Manual.doc:

Note: unsure of .doc format so I am using pdf

User instructions:

Necessary packages/dependencies:

- Python (I am on version 3.13.9)
- Pip
- Flask

Installation guide:

- Install project files
- Install Python globally
- Install Pip globally
- Create a virtual environment in the project directory
  - Run “python -m venv .venv”
- Activate virtual environment
  - Run “source .venv/bin/activate” (this is on linux, may be a different directory on windows/mac)
- in your active venv install Flask
  - pip install Flask

Execution guide:

Within the Stage_3 directory Run “python flask_game_engine.py” and navigate to the url specified in your terminal emulator

API instructions:

Components.py contains the following useful functions:

- legal_move(board, position, colour)
  - Returns if a move is legal

- check_outflanks(board, position, colour)
    - returns true, valid_directions. This represents if a move outflanks any stones and what directions it outflanks in
- initialise_board(size=8)
    - Returns a starting board state for a given size
- print_board(board)
    - Outputs a readable board state to a CLI

flask_game_engine.py contains the following useful functions:

- swap_player(player)
    - returns opposing player to argument
- change_outflanked_stones(board, move, player, direction)
    - returns a board with stones in a given direction which would be outflanked swapped in colour
- any_legal_moves(board, player)
    - returns true if any legal moves exist for a given player on a given board
- check_score(board)
    - returns winner, num_dark_stones, num_light_stones

Testing:

Unit tests:

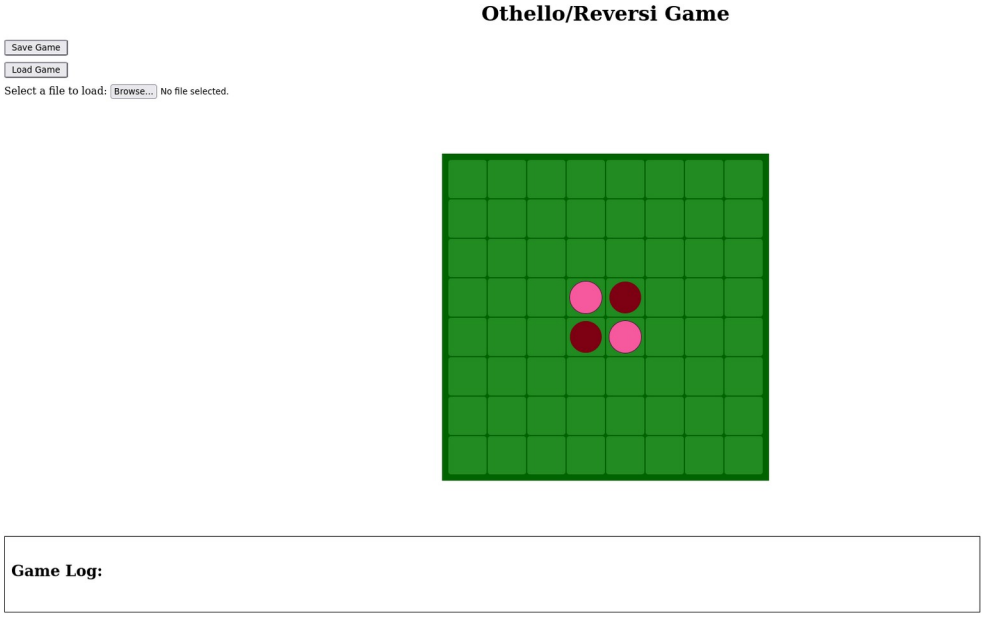
Unit tests can be found in tests.py, all functions with valid return values are tested under multiple varying criteria.

Integration tests:

For these tests I am going to lay out the feature being tested and present a series of before and after images.

Test	Expected result
Play legal move	Piece will be placed, outflanked stones will be swapped, bot will respond with a move, message box will be updated throughout

Before:

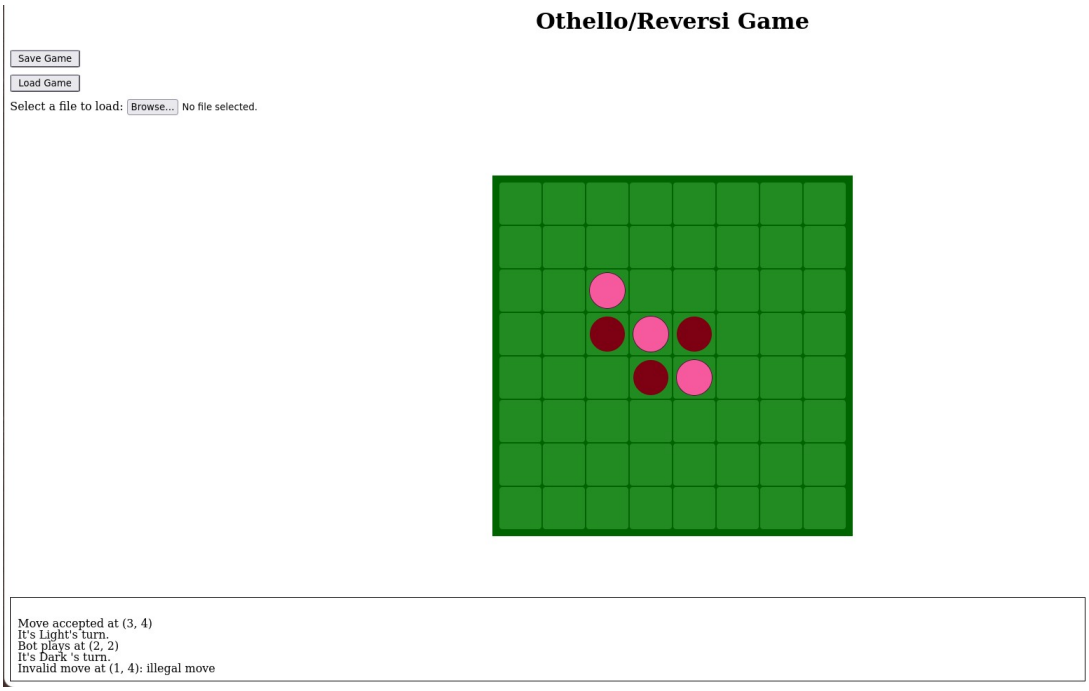


after:

Test	Expected result
Play illegal move	No pieces will change, message box will be updated

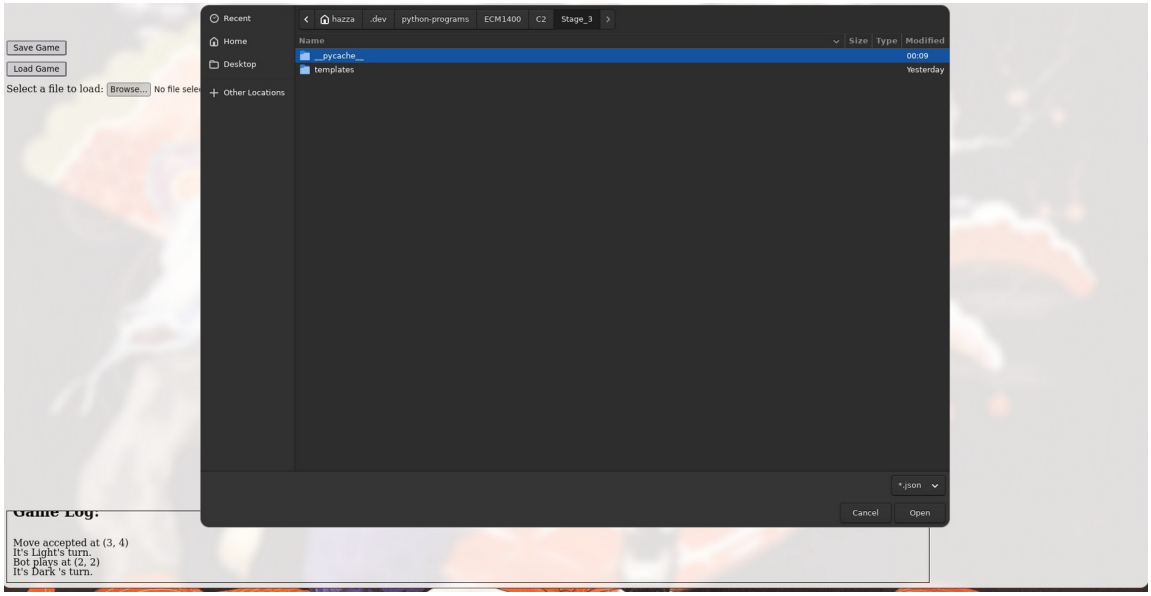
Before:

After:

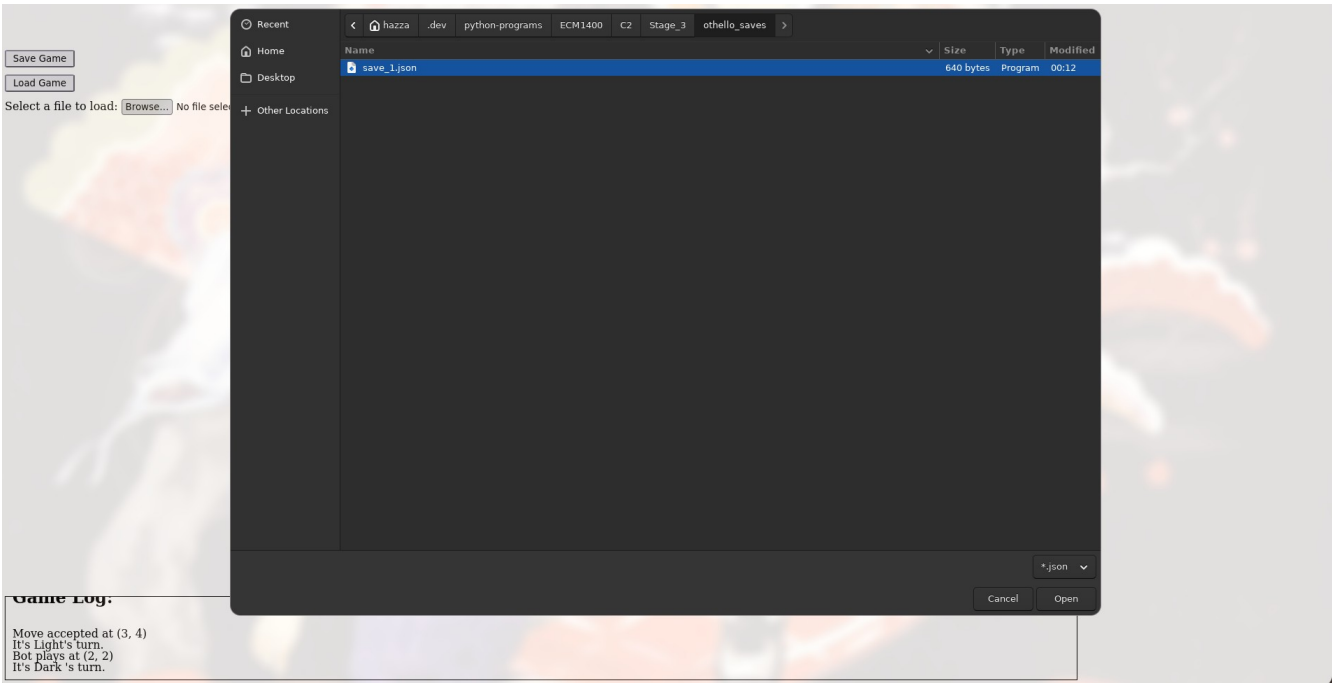


Test	Expected result
Save a game	Save directory will be created and save will be stored as a file

Before:

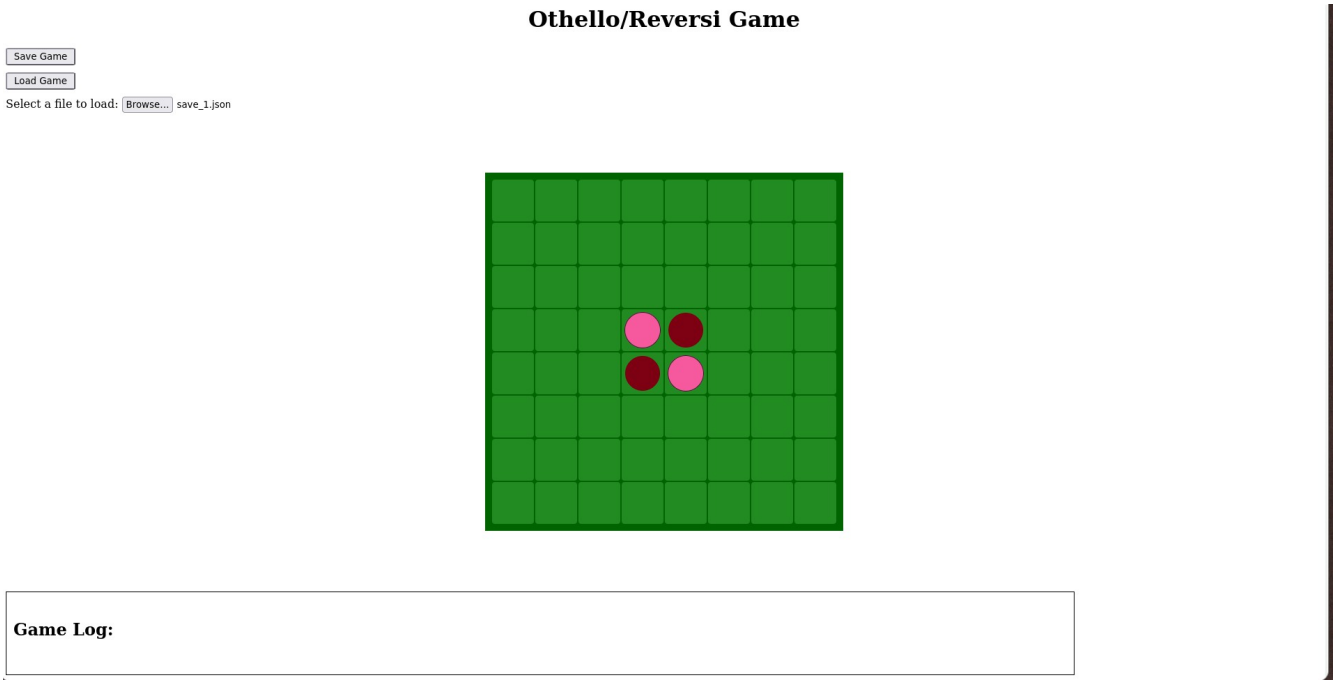


After:



Test	Expected result
load a game	Selected save file will be loaded, message box will be updated

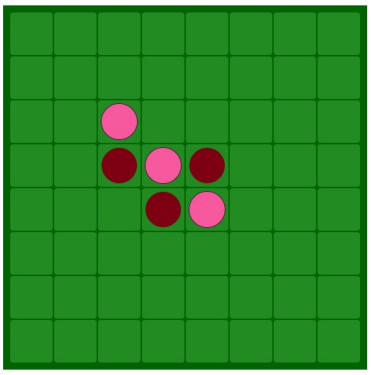
Before:



After:

Othello/Reversi Game

Save Game
Load Game
Select a file to load: Browse... save_1.json



Game Log:

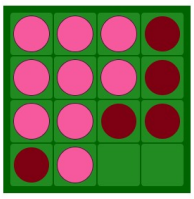
Save Loaded: move counter: 58
it's Dark 's turn

Test	Expected result
End a game	Game result will be outputted as an alert, board will reset itself

Before:

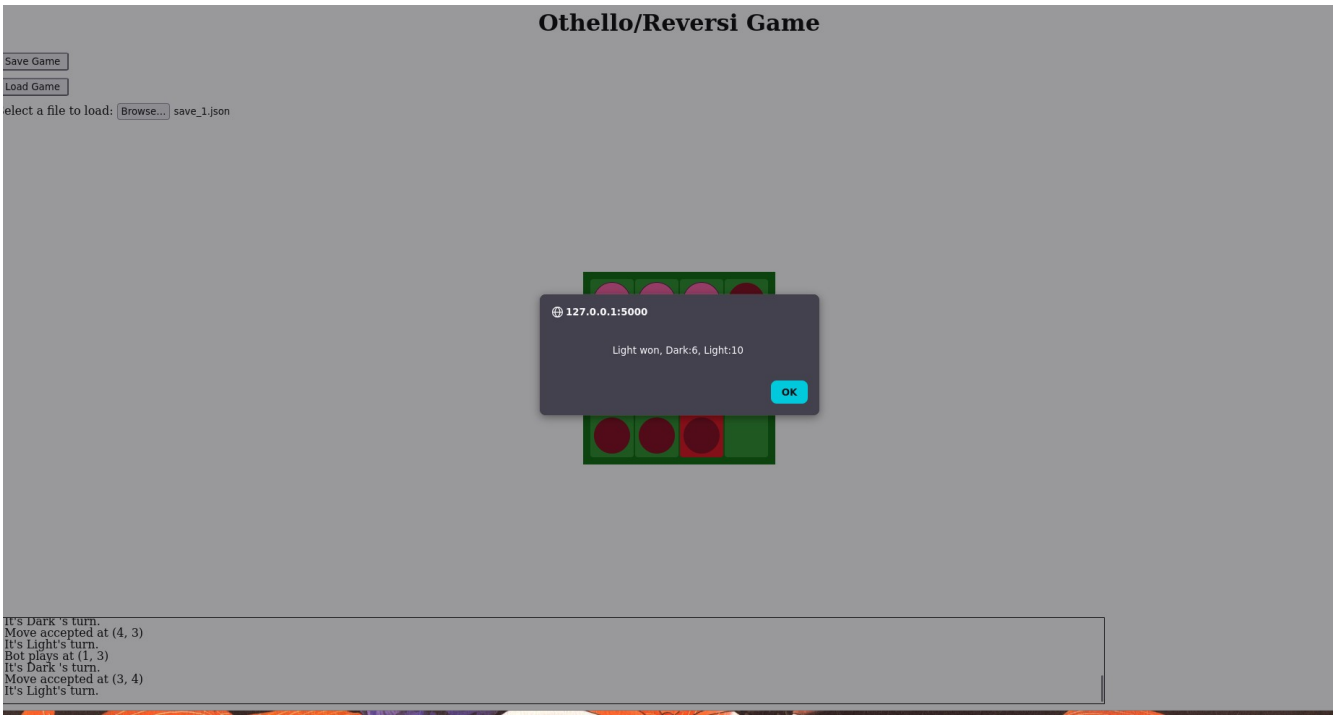
Othello/Reversi Game

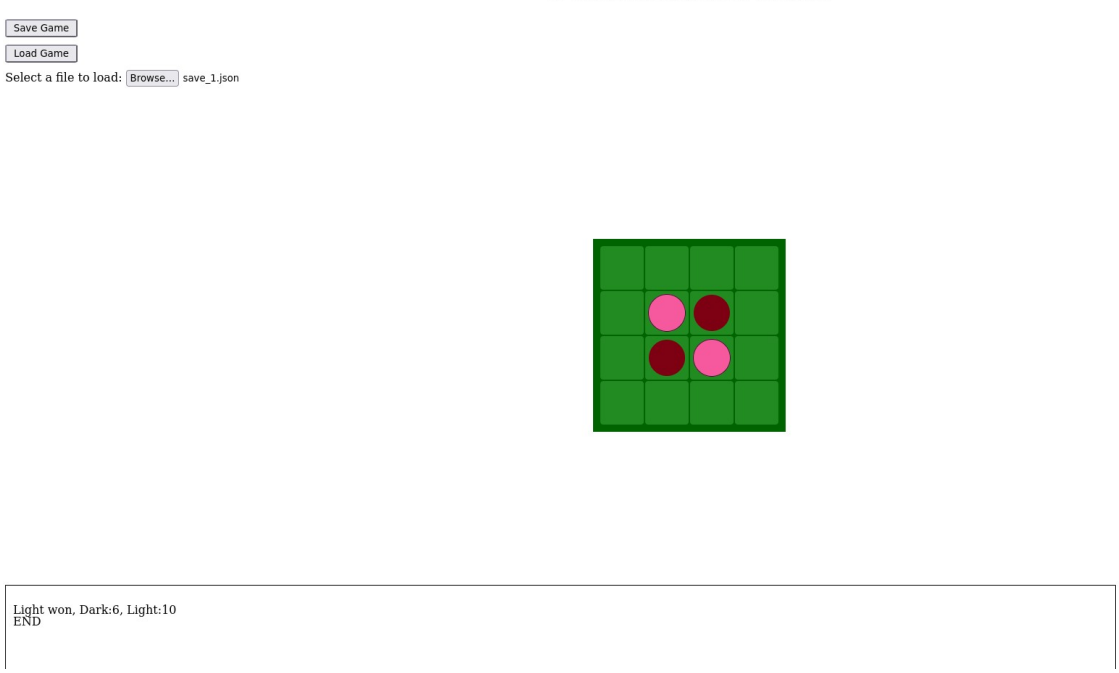
Save Game
Load Game
Select a file to load: Browse... save_1.json



it's Light's turn.
Bot plays at (3, 1)
It's Dark 's turn.
Move accepted at (4, 3)
It's Light's turn.
Bot plays at (1, 3)
It's Dark 's turn.

After:


Othello/Reversi Game
Save Game
Load Game
elect a file to load:
Browse...
save_1.json
127.0.0.1:5000
Light won, Dark:6, Light:10
OK
It's Dark 's turn.
Move accepted at (4, 3)
It's Light's turn.
Bot plays at (1, 3)
It's Dark 's turn.
Move accepted at (3, 4)
It's Light's turn.


Othello/Reversi Game
Save Game
Load Game
Select a file to load:
Browse...
save_1.json
Light won, Dark:6, Light:10
END