This manual contains information about how to install the project and its dependencies, how to run the CLI and Web version of the game provided and how to use the code for potential other implementations of the game.

System Requirements:

- Python (version 3.10 or above)
- Flask (external Python library can be installed via the pip command 'pip install flask')
- The project source code

Installation:

1. Download the project

To acquire the project on your device, the project source code and files can be downloaded in various ways:

- Download as a ZIP file from the main repository page (https://github.com/SpoonFish/Reversi-Project)
- Clone using the git command 'git clone https://github.com/SpoonFish/Reversi-Project'

2. Install dependencies

- If you do not already have Flask installed, run 'pip install flask' on the python installation you are using for the project

3. Project folder structure

Ensure the project has the same structure as defined below so that the game runs correctly

```
Reveri Project/
    Stage1/
      components.py
      game_engine.py
    Stage2/
      components.py
      flask_game_engine.py
      templates/
          index.html
    Stage3/
      components.py
      flask_game_engine.py
      templates/
          index.html
```

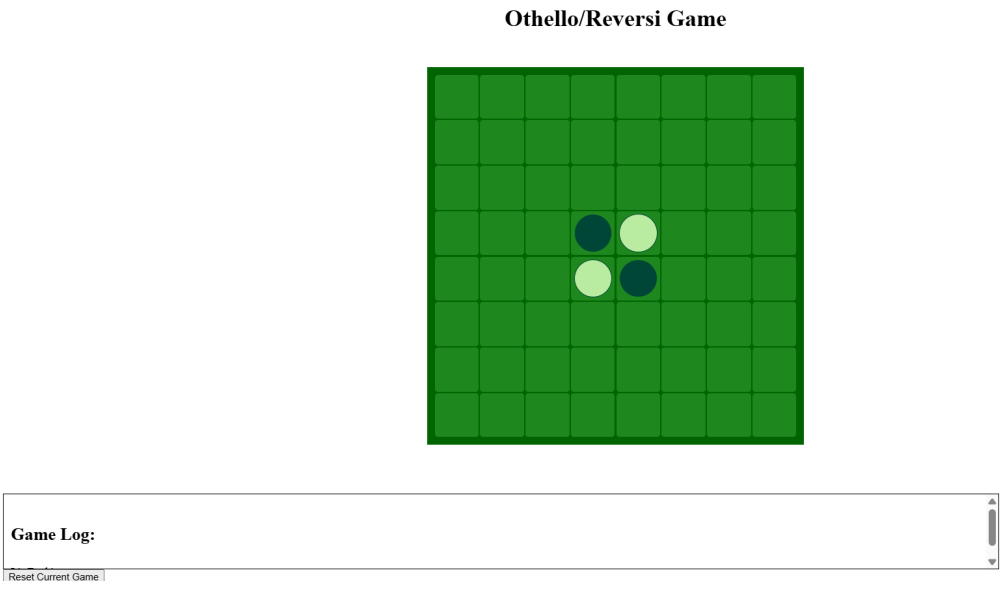
4. Running the applications

From the project root directory "Reversi Project" run the following commands to play the game.

For the CLI version run 'python Stage1\game_engine.py'. You should see this entered to the console:

```
Welcome to CLI Reversi! :)
   1     2     3     4     5     6     7     8
1  None  None  None  None  None  None  None  None
2  None  None  None  None  None  None  None  None
3  None  None  None  None  None  None  None  None
4  None  None  None  Dark  Light None  None  None
5  None  None  None  Light Dark  None  None  None
6  None  None  None  None  None  None  None  None
7  None  None  None  None  None  None  None  None
8  None  None  None  None  None  None  None  None
60 max moves left
Its Dark's turn
Enter x coordinate of move:
```

For the completed Web version run 'python Stage3\flask_game_engine.py'. Then open the web page locally hosted at 'http://127.0.0.1:5000/'. It should initially look like this:



5. Using the interface

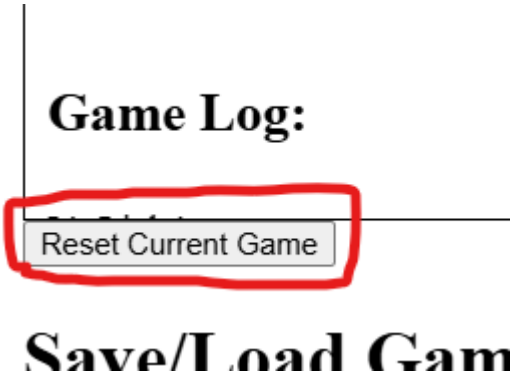
The green board displays where counters are on the board. Dark green counters belong to the player indicated as ‘Dark’ in the game log. This player will go first and is also the human player you control when playing against the AI. Light green counters belong to the player indicated as ‘Light’ in the game log.

The game log will show messages about events happening in the game. It will show whose turn it currently is. It will also show if a move is invalid and what valid moves have been made. When the game is over, the winner will be displayed in the log.

To make a move in the game, click an empty cell on the board where you want to place your counter. The move must be legal (it must outflank at least one of the other player's counters). The result of the move will be instantly displayed on the board. The colour of the counter placed depends on whose turn it is.

Turns are automatically passed when a player makes a legal move or if a player cannot make any legal moves on their turn. The game ends when the board is full and there is no space for more counters or if both players find themself in a position where they cannot make any legal moves.

To reset the game, press the 'Reset Current Game' button. This will return the board to its initial state and Dark will begin the game.

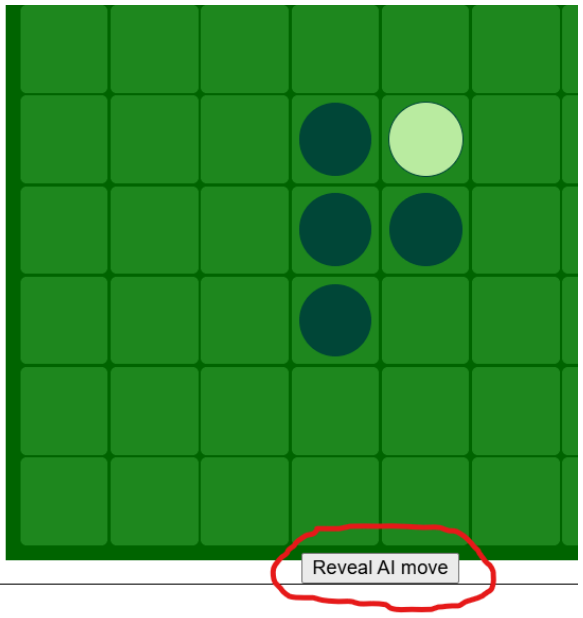


To save the current game, press the 'Save Game' button. This will download a file that can be used to load the game at a later date.

To load a game from a saved file, press 'Choose File' to select which file you want to use as the game save file to load. Then, press 'Load Game'. The board should be in the same state as it was saved and the turn of the player is reserved.

To play against the AI player, make your moves only as the Dark player. When it is Light's turn, a button 'Reveal AI move' will appear. Click this button to make the AI make the move for Light allowing you to play against the AI player.

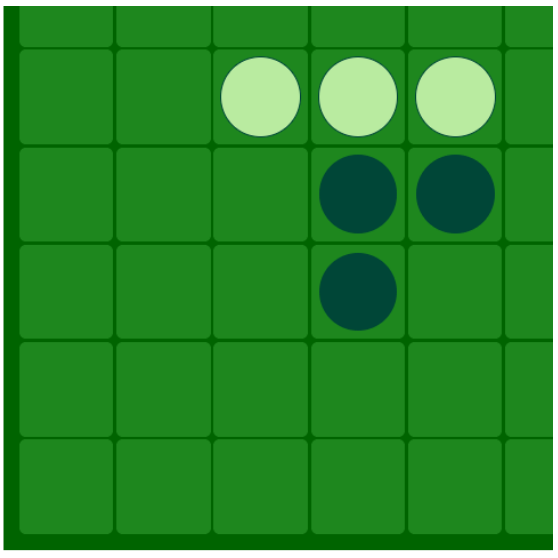
After your move:



It's Dark's turn.
Move accepted at (4, 6)
It's Light's turn.

Reset Current Game

After clicking the AI move button:



It's Dark's turn.
Move accepted at (4, 6)
It's Light's turn.
Move accepted at (3, 4)
It's Dark 's turn.

Reset Current Game

6. Using the API

Functions:

The functions in the project can be used in another implementation of Reversi. components.py contains some functions that can be used to debug, setup or check legal moves. flask_game_engine.py contains other functions needed for checking if legal moves are available, executing moves on the board, passing turns, calculating who wins the game

and AI move calculation. As long as data inputted into the functions is of the correct format, they can be used for the logic of the game Reversi in any other classic implementation. The ‘board’ must be a list of 8 lists containing 8 values for each cell (“None “, “Dark “ or “Light”). The current player’s turn is represented by either “Dark “ or “Light”. The coordinates must be integers between 1 and 8.

Web routes:
GET /

- Loads the main page for the game

GET /move?x=<num>&y=<num>

- Attempts to make a move at column x and row y on the board. Returns JSON data with the status of the response, the updated state of the board and a message about the move made.
- Example response:


```
{
  "status": "success",
  "player": "Light",
  “Message”: False,
  "board": [[“None “,”None “,”None “,”None “,”None “,”None “,”None “,”None “],[“None “,”None “,”None “,”None “,”None “,”None “,”None “,”None “],[“None “,”None “,”None “,”Dark “,”Light“,”None “,”None “,”None “],[“None “,”None “,”None “,”Dark “,”Dark “,”None “,”None “,”None “],[“None “,”None “,”None “,”Dark “,”None “,”None “,”None “,”None “],[“None “,”None “,”None “,”None “,”None “,”None “,”None “,”None “],[“None “,”None “,”None “,”None “,”None “,”None “,”None “,”None “],[“None “,”None “,”None “,”None “,”None “,”None “,”None “,”None “]
}
```


GET /save

- Downloads a JSON of the board, whose turn it was, and whether the game was won to the user’s device.

GET /ai_move

- Calculates an AI move depending on the current state of the board. Returns a response with a status and the coordinates of the calculated move to be used in /move.
- Example response:


```
{
  "status": "success",
  "x": 5,
  "y": 3,
```

}

POST /load

- Extracts game data from the provided save file to set the board to the state in the save file.

POST /reset

- Resets the game to its initial state and reloads the main page.

7. Limitations

For the web version, only an 8x8 board can be used. The CLI version can use any size board of an even integer >2 with a recommended enforced maximum of 16x16.

Appendix - Test information

Module	Function Tested	Test Name	Description	Expected Result	Result of Test	Outcome
components.py	initalise_board()	test_board_is_correct_size	Board initialises correctly with specified valid size	NxN board for sizes tested (4, 8, 10, 16)	Board generated was of correct size	PASS
components.py	initalise_board()	test_board_cells_initialised_as_none	All cells in the board except middle ones are “None “	All cells are “None “ except middle 4 cells	No cell checked was not “None “	PASS
components.py	initalise_board()	test_invalid_size_type	Input of invalid type (string) instead of int used in the function should cause Type error	Type error to be raised	Type error was raised	PASS
components.py	initalise_board()	test_invalid_size_odd	Odd number inputted into the function causes Value error	Value error to be raised	Value error was raised	PASS
components.py	initalise_board()	test_invalid_size_too_small	Number <4 inputted into the function causes Value error	Value error to be raised	Value error was raised	PASS
components.py	initalise_board()	test_invalid_size_too_large	Number >16 inputted into the function causes Value error	Value error to be raised	Value error was raised	PASS

components.py	legal_move()	test_initial_legal_moves_for_dark	The 4 legal moves at the start of the game should be valid	The moves to be considered legal by the function	True returned for all 4 move coordinates	PASS
components.py	legal_move()	test_initial_illegal_moves	Moves that are not legal at the start of the game should be invalidated by the function	The moves to be considered illegal by the function	False returned for all 4 move coordinates	PASS
components.py	legal_move()	test_move_on_occupied_square	A move on a cell that already has a counter should be illegal	The move is considered illegal by the function	False returned	PASS
components.py	legal_move()	test_move_out_of_bounds_does_not_crash	A move out of the range of the board size should be invalid and not crash	The function does not crash and move is considered illegal	False returned without crash	PASS
components.py	legal_move()	test_valid_flanking_move	A move that validly flanks an opponent's counter on an empty cell should be legal	The move is considered legal by the function	True returned	PASS
components.py	print_board()	test_print_board_output	The ASCII representation of the board should contain coordinate numbers and cells of dark, light and none	Output of the function contains numbers and expected phrases "None ", "Dark ", "Light"	"None ", "Dark ", "Light" and "1" found in the output	PASS
flask_game_engine.py	execute_move()	test_execute_move_flips	Counters outflanked by a move should be flipped to the correct colour	Light counter between newly placed Dark counter and other Dark counter should be turned to a Dark counter	The outflanked counter is flipped to the correct colour	PASS
flask_game_engine.py	pass_turn()	test_pass_turn	Passing a turn should change the current player to the	Turn goes from Dark player to Light player after	String used to store turn changed from "Dark " to "Light"	PASS

			other colour (Dark -> Light, Light -> Dark)	executing function once		
flask_game_engine.py	calculate_winner()	test_calculate_winner_dark_wins	Fill the board with more Dark counters than Light counters to see if Dark wins	Function should output Dark as the winner with the string “dark”	Function returned “dark”	PASS
flask_game_engine.py	calculate_winner()	test_calculate_winner_draw	When the board has an equal amount of dark counters and light counters, the winner calculated should be “draw”	Function outputs that the game ended in a draw by returning the string “draw”	Function returned “draw”	PASS
flask_game_engine.py	legal_move_available()	test_legal_move_available_true	When there are legal moves available, such as in the initial state of the game, the function should correctly decide that there are legal moves available	There are legal moves available so the function returns True	Function returned True	PASS
flask_game_engine.py	legal_move_available()	test_legal_move_available_false	When the there are no legal moves available, such as the case where all cells in the board have a counter, the function should correctly decide there are no legal moves	There are no legal moves so the function returns False	Function returned False	PASS
flask_game_engine.py	index()	test_index_route	The index page should load correctly and return status	Response status code is 200 and HTML data is correct	Response status code was 200 and byte sequence	PASS

			code 200		‘<html’ was in the HTML data	
flask_game_engine.py	save()	test_save_game_route	The request should return JSON data with correct game data and return status code 200	Response status code is 200 and JSON data contains game_state dictionary data	Response status code is 200 and JSON data contains game_state dictionary data	PASS
flask_game_engine.py	load()	test_load_game_route	Save files that are loaded should load the game state correctly preserving the current player’s turn and if the game has been won or not	Status code 302 returned from the POST request and the data in the file is correctly loaded to the game_state dictionary	Status code 302 returned from the POST request and the data in the file is correctly loaded to the game_state dictionary	PASS
flask_game_engine.py	reset()	test_reset_game_route	The current game should be reset to initial values no matter what state it is in when reset button is pressed. The game state is altered from its initial state before the test	State of the game is returned to its initial state with the player turn being set to “Dark “, the game being not won and the board reset to the starting state	Turn was set to “Dark “, game_won was set to False, board was set to initial state.	PASS
flask_game_engine.py	ai_move()	test_ai_move_route	The move the AI calculates should contain valid integer coordinates within the range of the board	Status code 200 returned and move provided is numerically valid	Status code 200 returned and move provided is numerically valid	PASS
flask_game_engine.py	move()	test_move_route_valid	Sending a move to the server should return a status of success if it is legal. It should also pass the turn to the other	Status in the response data is “success” and the turn was passed from “Dark “ to “Light”	Status in the response data is “success” and the turn was passed from “Dark “ to “Light”	PASS

			player.			
`flask_game_engine.py`	`move()`	test_move_route_invalid	Sending a move to the server should return a status of ‘fail’ if the move is illegal and explain that in a message	Status in the response data is “fail” and message in response data contains “Move is not legal”	Status in the response data is “fail” and message in response data contains “Move is not legal”	PASS