Functionalities: (banking account)

1) User account (create, update, delete)

a) Create: name, CNP (UNIQUE), phone number, email, password

The app should display a registration form with the following fields:

- i) first name: A string with less than 100 characters
- ii) last name: A string with less than 100 characters
- iii) CNP: the CNP must be valid and unique
- iv) phone number: the phone number must be valid and unique
- v) e-mail: the email must be valid and unique
- vi) password:
  - (1) The password will always be hidden when written — for each character, there will be a “*” in its place.
  - (2) the password must be between 8 and 100 characters
- vii) confirm password:
  - (1) The password will always be hidden when written — for each character, there will be a “*” in its place.
  - (2) the password must be between 8 and 100 characters
  - (3) the password and confirm password fields must be equal
- viii) the register button:
  - (1) Pressing this button will verify all fields and displays an error below each incorrect or incomplete field
  - (2) If all fields are correct, save the user’s details and redirect the user to the Login page
  - (3) When creating an account, their ID will be automatically created and assigned to their account.
- ix) the cancel button: Pressing this button will redirect the user to the Login page.

b) Update: name, phone number, email, password (everything from create, except the CNP) - based on the password. The app should display a form with the following fields:

- i) first name: Contains the user's current first name, can be modified by the user
- ii) last name: Contains the user's current last name, can be modified by the user
- iii) phone number: Contains the user's current phone number, can be modified by the user
- iv) e-mail: Contains the user's current e-mail address, can be modified by the user
- v) current password: Initially empty, the user must enter the current password to be able to update the existing information. The password will always be hidden when written — for each character, there will be a “*” in its place.

vi) new password: Initially empty. If the field is left empty the current password is not changed. The password will always be hidden when written — for each character, there will be a “*” in its place.

vii) confirm new password: Initially empty. If the field is left empty the current password is not changed. The password will always be hidden when written — for each character, there will be a “*” in its place.

viii) the Update button: Pressing this button will do the following:
   (1) verify that that all fields are valid, using the same rules as user account creation
   (2) check if the current password matches the user's password
   (3) if all fields are correct, update the user's information

ix) the Cancel button: Pressing this button redirects the user to the main screen

c) Delete: based on the password. In the update user form, there should be a button that redirects you to a window with the following items:
   i) a warning message, with the text: "Are you really sure you want to delete your user account? This action cannot be reversed!"
   ii) the current password field: Initially empty, the user must enter the current password to delete the user account. The password will always be hidden when written — for each character, there will be a “*” in its place.
   iii) the Delete button, with the text: "Yes, delete my user account". Pressing this button verifies that the password matches the user's current password. If the passwords are not equal, then an error message is displayed below the current password field. If the passwords are equal, the user's account is deleted, and the form is closed
   iv) the Cancel button. Pressing this button closes the window and returns the user to the the update form

2) User login

a) The user logs in using their email and password. The application should display a login form on the login screen with two clearly labeled fields: email and password. The password will always be hidden when written — for each character, there will be a “*” in its place.

b) There are two buttons below those 2 fields, one below the other. The first button and contains the text “Log in” and when pressed, validates the credentials above (email and password) against the database. If a user with the given email and password exists, the user is redirected to the main screen. If the given email and password do not match in the database to an existing account, the password field gets cleared and a message in red appears, reading “Wrong email or password”. The second button contains the text “Create an account”, which if clicked redirects the user to the User account creation page.

c) The password is encrypted with salt when saved in the database for security reasons explained below.
d) Password validation is done as follows: after the ‘log in’ button is pressed, we will check for an existing user in the database based on the email provided. Afterwards, if a user exists, we will take the PasswordSalt and concatenate it to the one typed by the user in the password field. Then, we will use an encryption algorithm/function to encrypt the newly concatenated string and validate it based on the HashedPasword stored in the database. This will ensure thorough security, since the actual password will never be stored in the database and even in case of a security breach, the potential hacker will need to know the exact encryption algorithm in order to be able to decrypt the hashed password.
e) Requirements division
    i) user email field
    ii) user password field
    iii) login button
    iv) sign up button
    v) credentials validation against the database
    vi) error message display if credentials are not correct + clear password field
    vii) redirection to main window if credentials are successful

3) User Log Out

a) On the main page, a user will have a log out button(in the upper right corner) which will invalidate his session upon button press and will redirect to the login/sign-up page
b) Session management, a user will be able to access the main page + all features if he has a valid session. A valid session is obtained by successful login. Also, by logging out, the session is invalidated.

4) Bank account

a) **Bank account creation (same thing as above)**
    i) The user should be able to choose which currency they desire to create an account in
    ii) The bank account will be given a unique IBAN (used as an ID) for the card/account
    iii) The page will display a list of all currencies available, the user must select one from them (by default the "Lei" currency will be selected, the user is free to choose any other currency). Above the list a label with the text “Select currency: ” is displayed. Every item of the list will consist of: an icon representing the currency displayed on the left side and the name of the currency on the right side.

iv) A text field will be displayed below the list, with the label “Custom name”. The user can edit the text field to set a custom name for the bank account when creating.

v) A button with the text "Cancel creation" will be displayed at the top left of the page, when the user clicks the button they will be brought to the main page of the application. Any previous currency selection will be discarded and no bank account is created.

vi) Another button with the text "Confirm creation" will be displayed at the bottom right of the page, when the user clicks the button a new bank account will be created. A confirmation message shall be displayed in the center with the text "Bank account successfully created" in order to confirm the success of the procedure. Afterwards, the confirmation message automatically disappears and the user is brought back to the main page of the application. The initial sum of a newly created bank account will be always set to 0 of the respective currency. A unique IBAN will be generated for the bank account (see below for the specifications of generating a unique IBAN - transaction section), such that the bank account can be identified for future transactions. The default values for the rest of the attributes of a bank account are: Custom name - “” (empty string, when displaying, the IBAN and currency will be used instead); Daily Limit (maximum amount a user can spend on a day) - 1000 (of the respective currency); Maximum amount per transaction (how much a user can spend per transaction) - 200 (of the respective currency); Maximum number of transactions per day - 10.

    (1) IBAN generation:

        (a) We will use only the romanian IBAN which has the following structure

            (i) Country -> RO

            (ii) IBAN Checksum: 2 digits

            (iii) SWIFT/BIC Code: 4 alphabet letters (for our bank would be SEUP)

            (iv) Account Number: 16 characters (letters and numbers) that are generated randomly

            (v) the IBAN is case insensitive (it does not matter if the letters are uppercase or lowercase)

        (b) IBAN Computation Process

            (i) Start with an incomplete IBAN

                1. The IBAN begins with "RO00" (where "00" is a placeholder for the checksum).

                2. Followed by the 4-letter SWIFT/BIC code.

                3. Followed by the 16-character account number.

            (ii) Rearrange the IBAN for computation

                1. Move the first four characters ("RO00") to the end.

2. The new order is: **SWIFT/BIC Code + Account Number + RO00**.

(iii) Convert letters to numbers
1. Each letter in the IBAN is replaced with its corresponding numerical value: A = 10, B = 11, C = 12, ..., Z = 35.
2. This applies to both the **SWIFT/BIC Code** and any **letters in the account number**.
3. "RO" is converted to "2724", and "00" remains "00".

(iv) Create a single large number
1. After converting all letters, the entire IBAN becomes a long numeric string.

(v) Compute the MOD 97 checksum
1. Divide the large number by 97 and take the remainder.
2. Subtract the remainder from 98.
3. The result is the **IBAN Checksum** (if it's a single digit, add a leading zero).

(vi) Replace "00" with the computed checksum
1. The final IBAN is now valid and ready for use.

b) **Bank account deletion**

i) This page is used only as a measure to prevent accidental deletion of bank accounts

ii) At the center of the screen the text "Are you sure you want to delete this bank account?" will be displayed, directly below the text there are two buttons with the text "No", respectively "Yes". The "No" button is on the left and the "Yes" button is on the right.

iii) If the user clicks the "No" button, they will be brought back to the main page.

iv) If the user clicks the "Yes" button, they will be brought to a Verify page where they must enter the credentials for security measures. There is one field, the field has a label “Password” above it and in the field the user must enter their password (the password is hidden, meaning as the user types it, each character is replaced with “*”). Below the “Password” field, there are two buttons with the text “Back”, respectively “Confirm”. The “Back” button is on the left and the “Confirm” button is on the right. When the user clicks the “Back” button, they are brought back to the “Bank account deletion” page. When the user clicks the “Confirm” button, a confirmation message will be displayed at the center of the screen with the text "Bank account successfully deleted", if the credentials are valid. If the credentials are invalid, a message will be displayed at the center over the page with the message “Wrong credentials. Try again.”. If the credentials are valid, The bank account is deleted from the database and shall not be further displayed (the application must make sure the proper update is performed for perfect synchronization). The user is brought back to the main page - the application must ensure that another bank account is selected as the current bank account (or selected to "None" when the user does not have any associated bank accounts).

c) **Bank Account Details**

   i) When the user clicks on the info button from the main page, they are brought to a page displaying various information about the selected bank account - the info page.

   ii) The info page will display:

      (1) Currency: this will be a text field consisting of the string representation of the currency (Ex.: "Lei", "Euro", "Dollar", etc.)

      (2) Sold: the bank account's current sum of money in the respective currency. This will be a text field consisting of the string representation of the sold, a float type (Ex: "100.39", "69.69", etc.)

      (3) Maximum number of transactions per day: will be a text field

      (4) Daily limit: will be a text field, the amount of money the user can spend a day before reaching a limit.

      (5) Maximum amount per transaction: will be a text field, is the maximum amount of money the user can spend per transaction

      (6) Custom name: the bank account’s associated custom name. This will be a text field (Ex: “Vacation”, “Car funds”, etc.)

      (7) IBAN: the unique identifier of the bank account. This will be a text field and the string representation of the IBAN will be displayed (Ex.: "RO45SEUP0000000000000000").

      (8) Blocked status: whether transactions are allowed to be performed from this bank account. This will be a yes/no type field: an icon of a white diagonal cross shape over a red background will denote a 'No' value (or any other diagonal cross shaped icon that may suggest the falsy nature of the status suffices); an icon of a white check mark shape over a green background will denote a 'Yes' value (or any other check mark shaped icon that may suggest the truthy nature of the status suffices).

   iii) The relevant pieces of information regarding the bank account (Currency, Custom name, Sold, IBAN, etc.) will be formatted as fields on two columns, with fields one above the other and the blocked status will be centered at the bottom. Above every field a description of the field will be present (Ex.: above the currency field where the currency of the bank account is displayed, the text "Currency" will appear).

iv) In the Info page (displaying various information about a bank account selected from the previous page - Bank account page), a button is displayed at the bottom center of the screen with the text "Back". When the user clicks the button, they are brought back to the Main Page where they can select another bank account for inspection.

d) **Bank account settings (update)**

i) The user will have a button next to bank account name which will take him to another screen/pop-up, where he will have forms for the daily limit, maximum amount of transaction, maximum number of transactions, if it is blocked or not, account custom name and can all be changed by the user in the original forms, where he will have to type the amounts or name to be updated or to check a box the block or unblock the account

ii) The settings apply to the current bank account: maximum number of transactions per day?, maximum/minimum sum of money per transaction, blocking/unblocking the bank account,

5) Loans (distopian credit system)

a) Main Loans Screen

i) There is a main loans screen where **all the active unpaid loans** are shown in a table. For each loan we can see the following: the loan ID, the currency, the amount taken from the bank, the amount to be given to the bank (with the tax), the date and time when the loan was taken, the date and time when it has to be paid, the number of months for the loan, the tax percentage. This table is ordered by creation date (the date and time when the loan was taken).

ii) Below this table are 2 buttons. One that contains the text “Take loan” which takes the user to the *Take a loan screen* when pressed, and another with the text “Pay loan” taking the user to the *Pay a loan screen* if clicked.

iii) A button below the other 2 has the text “Close” which closes the loans window.

b) Take a loan Screen

i) For this screen, the user has the following fields:

(1) The first field is the **bank account**, which is a drop-down list containing aggregated information of each of their bank accounts (IBAN, Currency, Amount of money). Depending on the bank account chosen, the currency of the given amount is inferred (e.g., if the bank account chosen is in euros, the currency for the given amount will be in euros). The user will receive their loan in this chosen bank account.

(2) The second field is a textbox in which the user can input their desired **amount** as a positive float number; its currency is the currency of the bank account chosen previously. This amount

cannot exceed 1.000.000 (one million). There is some text right below the amount field that clearly states: “Amount cannot exceed one million”.

(3) Another field will contain a drop-down list of all the possible **number of months** for the loan, which the user can select. The deadline for paying the loan is computed as the number of months selected added to the moment the loan is taken. The user should pay his loan by the end of the deadline. For this version of the app, there are no penalties for paying a loan after the deadline.

(4) Below the fields stated above, the **computed tax percentage will be shown**. This tax percentage is NOT monthly. The tax percentage gets applied once to get the amount the user needs to pay.

(5) **The amount the user needs to pay will be displayed below the tax percentage**, which will be the amount the user has to pay ONCE in full (they can’t pay portions of it over time) for the loan to be paid.

(6) Before the bank account, amount, and months are selected, we have the following values shown to the user: when there is no number of months chosen, the tax percentage is *N/A*; when no bank account or/and amount is chosen, the amount to be paid is *N/A*.

(7) There is a **button below that reads “Take loan”**. After being pressed, the data chosen/inputted will be validated. If there are any inconsistencies (amount is negative or exceeds one million) or any of the fields are empty, right below the button appears some text in red that reads **“Invalid data provided”**. If all the data chosen/inputted is valid, then the loan is saved, the user receives the amount of money in the bank account chosen, and the user is moved to the Main Loans Screen where he can see all of his loans, including the one he has just taken.

(8) Below all buttons will be a button with the text **“Back”** that, when pressed, moves the user to the Main Loans Screen. Data chosen/inputted will be cleared when going back.

c) Pay a loan Screen

i) For this screen, the user has the following fields:

(1) The first field is the **bank account**, which is a drop-down list containing aggregated information of each of their bank accounts (IBAN, Currency, Amount of money). The user will pay the loan with money from the chosen bank account.

(2) The second field is the **loan** the user wants to pay, which is a drop-down list containing the currency, the amount to be paid by the user, and the date and time when it needs to be paid. This list contains all the unpaid loans and the user can select only one of them. This is the loan that the user will try to pay.

(3) There is a **button below** the ones stated above**, that reads "Pay loan"**. After being pressed, the data chosen/inputted will be validated. If there are any empty fields, right below the button appears some text in red that reads **"Invalid data provided"**. If the data chosen is valid (no empty fields), but there is not enough money to pay the loan (if the currencies differ, the amount from the bank account gets virtually converted to the required amount before checking - only for checking, not actually converting it in the account), right below the button appears red text that reads **"Not enough funds"**. If all the data chosen is valid and there is enough money in the chosen bank account to pay it, then the loan is paid, the amount of money to be paid is taken from the chosen bank account, and the user is moved to the Main Loans Screen where he can see all of his loans, excluding the one he has just paid.

(4) Below all buttons will be a button with the text **"Back"** that, when pressed, moves the user to the Main Loans Screen. Data chosen/inputted will be cleared when going back.

d) The bank has infinite money. Meaning there is no limit to how much money people can loan.

e) When taking the loan, the loan is paid as a transaction from a bank's bank account to the user's account, with the transaction type "loan". When paying the loan, the loan is paid as a transaction from a user's account to the bank's bank account, with the transaction type "loan".

6) Homepage. Check sold (stonks team want to use, need a list with the user accounts to choose from which bank account/currency to take out to buy 'gold coins')

a) Check the sold button (in the selected bank account currency)
   - i) at the beginning the sold is hidden (we see the text 'Check Sold')
   - ii) after a click on that text, the text till turn in the actual bank account sold

b) New transaction button
   - i) will open a new window where the new transaction will happen

c) See transaction history button
   - i) this button will open the transaction history list for the selected bank account

d) New bank account button
   - i) will open a new windows there the new bank account creation will happen

e) Change bank account drop down menu (select which bank we want to see in the main window)
   - i) will open a drop down list with all the bank account the logged in user has

f) Transfer to another account button

        i) this will open a window in which the user will be able to transfer money from one of his bank account to another (also his bank account) and if the currency of the first bank account and the second one differ, the currency exchange will be made, else the transfer just occurs as it is(currencies of the source and destination accounts are the same)

    g) Go to the loan window
        i) will open a new window where the user will be able to take a loan from the bank. This window is showing the Main Loans Screen
    h) Go to user settings
        i) will open the user account update form
    i) Logout button
        i) will invalidate the user session and log the user out of the application
    j) Quit app button
        i) will terminate the main process of the app

7) Transactions history(type,…) → can be applied to the stonks part(we have a transaction on the currency exchange market)

a) Each transaction has: sender, receiver, time and date, category(would determined by the receiver of the transaction, ex. a grocery store), type, currency, amount
b) Verifiable: transactions between 2 mock accounts to test correctness
c) Test if the amount of the account permits the transaction (if i have 100 lei and want to send 101, I will not be able to, error pop-up)
d) Transaction means sending money from one place to another.(user to user, or user to institution (like a supermarket or online shop or whatever))
e) A transaction can happen between (only) 2 accounts!
f) Each transaction has a unique id
g) A transaction will go like this: A user initiates a new transaction from the account that is currently selected, and has to input the receiving account code. The currency that will be transferred is the currency that the currently selected account of the sender holds. The sum needs to be ≤ to the total sum in that currency account from that user. After the transaction is initiated, check if the receiving account holds the same currency as the sender, if not, make a conversion. (EUR -> RON, RON->EUR)
h) The page with the currently selected bank account will have a button labeled “Transaction history”. When pressed, this button will make a pop-up window appear. The window will contain a table-ish view of each transaction with the spent sum, and the name of the person/institution the sum has been directed to, as well as the date. For further details the user can click on one of the transactions and he will be redirected to a new page (in the pop-up window) with more details about the transactions, like the sending/receiving account number, category of the transaction, date and time of the transaction.
i) Allow the user to download their transactions in a csv file

j) Allow the user to see their transactions classified by their category either in a list view which has every single task that corresponds to the filtering, or in a pie chart that uses different colors for the categories.
k) Allow the user to add tags to each transaction (descriptive text that allows them to keep track of each transaction)
l) Allow the user to filter their transactions and only see the categories they want

8) Transactions

a) Each bank account has an IBAN which is unique (the bank account is represented, identified by this IBAN). This is how we distinct a bank account from another.
b) We consider that the bank has the IBAN: RO90BANK0000000000000005
c) All the transactions are made using the IBAN
d) A transaction is not possible (not valid) if:
    i) in the sender's bank account there are not enough money (the sender wants to transact more money than he has)
    ii) the sender's/ receiver's IBAN is not valid/ does not exist
e) We have different type of transactions
    i) from one bank account to another bank account
        (1) let X be the sender's bank account and Y be the receiver's bank account of the sum S of money
        (2) if the transaction is not valid, an error message will be shown and the transaction will not take place
        (3) if the transaction is valid, the sum S will be extracted from the bank account X and then will be added to the bank account Y.
    ii) from Bank to a bank account
        (1) when the user makes a loan from the bank
        (2) considering that the bank has infinite money, the money received from the loan are added to the given bank account
    iii) from a bank account to Bank
        (1) when the user pays a credit to the bank (a part of the loan that he made)
        (2) that sum of money is extracted from the bank account, if the user has enough money in the account, otherwise an error message will be shown and the transaction will not take place
f) If the transaction is made from a currency to another one, then the sender's sum will be converted into receiver's currency. The process is explained below.
g) GUI
    i) On the transactions page, there will be a list with different operations that can be made (each list element representing a link to a new page)
    ii) Send money to another account
        (1) after clicking on this link, we will be redirected to a new page

(2) there we will complete the following fields
    (a) receiver's IBAN
    (b) the sum of money that we want to send: a real number representing the sum of money in the account's currency
    (c) pay details: a string of maximum 50 characters
(3) we will have a button that checks that all the fields are completed and valid (the IBAN's and transaction's validity conditions are presented above)

iii) Pay a credit from the loan
    (1) after clicking on this link, we will be redirected to the specific page (as above)

iv) See currency exchange table
    (1) after clicking on this link, we will be redirected to a new page
    (2) the user will be able to see all the currency exchange rates

9) Currency exchange

a) Currency exchange happens if a transaction is made between two accounts with different currencies
b) We store in a table all the rates, from one currency to another. If we have a currency X and we want to convert it to the currency Y, we apply the formula: new_sum_of_money_in_currency_Y = sum_of_money_in_currency_X * rate_from_currency_X_to_currency_Y. The rate from currency X to currency Y might be different than the rate from currency Y to currency X.