Task Description: Simple Banking System with Modules

You are tasked with creating a Python program that simulates a simple banking system, split across two or more Python files using modules. The user will be able to deposit, withdraw, and check their balance, with all functions for transactions handled in a separate module. The main script will handle user input and interaction. You are required to use `try-except` to handle errors, such as insufficient funds or invalid inputs.

Deliverables

1. **Python Script 1 (`main.py`):**
    - Handles user interaction (deposit, withdraw, check balance, exit).
    - Imports and calls functions from the `bank.py` module.
    - Handles continuous program interaction with a loop and error handling.
2. **Python Script 2 (`bank.py`):**
    - Defines the core banking functions:
        - `deposit(amount)`
        - `withdraw(amount)`
        - `check_balance()`
    - Contains the logic for managing and updating the user's balance.
3. **Optional Script (`utils.py` - Optional):**
    - Could include utility functions like input validation, casting, or other helper functions to keep code modular.

Requirements List

1. **Modularization:**
    - Functions for banking operations (`deposit`, `withdraw`, `check_balance`) must reside in a separate module (`bank.py`).
    - The main script (`main.py`) should import these functions and handle the user interaction.
2. **Error Handling:**
    - Use `try-except` blocks in the main script (`main.py`) to handle invalid inputs (non-numeric values) and insufficient funds during withdrawal.
3. **Functions:**
    - `deposit(amount)` – Adds money to the user's balance (in `bank.py`).
    - `withdraw(amount)` – Subtracts money from the user's balance, checks for sufficient funds (in `bank.py`).
    - `check_balance()` – Displays the current balance (in `bank.py`).

4. **Main Function (main.py):**
    - Organizes the user input loop and calls the banking functions based on the user's action.

File Structure

bash
CopyEdit

```
banking_system/
│
├── main.py          # Handles user interaction
├── bank.py          # Contains core banking functions
└── utils.py         # Optional: Contains helper functions
```

Example User Test

User deposit using `main.py`:

```
Enter 'd' to deposit, 'w' to withdraw, 'b' to check balance, or 'exit'
to quit: d
Enter amount to deposit: 500
Deposited: 500.0
```

User withdraw using `main.py`:

```
Enter 'd' to deposit, 'w' to withdraw, 'b' to check balance, or 'exit'
to quit: w
Enter amount to withdraw: 200
Withdrawn: 200.0
```

User checks balance using `main.py`:

```
Enter 'd' to deposit, 'w' to withdraw, 'b' to check balance, or 'exit'
to quit: b
Current balance: 300.0
```

User enters invalid input for withdrawal:

```
Enter 'd' to deposit, 'w' to withdraw, 'b' to check balance, or 'exit'
to quit: w
Enter amount to withdraw: abc
Invalid input! Please enter a valid number.
```

User tries to withdraw more than balance:

```
Enter 'd' to deposit, 'w' to withdraw, 'b' to check balance, or 'exit'
to quit: w
Enter amount to withdraw: 500
Insufficient funds!
```

User exits program:

```
Enter 'd' to deposit, 'w' to withdraw, 'b' to check balance, or 'exit'
to quit: exit

Exiting... Goodbye!
```