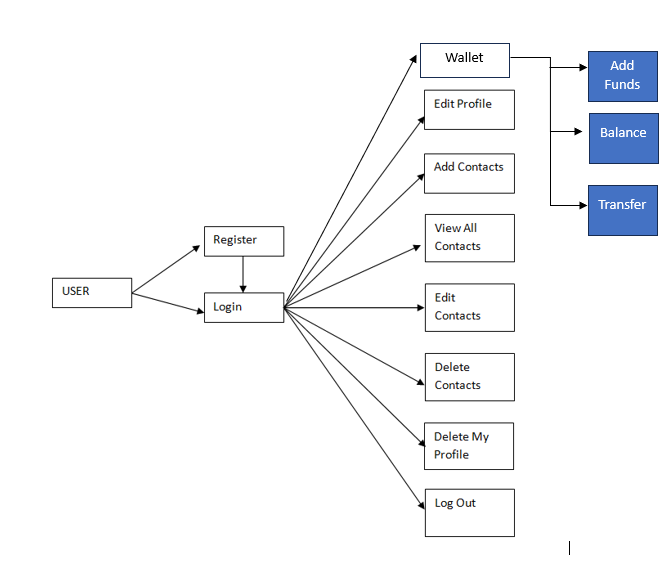
**Contact Transaction App**

The use case is to maintain contact details of users/profile where as a user/profile you could able to

1. Register
2. Login
3. Wallet Transactions
4. Edit Profile
5. Add Contacts
6. Edit Contacts
7. View Contacts
8. Delete Contacts
9. Delete Profile

Use Case Diagram



Business Requirement

1. User must be able to register his details first before he could login.
2. Register:
   1. User enters profile details like his email-id, name, password, phone number, date of birth which is recorded in the database and an unique id which is treated as a profile id will be auto-generated and will be shown to the user to login with the valid password
   2. Perform the validation on the input so that none of the details are empty
   3. After successful registration you could able to see the login page with the generated profile id

Validations:

ProfileId / Email-Id: Valid email-id must have @, .(dot) mandatorily

Name: Must be between 5 to 60 characters

Password: Must be strong password that includes special characters, numbers, upper case & lower case letters

Date of Birth: Valid date in past 18 years

Phone: 10 digits

1. Login:
   1. User can login by entering email-id and password if he has already registered
   2. Perform the validation on the input so that none of the fields are empty
   3. If login credentials are wrong, re-direct back to the Login page and ask the user to enter valid credentials
   4. If login credentials are right, go to the success page where user could able to perform operations shown in the above use case diagram like add contacts, delete contacts and etc.
2. Wallet:
   1. Every user can have a wallet through which they can transfer the amount to their contacts, doing that must update both the profile wallets i.e., the sender & receiver
   2. Add Funds must able to add the amount linked to the account of the user, if the amount trying to add is insufficient in the Account, show the appropriate message like Insufficient balance in Account
   3. Balance will show the wallet balance

Note: An Account must be maintained for each profile with some random amount to perform the transaction.

1. Add Contact:
   1. Logged in user will store new contact details like name & phone number
   2. When a user adds the new contact an unique contact id will be generated which may require to manipulate the contact details
   3. Logged in user must able to transfer amount to their contacts, which will be updated in their contacts wallet
   4. Contact details are stored along with the logged in user profile id
2. View All Contacts:
   1. Shows all the contacts of a logged in user in a table
3. Edit Profile:
   1. Logged in user must able to update his phone number, password
4. Edit Contacts:
   1. Logged in user must able to edit his corresponding contact book like phone, number, name (If Alex is the person logged in and he has some people in his contact book like Bruce or Charles, then Alex must able to edit their details like name, phone number, edit via contact\_id and profile\_id)
5. Delete Contact:
   1. Logged in user could able to delete people from his contact book
   2. Deletion of contact done via contact id to avoid the deletion of other profile contacts
6. Delete Profile:
   1. Logged in user must able to delete his profile permanently
   2. Automatically all his contact book entries must be deleted before performing the deletion of the profile
7. Log Out
   1. Terminate the session and redirect back to welcome page

Note:

* Backend must have all the API’s & methods to perform the above operations
* Write the test cases for each operations that you want to implement, follow TDD
* Follow the coding standards while naming the class, methods, variables and write the comments to describe your API’s & methods so that other developers could understand what exactly the code does
* Frontend must have the components to interact with the backend, which must be a single page application so that once you login only part of the page must be updated when user does any operation & it must not refresh entire

**Feedback from Nilesh:**

1. (Custom) Exception handling is must.
2. Login Log should be created.
3. Login Failed attempt should be captured.
4. Frontend should be developed using React micro front end architecture.
5. Caching should be implemented wherever applicable.
6. Front end validation should be cleared spelt
7. Database should have proper constraints / indexing / views etc