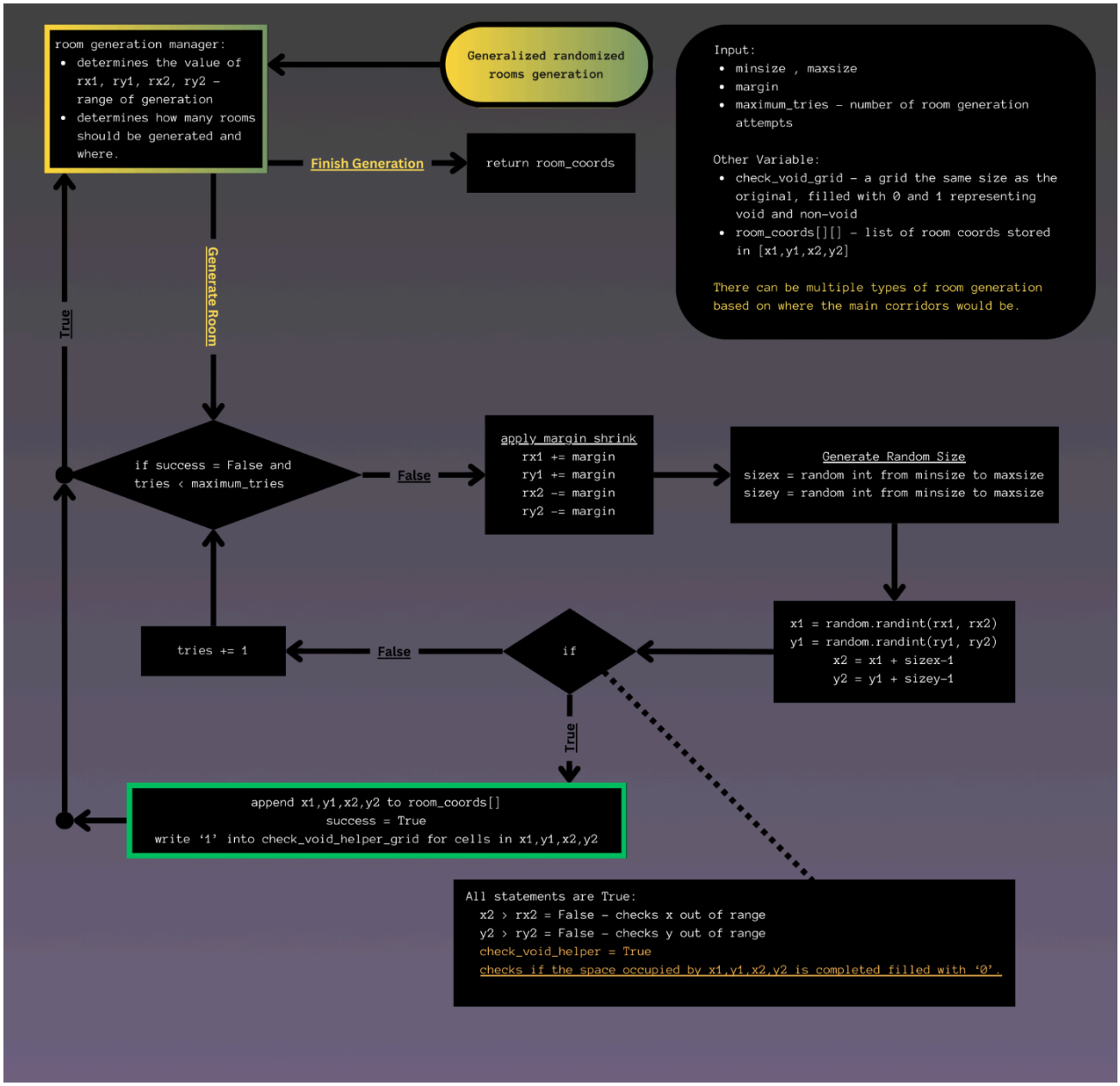
IB Computer Science IA Report

Battle Map Generator

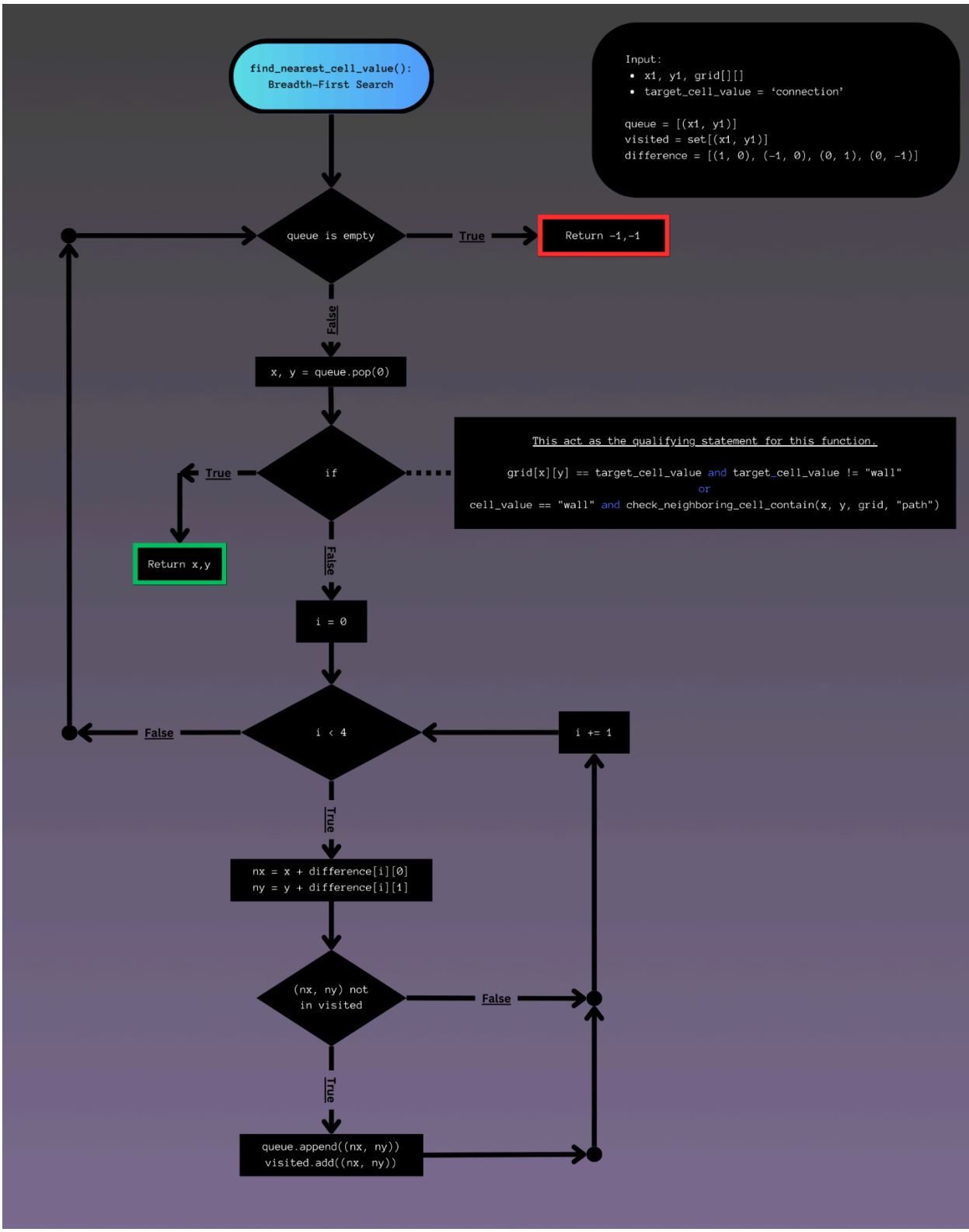
Criterion B - Design

1.0 System Flow Diagrams

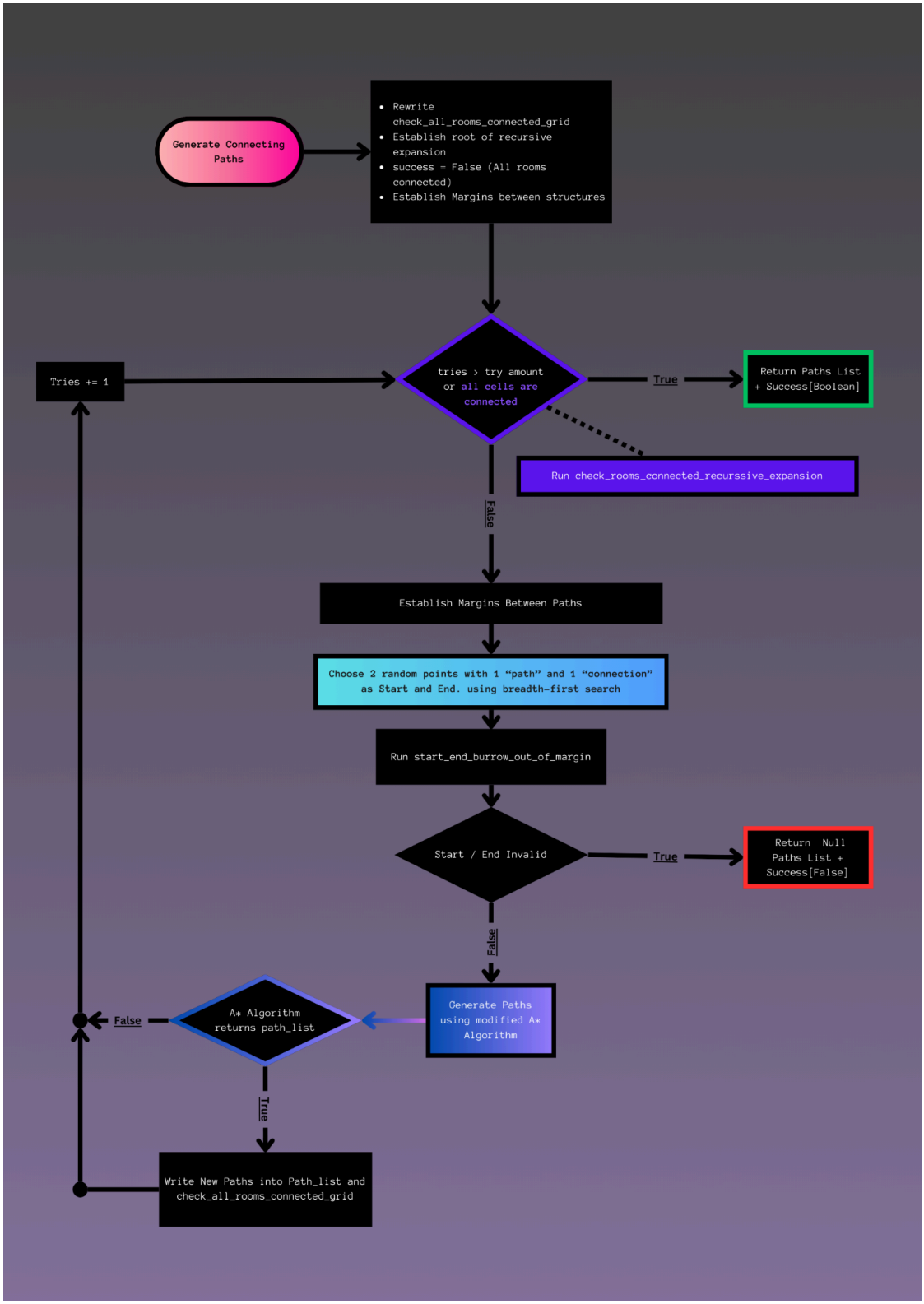
1.1 Check Void System



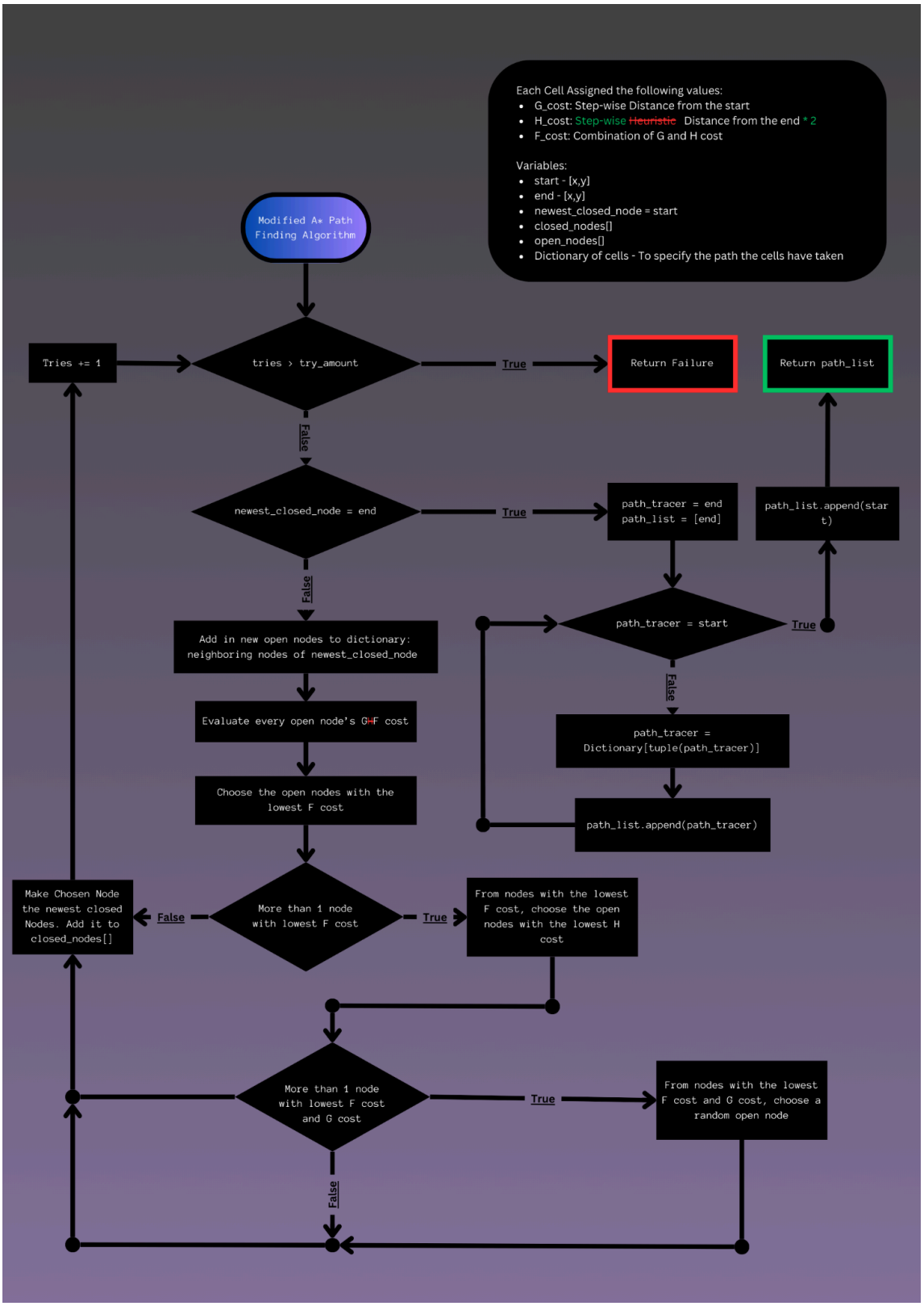
1.2 Breadth-first search(Amangeldi)



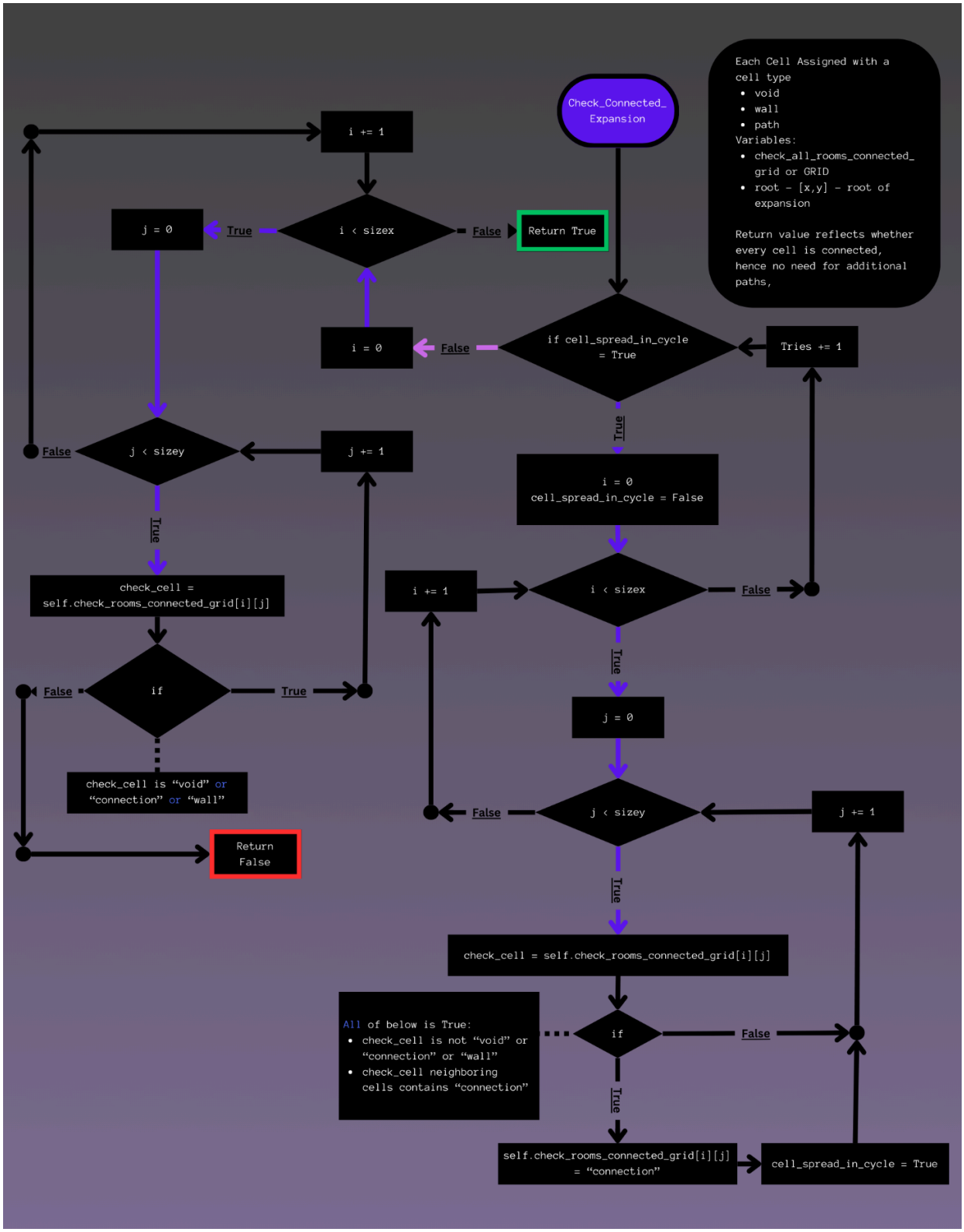
1.3 Main Path Generator in StructureOrganiser



1.4 Modified A* Path Finding Algorithm(Belwariar)

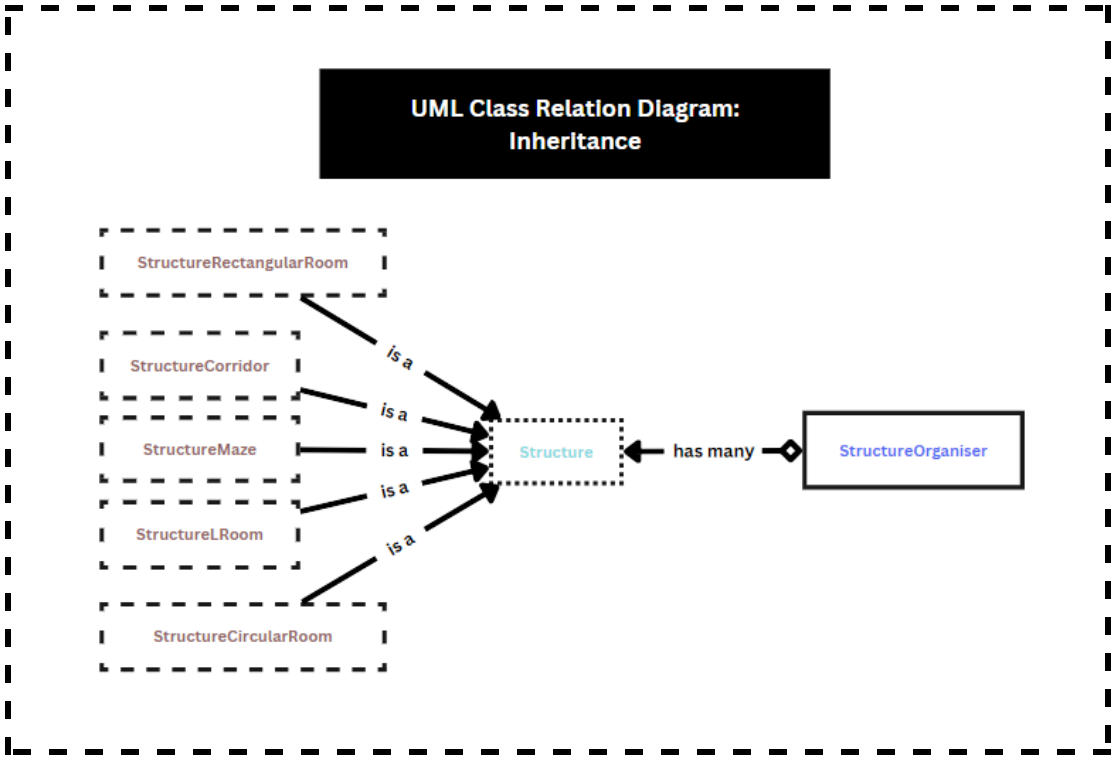


1.5 Check Connected Expansion

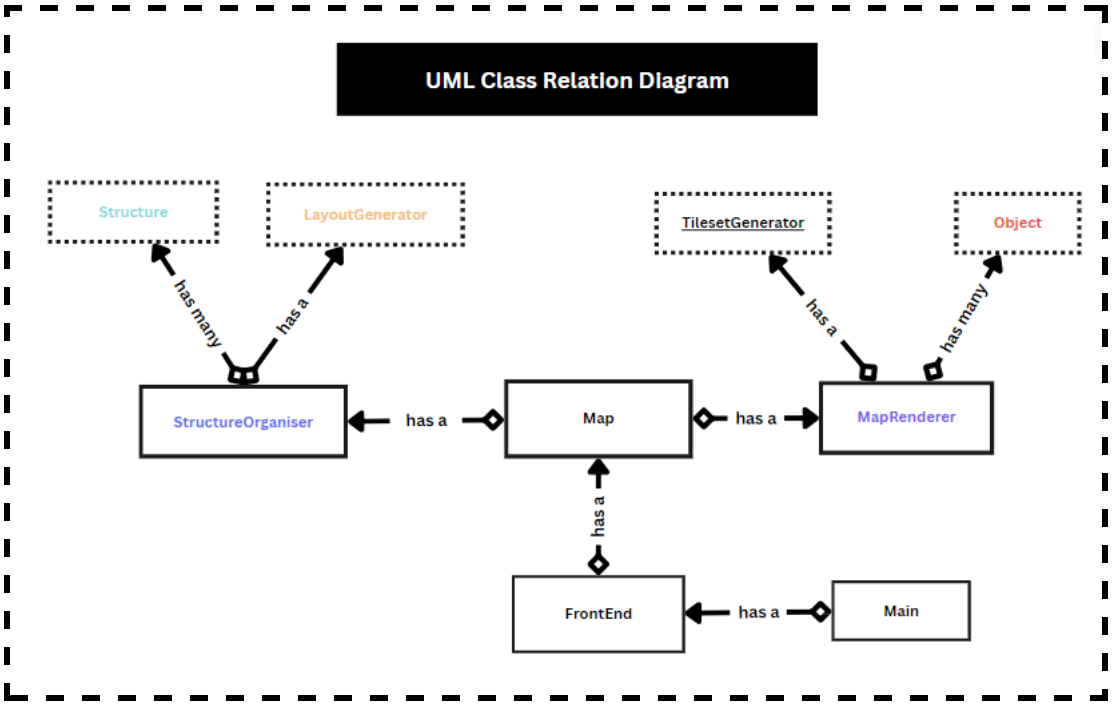


2.0 UML Diagrams

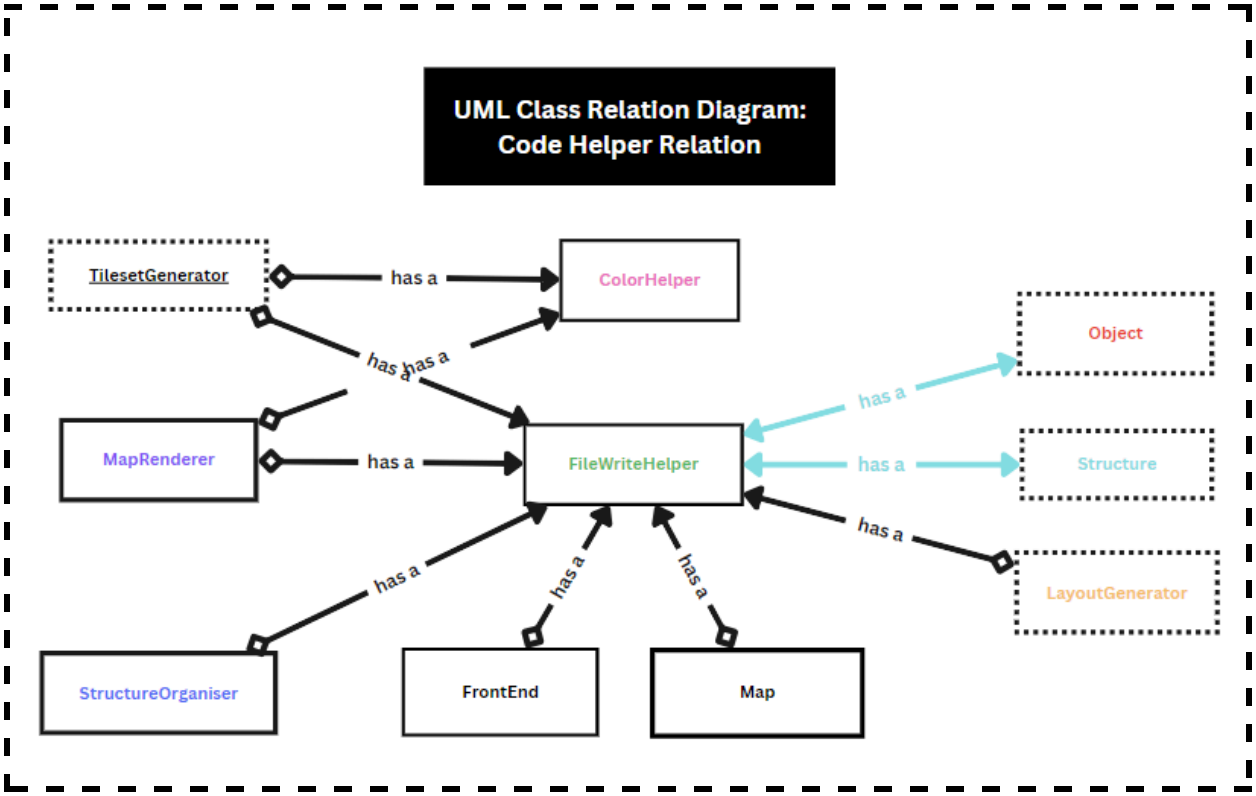
2.1 Structure Classes Inheritance



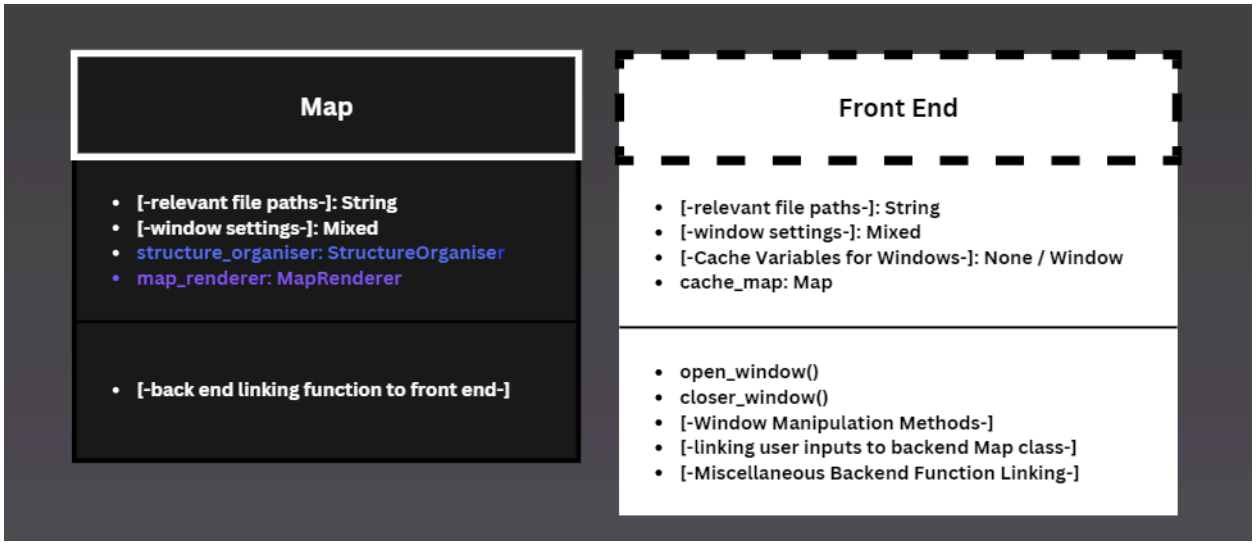
2.2 All Class Relationships



2.3 Code Helper Classes Dependency



2.4 Class Details



UML Diagram

FileWriteHelper

- no variables -

- csv_to_array(String: file_path, int: skip_line): return mix[][]
- write_array_to_csv(String: file_path, mix[]: data)
- clear_csv(String: file_path)

- load_properties(String: file_path): return mix{}
- save_properties(String: file_path, mix{}: dictionary)

- load_txt_as_string(file_path): return String
- append_to_txt(String: file_path, String: data)
- clear_txt_file(String: file_path)
- clear_folder(String: folder_path)

- write_array_of_dicts_to_json(String: file_path, mix{}: data)
- clear_json_file(String: file_path)

ColorHelper

- color_dict: String{}

- get_hue_from_range(int: number, int: min_value, int: max_value): return int
- hue_to_hex(String: hue): return String
- color_to_hex(String: color_name): String
- get_complementary_color(String: hex): return hex

LayoutGenerator

- [-relevant file paths-]: String
- [-window settings-]: Mixed

- grid: String[][]

- cell_types: String[]
- cell_types_file_path: String
- cell_types_CSV: mix[][]

- fwh: FileWriteHelper

- xy_order_helper(x1, y1, x2, y2): return x1, y1, x2, y2
- generate_room(int: x1, int: y1, int: x2, int: y2) #a function for each room type:
  - rectangular_room(int: N, int: E, int: S, int: W)
  - L_room #L-shaped(int: midx, int: midy, int: quadrant_missing)
  - maze
- grid_transformation(grid): return grid
  - rotation #each as separate functions, only by 90 degree
  - mirroring #each as separate functions: horizontal and vertical

StructureOrganiser

- [-relevant file paths-]: String
- [-window settings-]: Mixed

- structure_list: Structure[]
- path_list: int[][]

- path_radius: int
- path_margin: int

- mp: LayoutGenerator
- fwh: FileWriteHelper
- color_helper: ColorHelper

- check_void_helper_grid: int[][]
- check_rooms_connected_grid: String[][]

- Structure Organiser Methods
  - 1 room per chunk
  - Spine

- PathGenerator

- write_structure_list_into_layout_grid()
- write_path_list_into_layout_grid()

- write_structure_list_into_file()
- write_path_list_into_file()

Objects

- image_path: String
- image: Image
- x1: int
- x1: int
- y1: init
- x2: int
- y2: int

- update_image: void
- return_properties_as_dict(): Dictionary

MapRenderer

- [-relevant file paths-]: String
- [-window settings-]: Mixed

- texture_pack: String
- object_list = Object{}

- fwh: FileWriteHelper
- color_helper: ColorHelper
- texture_tile_set_generator: TileSetGenerator

- layout_render_format: String[][]

- cache_images: Image
  - structure_layer
  - layout
  - object_layer
  - rendered
  - display

- render_cache_image: void
  - structure_layer
  - layout
  - object_layer
  - rendered
  - display
- layout_to_render_format: String[][]

Structure

- x1: int
- y1: init
- x2: int
- y2: int
- layout: String[][]

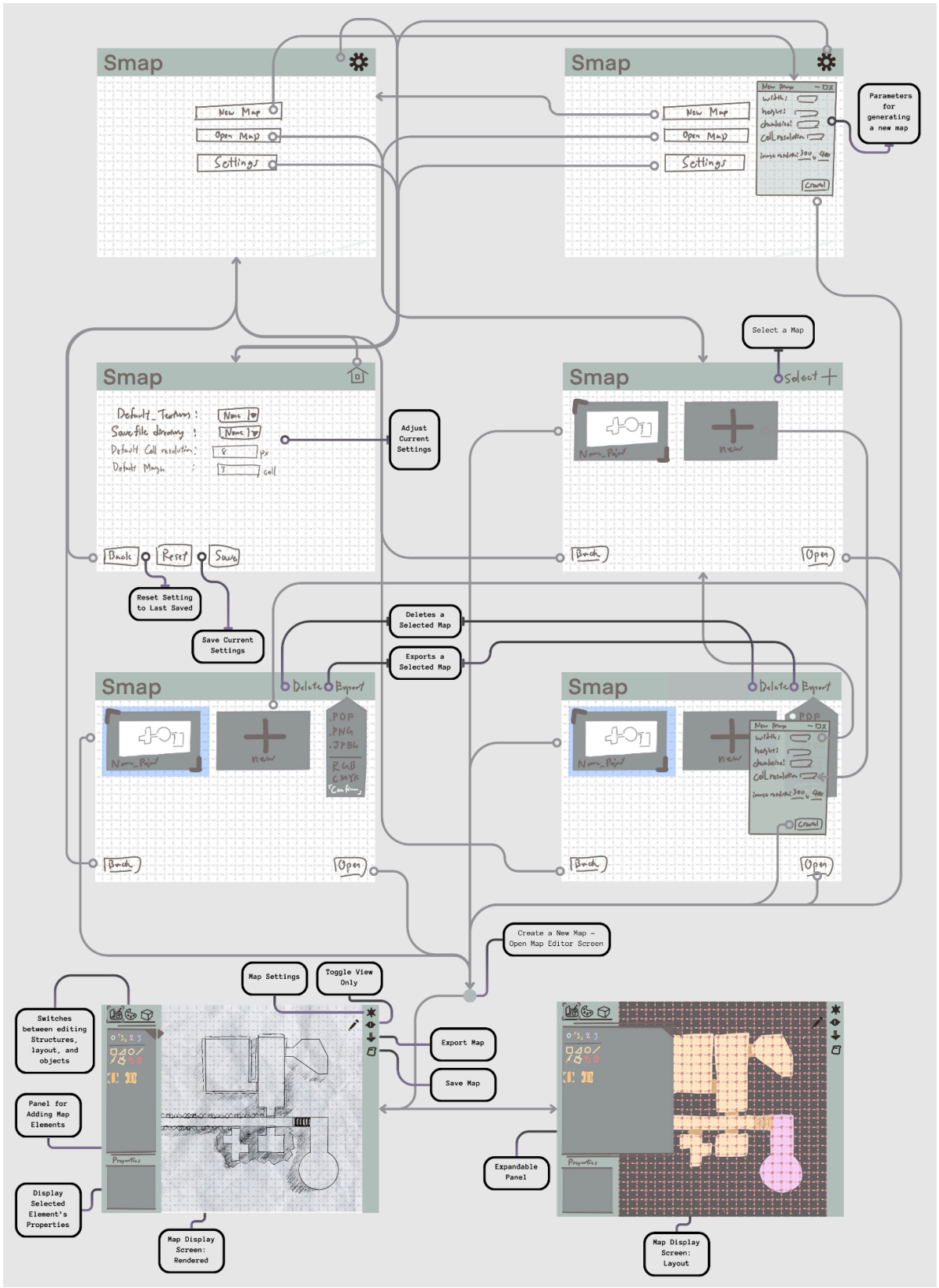
- generate_layout()
- return_properties_as_string(): String
- return_properties_as_dict(): Dictionary

TilesetGenerator

- fwh: FileWriteHelper
- color_helper: ColorHelper

- generate_tile_png(): Image
- generate_tile_png_from_format_Id(): Image

3.0 Screen Mock-Ups



4.0 Tables - csv files

4.1 cell_types_texture_hiearchy.csv

None	void	path	wall	entrance	hazard_1	hazard_2	hazard_3	water_1	water_2	water_3	connection
void	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
path	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
wall	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
entrance	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
hazard_1	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
hazard_2	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE
hazard_3	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE
water_1	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	FALSE
water_2	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE
water_3	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
connection	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

4.2cell_types_layout_hiearchy.csv

None	void	path	wall	entrance	hazard_1	hazard_2	hazard_3	water_1	water_2	water_3	connection
void	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
path	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
wall	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
entrance	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
hazard_1	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE
hazard_2	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE
hazard_3	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE
water_1	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE
water_2	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE
water_3	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE
connection	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE

4.3 cell_types.csv

index	structure_type	Variation [unimportant]	Hex color
0	structure_0	regular	000000
1	rectangular_room	regular/treasure_room/altar	33bb33
2	maze	regular	bbbb33
3	circular_room	regular/spiral_staircase	33bb33
4	corridor	regular/staircase/pillar_lined_corridor	3333bb
5	L_room	regular	bb33bb

4.4 Structure_types.csv

Index	Cell_Type	Hex color
0	void	000000
1	path	bf9f40
2	wall	333333
3	entrance	80bf40
4	hazard_1	ee2222
5	hazard_2	ee2222
6	hazard_3	ee2222
7	water_1	2222ee
8	water_2	2222ee
9	water_3	2222ee
10	connection	cb6ce6

5.0 Psuedo Code

5.1 Maze Generation: Recursive Backtracking(Abed-Esfahani)

```
MAZE = [][] #Maze is a bitmap populated with 1 and 0 -> path and walls

def recursive_backtracking(ROW, COL):
        MAZE[ROW][COL] = 0
        DIRECTIONS = [(0, 2), (0, -2), (2, 0), (-2, 0)]
        random.shuffle(DIRECTIONS)

        for DX, DY in DIRECTIONS:
                NEXT_ROW, NEXT_COL= ROW+ DX, COL + DY
                if 0 <= NEXT_ROW< len(MAZE[0]) and 0 <= NEXT_COL < len(MAZE) and
        MAZE[NEXT_ROW][NEXT_COL] != 0:
                MAZE[NEXT_ROW][NEXT_COL] = 0
                MAZE[ROW + DX,// 2][COL + DY // 2] = 0
                recursive_backtracking(NEXT_ROW, NEXT_COL)

recursive_backtracking(1, 1)
output MAZE
```

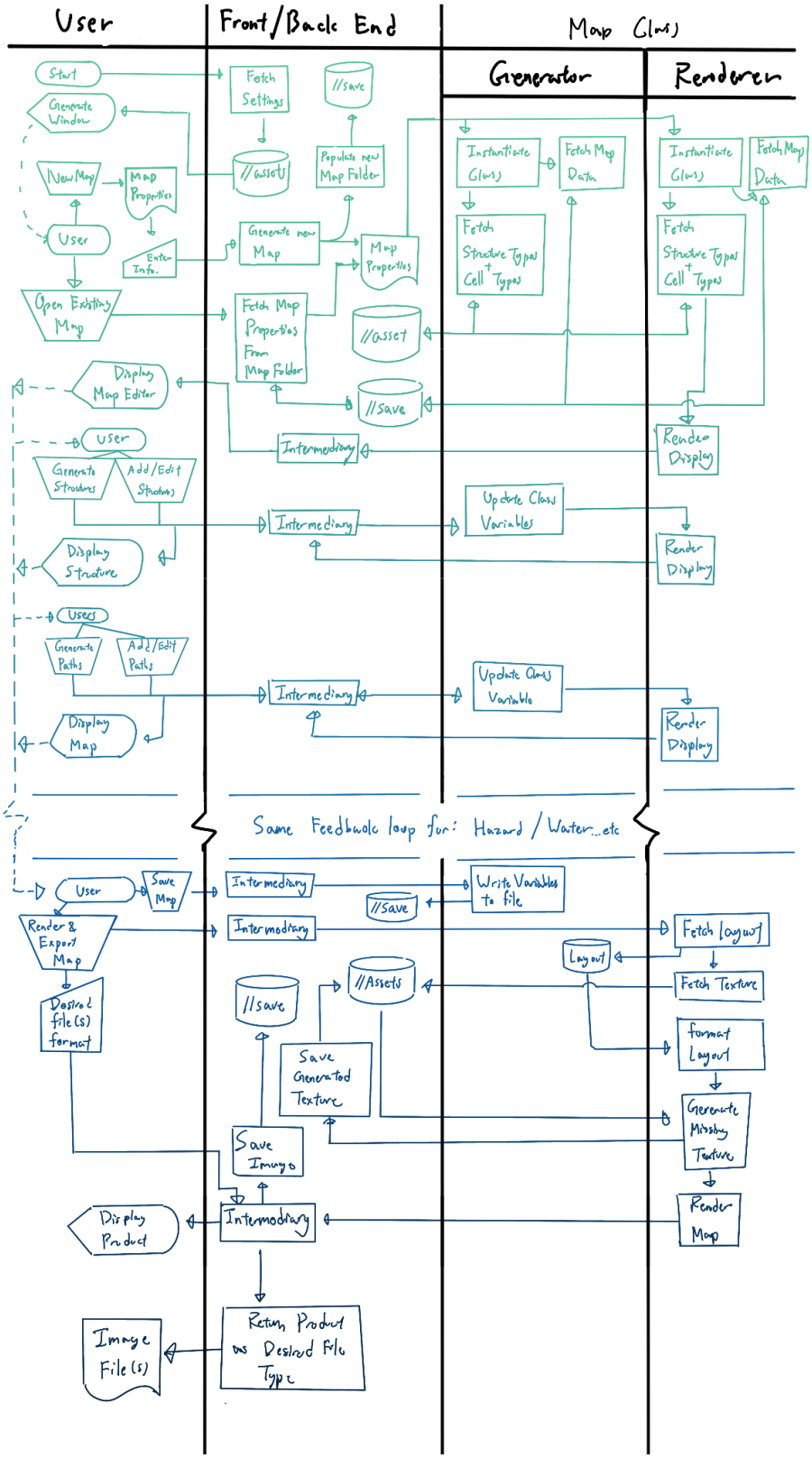
5.2 Path Finding Algorithm: A*（heuristic modified)(Belwariar)

```
def heuristic(NODE_1, NODE_2)
        return abs(NODE_1[0] - NODE_2[0]) + abs(NODE_1[1] - NODE_2[1])

def AStar(start, goal) // start, goal = (a,b), (c,d) coordinates
        // Initialize the open and closed lists
        OPEN_LIST = empty priority queue
        CLOSED_LIST = empty set

        // Initialize the start node
        G_SCORE[START[0]][START[1]] = 0
        F_SCORE[START[0]][START[1]] = G_SCORE[START[0]][START[1]] + heuristic(start, goal)

        // Add the start node to the open list
        OPEN_LIST.append(START)

        NODE_PATH = {}

        while OPENLIST is not empty
                CURRENT_NODE = OPEN_LIST.pop()
```

```
                // Check if the current node is the goal
                if CURRENT_NODE == GOAL
                        return reconstructPath(CURRENT_NODE)

                // Add the current node to the closed list
                CLOSED_NODE.append(CURRENT_NODE)

                // Explore the neighbors of the current node
                for each NEIGHBOR in currentNode.neighbors

                        if NEIGHBOR in CLOSED_LISR
                                break

                        NODE_PATH.append(NEIGHBOR: CURRENT_NODE)

                        // Calculate the tentative gScore for the neighbor
                        TENT_G_SCORE[NEIGHBOR[0]][NEIGHBOR[1]] =
G_SCORE[CURRENT_NODE[0]][CURRENT_NODE[1]]+ 1

                        // Check if the neighbor is not in the open list or the new gScore is lower
                        if NEIGHBOR not in OPEN_LIST or
TENT_G_SCORE[NEIGHBOR[0]][NEIGHBOR[1]] < G_SCORE[NEIGHBOR[0]][NEIGHBOR[1]]
                                // Update the neighbor's information
                                G_SCORE[NEIGHBOR[0]][NEIGHBOR[1]] =
TENT_G_SCORE[NEIGHBOR[0]][NEIGHBOR[1]]
                                F_SCORE[NEIGHBOR[0]][NEIGHBOR[1]] =
G_SCORE[NEIGHBOR[0]][NEIGHBOR[1]] + heuristic(NEIGHBOR, GOAL)

                        if NEIGHBOR not in OPEN_LIST
                                OPEN_LIST.append(NEIGHBOR)
        return NULL

def reconstructPath(NODE)
        PATH = []
        while NODE is not START
                PATH.APPEND(NODE_PATH[NODE])
                NODE = NODE_PATH[NODE]

  return PATH
```

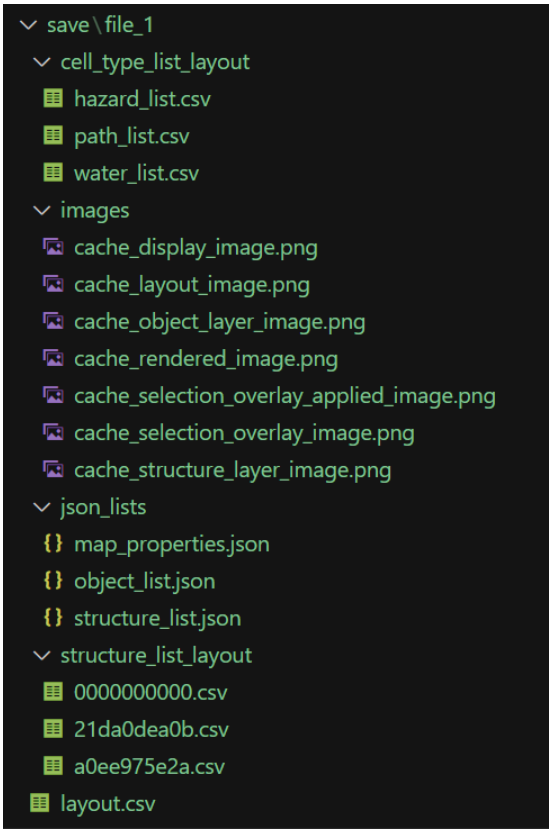
6.0 Others

6.1 ER Diagram



6.2 File Organisation Diagram - Map Save File

```
save_file
└── cell_type_list_layout
        └── hazard_list.csv
        └── path_list.csv
        └── water_list.csv
└── images
        └── cache_structure_layer.png
        └── cache_layout_layer.png
        └── cache_object_layer.png
        ……
└── json_lists
        └── map_properties.json
        └── object_list.json
        └── structure_list.json
└── json_lists
        └── map_properties.json
        └── object_list.json
        └── structure_list.json
└── structure_list_layout
        └── <structure id>.csv
        └── <structure id>.csv
        └── <structure id>.csv
        ……
```



6.3 File Organisation Diagram - Application File

```
Battle_map_creator
└── documentation_images
        └── Documentation <yyyy-mm-dd-hhmmss>.png
        └── Documentation <yyyy-mm-dd-hhmmss>.png
        …….
└── save
        └── save_file_1
        └── save_file_2
        ……
└── src
        └── assets
                └── csv_files
                        └── cell_type_texture_heiarchy.csv
                        └── cell_types.csv
                        └── structure_generation_methods.csv
                        └── structure_types.csv
                └── icons
```

```
                    └── object_icons
                        └── pillar_stone.png
                        └── cubboard_oak.png
                        .......
                    └── window_icons
                        └── blueprint_icon.png
                        └── painter_icon.png
                        ......
                └── json_files
                    └── file_paths.json
                    └── generation_setting.json
                    └── settings.json
                    └── window_settings.json
                └── textures
                    └── default
                    └── debug
                    └── textures_1
                    └── textures_2
                    ......
        └── front_end
            └── main_front_end.py
            └── map_class.py
        └── helpers
            └── color_helper.py
            └── file_write_helper.py
        └── map_generator
            └── structure_organiser.py
            └── layout_generator.py
            └── structures.py
        └── map_renderer
            └── map_renderer.py
            └── object.py
        └── textures_generator
            └── texture_tile_set_generator.py
        └── main.py

└── tests        # developmental tests
        └── test_1.py
        └── test_2.py
        .......
```

7.0 Testing Plan

Feature	Action	Expected Result	Success Criterion
Easy implementation	1 - Download the source code from the git hub and run the code as per the instructions.	Application Opens successfully.	1
Homescreen	2 - Run program	Homescreen appears	2
Application feature navigation	3 - Navigate to each screen.	Able to access the settings screen, new map screen and open map screen	2
Adjust base program settings	4 - Adjust screen appearance. Check the change in settings.json file.	Window appearance changes after saving. Settings are saved to program	3
An existing map file can be navigated with previews.	5 - Drag and drop the map folders into /save, then go to the open map screen.	The imported maps appear on the Open Map Screen.	4, 5
Open Existing maps	6 - Open an existing map.	An existing map is opened on the Map Editor Scree	4
Add and Delete Cells	7- Use the provided window panels to add and delete cells	C ells are added and deleted.	6
Add and Delete Structures	8 - Use the provided window panels to add and delete structures	Structures are added and deleted.	6
Saving the Map	9 - Save the map and exit the editor. Check the changes in /save folder and relevant map data.	Map data is saved.	7
Exporting the Map	10 - Navigate to the Open File screen and export the map as PNG	The map is rendered as An RGB PNG file	7
Importing texture	11 - Drag and drop the custom texture's folder into	-to be continued-	8

	/assets/textures		
Empty Map Creation	12 - Create a new map by entering numbers which differ from the provided parameters.	A new map is created and opens up the map editor screen.	9
Automatic Structure Placement	13 - Enter the parameters necessary for structure placement, and press the button.	Structures of variety are placed automatically on the map.	10
Automatic Pathway Placement	14 - Press the “Generate Paths” button.	Paths are generated to connect all existing structures.	10
Fast generation and use of imported texture.	15 - Export the map as a PDF in CMYK.	The map is rendered using the custom textures as a CMYK PDF file	11, 8, 7