WALK THROUGH OF MONOPOLY

Legend:

- .ftl file
- Handler
- game data from frontend to backend
- game data from backend to frontend

1. User starts program, Main Menu appears (FrontHandler)
    a. User clicks on New Game -> See step 2
    b. User clicks on Customize Board -> See step 3
    c. User clicks on Load Game -> See step 4
    d. User clicks on Exit -> See step 5
2. User is at Game Options page (OptionsPageHandler)
    a. User specifies the number of human and AI players
        i. Total number of players must be between 2 and 6/7/8
    b. User specifies the Game Mode (Standard or Fast Play)
    c. User specifies the Board Theme
        i. reads from folder of saved Board Themes
        ii. default Board Theme already existent (not as a file, but as an option)
    d. User clicks on Customize Board -> See step 3
    e. User clicks on Done -> See step 6
3. User is at Customize Board page (CustomizePageHandler)
    a. Design for this page hasn't been finalized yet. Some options:
        i. Have the standard board on the screen, user can click on each square and change its name (and color if it's a property)
            1. color change would have to be reflected for every property in the monopoly
        ii. Have each of the 8 monopolies laid out (one square per Monopoly). User can click on one of the 8 squares, then they are brought to a new page/popup with each of the properties within that monopoly. They can then customize the properties from there
    b. Whatever the design, it may be useful to have the board represented by two JS Objects (one for names, one for colors) formatted as follows:
        i. Names: {sq1: name1, sq2: name2, …}
        ii. in reality: {sq0: "GO", sq1: "Mediterranean Avenue", …}
        iii. Colors: {monopoly1: color1, monopoly2: color2, …}
        iv. in reality: {purple: [r,g,b], lightBlue: [r,g,b], …}
    c. To get the objects initially, it may be useful to get the default names and colors from the backend since all of the information for that is already there, so we don't have to hardcode it on the frontend.
    d. User clicks Done, and they are brought back to Game Options (or Main Menu) -> See step 2 (or 1)

        i. When Done is pressed, the customized theme should be saved into the folder containing the Board Themes (SaveThemeHandler)
            1. This can be handled by sending the JSON object to the backend, parsing it, creating a BoardTheme, and saving that
        ii. If they are brought back to game options, the User's newly created theme should be selected as the Board Theme
4. Will complete later since it isn't strictly a requirement
5. Application exits
    a. Suggestion: Close the window but leave the program running
6. User is brought to the Game screen (NewGameHandler)
    a. Specified game options are given to the backend. The backend uses the settings to create the game (and board, players, etc.). The backend responds to the post request with a GameState to represent the start of the game.
    b. Do we want a popup/textbox displaying whose turn it is?
    c. User clicks on Roll -> See step 7
    d. User clicks on Trade -> See step 8
    e. User clicks on Manage Properties -> See step 9
    f. User clicks on Pause -> See step 10
7. Tell the backend to roll (RollHandler)
    a. Backend will already know who the current player is
    b. Backend responds with the number that appeared on the dice, and the new GameState
        i. Currently, the backend responds with the dice, and whether or not the player went to jail. In this case, I'm assuming whoever wrote this figured the frontend would move the player the correct number of spaces on its own. However, this means that both the frontend and backend would be doing this action, when really only the backend needs to do so. The frontend can use the new GameState to redisplay the board?
    c. The backend will need to send more info in the response (e.g. whether or not the user went to jail). Other things the frontend will need to know at this point:
        i. Did the User land on an unowned Ownable?
        ii. Did the User land on an owned Ownable?
        iii. Did the User land on Chance or Community Chest?
        iv. Did the User land on a TaxSquare?
    d. Based on what happened to the user after the roll the frontend will need to do different things
        i. If the User landed on basically anything except an unowned Ownable (e.g. an owned Ownable, a Chance/Community Chest Card, a TaxSquare, Free Parking, GO, Go To Jail) or if the user rolled 3 straight doubles, either:
            1. nothing should happen, except any changes (e.g. differences in balances, moves on the board) made should be reflected by

updating the GameState and redrawing the Board/refreshing Player info
2. A popup detailing what happened (detailed message could be given from backend depending on what happened) should appear, and the user can dismiss it by pressing OK or something
3. Edge Cases to worry about later:
    a. Effect from roll causes player to lose more money than they currently have. The backend should communicate this and the User should be forced to Mortgage any properties if they have them, or they should be notified that they are bankrupt and out of the game.

ii. If the User lands on an unowned Ownable, they should be prompted to make a decision as to whether or not they want to buy the property
1. For an AI, the backend would handle this without re-prompting (this walkthrough is for human users)
2. For a user, the frontend should have a popup prompting the User to make a decision, and the User cannot make the popup disappear unless they select “yes” or “no”
3. The User will decide whether or not to buy the Ownable -> see step 11

8.
9.
10.
11. The User decides whether or not to buy the property (BuyOwnableHandler)
    a. If the User clicks “no”, dismiss the popup, but no communicate with the backend is necessary
    b. If the User clicks “yes,” send the User’s decision to the backend.
    c. The backend should respond with the new GameState (which will include the updated player information)
        i. If the User doesn’t have enough money to purchase this property, the backend should communicate this and the user should be allowed to mortgage any Ownables to give them enough money to buy the Ownable they landed on