User Creation & Login/Auth Process Task

Summary: A user is able to visit a login/signup page that will allow them to login or if they are not an existing user, sign-up as a new user. In either case, the user will have a basic profile information page after sign-up or login, which will also be editable. Also, if the user has forgotten their password, they will be able to get a reset password email link and by following it, be able to reset the password. Below, we have put forth more detailed description and instructions in order to avoid any confusion & have attempted to clarify everything we could think of!

Please follow ALL the instructions below very closely and check-mark each to be sure you have followed them before submission:

1- Implementation of functionalities for new or existing users:

A. Sign-Up: New users will be prompted to sign-up with (1) Google; OR (2) their own email and password (See 2-B below). If a user chooses the Google option, they will be connected to Google auth api and be registered with Google's basic stored information. On the other hand, if the user chooses to enter their email, then they must enter their email, a password, and proceed to 2-C (see below). Note that username should be an email only. Also note that Google auth integration is a major part of this challenge, it must be implemented and work solidly, otherwise, your submission will not be evaluated.

B. Profile Info, Setup, or Edit: Once the user is registered in 2-B (just with email & pass or with the Google api), they will then be proceeded to enter profile information in 2-C. Once the profile information entry is saved, the user will be directed to 2-D. From here, If at any point, a user chooses to edit profile info, they will be directed back to 2-C again and proceed to edit previously entered data (fetch previously populated data in the fields) or cancel. If a user chooses to cancel, they will be directed back to 2-D. Note that if the user is an existing user and logs-in (See 1-C below), they will also be directed to 2-D. Also note that if the user edits their email while in 2-C the registered email address/username is replaced with the newly entered email and the user must enter this new email upon future login in 2-A. However, the email accepted/obtained through Google auth cannot be edited (show email in its field but disabled for this scenario).

C. Login/Authentication: Using 2-A, existing users will be able to login with Google account api or their designated username and password depending on how they signed up in the first place. Once a user successfully logs-in, the user will be directed to 2-D. See 1-B for description of all related functionality in Profile Info, Setup, or Edit. If the user does not exist, display a message that the username entered does not exist. If the user password is entered incorrectly, display a message that password is incorrect. If a user logs out in 2-D, session/auth will expire and the user will be directed to 2-A. Additionally, see related important note/reminder in section 4 below.

D. Password Reset: If a user is registered but forgot the password, they can request a reset password in 2-A. Once proceeded to reset, the user will be directed to 2-E and enter their registered email/username. After the user has provided the email & requested reset, email the user a reset password link that expires in 15 minutes after it's sent. If the email does not exist, display an error message to the user in 2-E that the email does not exist. If an email link is sent, display a success message to check their email in 2-E . The user should be able to reset the password through the

emailed link in 2-F. If the link followed by the user has expired, display a message to the user in 2-F that the link has expired and to request a new reset with link to 2-E. A tip: test this aspect of the task after implementation on your deployed application on the web and send the reset link to different email providers to be sure it's working (so, don't just rely on a local machine testing and assume it will work for web users).

2- Implementation of necessary UI / Pages, Links, or Buttons for the required functionalities:

A. "Login": This page/url is considered the home page or starting page and the user will have the functionality to login with the username and pass fields shown or with Google login button. This page will also have two additional buttons or links: "New User Sign-Up" directing to 2-B and "Forgot Password" directing to 2-E.

B. "Sign-Up": In this page/url, the user will have the username and pass fields and the Google sign-up/login button. Additionally, his page will also have a button or link for "Existing User Login" which will direct the user back to 2-A.

C. "Enter Profile Information": This page/url serves as both the registration of contact/profile information and also the subsequent editing of existing profile entries. Four fields in this form will be provided for: full name, address, telephone, and email (each with a single entry line/field only). This page will also have two buttons: "Save & Continue" and "Cancel" … Save & Continue will save entries and direct the user to 2-D. The Cancel button will ignore the new/edited entries and direct the user to 2-D.

D. "Main Profile": This is the main profile page/url displaying the user's contact/entries in view or display mode. This page will have two buttons: "Edit" and "Logout". If a user chooses to "Edit", they will be directed to 2-C. If a user chooses "Logout", session/auth will expire and the user will be directed to 2-A (also see notes/reminders in section 4 below)

E. "Forgot Password": This page/url will have one field for the user to enter their previously entered username/email address and two buttons "Send Reset Link" and "Back to Login". The "Send Rest Link" button will execute that action and display proper confirmation and the "Back to Login" button will direct the user to 2-A.

F. "Reset Password": This page/url is rendered only following the link provided and sent to the user's entered email address. User will be shown the field for a new password and a button "Reset." Once "Reset" action is taken, the new password is saved and the user will be directed to the "Login" page. At your option make the user retype the new password to make sure it's a match and to avoid mistakes.

3- Implementation of Unit Testing:

A. Implementation of Unit Testing is not required for ALL functionalities listed or for all algorithms used but you must show us some aspects of unit testing skills for some of the core functionalities/algorithms as you choose. We do not require testing for all in the interest of time but would like to see some implementation samples showing that you are capable of handling this aspect of programming.

4- A few reminders/notes on security & authentication (very important aspect of this task)

A. Please make sure that the user can not view 2-D or 2-C without the login or signup process (See 2-A and 2-B) even by tricking or conducting unusual interactions with the app! Such unusual user interactions can include but are not limited to: (1) clicking the back button on the browser (one or several times) or refreshing form/page to view previous 2-D or 2-C without an active session/authentication; (2) copy and paste a url for 2-D or 2-C in browser and view the content of these pages without authentication; etc. Also, a reminder that the user needs to be re-authenticated after logging out in 2-C and be directed to 2-A. In sum, if you allow users to see content of 2-D or 2-C without authentication or an active session (even for a split second), your implementation will be considered flawed and incomplete.

5- Required technologies and technical specifications in implementing this challenge:

A. Back-End REST API: **Use Golang only**

B. Database: **Use mySQL only**

C. Front-End: **Golang**

   a. Create a server-rendered application which will communicate through REST API.
   b. Use Golang for frontend using Go templates. The communication with REST API should be with Golang. Don't create a javascript single page application (SPA) or use any of the popular js frameworks (react, angular, vue, etc). Use vanilla javascript/Es6 only to make UI dynamic when or if needed.
   c. This is mainly a back-end challenge, so do not spend too much time on making the frontend or UI beautiful, although it should be functional and user friendly based on the UI requirements given above.

D. DO NOT use any Golang frameworks. Only the standard Go library can be used. 3rd party libraries such as the Gorilla tool kit are okay if needed.

E. MUST create the REST API. If you don't have the separate API for all the functionalities, your submission will not be considered. Please note that the overall organization, structure, and architecture of the application is very important in our evaluation. To clarify, separate API from the frontend, means that if you were to change the frontend technology to something entirely different, the API would stand independently and there would be no need to modify it.

6- How to turn in the results:

A. Test your Implementation First: please test your application thoroughly for all the functionalities as instructed to make sure it all works according to the instructions before submitting or providing the link. Note that your application may run perfectly on a local machine setting but endup with errors or issues when deployed or viewed as a user on the web, so we encourage you to test your app locally and as deployed on the web.