Virtual Poker Chips

Nguyen Nguyen

Junyang Lu

Ayushman Satpathy

Gabriella Wang

Christian Michael Dela Cruz

We will be determining the function points for use case 4, which in order to remind everybody, is about customizing the game and user's experience.

For users:

- Players select the "customize" button on the client
- Players make the desired customization for their game
  - This includes card colors, card patterns, board color
- Players click "Accept changes" on the website

For hosts:

- Hosts select the "customize" game button on the client
- Hosts modify the rules of the game, this includes:
  - Maximum, minimum player count
  - Maximum, minimum player bet
  - Number of rounds played
- Hosts click "Accept changes" on the website

After that, both methods work the same for either cases:

- The request is sent to FastAPI
- FastAPI updates the result in local database
- Next gets result from FastAPI
- Next displays results (showing new customization)

We will be using these criteria's to evaluate our function points:

- **External Inputs (EIs) -** Data coming in from the frontend or API requests
- **External Outputs (EOs) -** Data being sent out, such as API responses from backend
- **External Inquiries (EQs) -** Request and responses without internal data modification, inlcuding the intermediate validation steps
- **Internal Logical Files (ILFs) -** Data getting maintained internally, such as player customization, host room settings, room player count, etc.

- **External Interface Files (EIFs)** - Data used from a third party

CreateValidationRequest():

- **Summary:** send an API request to FastAPI when "Accept changes" button is made on the frontend
- External Inputs:
  - User input to validate the customization rules they requested
  - This is quick to implement
  - 3
- External Output:
  - Returns the customization rules applied or an error
  - This should be on similar level to EI
  - 3
- External Inquiries:
  - Session validation, checking if the request is permitted or not
  - This is trickier to work on, may require to setup authentication on the app
  - 6
- Total : 3 + 3 + 6 = 12

SettingsCase():

- **Summary**: Filters the incoming request to "User" or "Room" and call the appropriate methods for them
- External Inputs:
  - Input is user input requested in **CreateValidationRequest()**
  - This is quick to make
  - 3
- External Inquiries:
  - Calls **validateUser()** or **validateRoom()** depending on the request
  - This is simple to make
  - 3
- Total: 3 + 3 = 6

ValidateUser()

  - **Summary:** check if the Users customization request is valid and contains appropriate inputs or not
  - External Inputs:
    - Inputs includes hand colors, board colors and hand presets / patterns the user requested
    - 3
  - External Inquiries:
    - Input validation using basic arithmetic checking, or datatype check

        - Very quick to do
        - 3
    - Internal Logical Files:
        - Database is sent a request to update the User's customization rules, **UpdateUser()**
        - The sending part is not complicated
        - 7
    - Total: 3 + 3 + 7 = 13

ValidateRoom()

    - **Summary:** check if the Room customization request is valid and contains appropriate inputs or not
    - External Inputs:
        - Inputs includes mininum and maximum player count, bets and game's total rounds
        - 3
    - External Inquiries:
        - Input validation using basic arithmetic checking, or datatype check
        - Very quick to do
        - 3
    - Internal Logical Files:
        - Database is sent a request to update the User's customization rules, **UpdateRoom()**
        - The sending part is not complicated
        - 7
    - Total: 3 + 3 + 7 = 13

UpdateUser()

- **Summary:** Update user requested rules to the database
- External Inputs:
    - Input is User's customization rules, including hand and board colors and presets
    - 3
- External Output:
    - Response sent from database methods indicating a successful operation or not to **BroadcastUserSettings()**
    - Ties in with ILF, requires some more complex functionality
    - 5
- Internal Logical Files:
    - User database update the user's customization rules with the requested one, or raise an error in case of exception

- Handles more complicated exception, more testing required for databse updating and request responses
- 15

- Total : 3 + 5 + 15 + 23

UpdateRoom()

- **Summary:** Update user requested rules to the database
- External Inputs:
  - Input is Room's customization rules, similar to **ValidateRoom()**
  - 3
- External Output:
  - Response sent from database methods indicating a successful operation or not to **BroadcastRoomSettings()**
  - Ties in with ILF, requires some more complex functionality
  - 5
- Internal Logical Files:
  - Room database update the room's customization rules with the requested one, or raise an error in case of exception
  - Handles more complicated exception, more testing required for databse updating and request responses
  - 15
- Total : 3 + 5 + 15 + 23

BroadcastUserSettings()

- **Summary:** Apply user settings only to the user requesting it
- External inputs:
  - Input is the response we receive from **UpdateUser()**
  - Not too complicated
  - 4
- External Output:
  - Broadcasting update to a singular player
  - Uses WebSocket with less emphasis on synchronization between players
  - 5
- Total : 4 + 5 = 9

BroadcastRoomSettings()

- **Summary:** Apply user settings only to the user requesting it
- External inputs:
    - Input is the response we receive from **UpdateRoom()**
    - Not too complicated
    - 4
- External Output:
    - Broadcasting update to a singular player
    - Heavy emphasis on Websocket synchronization to players, rules applying to everybody
    - 7
- Internal Logical Files:
    - Updating room rules for the room every time host requests a change
    - Complicating Websocket methodologies
    - 15
- Total : 4 + 7 + 15 = 26