Tables:

AdminEmailCredential

Purpose: Used to store email address and password that will be used by the automated email systems of the project. These credentials must be a valid email to have any email functions working.

Columns:

-EmailAddress: Email address for automated emails

-Password: Password for automated emails. This value is not hashed unfortunately because we need to use the password to send emails rather than just checking the hashed value of a user vs the stored table value. I could not find anywhere that gave a better solution for security and both Stu and our IT department said this was the only solution available to us for now.

Biography

Purpose: Used to store Biography info of Mentor accounts that can be viewed by Mentees. Biographies are unique per User and enforced with unique UserID

Columns:

-EntryID: Unique identifier for this table. Auto-increments when new Biography is created.

-UserID: UserID of User account this entry is tied to.

-Details: Varchar(4000) for storing large text entry from user

Conversation

Purpose: Conversations are the core table for sending messages between users. Every new connection between a mentor/mentee creates a new Conversation table entry.

Columns:

-ConversationID: Unique identifier for this table. Auto-increments when new Conversation

-OpenDateTime: DateTime when conversation was created

-status: 0=conversation was accepted by mentor; 1=conversation was set to pending by mentor; 2=conversation is unreviewed(Default when conversation is started); 3=conversation was denied by mentor

ConversationUser

Purpose: Small table to link a conversation to a user. Users should only be able to see or respond to conversations where they have a related ConversationUser.UserID value tied to the ConversationUser.ConversationID value on this table.

Columns:

-ConversationID: ConversationID that matches the conversation in question

-UserID: UserID that matches the user in question

EmailVerification

Purpose: Store unique hash to verify a user's email account on creation.

Columns:

-Email: email address claimed by the user. This is the email that will be sent a verification link.

-Hash: a unique generated hash tied to the email to verify the user with when they click the link

-validated: 0=default email link has not been clicked on and validated; 1=link was clicked and email validated.

EmailChangeVerification

Purpose: Works the same way as EmailVerification but has extra columns to handle verification and to notify the old account

Columns:

-OldEmail: Email account tied to the User before the update. This account will be notified that an email change was requested

-Email: The requested Email account. This account will be sent a verification link.

-Hash: a unique generated hash tied to the email to verify the user with when they click the link

-validated: 0=default email link has not been clicked on and validated; 1=link was clicked and email validated.

Mentee

Purpose: Stores User information for the Mentee that will be used more user side. Tied to the User table by UserID. This is a student account that wishes to match with mentors. They will use SSO to log in and we will not store or manage their password. Leave account validations to SSO. All these fields can be grabbed from SSO.

Columns:

-UserID: Unique number generated by the User table to identify a User

-EwuID: EWU ID number tied to the student. It should be an 8 digit number

-FirstName: Mentee's first name

-LastName: Mentee's last name

-Email: Mentees student email

Mentor

Purpose: Stores User information for the Mentor that will be used more user side.

Columns:

-UserID: Unique number generated by the User table to identify a User

-FirstName: Mentor's first name

-LastName: Mentor's last name

-Email: Email address tied to the Mentor's account. This is currently always defaulting to their login info but does need to if a use case is found.

-CurNumMentees: Value used to keep track of how many mentees a mentor is in charge of currently

-MaxNumMentees: Used to enforce the maximum allowable number of mentees a mentor can have at the same time. Originally this was static, but our client wanted the mentors to be able to change this value as time went on.

MentorPair

Purpose: Works similarly to the ConversationUser. It pairs a mentorID with a menteeID to allow a relationship between the two.

Columns:

-MentorID: UserID of mentor in the partnership

-MenteeID: UserID of mentee in the partnership

Message

Purpose: Stores message information to be used in conversations

Columns:

-MessageID: Unique identifier for the message table to be able to reference a specific message

-ConversationID: ConversationID from the conversation table. This is how we link messages to a given conversation and allow populating a conversation from just the message table.

-OwnerID: UserID of the user who posted the message

-ParentMessageID: MessageID of the message who this message is replying to. Insert 0 for this value if it is the first message of a conversation. This helps message ordering and readability.

-OpenDateTime: DateTime generated when the message is first sent

-ModifiedDateTime: DateTime generated when a message is altered. Defaults to OpenDateTime on creation. Currently un-used because it was designed before us and our client decided that altering a message could cause legal issues for the university unless we kept records, so we opted to not allow editing of messages.

-Content: Varchar(4000) for the body of the message.

PasswordChangeVerification

Purpose: Used to facilitate resetting a mentor's account password. Works similar to the EmailChangeVerification and EmailVerification tables. This was the table they were templated from and so it is the cleanest working.

Columns:

-Email: Email address that requested changing their password

-Hash: Unique hash to authenticate user once they click the verification link

-validated: 0=link has not been clicked; 1=link was clicked and is now expired because the mentor already reset their password.

Reports

Purpose: Used to gather information on a User when another User wishes to report them to the admins for a breach in the terms and conditions.

Columns:

-EntryID: Unique identifier for this table

-ReporterID: UserID of the user who put in a report about another user

-ReportedUserID: UserID of the user who was reported

-Details: Body of the message that the reporter can fill out to explain the reason for reporting this user

-ReportDate: DateTIme for when this report was submitted.

-IsReviewed: 0=report is new and has not been judged by an admin yet; 1=report was viewed by an admin and they have decided to either ban the user or judged it to be an incorrect justification for a ban. Either way, the report is kept for records but will not appear in the admins tool box for pending reports.

User

Purpose: Stores user information that is primarily used on the back end. This is the most central table in the database and is where all account creation attempts start.

Columns:

-UserID: Auto-generated 8 digit number that uniquely identifies a user and is used in many tables/pages to track the user.

-UserName: the username a user uses to log in. For mentors this will be their email, for mentees this is their SSO login.

-PasswordHash: Hash value of a User's password to validate them on login.

-IsMentor: 0=this user is a mentee; 1=this user is a mentor; 2=this user is an admin

-IsEnabled: 0=user account is disabled, this is the default value before a user validates their email address; 1=user account is enabled and ready for use; 2=user has been banned by an admin for an infraction of the terms and conditions

The following tables are all designed the same. They are all tables to hold Profile info for a mentor. Each table corresponds to a different profile item our client wanted a Mentor to be able to fill out and display to a potential mentee. These details are outlined in the SRS. Only Education is filled out for simplicity as the others work identically.

Education

Purpose: Store profile information about a user

Columns:

-EntryID: Unique Auto-generated identifier for this table

-UserID: UserID of the user who owns this entry

-DegreeName/UniversityName: varchar(100) fields for small user data

-Details: varchar(4000) optional room to elaborate on the above information

UserAffiliation

WorkExperience

Skill

PrimaryCommunication

MentorAffinityGroup

MentorAvailability

Expertise

Notes to be made about the database:

-The database and it's tables were designed extra loose with their rules to allow the project to expand in scope later if we wanted. Some of this came in handy and we implemented it, while others never got finished or brought up. For example, all User accounts have a UserID and all profile items have a UserID even though only Mentors will ever have profile information. This was left vague to allow mentees to have profiles later if needed. We also wanted to limit the amount of times the User table itself was queried so we split some information into the mentor/mentee tables that would have been better in the User. We would change this if we were to start over again.

-The EmailChangeVerification and EmailVerification were made very quickly and rough to get the features down before the end of our project so they have some flaws and are not well designed, but they work.

-Several of the User Profile tables are currently not being used. This is because we never got around to implementing them on the profile or account creation page just to save on complexity. They can be easily tacked on using the existing logic from profile/account creation and all procedures should be in place to support them like other profile info tables.

-Storing the password for administrator automated email system in plain text was something I did not want to do, but all my research seemed to show there is not a better way that still allows you to update the credentials from the website admin console. The more secure option is to store the credentials as hard coded variables in a php file outside the browsable directory, but to edit those credentials, you need terminal access to the server and an amount of Linux knowledge we did not want to expect from our client. Stu and EWU IT approved this step before we presented on it.