ChangeUserPassword—Takes in a UserID and password hash. Will replace the PasswordHash entry on the User table. Returns Confirmation as -1 if no user exists or 0 if success

CheckEmailChangeVerification— Takes in an Email address and unique random hash generated for that email. Returns Confirmation as 0 if the Hash did not match an existing entries hash (Hash is out of date/been replaced), 1 if the hash has already been used to change an email. 2 if no entry in the EmailChangeVerification table matches the input email. 3 if an email and hash match was found. If it returns 3, that means the entry now set validated to 1, meaning all future calls of this procedure should return 1.

CheckEmailVerification—Works identical to CheckEmailChangeVerification except with the EmailVerification table.

CheckIsEmailAvailable – Takes in an email address. Returns Confirmation as 0 if a User is already using that Email for their username. 1 if the email was not used yet (Email can be used for a user)

CheckPasswordChangeValidated – Takes in an Email and a unique random hash generated for that email. Returns Confirmation of 0 if the hash did not match an existing entries hash, 1 if that hash was already used to change a password, 2 if no entry with that email was found in PasswordChangeVerification table, 3 if the email and hash match an entry.

CheckUserVerified – Takes in UserID. Returns validated column from the EmailVerification table where the input email matches a table entry. Returns -1 if that users email is not found in the EmailVerification table.

CreateConversation – takes no input. Returns ConversationID as the ID of a newly created conversation table entry.

CreateConversationUser – Takes in a UserID and ConversationID to create a ConversationUser table entry. Returns Confirmation of 0 if successful, -1 if no user matches that UserID, -2 if no conversation matches that ConversationID

CreateEmailChangeVerification – Takes in Email, Hash, and OldEmail where Email is the new email a user wants to change their login info to, hash is a unique value generated to ensure a valid reset link was clicked, and oldemail is the users current email address before changing. It will create an EmailChangeVerification table entry if none exist, or change the has and validated column if one does exist. No return value.

CreateEmailVerification – Works identical to CreateEmailChangeVerification except it does not use the OldEmail input and it works with the EmailVerification table.

CreateMentee – Takes in a firstname, lastname, email, Username, and EWUID where all this info is grabbed from their SSO session. It will create a new User table entry and Mentee table entry with the given info.

CreateMentor – Takes Firstname, Lastname, Email, and MaxNumMentees where this info is acquired from a Mentor creation form. MaxNumMentees is the integer value maximum number of consecutive mentees a mentor wants. Creates User and Mentor table entry with the info, the username of this mentor will be the same as the email.

CreateMentorPair – Takes MentorID and MenteeID that are UserIDs of a mentor and mentee. Will create a MentorPair table entry

CreateMessage – Takes a ConversationID, OwnerID, ParentMessageID, and MessageContent where ConversationID is the Conversation ID of the conversation this message is posted in, OwnerID is the UserID of the user that posted the message, ParentMessageID is a MessageID that this message is in reply to (put 0 here if not replying to a previous message). Will create a Message table entry.

CreatePasswordChangeVerification – Works identical to CreateEmailVerification except works with the PasswordChangeVerification table.

DeleteMentee – Takes UserID of a mentee and deletes the User table and mentee table of this user IFF the UserID is a Mentee ID.

DeleteMentor – Takes a UserID of a mentor and deletes all profile information about this mentor including mentor and user tables IFF the UserID is a Mentor ID.

EditMentorMaxNumMentee – Takes UserID of a mentor and an int to denote the maximum number of mentees a mentor wants to have. Edits the Mentor table entry.

EditMessageContent – Takes a MessageID and Content, edits the Message with given MessageID's content with the input content. Updates the ChangeDateTime on the message to the current time.

GetAdminEmailCredentials – No input, returns the table entry of AdminEmailCredential for Email and Password used in automated emails.

GetBannedUsers – No input, returns UserID, UserName from the User table where the IsEnabled column = 2 (Banned)

GetConversation – Takes in ConversationID and returns OwnerID, Content, ModifiedDateTime from all messages that share the conversation ID.

GetConversationExists – Takes in UserID and ViewID where UserID is a mentee's UserID and ViewID is a Mentor's ID. Will return ConversationID if the 2 userIDs share a conversation, or -1 if they do not.

GetConversationStatus – Takes in ConversationID and returns the Status column for that conversation, -1 if no conversation exists.

GetConversationUsers – Takes ConversationID and returns list of UserIDs of Users that are in that conversation (It checks the ConversationUser table)

GetIndividualReport – Take EntryID where EntryID is the EntryID of a specific report and returns theReport table entry that matches the EntryID

GetMentee – Takes a UserID and returns the Mentee table entry that matches that UserID, -1 if the UserID was not found in the Mentee table

GetMenteeUserID_EwuID – Takes an EWUID and returns the UserID of a Mentee that has that EWUID, -1 if no mentee found with that EWUID

GetMenteesList – Takes MentorID that is a UserID of a mentor. Returns a list of UserIDs of all Mentees in MentorPair table that are paired to that MentorID

GetMentor – Takes UserID of a Mentor and returns the Mentor table entry for that UserID. -1 if no Mentor found with that UserID.

GetMentorsList – Works same as GetMenteesList but returns a list of mentors ids from a given mentee id.

GetMessage – Takes MessageID and returns the table entry for that message.

GetMessageContent – Takes MessageID and returns the content of that table entry

GetMessageDateTime – Takes MessageID and returns the ModifiedDateTime of that table entry

GetPendingMentors – No input, returns list of UserName, UserID of mentors that are enabled and have validated their emails (EmailVerification table validated column = 1)

GetRecommendationList – No input, Returns list of unique words used by mentors in all their related profile table entries to be used in search suggestions.

GetReportsAll – No input, returns list of ReportIDs of all Reports

GetReportsNew – No input, returns list of ReportIDs of all Reports that have not been reviewed.

GetReportsReviewed – No input, returns list of RteportIDs of all reports that have been reviewed.

GetUser – Takes in UserID, returns UserID, UserName, IsMentor, IsEnabled, Concat(firstname, ‘ ‘, lastname) for the given user, -1 as UserID if no user found.

GetUserFullName – Takes UserID, returns Concat(Firstname, ‘ ‘, lastname) as Name for user.

GetUserID_UserName – Takes UserName and returns UserID of the user with that Unique UserName.

GetUserPassword_UserName – Takes UserName and returns the Users password Hash for login verification.

ReportUser – Takes ReporterID, ReportedID, Details where the reporterID and ReportedID are userIDs and Details is a message content explaining the reason for the report. Returns Confirmation of 0 if success, -1 if reporter already has an unreviewed report in for that reportedID.

SearchMentor_All – No Input. Returns list of UserID, concat(firstname, ‘ ‘, lastname),jobname,univeristyname of all users that are mentors and enabled.

SearchMentors – Takes in a varchar(100) of a word or sentence. Returns a list of MentorIDs that have a profile table entry containing that word or sentence.

SetAdminEmailCredentials – Takes email and password, Updates the AdminEmailCredential table to store this info to later be used for automated emails.

SuspendUser – Takes in userID, Updates that user's user table IsEnabled entry to = 2. Returns Confirmation as -1 if no user found, 0 if success.

ValidatePasswordChange – Takes in Email, Updates the PasswordChangeVerification table entry for that email's validated column = 1. Returns -1 if no entry found, 0 if success.

NOTE: The following procedures were made to be as identical as possible to save on complexity.

PROCEDURES: SETUSERs – Takes UserID, entryName, Details for all entries and creates a new table entry in the Table with the same name as the Suffix portion of the procedure for that user. Work and Education have 2 name slots for JobName/CompanyName and DegreeName/UniversityName instead.

- SetUserAffiliation
- SetUserAffinityGroup
- SetUserAvailability
- SetUserBiography
- SetUserDisabled
- SetUserEducation
- SetUserExpertise
- SetUserPrimaryCommunication
- SetUserSkill
- SetUserWorkExperience

PROCEDURES: GETUSERs – Takes a UserID and returns a list of the table entries for a given table that match the userID.

- GetUserAffiliation
- GetUserAffinityGroups
- GetUserAvailability
- GetUserBiography
- GetUserEducation
- GetUserExpertise
- GetUserPrimaryCommunication
- GetUserSkills
- GetUserWorkExperience
- GetUsersConversations

PROCEDURES: EDITUSERs – Takes EntryIDIn, Name, and Details (Identical format to the SetUser procedures, except replace UserID with EntryID of a specific entry to edit). It will replace the entries values with your input values.

- EditUserAffiliation
- EditUserAffinityGroup
- EditUserAvailability
- EditUserBiography
- EditUserEducation
- EditUserEmail
- EditUserExpertise
- EditUserFirstName
- EditUserLastName
- EditUserPrimaryCommunication
- EditUserSkill
- EditUserWorkExperience

PROCEDURES: DELETEUSERs – Takes an EntryID and deletes the selected table's entry with the given entryid

- DeleteMentorPair (Unique, takes mentorID and MenteeID and deletes that entry)
- DeleteUserAffiliation
- DeleteUserAffinityGroup
- DeleteUserAvailability
- DeleteUserBiography
- DeleteUserEducation
- DeleteUserExpertise
- DeleteUserPrimaryCommunication
- DeleteUserSkill
- DeleteWorkExperience

PROCEDURES: SETCONVERSATIONs – Takes a conversationID and sets the given conversation status to the following values.

SetConversationAccepted: 0

SetConversationDenied: 3

SetConversationPending: 1

SetConversationUnreviewed: 2