User's guide :

1) In order to understand how to run the different algorithms, here is a short explanation of the structure of the project :
The modules :

Each algorithm has it's own module.
There are modules for the game board, the State of the game, and the State for the algorithm which learns an automaton (StateWithAuto).
The module Show is in charge of the display (GUI) .
The module Dfa is used for learning an automaton.
The module StateGenericFunctions holds different functions which are in use of more than one algorithm, and therefore are implemented in a more generic way .
The module Main contains all the "main" functions for the different algorithms together: for each algorithm, it's appropriate "main" initializes the variables it needs, and then runs it .

2) In order to run a specific algorithm, you should uncomment the appropriate code in the Main module (and, of course, leave the rest commented).

3) Changes for the board game are done in the Board module as follows :

Constant variables - credits for various actions, size etc.- all of these can be changed .

InitRoom function - the code block for adding walls in the room is commented. If you want to add walls, notice that you might need a big room. The walls which are added by the commented code require room of 6X9 (roomHeight X roomWidth) .

Regarding the scatteringStains and scatteringFruits functions - the commented code in both adds stains and fruits randomly. You can add them manually, in a similar way that one stain and one fruit are put right now, in the end of those functions. Notice, that you should put them within the room boundaries .

4) There are two reward functions in StateGenericFunctions module named computeReward and computeReward2. The second one can be used for policy iteration algorithm, and in order to use it, you need to change the last parameter for PolicyIteration.initModule in the Main module to "StateGenericFunctions.computeReward2 . "
It might produce different policies (depends on the reward values in Board module, and the size of the board etc.) .

5) The last line (Show.showRoom...) in Main module is used by all algorithms. Note not to comment it, unless you don't won't the GUI .

6) Automata Learning algorithm:

6.1) The automata synthesizer function is computationally heavy. Take it into account when setting the room's measures. See 6.4 .
6.2) The algorithm doesn't support fruit picking. Don't put them in the room while running it. Therefore, the basket is unnecessary for this algorithm, and it can be removed to make more space (recall from 6.1 that large rooms are not recommended for this algorithm.
This can be done by commenting all basket references in Board module, except for the basket initialization (there is a note in the code in the appropriate line) .
6.3) The algorithm gives an extra reward for a stain which is cleaned when the robot came to the dirty spot from underneath.

Therefore, if all the stains are in the lowest row, there will be no accepted words. The automata synthesizer won't be
able to generate an automaton for this case (it can't generate an automaton which doesn't accept any words) .
In order to change this "special" reward, you need to change the conditions in the method computeReward_NonMarkovian in AutomataLearning module .

6.4) It worked fast on room of 5X5 (walls included) with one stain in the center .