Comp 4350
Group 8
Favour Trader Web
Epics and User Stories Outline.

Epics:

These are our highest level, broadest user stories to be broken up into smaller ones (features).

1. As a skilled individual, I would like to create a profile that specifies the skills/services that I can offer, the skills/services that I am seeking and other details about myself so that others can decide if they would like to offer a trade to me.

2. As a skilled individual, I would like to search for and view other skilled individuals in my area that match well with my possessed and desired skills/services so that I can decide if I would like to offer a trade to them.

3. As a skilled individual, I would like to be able to offer, accept or negotiate a trade with another skilled individual so that we can mutually benefit from each-other's talents and skills.

4. As a current user, I would like to manage my secure info such as my email and password so that I keep my account security robust.

5. As a regular user, I would like to be able to save other users as friends so that we can regularly trade with each other.

6. As a Facebook user, I would like to create an account and login using facebook so I do not need to add to the amount of email/passwords I have to remember.

User Stories (features):

These are specific features of the app sorted under our epics. They are made up of several developer tasks/requirements (these will range from front to backend tasks) and will from now on correspond to a single pull request.

1. Epic 1

    1.1. As a potential user, I would like to be able to sign up using my name, an email and password, so that I can build my profile and start trading.

        1.1.1. Ensure a post route /api/users/register that accepts a body with first name, last name, email and password, saves the new user to the database and returns a json body containing a success bool, message and the new user.

- 1.1.2. Create a RegisterForm react component with first name, last name, email and password fields. Within the component, validate that each field is provided and passwords match before posting.

- 1.1.3. Have the RegisterForm post to our endpoint. If success: false is returned in the response, display in the the message field. If successful, direct the user to the profile update page to finish their profile.

1.2. As a user, I would like to customize my profile to represent myself as a skilled individual so that I can secure trades and mutually benefit with others.

- 1.2.1. Ensure there is a secure /api/users/:id/profile get route that returns the 'users': firstName, lastName, location, picture, about and populated has/wants arrays.

- 1.2.2. Make a ProfilePage react component with a 'uID' prop that will specify the :id param for our fetch call. It will then displays a user's profile picture, full name, location, about section, skill section (this will be its own feature, you will pass the has/wants to the SkillTable component). Additionally if the page does not belong to logged in user, render a Offer A Trade button that will (See provided mockup).

- 1.2.3. Ensure there is a secure /api/users/update/profile post route that takes the appropriate user profile fields from the req.body and updates them in db. Return a success bool, message and the updated user.

- 1.2.4. Add profile edit button that should only appear if the UserProfilePage belongs to the currently logged in user. Clicking this button will swap-in a form identical in composition to the normal screen, only replacing each field with an input and adding "Change Password" (this will be a separate feature so leave it dead), "Save" and "Cancel" at the bottom.

- 1.2.5. Validate that the user has a name and location set before sending, they can empty their about or profile picture if they please. Post our updates to the route, if success:false keep form open and display message. If successful, close the form and refresh the page with updated user info.

1.3. As a user I would like to update my password so that I can maintain my account's security.

1.3.1. Ensure there is a secure /api/users/update/password post route that reads a “currentPass” and “newPass” field from the req.body. Check the to see if currentPass matches, if true update password with newPass and return success true with message. If false, do not update and return success false with a message.

1.3.2. Make a PassChangePage and a PassChangeForm component that has input for currPass, newPass and verifyNewPass and submit/cancel button.

1.3.3. Verify all fields are filled and that newPass/verify match before posting. On submit, post to the endpoint, if fails: display message, if success: redirect to back user profile page. On cancel, redirect to user profile page.

1.4. As a user I would like to display and update the skills/talents I possess and I am seeking, as well as provide a further description on each, so that I can better match with users for a trade. (This is a big ticket and might need to be shared or split up)

1.4.1. Make a SkillsTable react component, with has and wants props, these will be passed their respective populated arrays from the parent UserProfilePage (this cuts down on the number of separate fetches we have to rely on)

1.4.2. The table will have has column and wants column (rows don't mean anything). In each column will be a list of SkillItem components (more on these up next) *(See mockup)

1.4.3. Ensure there are secure /api/users/skills/(has & wants) post routes that take an object identical to those of in the has/wants arrays in he user model (eg. [{ skill: skillId, description: String}]) and adds it to the array if that skill doesn't already exists. If failed: {success:false, message}, If added: {success:true, message, and the full newly updated array}.

1.4.4. If the table belongs to the currently logged in user. Display a green font-awesome + icon in the column header. Clicking this will render a NewSkillForm component in the header. This component will have a ‘skillset’ prop identifying if it for ‘has’ or ‘wants. Visually it will have a drop down menu for choosing the skill (populated via our api/skills/all endpoint), a text input for the description and of course submit/cancel buttons.

            1.4.5. Validate that the user has chosen a skill, description is optional. On submit, post to the api/users/skills/{skillset}. If failed: close form and display the message. If succeeded, close form and populate the list with the updatedArray.

            1.4.6. Next, Ensure there are secure /api/user/skills/(has & wants) delete routes. They will take a skillID from the req.body and attempt to delete it from the user's respective skill list. It will return the same essentially as our post route.

            1.4.7. If the table belongs to our user, each SkillItem component will show a red font-awesome trash can icon on the far right. Clicking this will delete using our new endpoint. If failed, display the error message, if success: the new resulting array will have to be bubbled up to the parent SkillTable and refreshed. This can be done several ways here is one example to get you started

    1.5. As an existing user, I would like to be able to login using my email/password and logout so that I can access my account. (Pretty much done)

            1.5.1. Ensure there is a POST /api/users/login route that checks the user's username and password and return a jwt access-token (already exists)

            1.5.2. Create LoginPage and LoginForm components, inspiration is pretty much just the instagram web login *(see Mockup)

            1.5.3. Validate that both username and password are provided before wasting a fetch call.

            1.5.4. On submit, post to the login route, if fail:display error, if pass: store token and redirect to the MainPage (by default this is users who 1 to 1 match with your has/wants).

    1.6. As a reliable user, I would like to be able to display on my profile how many trades or favours I have successfully completed and recieve/display reviews from users I have made trades with.

2. Epic 2

    2.1. As a user, I would like to view a listing of all other users in my city with 3 filtering options. Either they want what I have or Have what I want or with wants for skills that I have AND have skills that I want so that I can find the best possible matchup for a trade.

2.1.1. Ensure there is a secure GET /api/users/matches + (optional query string) routes that return a response with {success, message, matches: [{ uID, firstName, LastName, profilePic, about}]. With no query, all users in your location (city,state,country) will be returned. With “/matches?filter=both” users who want want you have and have what you want are returned. With “?filter=has” all users who have what you want are returned. Finally, with “?filter=wants” all users who want what you have are returned.

2.1.2. Create a MatchCard component that shows the user’s displayPic, firstName, last Initial, and the first idk 200-300 chars of their about. Also a button or link that says view profile. (pass in the user’s Id in the ‘uID’ prop)

2.1.3. The ResultsPage will be passed a prop “filter” that will determine which filter strategy to place in the onMound fetch. It will have a Header that displays the post response message if successful, if not it will say “Sorry :(we couldn't find any matches, try updating your has and wants”. The page’s body will render all of your matches into MatchCards. (In the future we may want pagination and “how many per page” but that is lower priority)

2.1.4. Hitting a matchcard’s view profile button/link redirects you to the user’s profile by Redirecting to the ProfilePage component and passing it the userId in it’s props.

- Note: Until 2.1 is complete it is the only really high priority feature within the epic. Once it is complete other features such as pagination and more location precise filtering are unblocked.

2.2. (BLOCKED by 2.1) As a user, I would like to be able to filter my matches by their in distance from my actual location so that my matches are more convenient and I can connect with others in my community.

2.3. (BLOCKED by 2.1) As a user, I would like to be able choose how many match cards are displayed on my page and browse pages of matches so that I can more effectively view matches.

2.4. (BLOCKED by 2.1) As a user, I would like to be able to filter/sort my matches by their rating and/or how many contracts they have completed so that I can ensure I am trading with reliable users.

3. Epic 3

    3.1. As a user, I would like to be able to create and offer a trade to another user so that they can decide if they would like to make a trade with me.

    3.2. As a user, I would like to be able to view lists of my active, offered, requested or complete trades so that I can keep track of what trades I have.

    3.3. As a user, I would like see an overview of a trade keep track of it’s status.

    3.4. As a user, I would like to be able to see lists of a trade terms and for an active trade, see completion status of term as well as indicate when I have completed my terms so that I can decide whether or not I want to accept a trade and keep track of its completion if I do.

4. Epic 4

    4.1. As a current user, I would like to have settings page so I can view my options for managing my security and privacy.

    4.2. As a current user, I would like to be able to change my password so that If I can keep my account secure.

    4.3. As a current user, I would like to be able to change my email so that I can change email accounts.

    4.4. As a current user, I would like to manage which other users are blocked from seeing my account so that I can feel safe If another user is harassing me.

5.

- NOTE: When you are creating and testing new components, they may not have an exact home yet. I recommend creating a sandbox page in the router where you can develop/test your components, just remember to remove it from your PR.