Disable Specfied User

User → Controller.cs: Request to disable user by a different user

Controller.cs → RDSFactory: factory = new RDSFactory()
RDSFactory → RDSFactory: Constructor()
RDSFactory → Controller.cs: return RDSFactory

Controller.cs → SqlDAO: IDataSource factory.GetDataSource(string DbName, string connInfo)
SqlDAO → SqlDAO: Constructor(string connInfo)

Data Store Timed Out
SqlDAO → SqlDAO: Catch SqlException
SqlDAO → Controller.cs: return "Data Source Timed Out"

Data Store Online
SqlDAO → Controller.cs: return SqlDAO

Controller.cs → SystemAccountManager: manager = new SystemAccountManager(IDataSource conn)
SystemAccountManager → SystemAccountManager: Constructor(IDataSource conn)
SystemAccountManager → Controller.cs: New SystemAccountManager

Controller.cs → UserAccount: new UserAccount(string username, string password DateTime RegistrationTimeStamp)
UserAccount → UserAccount: Constructor(string username, string password, DateTime RegistrationTimeStamp)
UserAccount → Controller.cs: New UserAccount

Controller.cs → SystemAccountManager: manager.DisableUserRecord(user:UserAccount)
SystemAccountManager → SqlDAO: isAdmin(user:UserAccount)
SqlDAO → SqlServer(MariaDB): SqlRead confirmAdmin()
SqlServer(MariaDB) → SqlServer(MariaDB): Select * from Admins where Username = @UserName and Password = @Pwd

Unauthorized User
SqlServer(MariaDB) → SqlDAO: SqlException
SqlDAO → SqlDAO: Catch SqlException
SqlDAO → SystemAccountManager: return false
SystemAccountManager → Controller.cs: return "Access Denied: Unauthorized"
Controller.cs → User: return "Access Denied: Unauthorized"

Unauthorized User
SqlDAO → SystemAccountManager: return false
SystemAccountManager → Controller.cs: return false
Controller.cs: return "Access Denied: Unauthorized"
Controller.cs → User: return "Access Denied: Unauthorized"

Authorized User
SqlServer(MariaDB) → SqlDAO: return true
SqlDAO → SystemAccountManager: return true
SystemAccountManager → SystemAccountManager: isInputValid(user: UserAccount)

Invalid Information
SystemAccountManager → Controller.cs: return "Invalid input"
Controller.cs → User: return "Invalid input"

Valid Information
SystemAccountManager → AccountService: AccountService.DisableUserRecord(user:UserAccount)
AccountService → SqlDAO: SqlDAO.DisableUser(user:UserAccount)
SqlDAO → SqlServer(MariaDB): isUser(user:UserAccount)
SqlServer(MariaDB) → SqlServer(MariaDB): Select * From Users where Username = @UserName OR email = @Email

data store timed out
SqlServer(MariaDB) → SqlDAO: SqlException
SqlDAO → SqlDAO: Catch SqlException
SqlDAO → AccountService: return "Database timed out"
AccountService → SystemAccountManager: return "Database timed out"

return "Database timed out"

return "Database timed out"

data store is online

User does exist

return True

Update users Set isActive = 0
where UserName = @UserName

return 1

return "User is now disabled"

return "User is now disabled"

return "User is now disabled"

return
"User is now disabled"

User does not exist

return False

return "User does not exist"

return "User does not exist"

return "User does not exist"

return
"User does not exist"