Function Requirement1: RegisterUser

Purpose:

- registers a new user on the platform and automatically creates associated wallet and portfolio entities

Inputs:

- FirstName(String): User’s first name
- lastName(string): User’s last name
- email(String): User’s email address
- userName(String):User username
- Password(String): User password

Outputs:

- RegistrationSuccessful(Boolean): indicates if registration was successful
- message(String): confirmation message or error description
- UserID(Integer):Unique identifier of the newly create user if account creation is successful

Process:

1. **Validate input data**
    - Check that all required fields are provided and correctly formatted
    - Check that email and username are unique
2. **Create User entity:**
    - Hash the password for security
    - Insert a new record into the user entity with the provided details
3. **Create User Entity:**
    - Automatically create a wallet associated with the userID
    - Initialize balance to $0
4. **Create portfolio Entity:**
    - Automatically create a Portfolio associated with the new userID
    - Initialize TotalShares to 0
5. **Return Success Response:**
    - If all steps are successful, return registrationSuccess as true with a confirmation message and the UserID
    - If any steps fails, return registrationSuccess as false with the correct error message

Dependencies:

- User Entity: for string user details
- Wallet Entity: for creating user wallets
- Portfolio Entity: For creating user portfolios.

Function Requirement2: DepositFunds

Purpose:

- Allows a user to deposit funds into their wallet

Inputs:

- UserId (Integer): Unique identifier of the user
- Amount (float) : amount to deposit
- PaymentMethod (String): Method of payment

Outputs:

- depositSuccess (Boolean): Indicates if the deposit was successful.
- message (String): Confirmation message or error description.
- newBalance (Decimal): Updated balance of the user's Wallet (if successful).

Process:

1. **Validate Input Data:**
    - Ensure the amount is positive and meets any minimum deposit requirements.
    - Verify that paymentMethod is supported.
2. **Process Payments:**
    - Interact with the payment gateway to process the deposit.
    - Confirm successful payment.
3. **Update Wallet Balance:**
    - Increment the user's Wallet Balance by the amount
4. **Record Transaction:**
    - Create a Transaction of type 'Deposit' with relevant details.
    - Set transactionStatus to Completed.
5. Return Success Response:
    - Return depositSuccess as true with a confirmation message and the newBalance.
    - If any step fails, return depositSuccess as false with an appropriate error message.

Dependencies:

- Wallet Entity: For updating the balance.
- Transaction Entity: For recording the deposit.
- Payment Gateway Integration: For processing payments.

Function Requirement3: MakeInvestment

Purpose:

- Facilitates the purchase of shares in a Property by a user, updating relevant entities accordingly.

Inputs:

- userId (Integer): Unique identifier of the user.
- propertyId (Integer): Unique identifier of the property.
- numberOfShares (Integer): Number of shares the user wants to purchase.

Outputs:

- investmentSuccess (Boolean): Indicates if the investment was successful.
- message (String): Confirmation message or error description.
- investmentId (Integer): Unique identifier of the new Investment (if successful).

Process:

1. **Validate Input Data:**
    - Ensure userId and propertyId exist.
    - Check that numberOfShares is positive.
    - Confirm the Property has enough availableShares
2. **Calculate Total Investment Amount**
    - total Investment = numberOfShares × SharePrice of the Property.
3. **Check Wallet Balance:**
    - Verify that the user's Wallet Balance is ≥ Total Investment.
4. **Deduct Funds from Wallet**
    - Subtract Total Investment from the user's Wallet Balance.
5. **Record Transaction:**
    - Create a Transaction of type 'Investment Purchase' with relevant details.
    - Set transactionStatus to 'Completed'.
6. **Update Property Shares:**
    - Decrease the Property's availableShares by numberOfShares
7. **Update Portfolio:**
    - Increment totalInvestmentAmount in the user's Portfolio by Total Investment.
8. **Return Success Response:**
    - Return investmentSuccess as true with a confirmation message and investmentId.
    - If any step fails, return investmentSuccess as false with an appropriate error message.

Dependencies:

- User Entity: For user validation and Wallet access.
- Property Entity: For share availability.
- Wallet Entity: For balance deduction.
- Investment Entity: For creating investment records.
- Transaction Entity: For recording the purchase.

Function Requirement4: WithdrawFunds

Purpose:

- Allows a user to withdraw funds from their Wallet

Inputs:

- userId (Integer): Unique identifier of the user.
- amount (Float): Amount to withdraw.
- paymentMethod (String): Method of withdrawal

Outputs:

- withdrawalSuccess (Boolean): Indicates if the withdrawal was successful.
- message (String): Confirmation message or error description.
- newBalance (Decimal): Updated balance of the user's Wallet (if successful).

Process:

1. **Validate Input Data:**
   - Ensure the amount is positive and meets any minimum withdrawal requirements.
   - Verify that paymentMethod is supported.
2. **Check Wallet Balance:**
   - Confirm that the user's Wallet Balance is ≥ amount withdrawing.
3. **Process Withdrawal:**
   - Initiate the withdrawal through the specified paymentMethod.
4. **Update Wallet Balance:**
   - Subtract amount from the user's Wallet Balance.
5. **Record Transaction:**
   - Create a Transaction of type 'Withdrawal' with relevant details.
   - Set transactionStatus to 'Completed'.
6. **Return Success Response**
   - Return withdrawalSuccess as true with a confirmation message and newBalance.
   - If any step fails, return withdrawalSuccess as false with an appropriate error message.

Dependencies:

- Wallet Entity: For balance verification and updates.
- Transaction Entity: For recording the withdrawal.

Function Requirement5: SellShares

Purpose:

- Facilitates the sale of shares from a user's Investment, updating relevant entities accordingly.

Inputs:

- userId (Integer): Unique identifier of the user.
- investmentId (Integer): Unique identifier of the investment from which shares are being sold.
- numberOfSharesToSell (Integer): Number of shares the user wants to sell.

Outputs:

- saleSuccess (Boolean): Indicates if the sale was successful.
- message (String): Confirmation message or error description.
- transactionId (Integer): Unique identifier of the sale Transaction (if successful).

Process:

1. **Validate Input Data:**
    - Ensure userId and investmentId exist.
    - Confirm that numberOfSharesToSell is positive.
    - Verify that the user owns the specified Investment.
2. **Check Share Availability:**
    - Ensure the user has ≥ numberOfSharesToSell in their Investment.
3. **Calculate Sale Amount:**
    - Total Sale Amount = numberOfSharesToSell × SharePrice of the Property.
4. **Create transaction**
    - Insert a Transaction of type 'Investment Sale'
5. **Update Investment:**
    - Decrease numberOfSharesPurchased by numberOfSharesToSell
    - If numberOfSharesPurchased becomes 0: Set investmentStatus to 'Sold'.
    - Update investmentAmount accordingly.
    - Update lastUpdatedDateTime.
6. **Update Wallet Balance:**
    - Add Total Sale Amount to the user's Wallet Balance.
7. **Update Portfolio:**
    - Decrease totalInvestmentAmount by the original investment amount of the sold shares.
8. **Return Success Response:**
    - Return saleSuccess as true with a confirmation message and transactionId.
    - If any step fails, return saleSuccess as false with an appropriate error message.

Dependencies:

- User Entity: For user validation and Wallet access.
- Investment Entity: For share ownership verification.
- Property Entity: For share price and availability.
- Wallet Entity: For updating balance.
- Transaction Entity: For recording the sale.