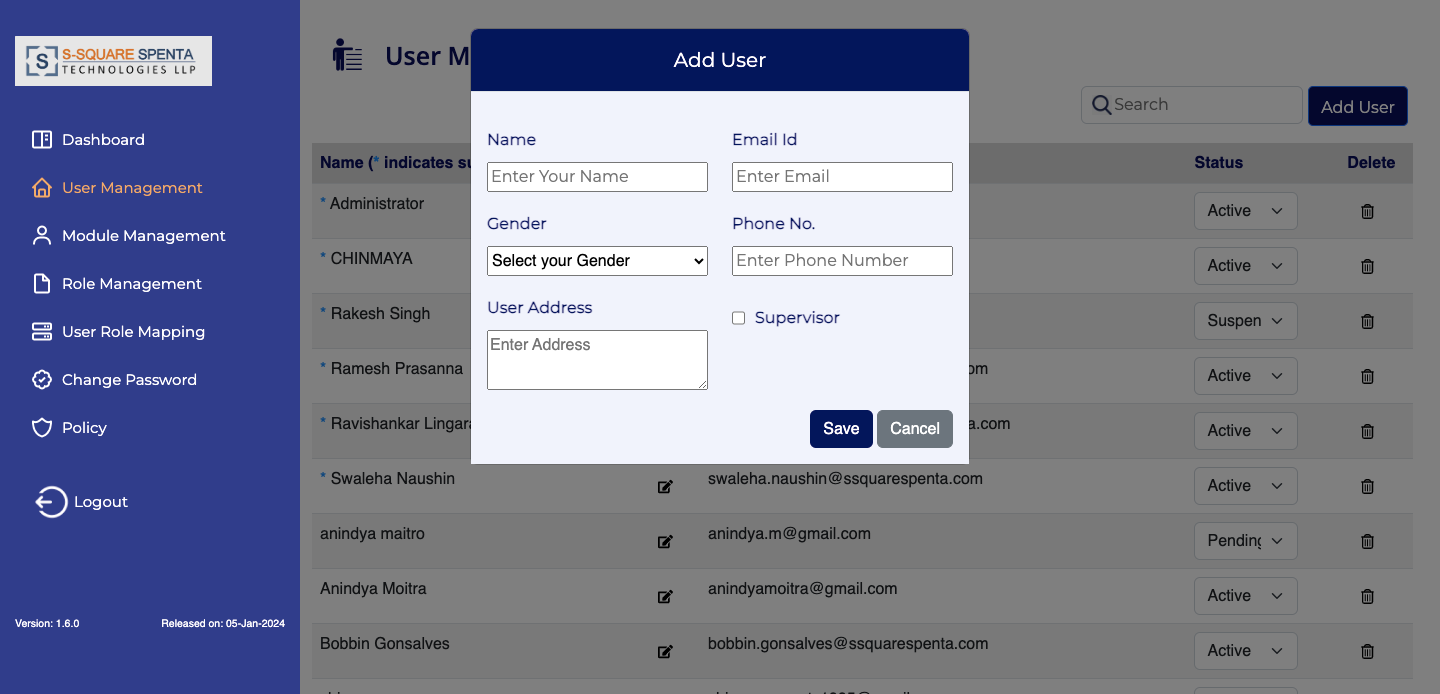
## Entity Relationship Diagram

ER diagram has been prepared separately and shared.

# LLD: User Management

## Create User

**Screen Design**



**Inputs**

* *name*: Name of the new user
* *email*: email address of the new user
* *gender*: gender of the user
* *phone*: phone number of the user
* *user address*: address of the user
* *supervisor*: check box to make the new user as supervisor

**Validations**

* email is mandatory and unique
* password is mandatory
* name is mandatory

**Processing**

* Only admin can create new Users.
* Admon inputs the data to the form.
* When the user inputs the email address system checks the users table for the email using ajax request. if found it will show error(This email already exists)
* Once submitted the user is created with role (staff or supervisor)
* Email is sent to admin for approval

**Output**

* New user is created with password=Password@123

## Register User

**Screen Design**

**Inputs**

* *name*: Name of the new user
* *email address:*email address if new user
* *phone*: Phone number of the new user
* *password*: password of the new user
* *confirm password*: confirm password of the new user
* *User Address:* Address of the new user
* *Gender:* Gender of the user
* Supervisor: check box to set the user registered as supervisor.

**Validations**

* email is mandatory and unique
* password is mandatory
* name is mandatory

**Processing**

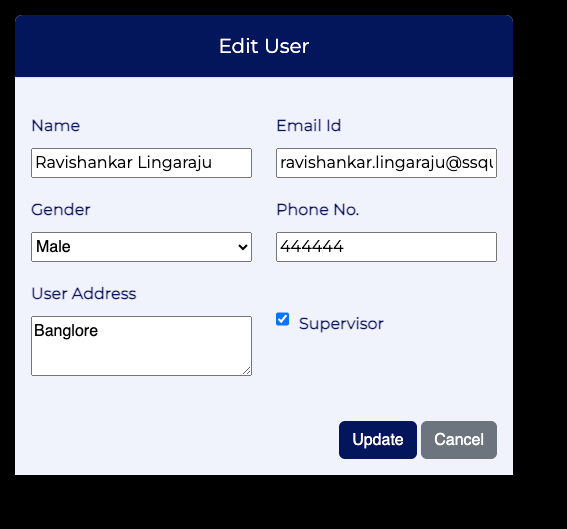
* User clicks the register link and the user registration page shows.
* User inputs the data to the form.
* When the user inputs the email address system checks the users table for the email using ajax request. if found it will show error(This email already exists)
* Once submitted the status is set as pending and user is created with role (staff or supervisor)
* Email is sent to admin for approval

**Output**

* User registered with pending status with the submitted data.

## EDIT User

**Screen Design**



**Inputs**

* *name*: Name of the user
* *email*: email address of the user
* *gender*: gender of the user
* *phone*: phone number of the user
* *user address*: address of the user
* *supervisor*: check box to make the user as supervisor

**Validations**

* email is mandatory and unique
* password is mandatory
* name is mandatory

**Processing**

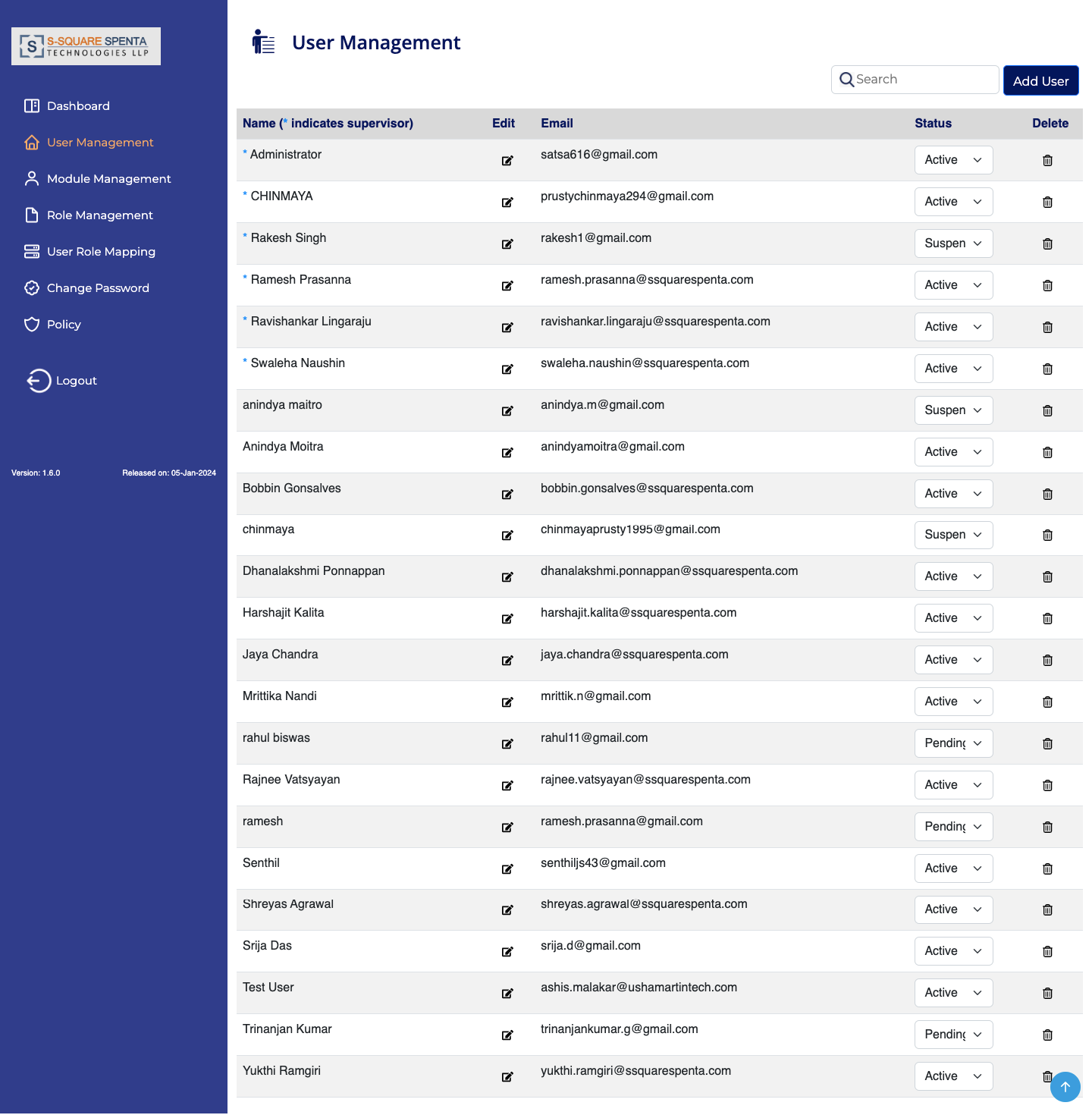
* Admin clicks the edit icon from the user listing screen of the user management.
* Admin changes the data of the form.
* Once submitted the user is updated.

**Output**

* User data is updated.

## Suspend User

**Screen Design**



**Processing**

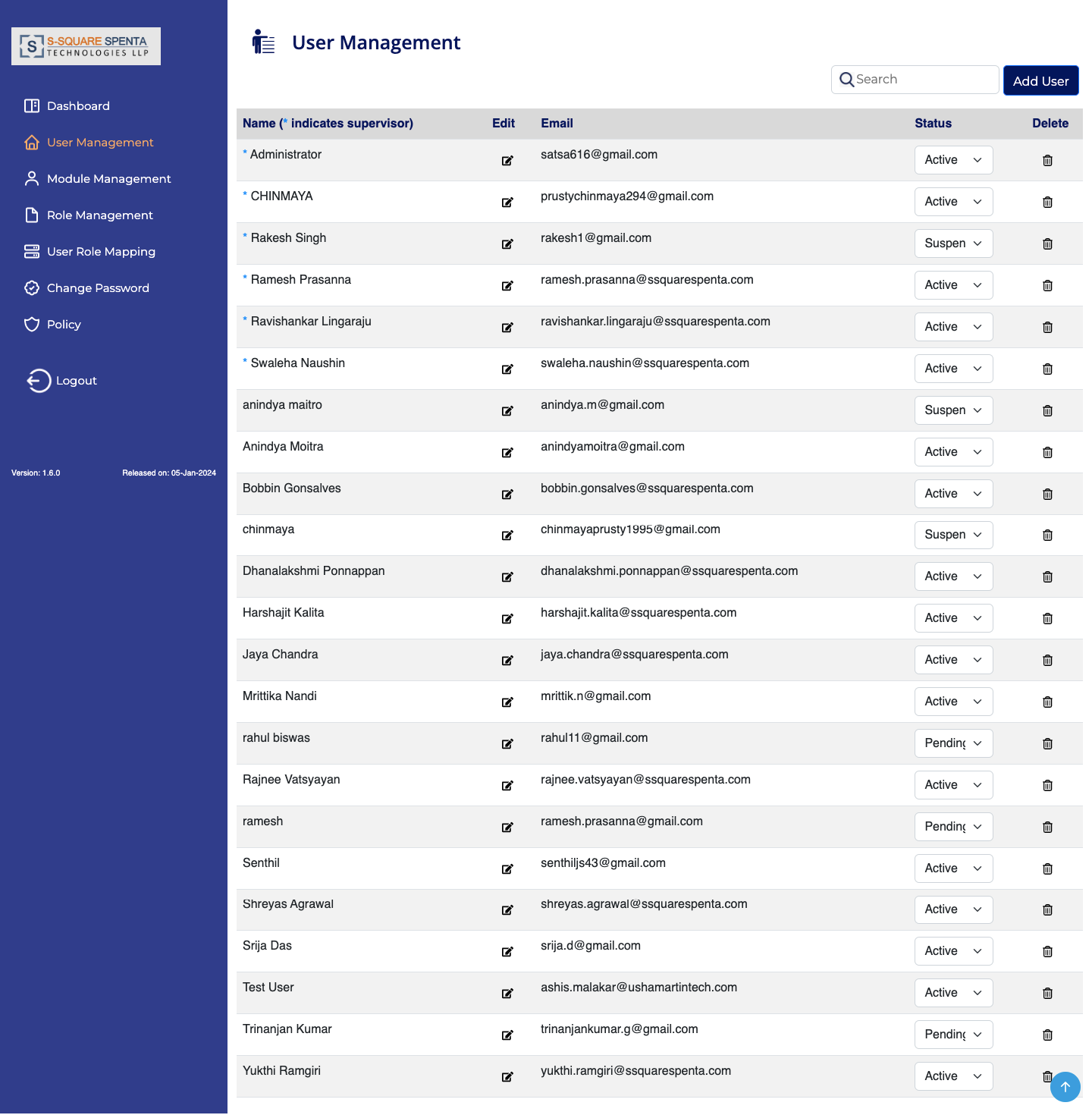
* Admin user selects **suspended** from the **Status** dropdown list of the user management screen
* On changing the dropdownlist the page submits and the status of the user in users table is updated as suspended

**Output**

* User is suspended and not able to login from the mobile app or web portal.

## Resume User

**Screen Design**



**Processing**

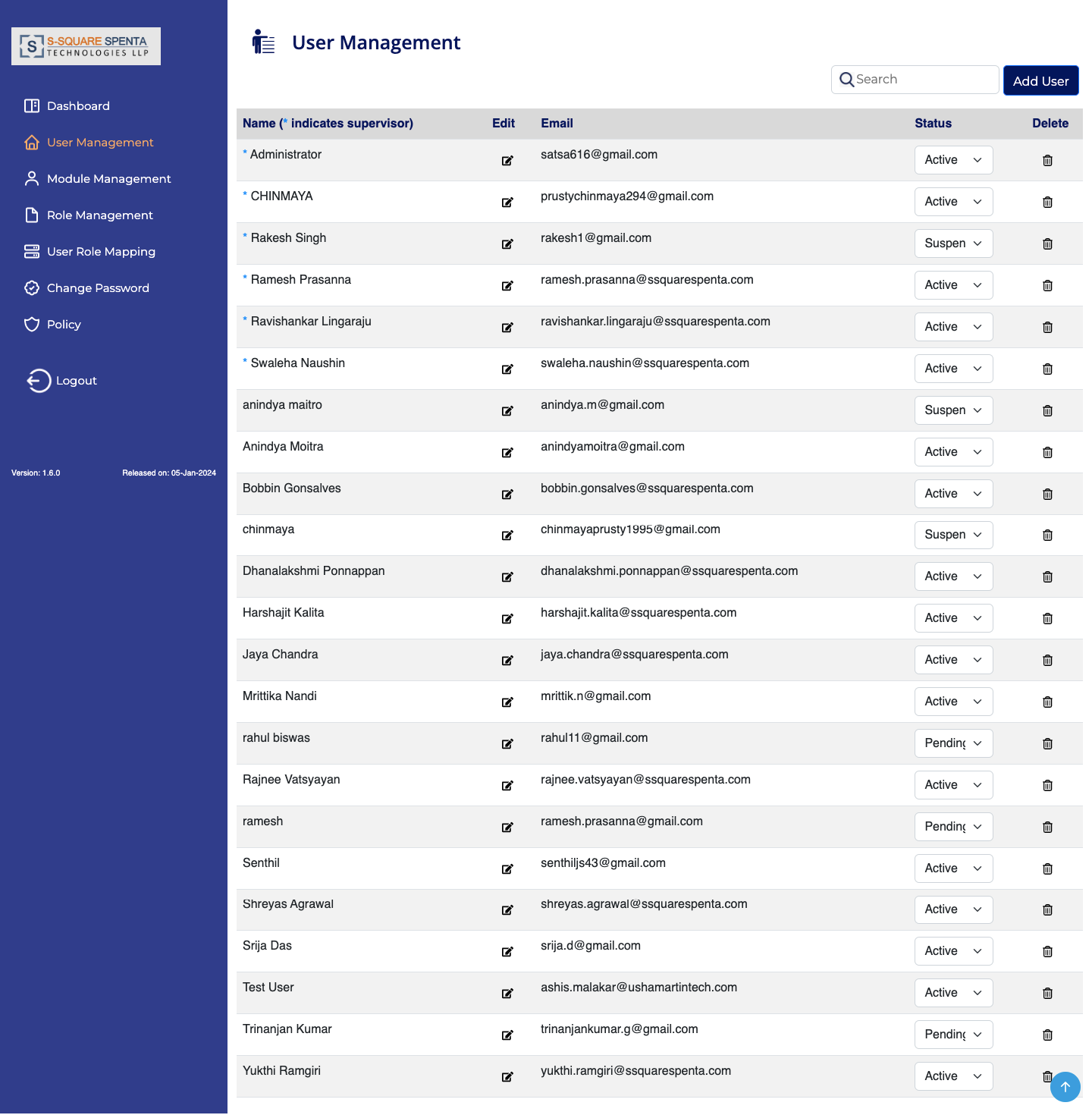
* Admin user selects **Active** from the **Status** dropdown list of the user management screen
* On changing the dropdownlist the page submits and the status of the user in users table is updated as Active.

**Output**

* User is activated and will be able to login from a mobile app or web portal.

## Delete User

**Screen Design**



**Processing**

* Admin user clicks delete button from the user management screen
* The user will be deleted from the users table.

**Output**

* Users will be deleted.

## User Authentication - Local Users

**Screen Design**



**Inputs**

* *Email address:* email address of the user who wants to login
* *Password:* Password for the user who wants to login

**Validations**

* *Email address:* should be mandatory and a valid email address.
* *Password:* The password field must be at least 6 characters

**Processing**

* User inputs username and password
* User clicks the submit button.
* The system checks in users table in usermgmt database using
  + if (Auth::attempt(['email' => strtolower($request->email), 'password' => $request->password,'status'=>'active']))
* If successful, it navigates to the dashboard.
* Else it navigates back to the login page with message “Incorrect User ID or password”.

**Output**

* Navigate to the dashboard on successful login.

## User Authentication - Module Users

**Screen Design**



**Inputs**

* Clicking on Modules (OMS, UCP, ATS)

**Validations**

* Checks if the user is a supervisor or not.

**Processing**

* When a user clicks the module ATS it checks the roles and permissions for the module and navigates to the dashboard or home page of the module. The login happens using Laravel’s passport service.

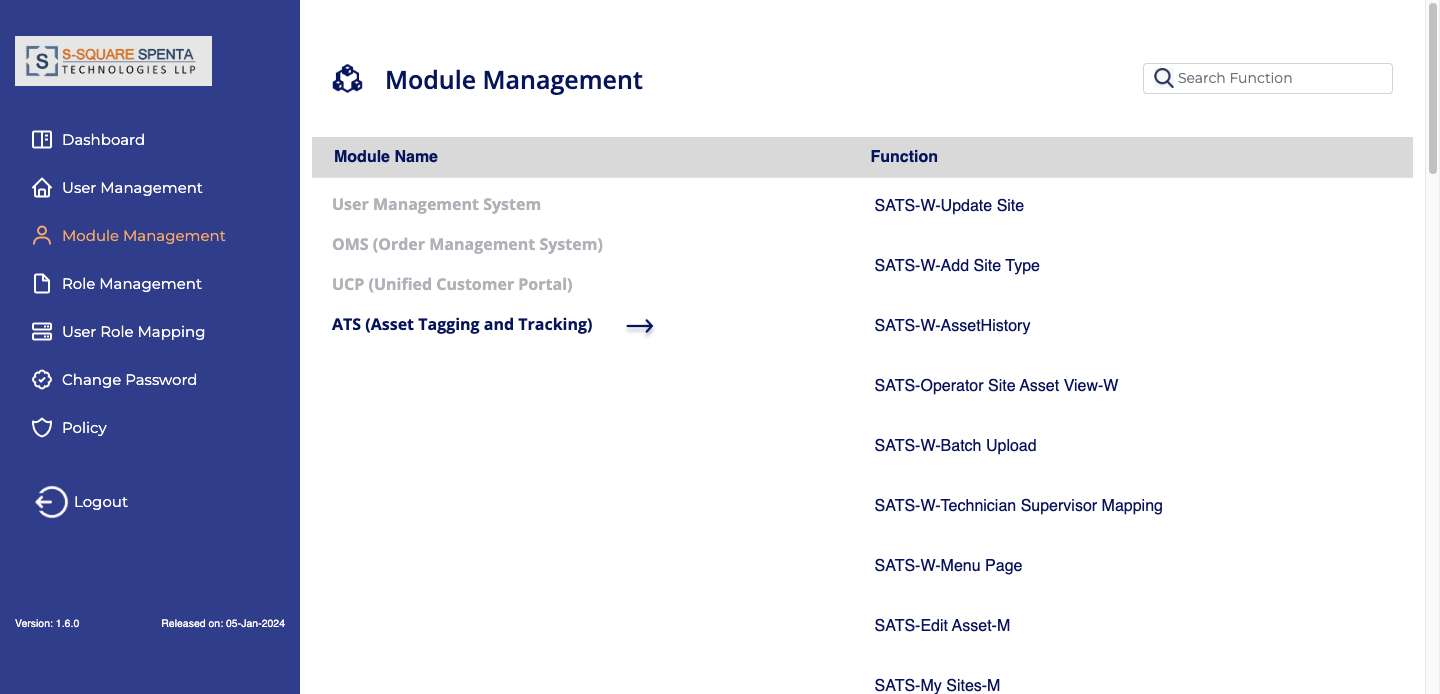
**Output**

* Login to the modules

# LLD: Module Management

## View Module and Module Functions

**Screen Design**



**Processing**

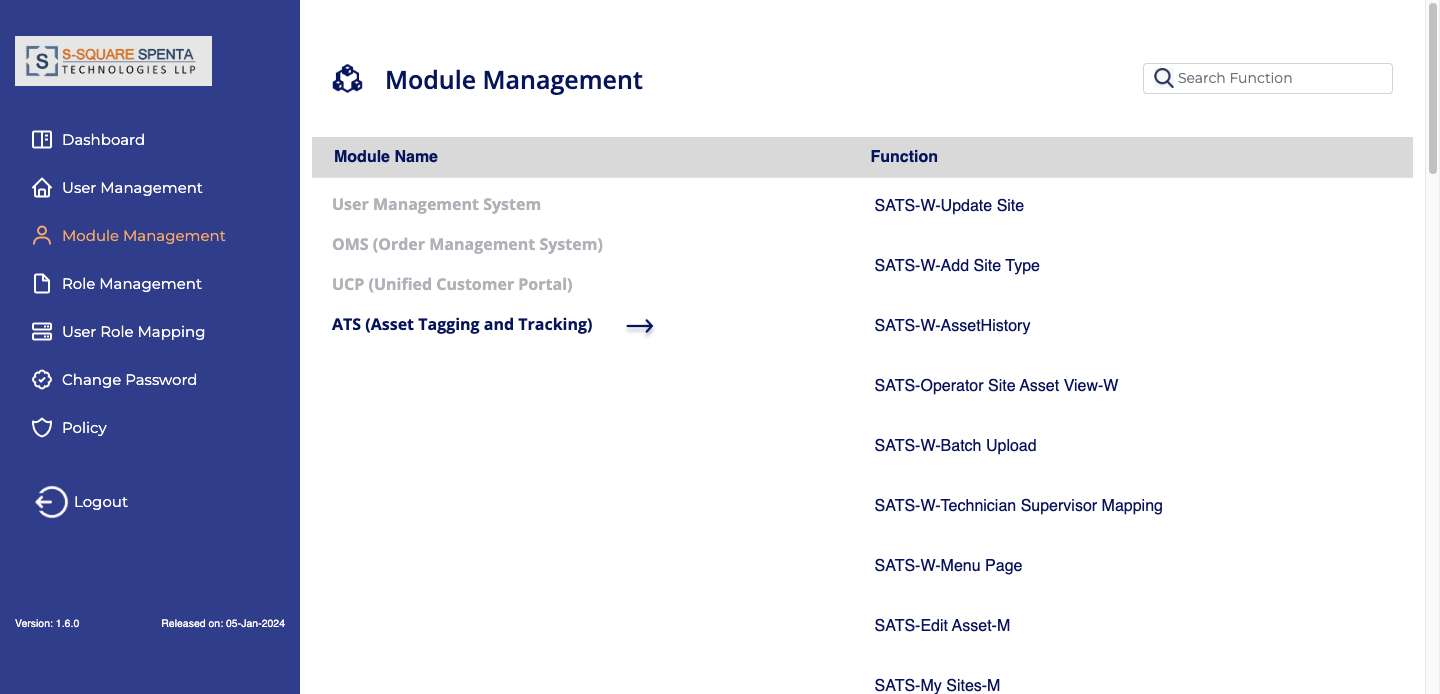
* User clicks Module Management link from the left panel
* .User will see the list of modules pre populated in ***modules*** table
* On clicking the modules the functions related to the modules will be shown from the pre populated ***functions*** table

**Output**

* Modules and function listing.

## Search Functions

**Screen Design**



**Processing**

* User clicks Module Management link from the left panel
* .User will see the list of modules pre populated in ***modules*** table
* On clicking the modules the functions related to the modules will be shown from the pre populated ***functions*** table
* To search any function the user types the keywords of the function name and auto dropdown with suggestions appears.
* On selecting the dropdown function appears.

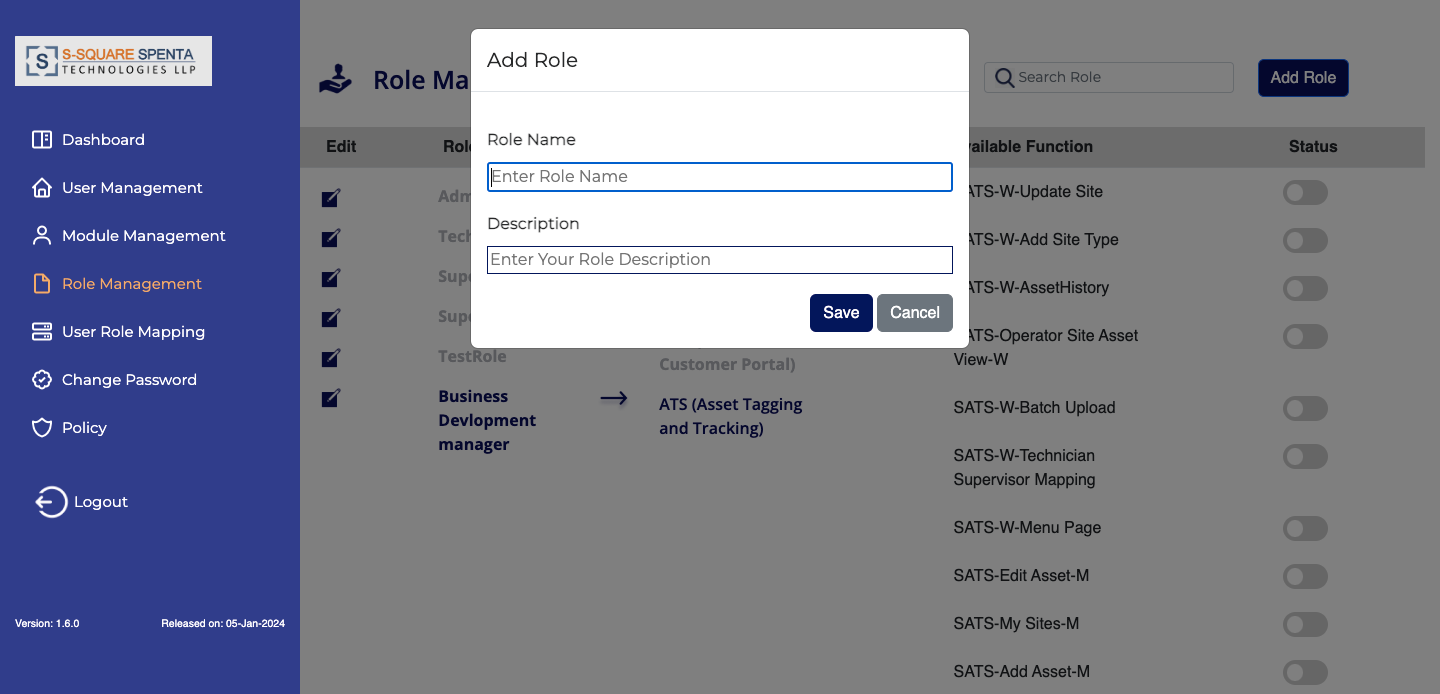
**Output**

* Function search result displayed.

# LLD: Role Management

## Create Role

**Screen Design**



**Inputs**

* *Role name:* Name of a new role
* *Description:*Description of the new role

**Validations**

* *Role name*: should be mandatory and unique.

**Processing**

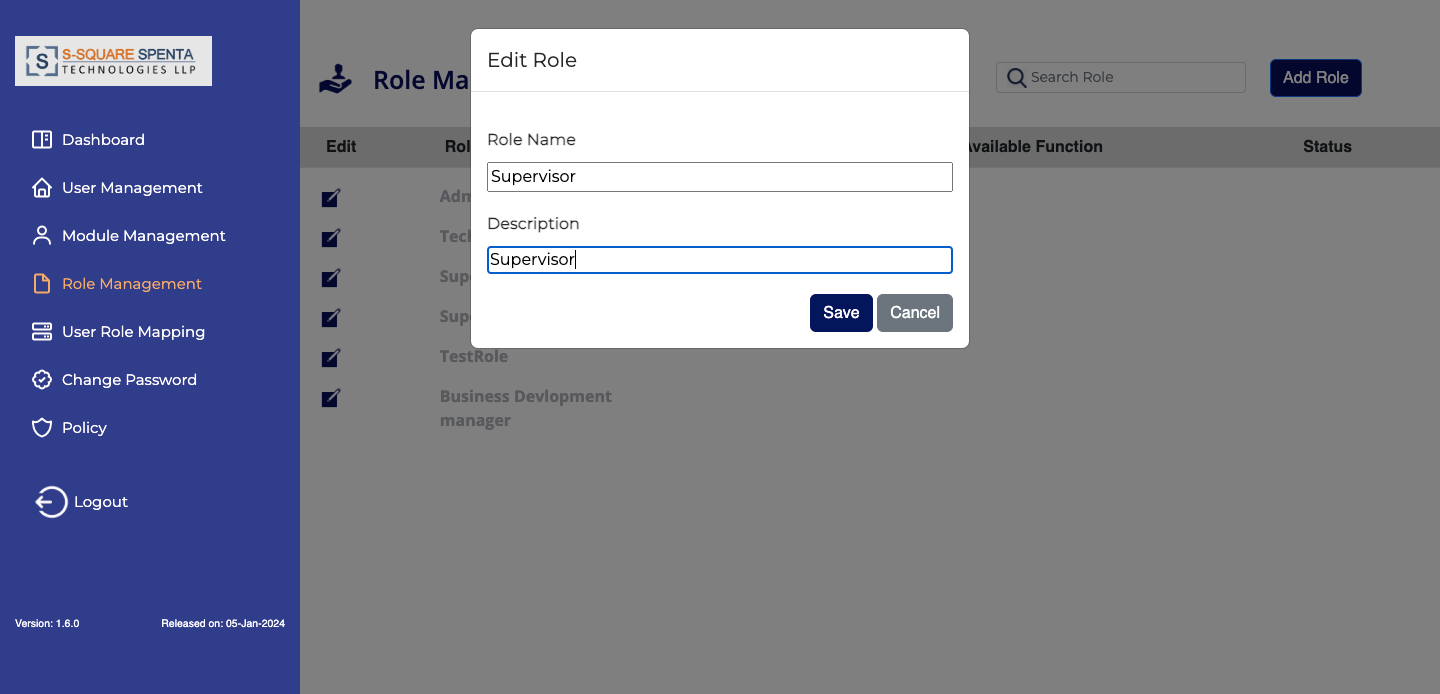
* User clicks Role management link from the left panel and clicks the add role button.
* One Popup opens.
* User inputs Role Name and Description in the popup windows and clicks the Save button.
* The system checks in the rolestable in the usermgmt database and creates a new role.

**Output**

* New role is created.

## Edit Role

**Screen Design**



**Inputs**

* *Role name:* Name of a role to be changed
* *Description:*Description of the role to be changed

**Validations**

* *Role name*: should be mandatory and unique.

**Processing**

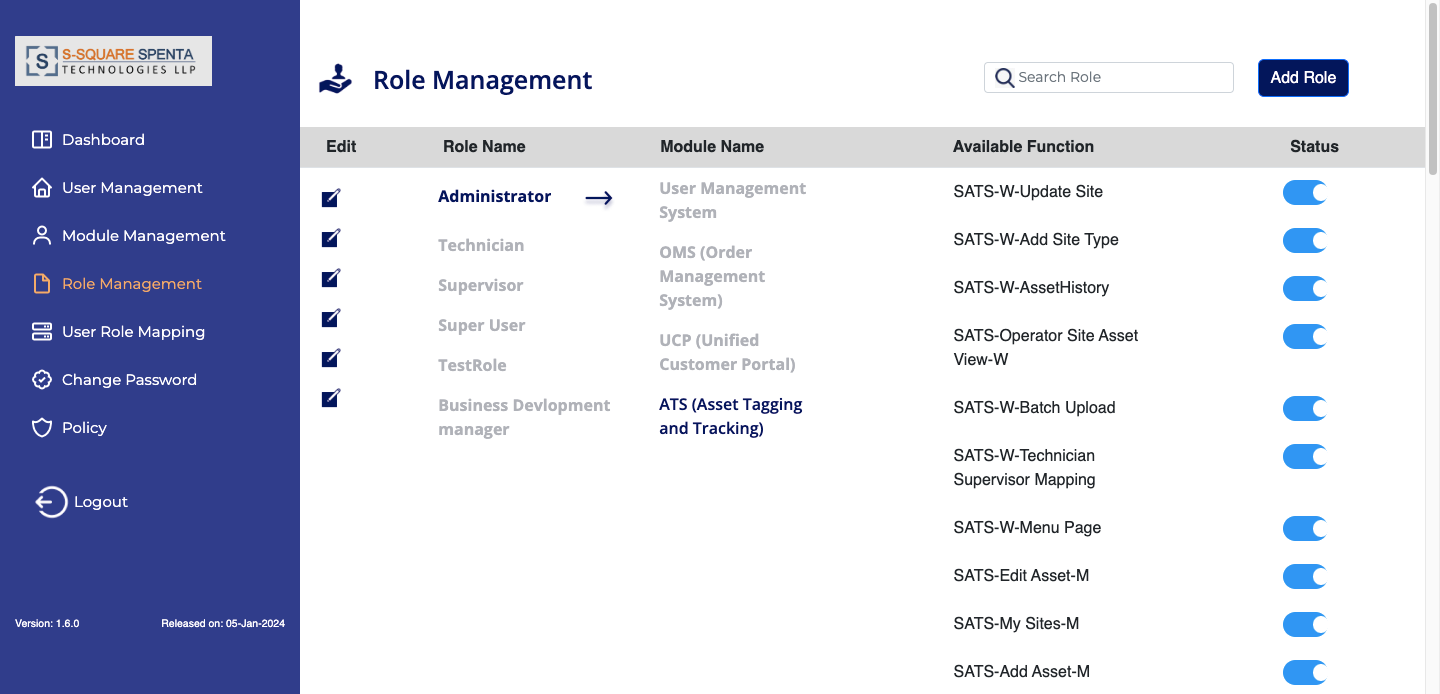
* User clicks Role management link from the left panel
* from the list of the role user clicks edit button.
* One Popup opens.
* User updated Role Name and Description in the popup windows and clicks the Save button.
* The system checks in the rolestable in the usermgmt database and updates the role.

**Output**

* Role is updated.

## Assign / De-assign Module Function

**Screen Design**



**Processing**

* User clicks Role management link from the left panel
* User clicks any role from the list.
* It will show list of modules associated with the role.
* User clicks any module then the function associated with the module will show in the available functions column.
* In the status column one clicking on the slider changes the status of the function

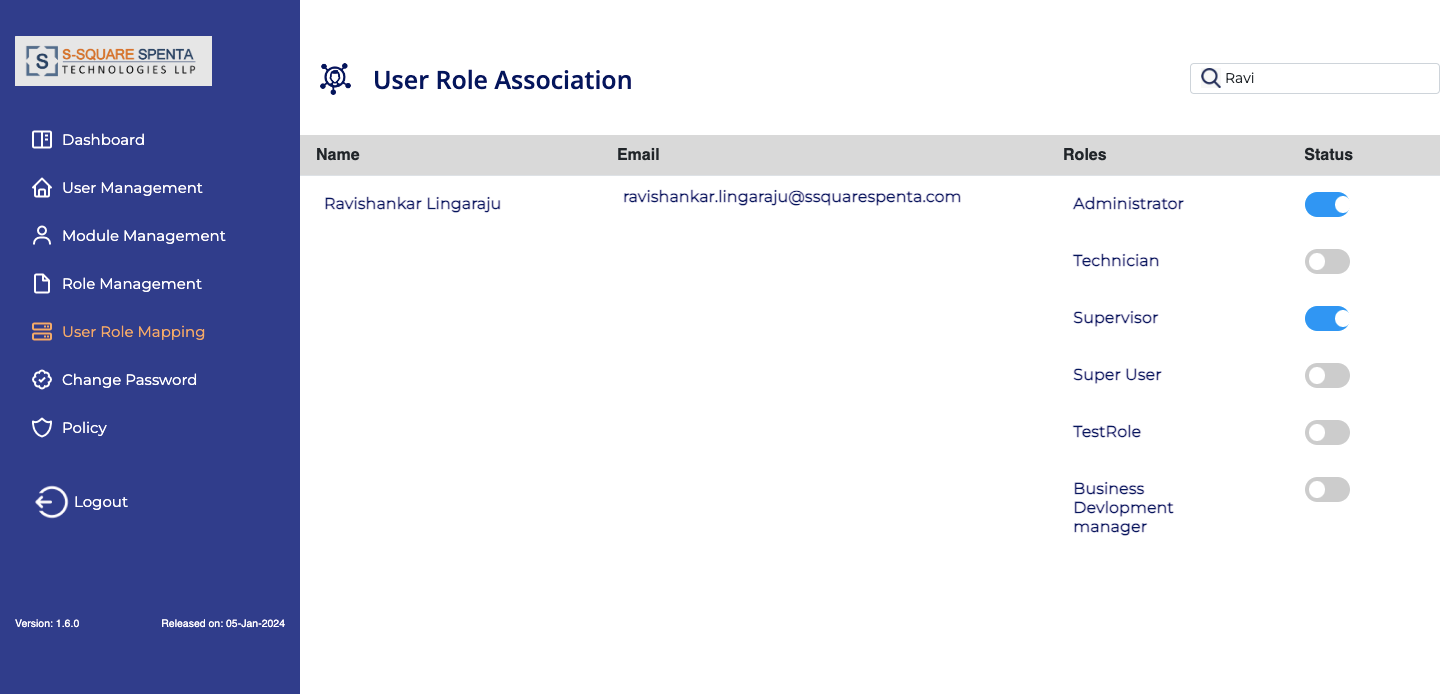
**Output**

* The function status is updated for any module.

# LLD: User-Role Mapping

## Search User

**Screen Design**



**Inputs**

* *Search name*: Name of a user to be search

**Processing**

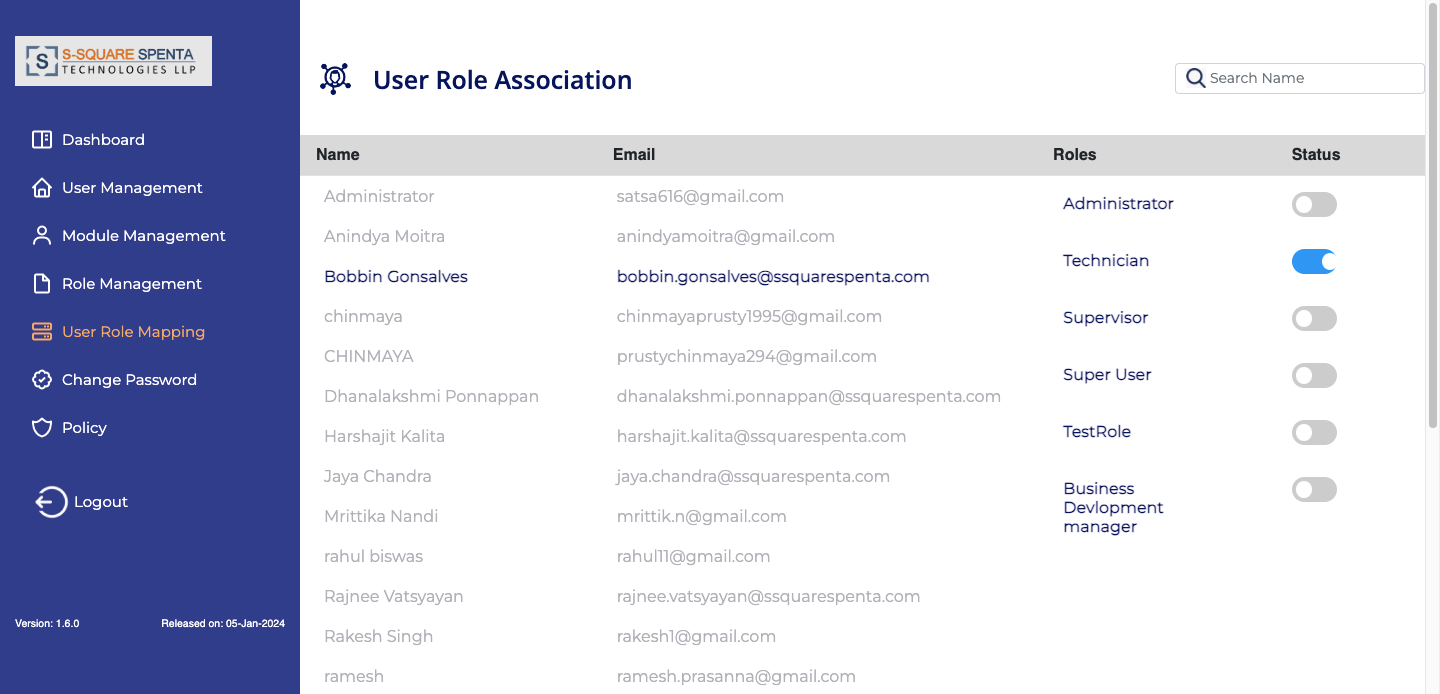
* User clicks User Role Mapping link from the left panel
* From the User Role Association page user types text in the search box.
* Auto suggestion dropdown appears
* On selecting any name displays the name with roles and status of the user.

**Output**

* Roles and user mapping is shown with status.

## Assign Role(s)

**Screen Design**



**Processing**

* User clicks User Role Mapping link from the left panel
* From the User Role Association page user selects any user.
* It shows all roles and status.
* On clicking the status the role is assigned or de-assigned to the user.

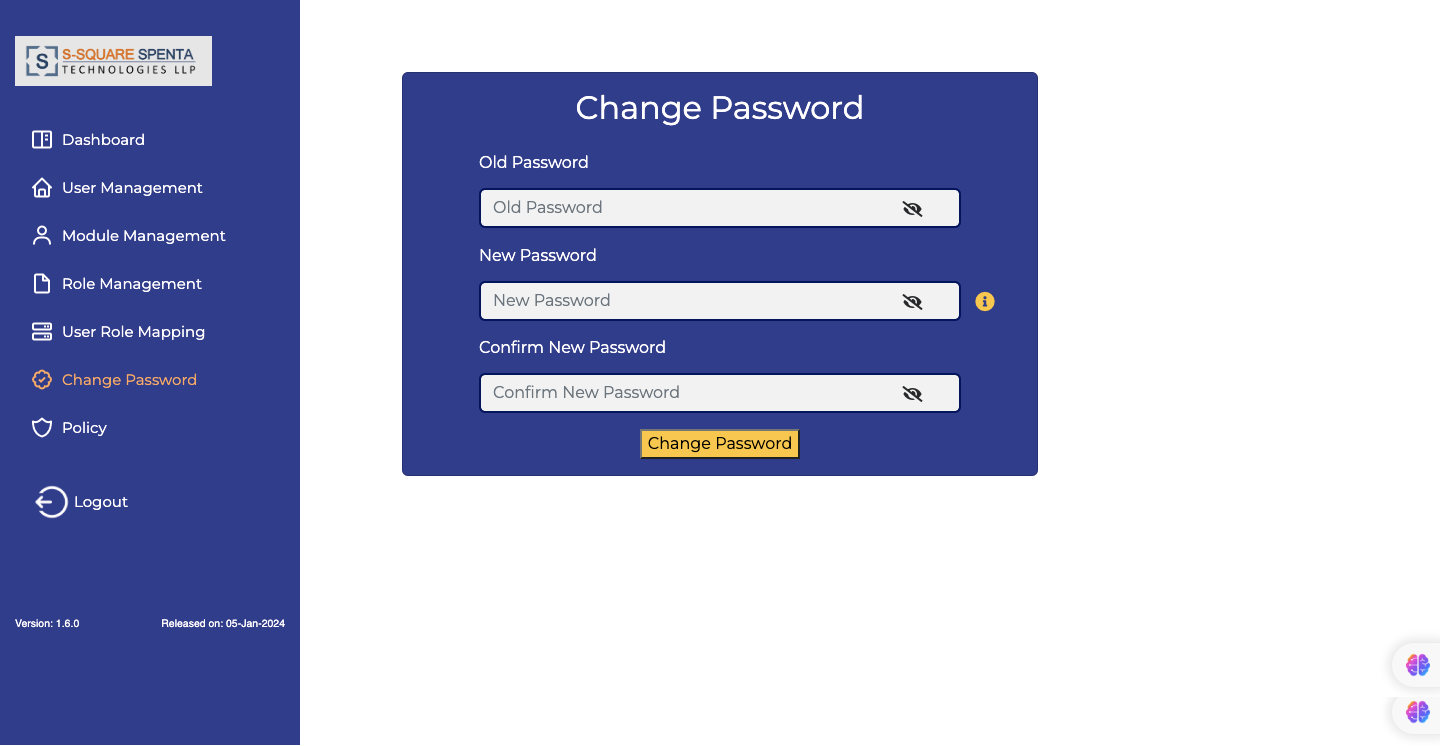
**Output**

* Role is assigned to the user.

# LLD: Change Password

## CHANGE PASSWORD

**Screen Design**



**Inputs**

* *Old Password:* existing password of the login user
* *New Password:* New password which needs to be set.
* *Confirm New Password:* Confirm the new password.

**Validations**

* The validation will be done according to the password policy set.

**Processing**

* When the user gives an old password and new password then the system checks the validation and matches the user password using auth component.

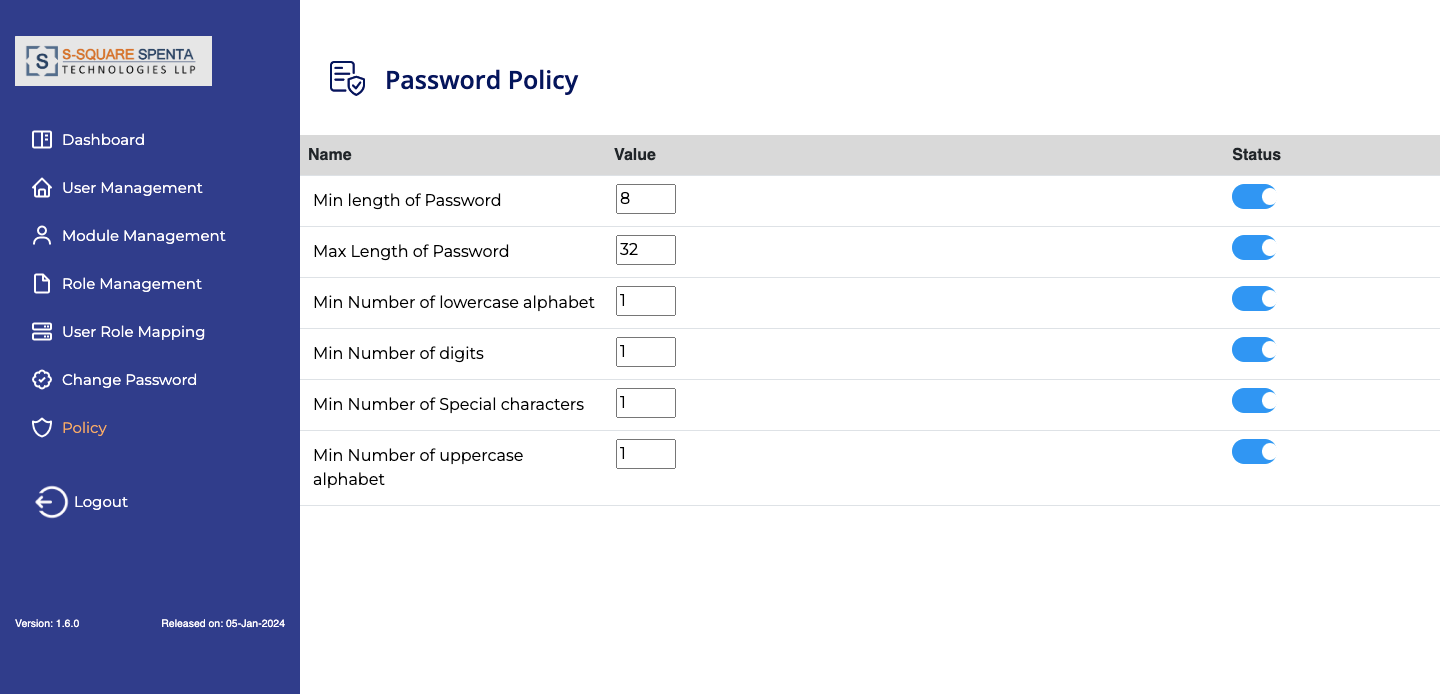
**Output**

* Password is changed.

# Password Policy

## Policy Configuration

**Screen Design**



**Processing**

* User select the policy link from the left panel
* The password policy screen opens with the prepopulated validation from the **password\_policy** table.
* User can make the policy active of deactive by clicking the status button
* user can change the value from the text box and it updates the table using ajax.

**Output**

* Password policy updated.