# Abstract

Objective of User Management Component is to manage application users. Thisservice need to be built as a generic service that allows us to manage (CRUDOperations) users. There is a need to build a user interface also to support this capability.

# Scenario

The objective of this component is 2 fold:

1. Allow administrators to define/create a new user needed for an application via an admin user interface. This service should capture various information about the user like his personal details and contact details apart from associating the user to a specific role. Admin Users also need to have an ability to edit an already created user’s data, make an existing user inactive and to view the users as needed.
2. Allow the users that are created to login to the application.

# Component Features

* OOB Super Admin user to be made available for logging into the application.
* Super Admin should have an ability to create different user types. For example something like Admin User, A
* pplication User, System User, …
* Ability for a super admin to create other usersof any type.
* User should have the following fields
  + User Login Id – Unique ID
  + User Password – One Time Generated Password (not to be displayed to the admin user)
  + First Name
  + Last Name
  + User Type
  + User Role => Should be selectable from a predefined list of roles defined in DB
* Any newly created admin user should be able view the user creation page
* Create another page with just a welcome message. All other user types when logged in to be taken to this welcome message page.
* Allow users to edit an already create user’s first name, last name and role.
* Allow admin users to make an existing user inactive upon which the user should not be able to login anymore.
* Allows users to view the list of all users in a grid and select any user to go to the detail view for edit/making an user inactive.

Authorization requirements

* Admin users are only expected to have access to admin screen and the associated backend and
* All others users are expected to have access to the welcome screen only.

# Assumptions

* Role could be managed in a separate table in the user database itself.

# Dependencies & Constraints

* All services must be reusable and should be easily integratable into any application.
* Keep the authentication part in a separate service.
* Leverage open source libraries where possible (for example spring boot, Spring Cloud, Maven) while building the project. For user interface using technologies like Angular, React JS or Vue JS.
* Ensure that the database scripts needed for the creation of the database and any startup metadata like the configuration categories are created as part of the startup using tools like Liquibase.
* Protect both front end and back end points using an appropriate mechanism. To support this requirement roles need to be created and there is a need to have a login page to authenticate the users.

# Exception Handling

* Use Http Status Code 409 for any validation failures and 500 for any other system failures.
* Ensure that the user gets an appropriate error message in the user interface in case of validation errors
* Ensure that all exceptions are logged in to a log file.