ATM Use Case

ID: X
Name: ATM Withdrawal
Description: A user can withdraw cash from an ATM.
System: Automatic Teller Machine (ATM)
Preconditions:

User must have a checkings or savings account with the bank.
User must have a valid debit card.

Actors:

User (accessing the ATM)
System (the ATM)

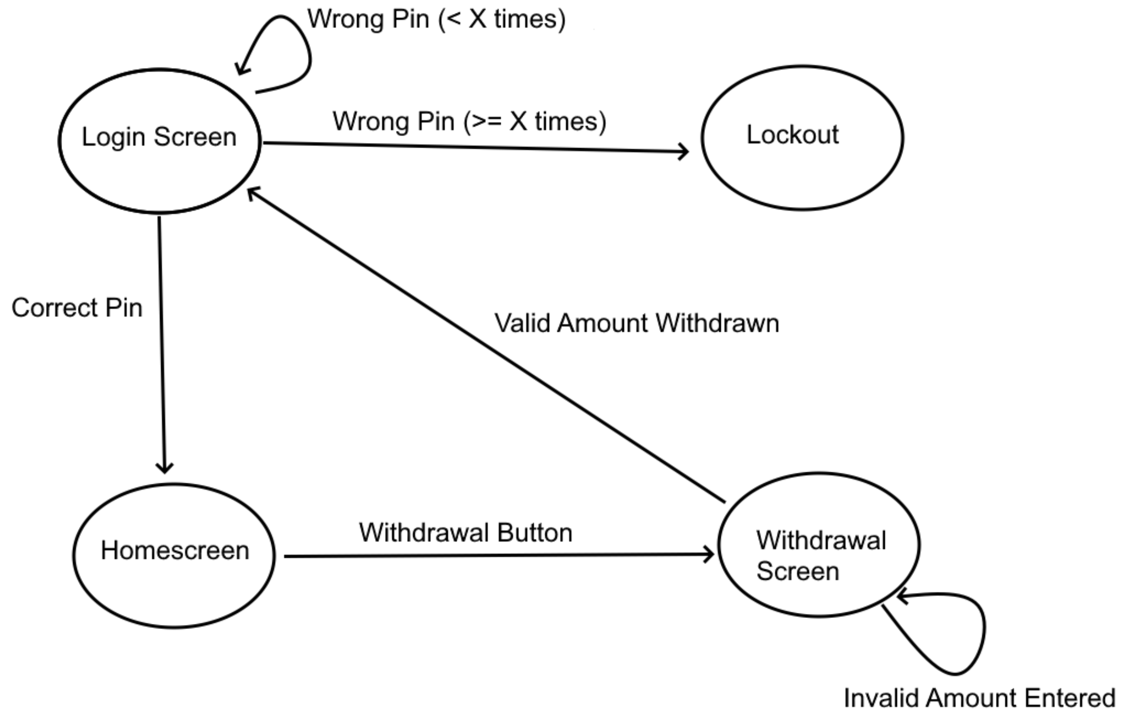
Scenario:

1. User inserts card into ATM.
2. User enters PIN.
3. User presses the Withdraw button.
4. User enters Withdrawal amount.
5. User hits the confirm button.
6. Card is returned to User.

2a. User enters incorrect pin, system shows error.
2b. User entered incorrect pin X times, system locks further attempts.

4a. Withdrawal amount exceeds available balance, system rejects.
4b. Withdrawal amount is an invalid number (ex. 0 or negative), system rejects.

State Transition Diagram



Decision Tables

Opted to split this into 2 tasks (Login, Withdrawal) since any failure in the Login will prevent the Withdrawal action from ever being done. Extra table for Account Locking since the counter for that is separate from the simple “Account Lock” state that Login sees.

Login

PIN	Account Locked	Result
Correct	False	Success
Correct	True	Fail
Incorrect	False	Fail
Incorrect	True	Fail

Account Lock

Attempts	Result
<= X	Not Locked
> X	Locked

Account Lock Partitioning:

Attempts = 0
Attempts = X - 1
Attempts = X

Withdrawal

Withdrawal Amount	Valid Amount (Amt > 0)	Result
Amt ≤ Balance	True	Pass
Amt > Balance	True	Fail
Amt ≤ Balance	False	Fail
Amt > Balance	False	Fail

Withdrawal Partitioning:

Amt = 0
0 < Amt < Balance
Amt = Balance
Balance < Amt < 0 is impossible since Balance cannot be < 0, discard this.

Food Delivery Use Case

ID: X
Name: Order Delivery
Description: A user can place an order of food for delivery to their address.
System: Food Delivery App
Preconditions:
User has an account
Actors:
User (accessing the food delivery app)
System (the food delivery app)
Techniques:

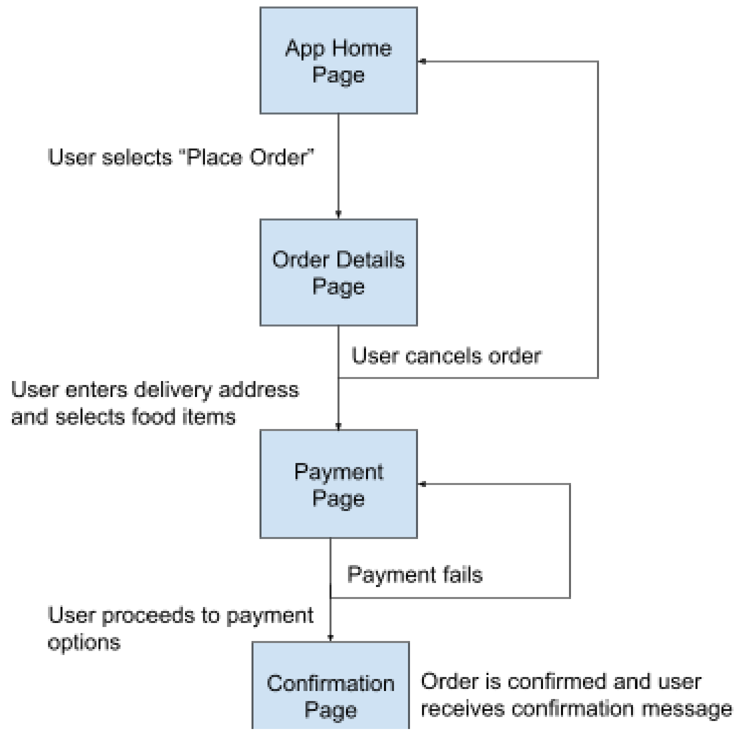
- **Exploratory Testing:**
    - Exploring the food delivery app's features and functionalities using current food delivery apps on the market (Uber Eats, Doordash, etc…). We will navigate through these apps' interface and try different scenarios, like placing orders, modifying details, and checking for errors.
    - Examples:
        - Place an order for food from various restaurants available on the app.
        - Modify order details (e.g., add/remove items, change delivery address) during the ordering process

- **Decision Tables:**
    -

Test case	User has an account	User places an order	Expected outcome
1	Yes	Yes	Order is successfully placed
2	No	Yes	Error message prompts user to log in or create an account
3	Yes	No	No action taken, user remains on the app homepage
4	No	No	Error message prompts user to log in or create an account.

    -
- **Boundary Value Analysis:**
    - Test case 1: User inputs delivery address with minimum characters allowed.
    - Test case 2: User inputs delivery address with maximum characters allowed.
    - Test case 3: User places an order with the minimum allowed quantity of items.
    - Test case 4: User places an order with the maximum allowed quantity of items.
    - Test case 5: User enters a delivery time just before the minimum allowed time.
    - Test case 6: User enters a delivery time just after the maximum allowed time.

- **State Transition Diagrams:**


App Home Page
User selects "Place Order"
Order Details Page
User cancels order
User enters delivery address and selects food items
Payment Page
Payment fails
User proceeds to payment options
Confirmation Page
Order is confirmed and user receives confirmation message

- **Equivalence Partitioning**
    - Dividing input conditions into equivalence classes, like valid and invalid delivery addresses or valid and invalid payment methods. By testing representative values from each partition, we can ensure that the app handles different types of inputs consistently and accurately
        - Test case 1: User inputs a valid delivery address.
        - Test case 2: User inputs an invalid delivery address.
        - Test case 3: User selects food items from available menu.
        - Test case 4: User attempts to select food items unavailable.
        - Test case 5: User selects a valid payment method.
        - Test case 6: User attempts to use an unsupported payment method.

Instructions

All team members will work together to plan and design their test cases.
Once the team has finished their Decision Table, BVA, Equivalence Paritioning, etc.
Work can then be separated out to have each member work on writing a few test cases.

The techniques that associates should be employing for this activity are:

Exploratory Testing (using similar existing web apps - for example if a team has an e-commerce-themed use case, they can do exploratory testing with websites like Amazon, Barnes & Noble, etc.)
Decision Tables
Boundary Value Analysis
State Transition Diagrams
Equivalence Partitioning
The test cases will not be executed in this activity. The focus is on utilizing and familiarizing oneself with different testing techniques.