AOS –
User Manual

Introduction 3
Scenarios 3
  Activation and Deactivation Sever 3
  Creating New Project 4
  Add New Skill 8
  Edit Skill 11
  Edit Environment File 13
  Project Initialization Request 16
  Send a Get Execution Request 18
  Request to show Server's Logs 25
  Request to Stop the Robot 28
  Request the Generated Code 29

Introduction

For a robot developer in the robotics industry, building autonomous robots can be difficult and cumbersome. To solve that issue, AOS server was created. Our system, which is built on top of the server makes it even more convenient. With AOS-GUI, the developer can create a new project from scratch, operate the robot and debug its actions.

Our system is desktop application. Instructions for downloading the application can be found here.

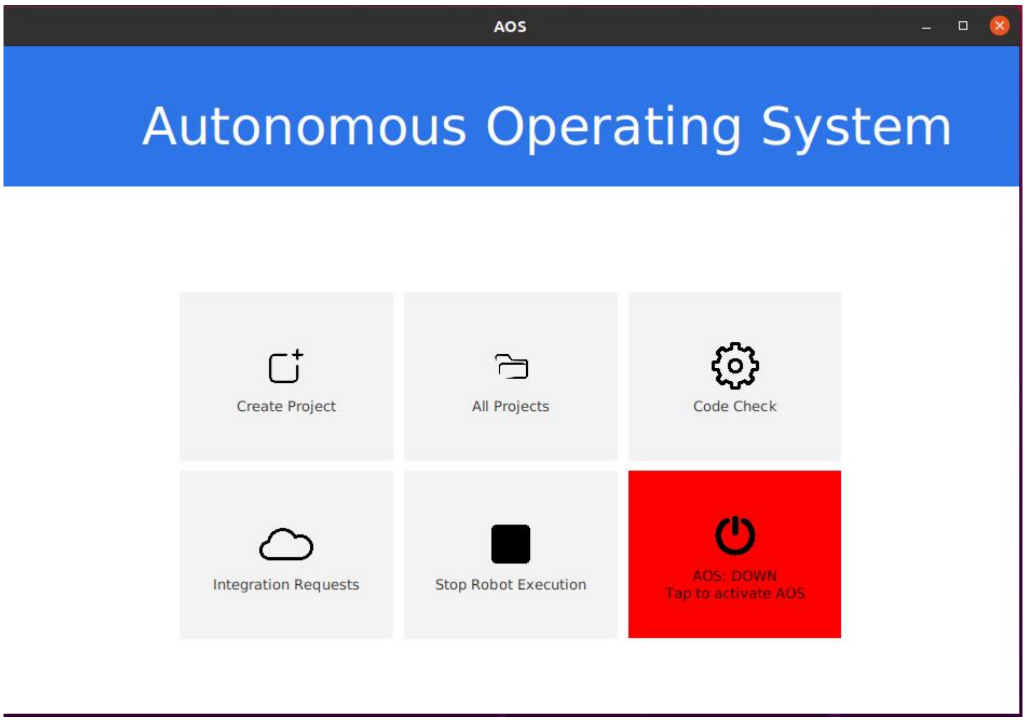
The following manual describes the optional scenarios in our system, with detailed instructions for each scenario, so you could use the system as painlessly and effortlessly as possible.

We assume that you have basic familiarity with the AOS server. If not, feel free to use the manual provided here, and any other relevant documentation on the AOS-WebAPI repository.

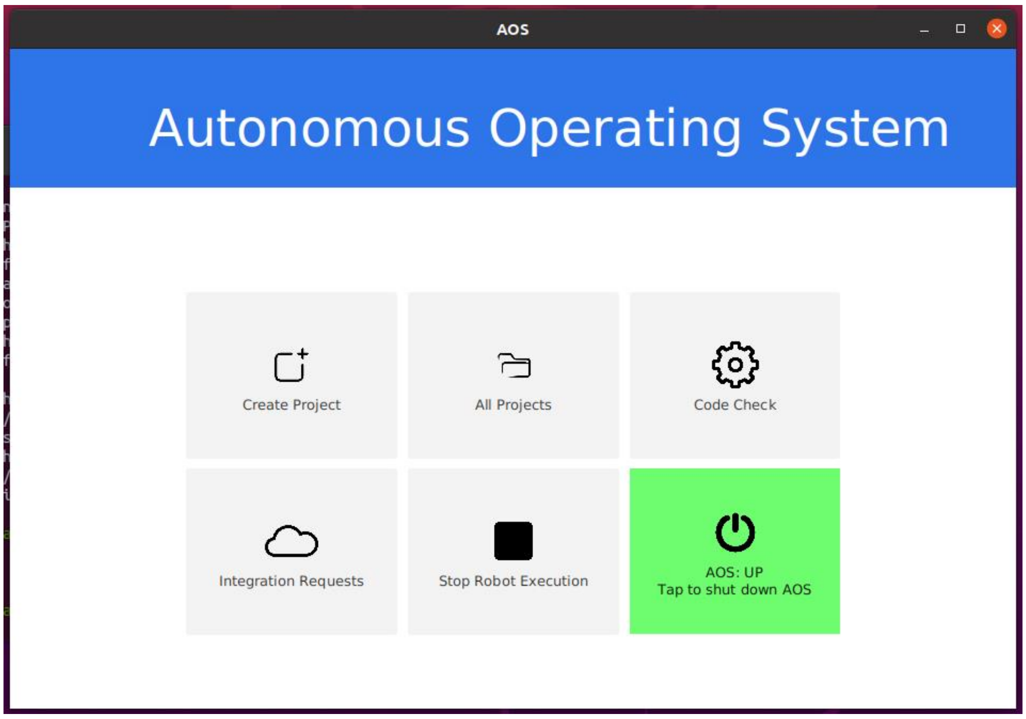
Scenarios

Activation and Deactivation Sever

1. Start the application, you should see the home screen



2. To activate the server, click on the AOS: DOWN red button.
3. The button should be green, and the server is up.

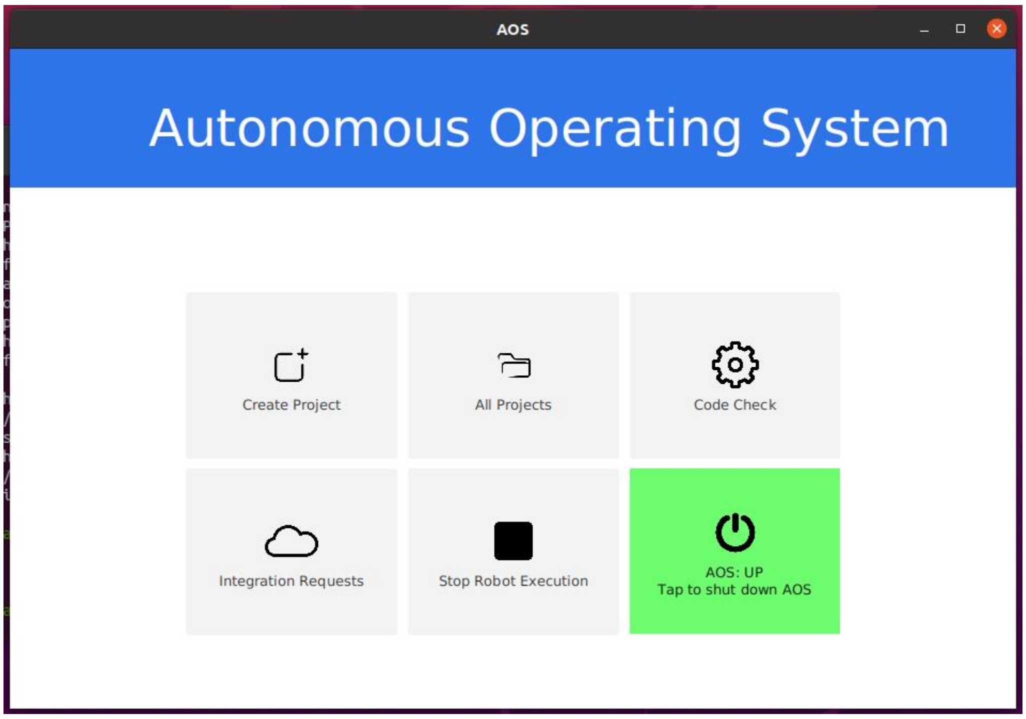


4. To Deactivate the server Press on the green AOS: UP button.
5. The button should change back to red, and the server has exited normally.

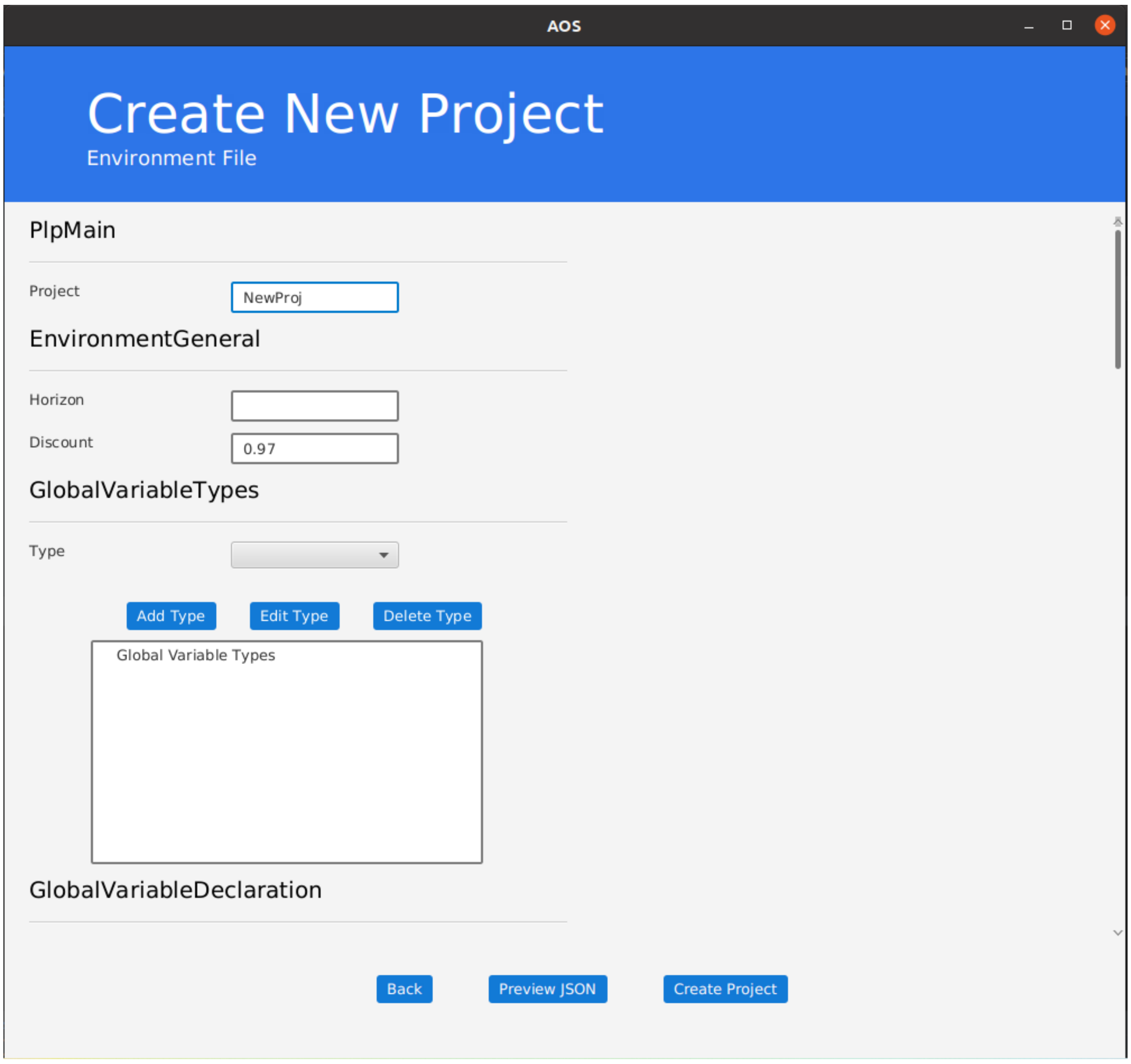
Creating New Project

Goal: basic scenario in the system, allows us to create new project in a dedicated form, insert all the needed parameters separately, and preview the JSON of the env file that will be generated once the creation is complete.

1. Start the Application, if not already started, in the home page press on the create project button, as seen below.

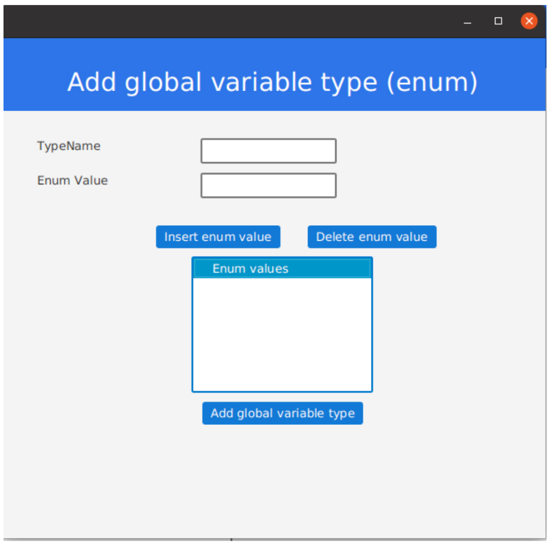


2. A new window will open. Open

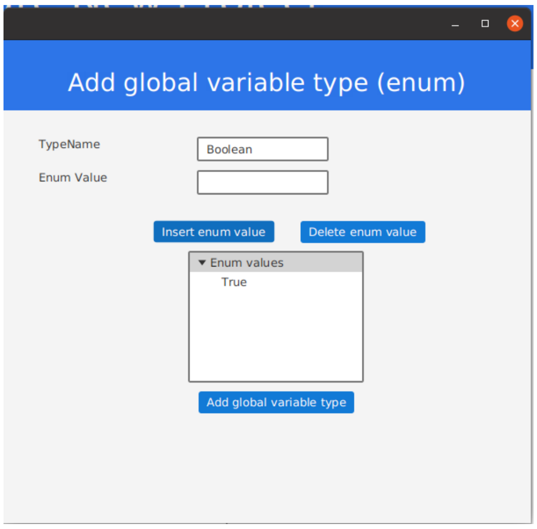


Project	NewProj
Horizon	
Discount	0.97
Type	

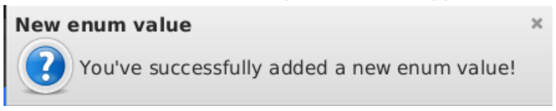
3. Enter a new name for the project and enter in the Horizon field a number.
   The "GlobalVariableTypes" section enables users to define custom state variable types. More specifically, Enums and compound data structures that aggregate multiple data items.
   You can choose the type in the 'Type' dropdown.
   For Enum type, you need to insert the name and the Enum values (more than 1). For example:



   Once you insert new Enum value, you will see it in the list. To finish, press Add global variable type and you will be redirected to the previous screen (create new project).



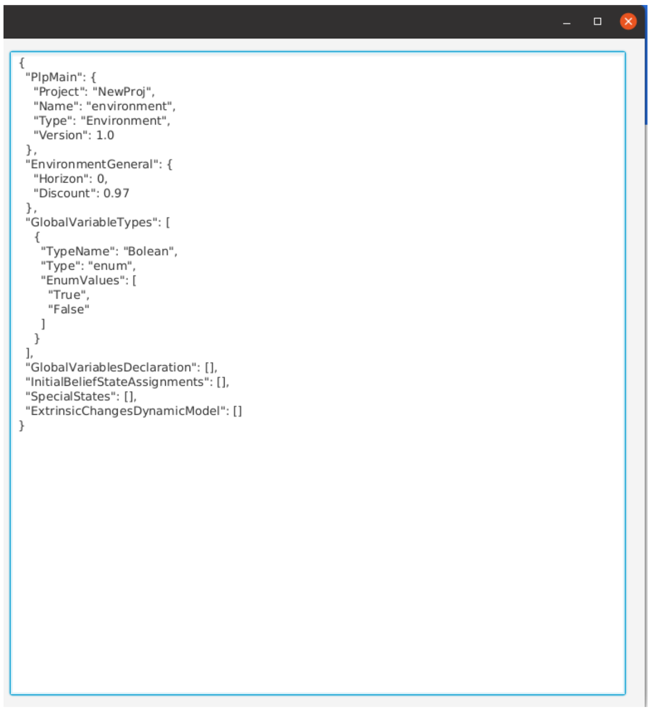
4. Once you add a new globalVariable, A pop up message should appear in the top right corner of the screen. For example: (Enum type)



New enum value

You've successfully added a new enum value!

   You can also edit the variable, add more values if it is Enum type, or delete it.
5. Once you finished adding all the needed parameters, you can press on Preview JSON, and see the environment file that will be created:



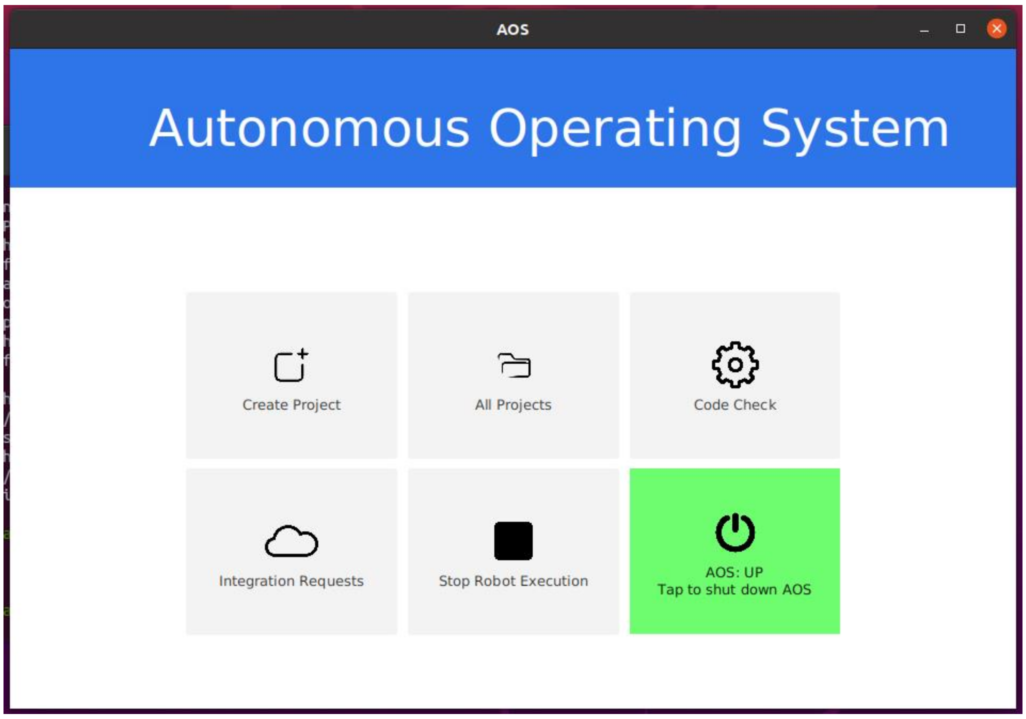
```
{
 "PlpMain": {
  "Project": "NewProj",
  "Name": "environment",
  "Type": "Environment",
  "Version": 1.0
 },
 "EnvironmentGeneral": {
  "Horizon": 0,
  "Discount": 0.97
 },
 "GlobalVariableTypes": [
  {
   "TypeName": "Bolean",
   "Type": "enum",
   "EnumValues": [
    "True",
    "False"
   ]
  }
 ],
 "GlobalVariablesDeclaration": [],
 "InitialBeliefStateAssignments": [],
 "SpecialStates": [],
 "ExtrinsicChangesDynamicModel": []
}
```

6. You can continue and change some parameters or press on 'create project'.
7. A pop-up message will appear, indicating the project was created.
8. To return to the main screen. Press the 'Back' button.

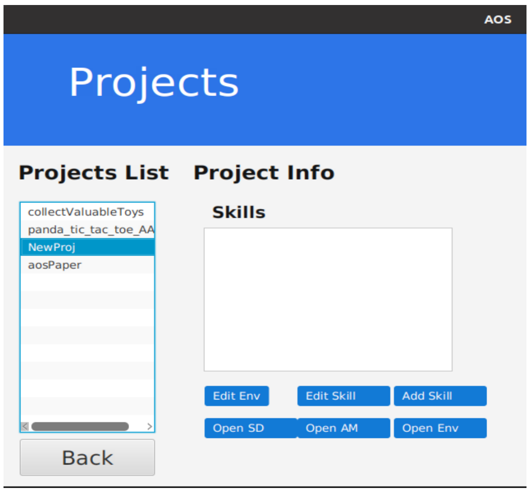
Add New Skill

Goal: add new skill to existing project, or newly created project that contains only env file. For each skill, we need to define the SD and AM parameters.

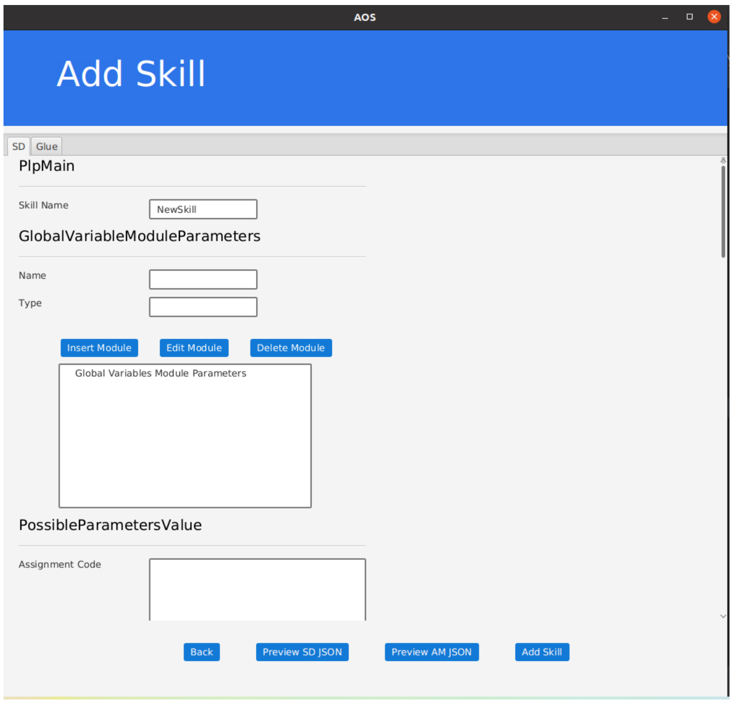
1. In the Home Page press All Project Button.



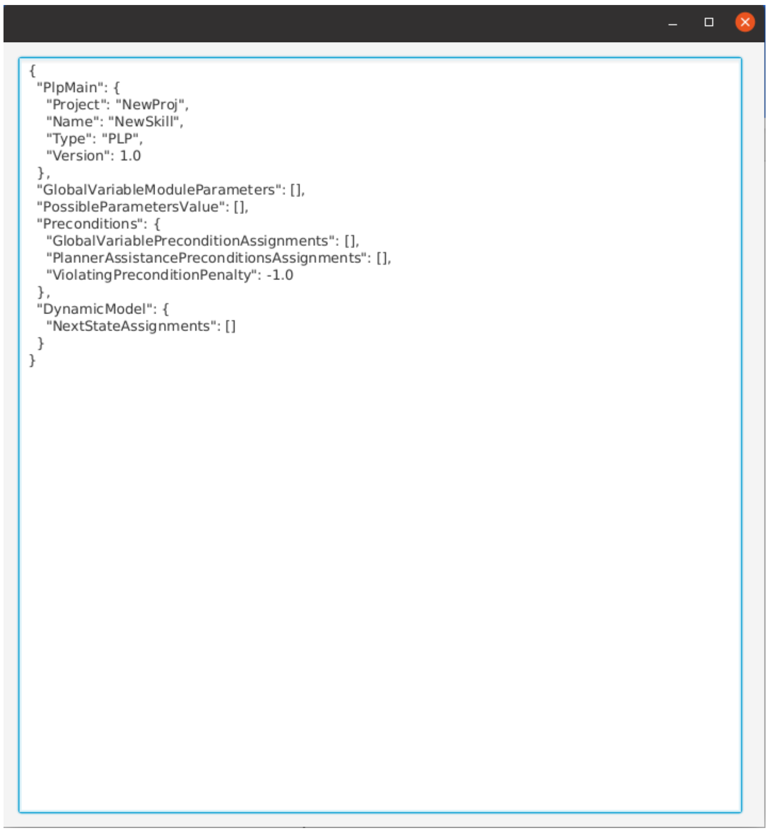
2. A new Window will open, in the project list select the project to add skill to, and afterward press the Add Skill button.



3. A new window will open. Fill in the desired parameters on each tab (SD,AM).

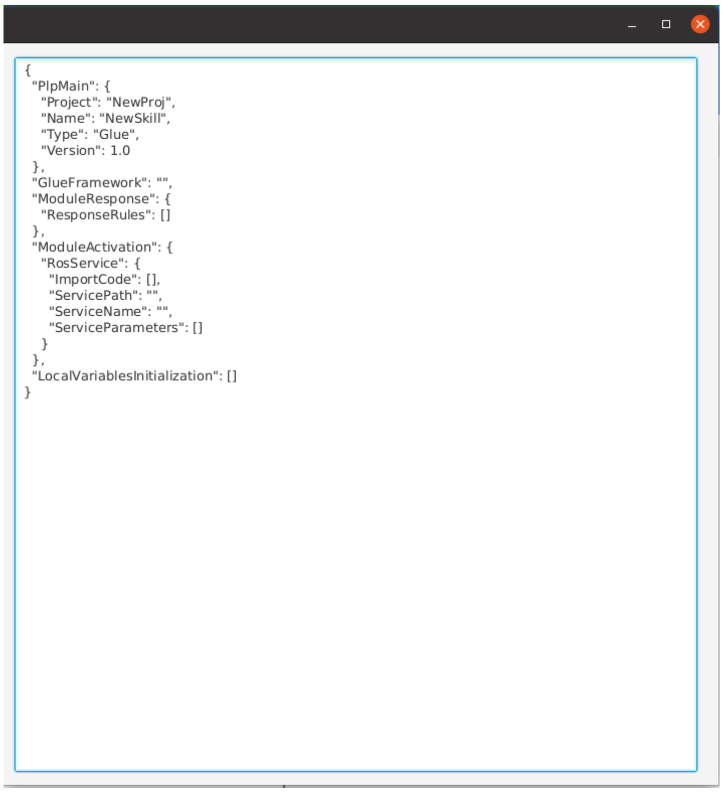


4.
5. You can press on the preview SD JSON button and a new window will open with a text box containing the JSON structure of the SD file. Exit by pressing the X button.



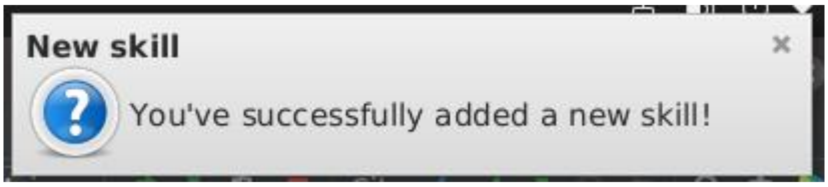
```
{
  "PlpMain": {
    "Project": "NewProj",
    "Name": "NewSkill",
    "Type": "PLP",
    "Version": 1.0
  },
  "GlobalVariableModuleParameters": [],
  "PossibleParametersValue": [],
  "Preconditions": {
    "GlobalVariablePreconditionAssignments": [],
    "PlannerAssistancePreconditionsAssignments": [],
    "ViolatingPreconditionPenalty": -1.0
  },
  "DynamicModel": {
    "NextStateAssignments": []
  }
}
```

6. Same for AM, you can press Preview AM JSON Button and a new window will open with a text box containing the am JSON structure. Exit using the X button.



```
{
 "PlpMain": {
   "Project": "NewProj",
   "Name": "NewSkill",
   "Type": "Glue",
   "Version": 1.0
 },
 "GlueFramework": "",
 "ModuleResponse": {
   "ResponseRules": []
 },
 "ModuleActivation": {
   "RosService": {
     "ImportCode": [],
     "ServicePath": "",
     "ServiceName": "",
     "ServiceParameters": []
   }
 },
 "LocalVariablesInitialization": []
}
```

7. Once you press the Add Skill button, the following pop up message will appear.



New skill

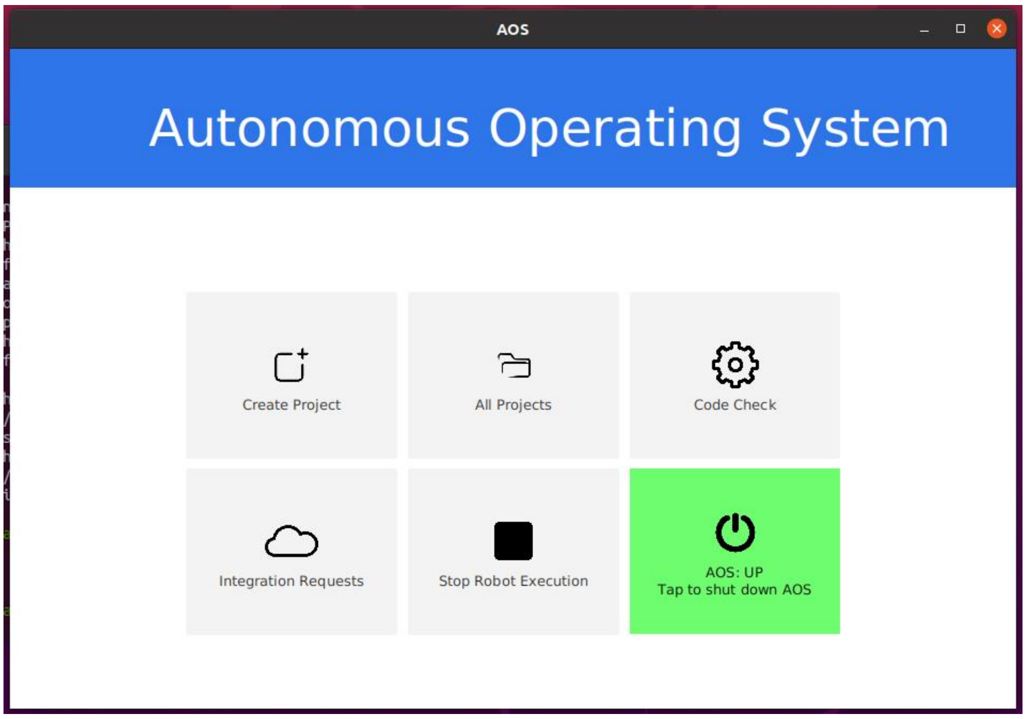
You've successfully added a new skill!

8. Exit By pressing the Back button.

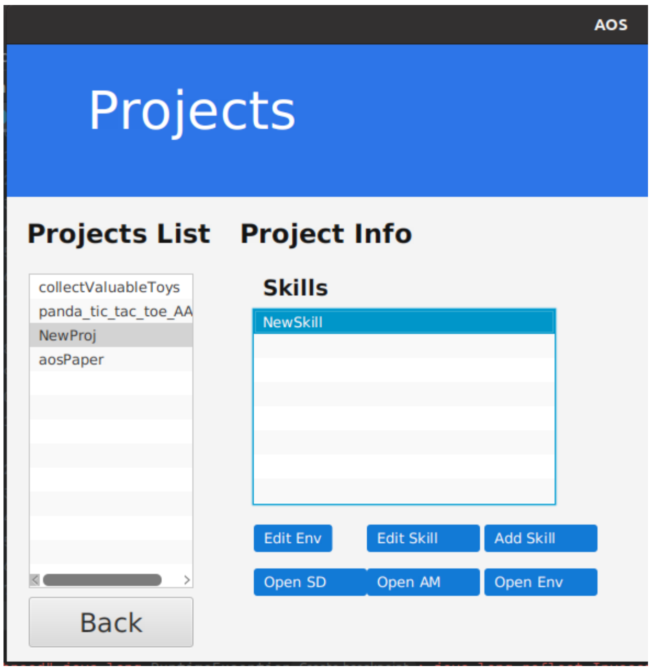
Edit Skill

Goal: Edit skills after they were already created (or loaded into the projects folder) to ensure maximal flexibility when working with the system.

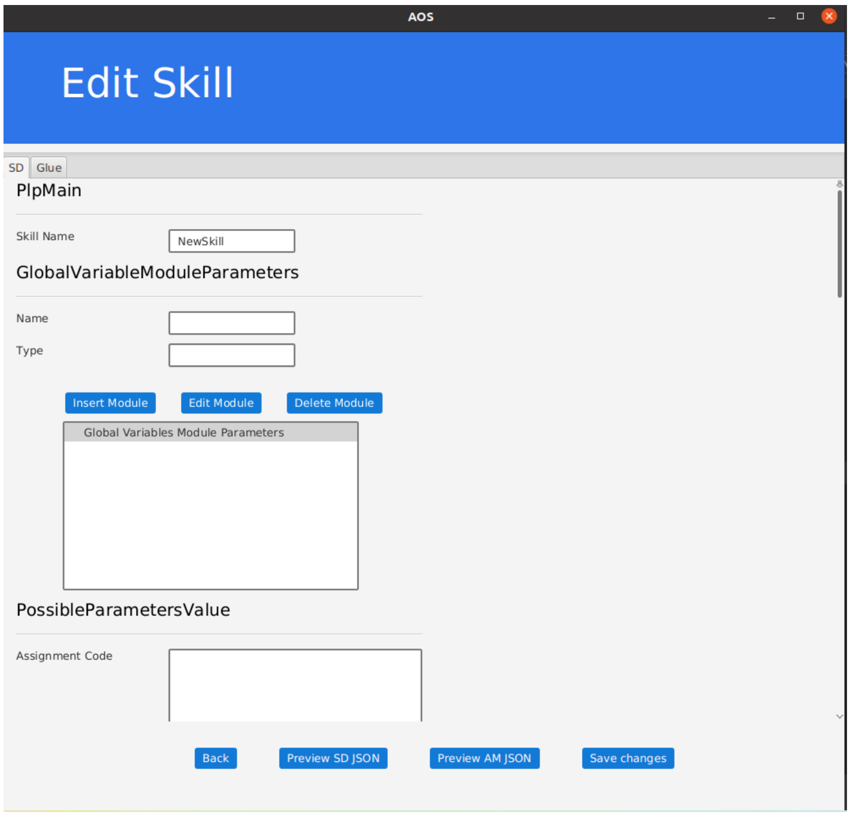
1. In the Home Page press All Project Button.



2. A new Window will open, in the project list select the project to edit skill from, select the skill in the skill list and press edit skill.



3. A new Window will open, with a form to change the SD and Am.



PlpMain

Skill Name: NewSkill

GlobalVariableModuleParameters

Name:

Type:

Global Variables Module Parameters

PossibleParametersValue

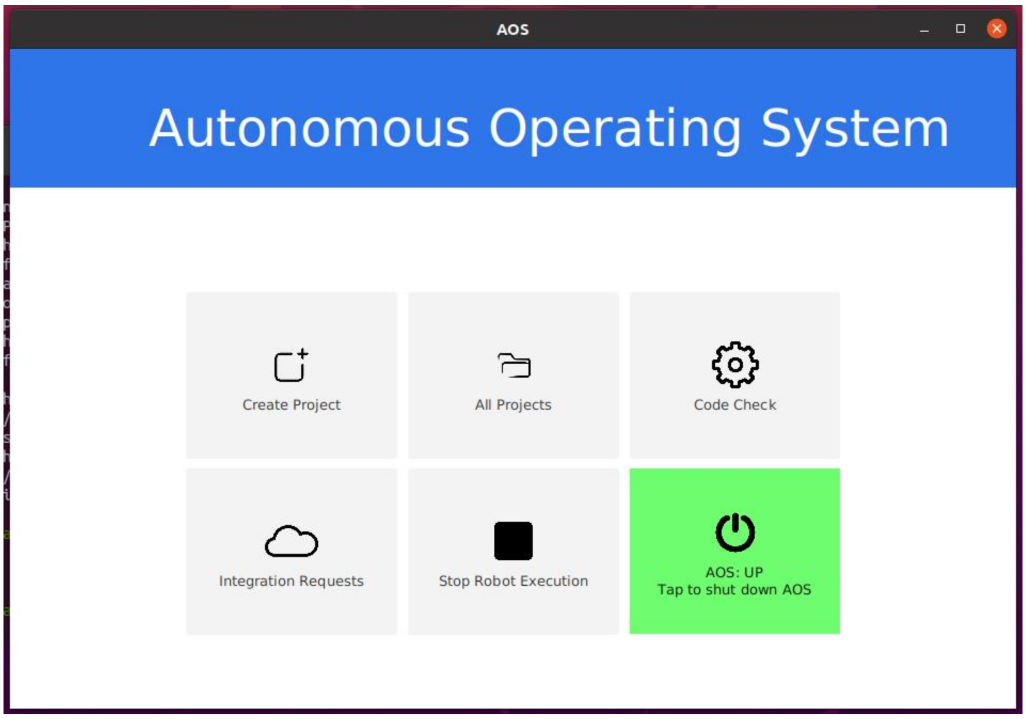
Assignment Code:

4. click the save button to save the changes. A pop-up message will appear.
5. Press the back button to return to the project page.

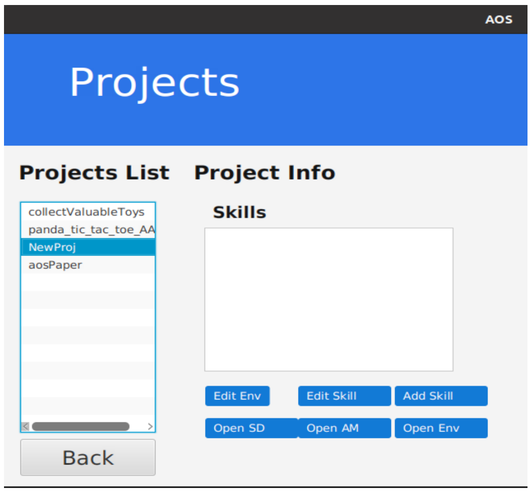
Edit Environment File

Goal: Edit env file after it was already created (or loaded into the projects folder) to ensure maximal flexibility when working with the system.

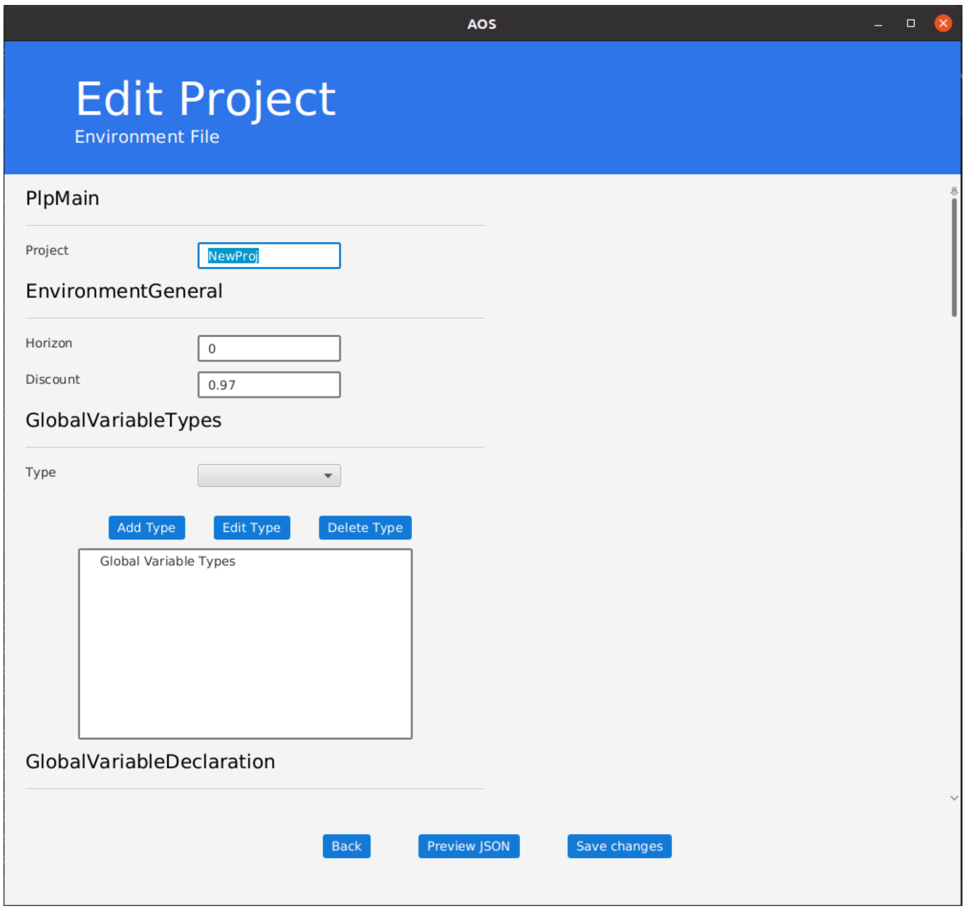
1. In the Home Page press All Project Button.



2. A new Window will open, in the project list select the project to edit the environment file, and press on the Edit Env button.



3. A new window will be open with form to the change the environment file.



AOS

Edit Project

Environment File

PlpMain

Project: NewProj

EnvironmentGeneral

Horizon: 0

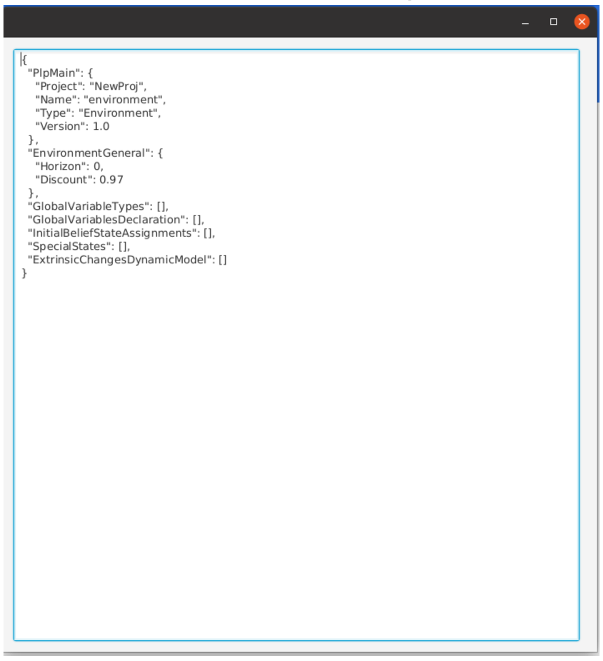
Discount: 0.97

GlobalVariableTypes

Type:

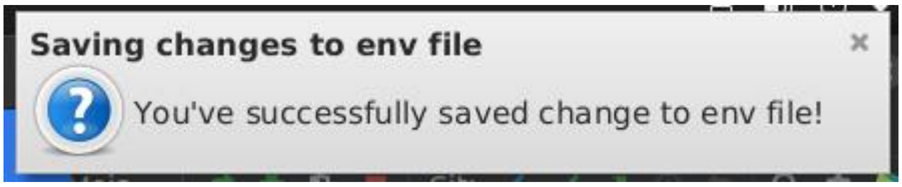
GlobalVariableDeclaration

4. You can press the Preview JSON button. A new window will open with textbox containing the structure of the ENV file. Exit using the X button.



```
{
  "PlpMain": {
    "Project": "NewProj",
    "Name": "environment",
    "Type": "Environment",
    "Version": 1.0
  },
  "EnvironmentGeneral": {
    "Horizon": 0,
    "Discount": 0.97
  },
  "GlobalVariableTypes": [],
  "GlobalVariablesDeclaration": [],
  "InitialBeliefStateAssignments": [],
  "SpecialStates": [],
  "ExtrinsicChangesDynamicModel": []
}
```

5. Click Save Changes, A pop-up message will appear.



Saving changes to env file

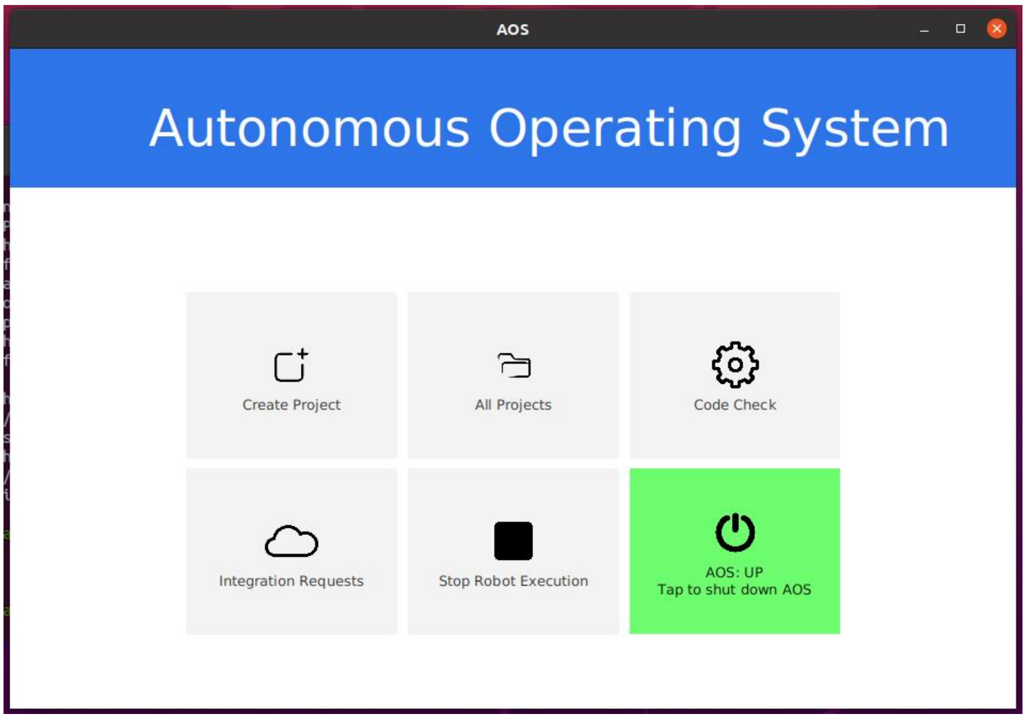
You've successfully saved change to env file!

6. Press back button to return to the projects window.

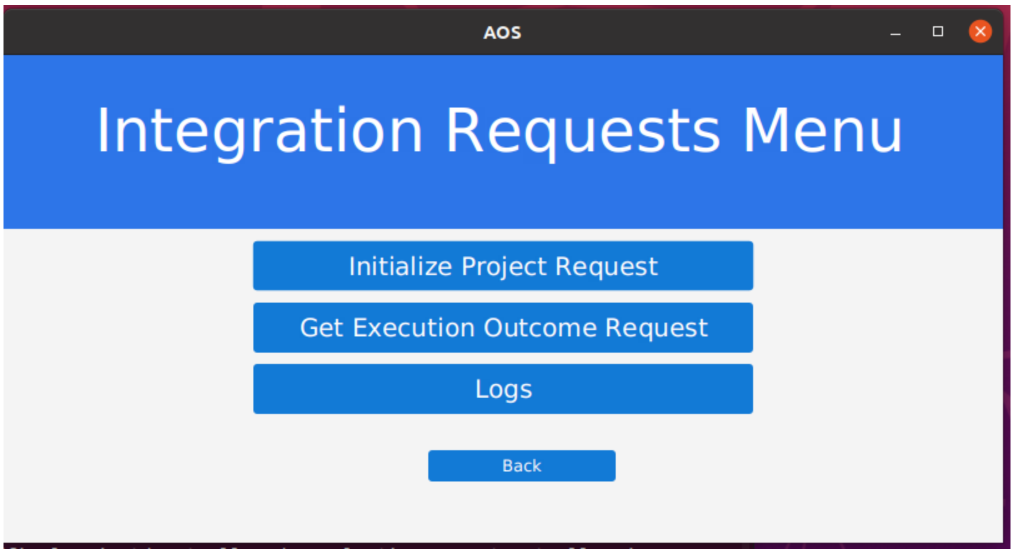
Project Initialization Request

Goal: send project initialization request to the server, in order to run the robot (in inner simulation for example)

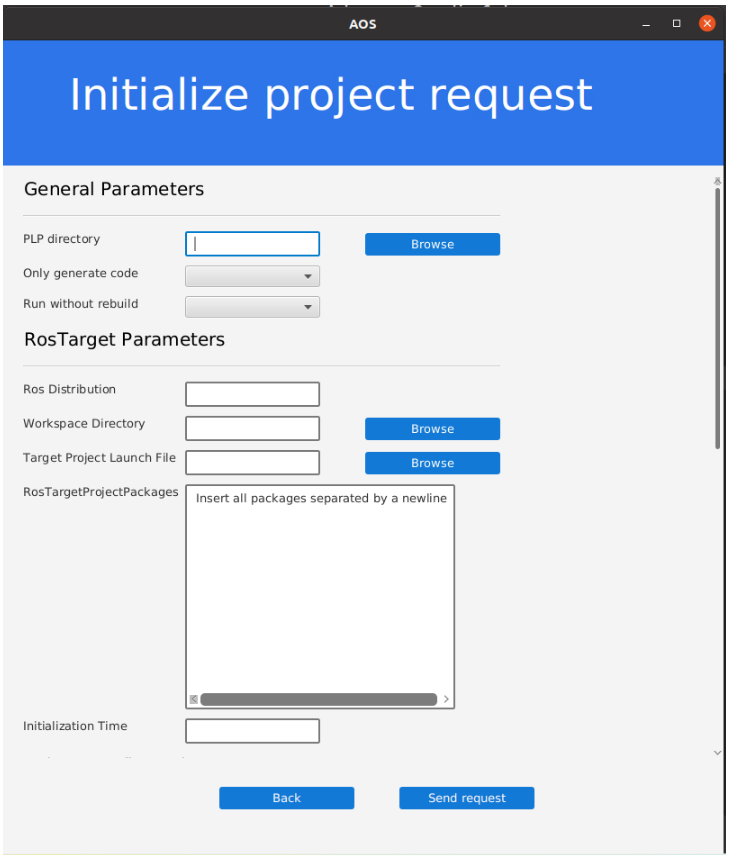
1. In the main screen make sure the server is up, if not Activate it.
2. Press on the Integrations request button as seen below.



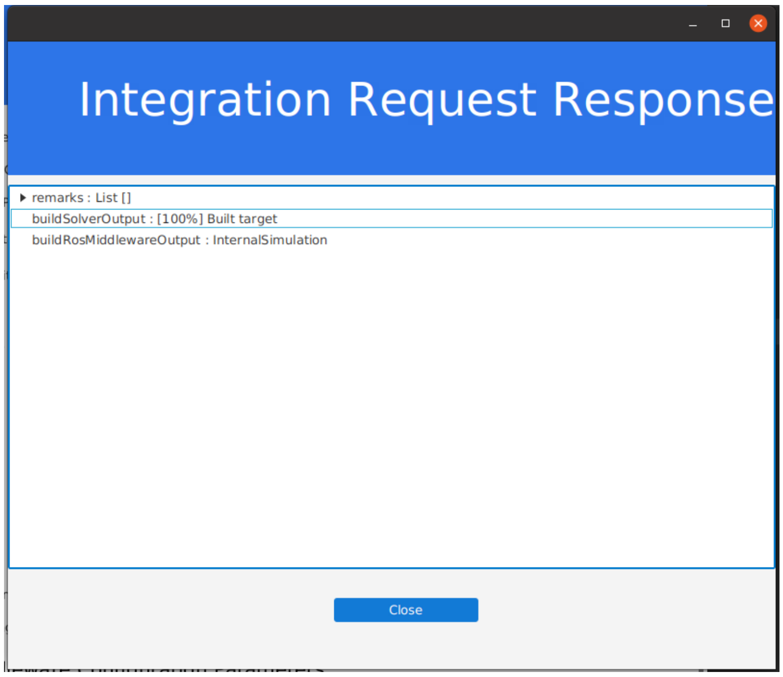
3. The window will change to a list with the integration requests press on the Initialize project request.


AOS
Integration Requests Menu
Initialize Project Request
Get Execution Outcome Request
Logs
Back

4. The windows should change to the initialization request form as seen below.



5. Fill the form with the desired parameters. Then, press 'Send Request' button.
6. A new window should open with the results. In the buildSolverAction it should state 100% as seen below in case of success build.



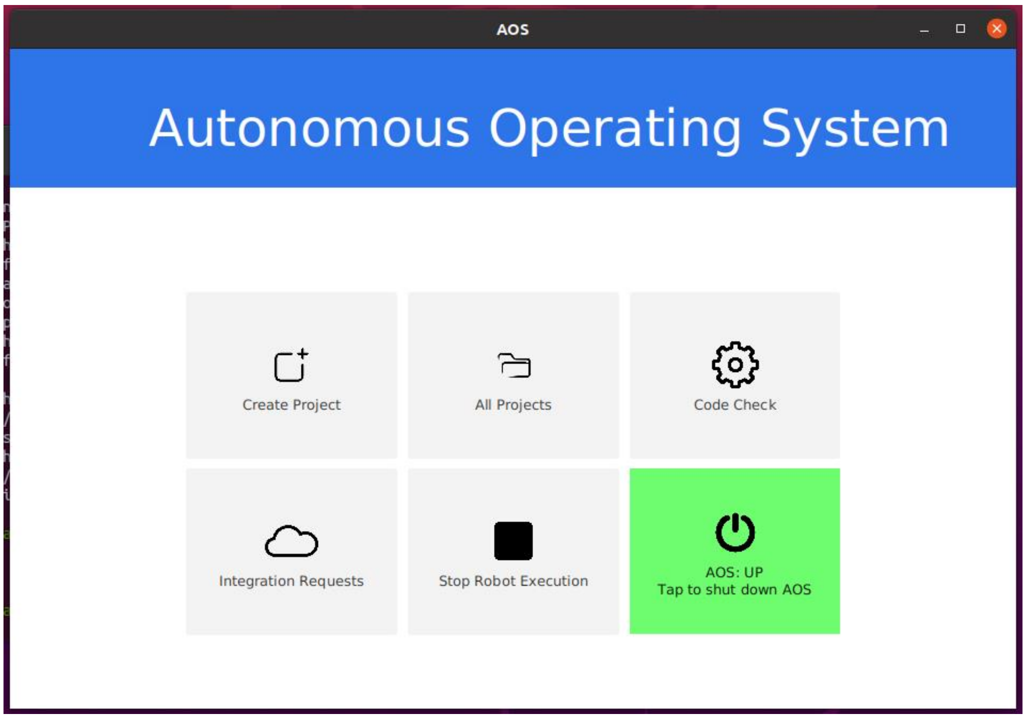
```
▸ remarks : List []
buildSolverOutput : [100%] Built target
buildRosMiddlewareOutput : InternalSimulation
```

7. Press the Close button and make sure the window has closed.
8. Press on the back button on the form windows to return to the Request list window.

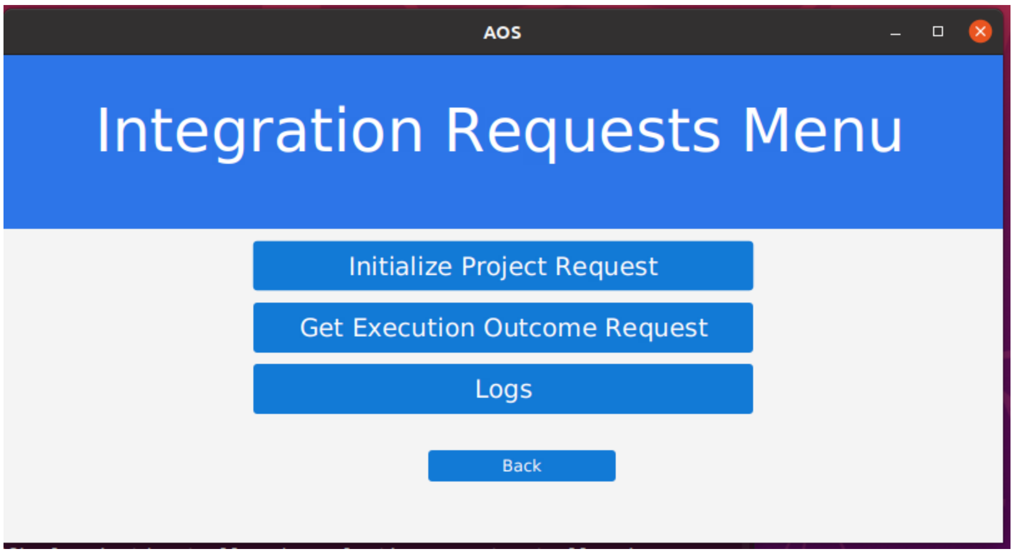
Send a Get Execution Request

Goal: get the belief state of the robot for debugging purposes, visualization and more.

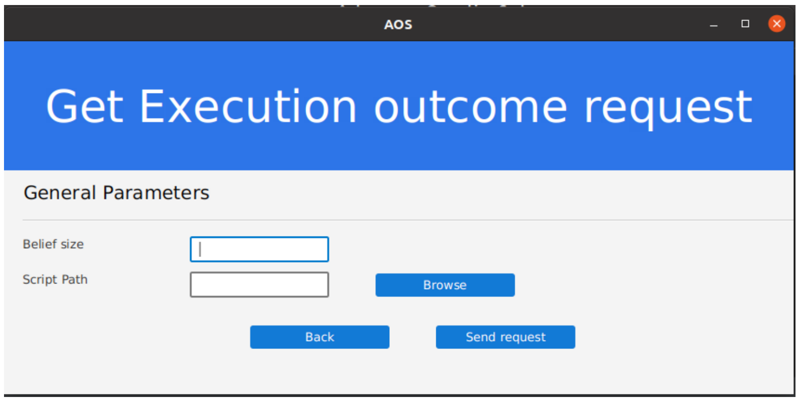
1. Make sure the server is up and initialization project request ran successfully.
2. Press on the Integration requests button in the home page.



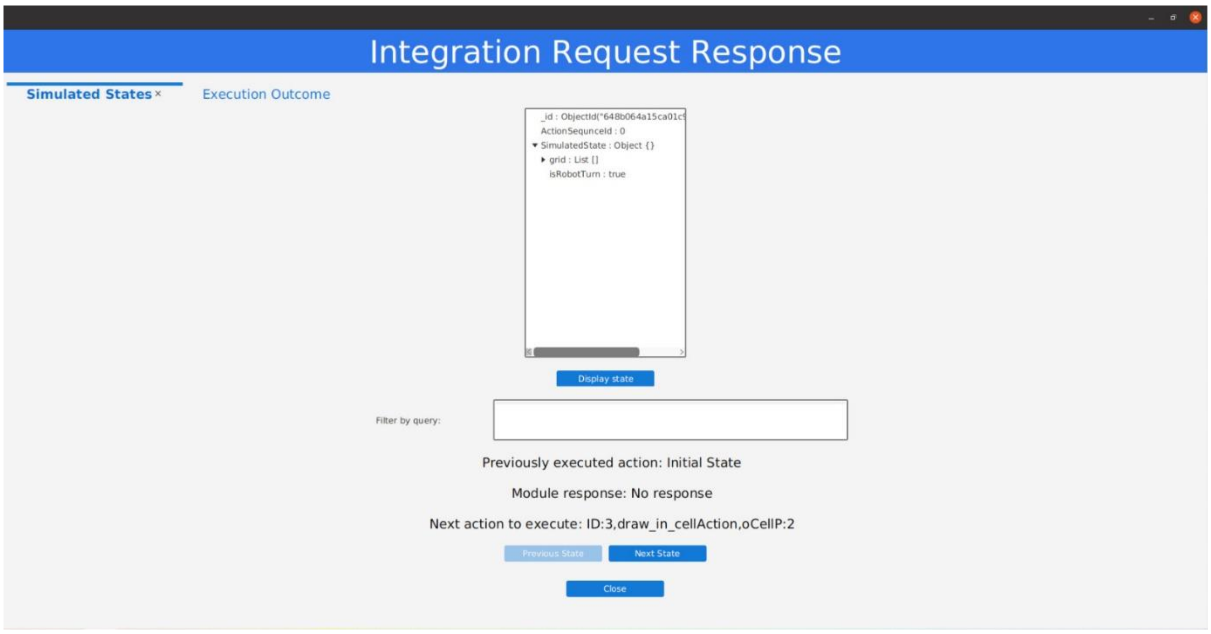
3. In the Integration request menu choose the Get Execution outcome.


AOS
Integration Requests Menu
Initialize Project Request
Get Execution Outcome Request
Logs
Back

4. In the form window of the request enter the desired parameters. If you want to display the robot states, make sure to upload a .py script in the following structure.



5. Press on Send Request.
6. A new window should open with the robot initial belief state. For example:
   - Next/previous state buttons: for navigating between states.
   - Display state – to show graphical representation of the states.
   - Close – close the window and return to the window in step 4.



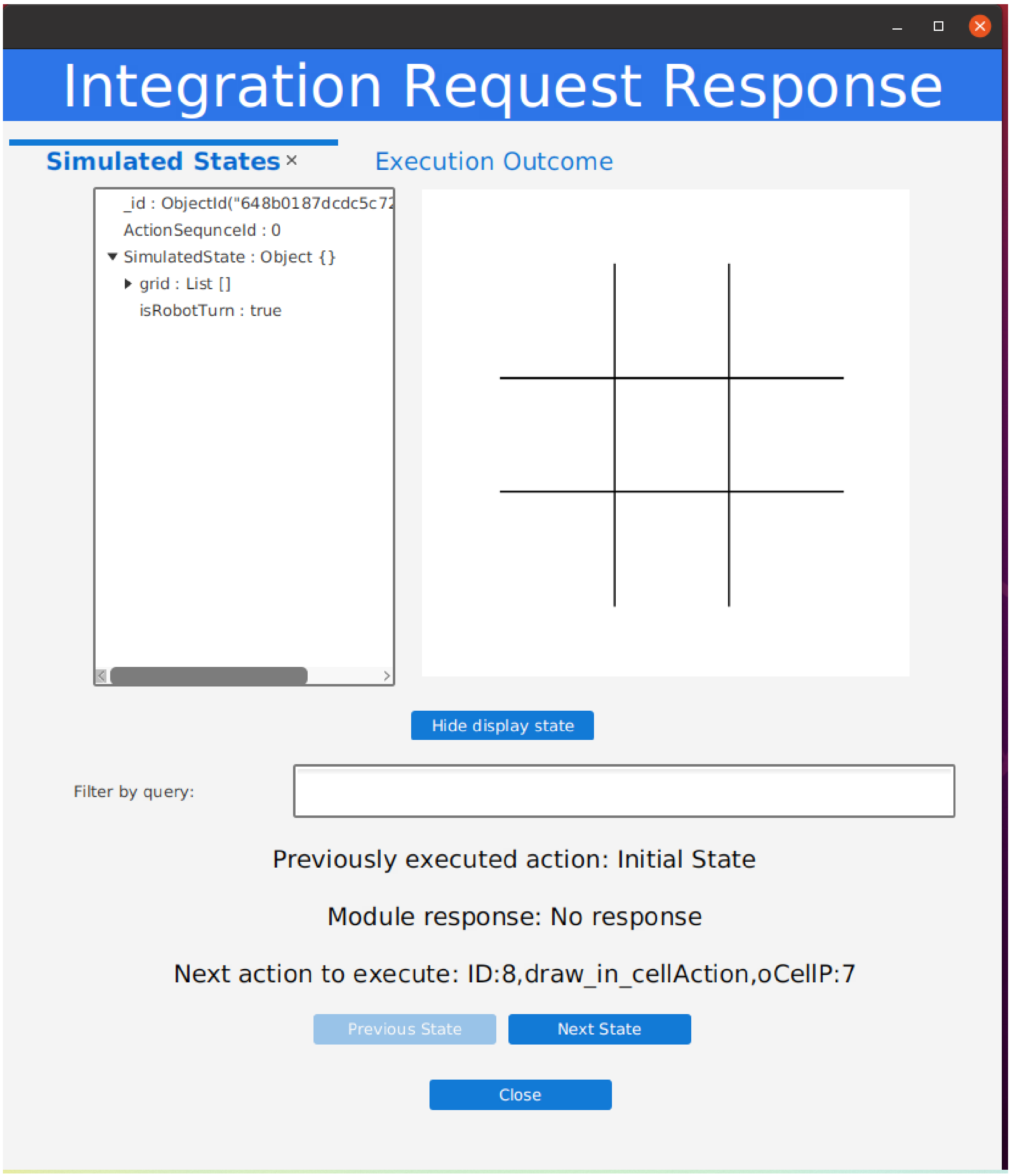
```
_id : ObjectId("648b064a15ca01c
ActionSequnceId : 0
▾ SimulatedState : Object {}
  ▸ grid : List []
    isRobotTurn : true
```

Previously executed action: Initial State

Module response: No response

Next action to execute: ID:3,draw_in_cellAction,oCellP:2

7. When pressing on Show Display, the window will display the current belief state of the robot in graphical representation. For example:



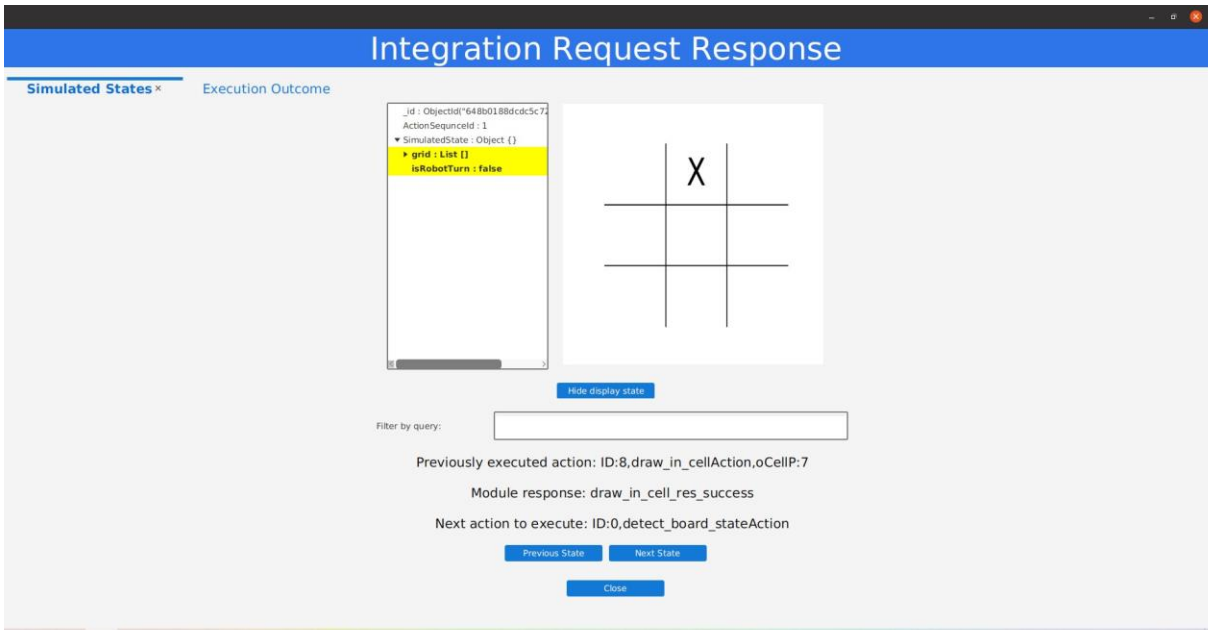
```
_id : ObjectId("648b0187dcdc5c72
ActionSequnceId : 0
▼ SimulatedState : Object {}
  ▶ grid : List []
    isRobotTurn : true
```

Previously executed action: Initial State

Module response: No response

Next action to execute: ID:8,draw_in_cellAction,oCellP:7

8. When pressing on Next State button the lines on grid and isRobotTurn should be highlighted ,and the display will update to the current belief state. For example:



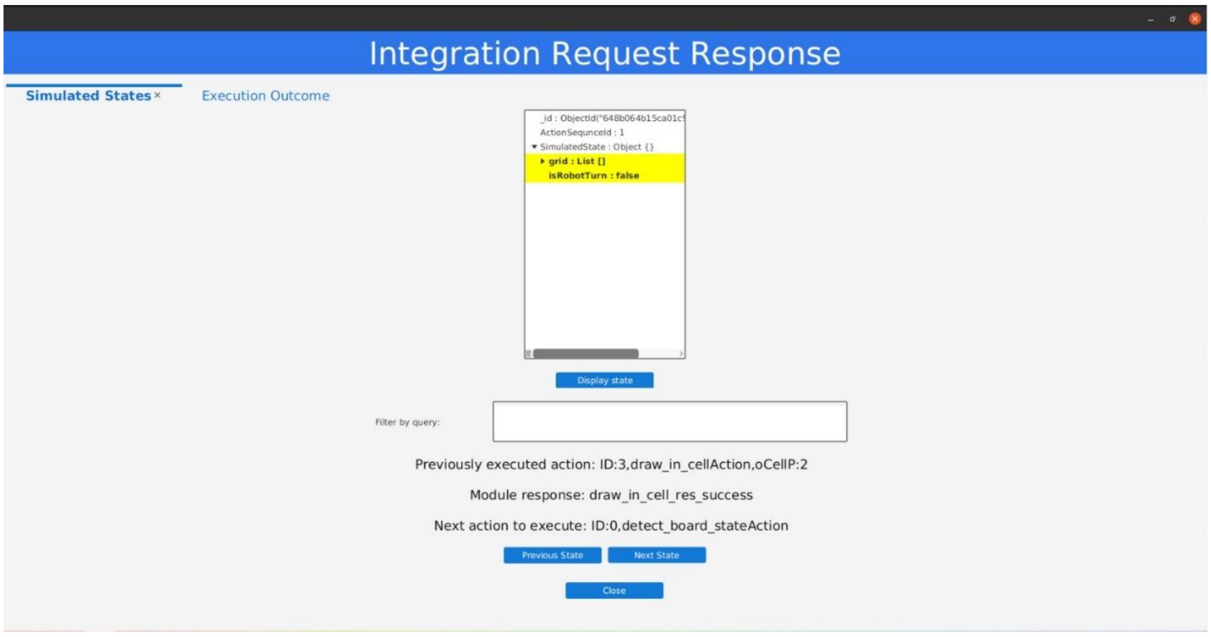
```
_id : ObjectId("648b0188dcdc5c72
ActionSequnceId : 1
▾ SimulatedState : Object {}
  ▸ grid : List []
    isRobotTurn : false
```

Previously executed action: ID:8,draw_in_cellAction,oCellP:7

Module response: draw_in_cell_res_success

Next action to execute: ID:0,detect_board_stateAction

9. When pressing on Hide Display we return to see the raw state without the image.



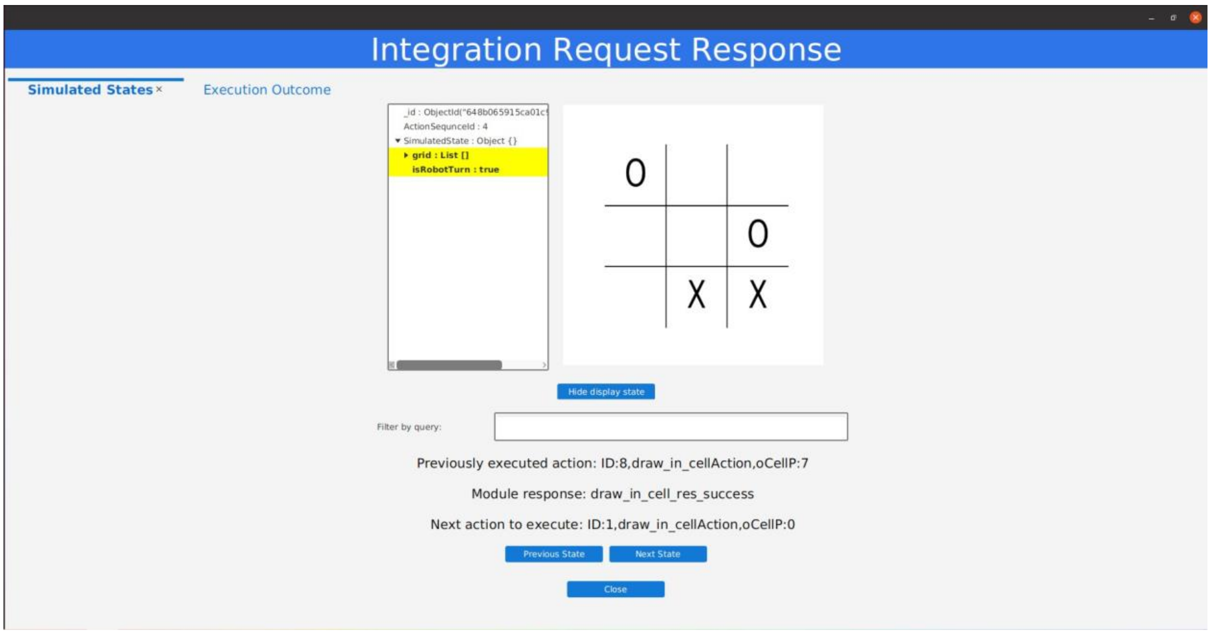
```
_id : ObjectId("648b064b15ca01c
ActionSequnceId : 1
▾ SimulatedState : Object {}
  ▸ grid : List []
    isRobotTurn : false
```

Previously executed action: ID:3,draw_in_cellAction,oCellP:2

Module response: draw_in_cell_res_success

Next action to execute: ID:0,detect_board_stateAction

10. Navigate between other belief states updated the image. For exmale:



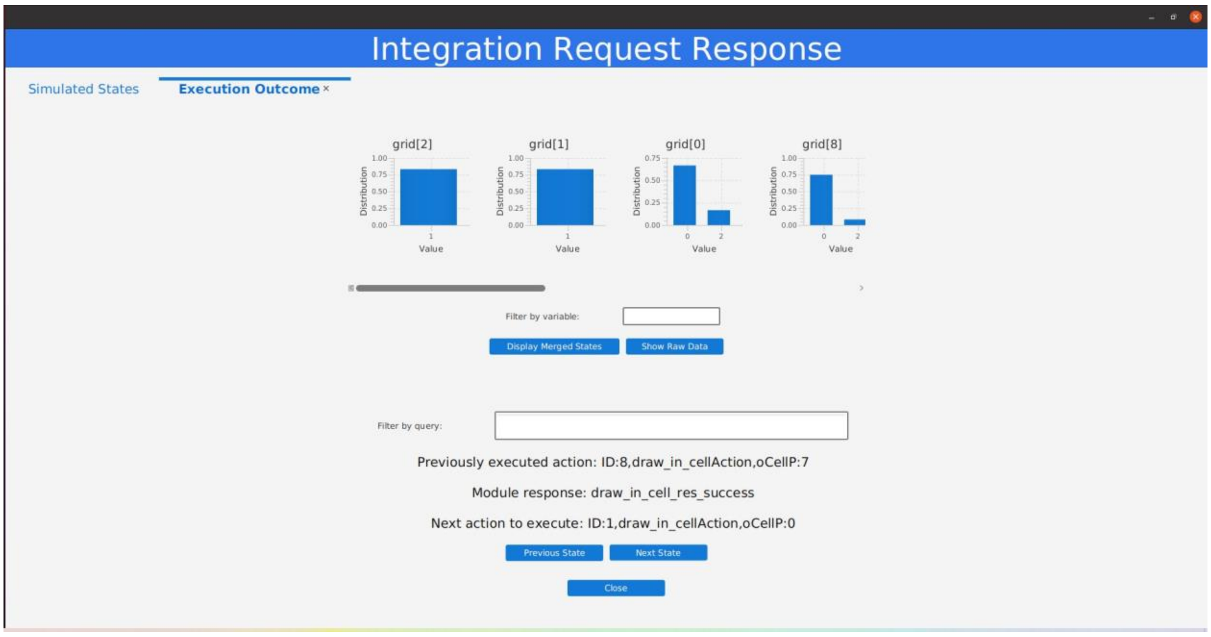
```
_id : ObjectId("648b065915ca01c
ActionSequnceId : 4
▾ SimulatedState : Object {}
  ▸ grid : List []
    isRobotTurn : true
```

Previously executed action: ID:8,draw_in_cellAction,oCellP:7

Module response: draw_in_cell_res_success

Next action to execute: ID:1,draw_in_cellAction,oCellP:0

11. When clicking on the Execution Outcome tab, the layout should change to the tab the presents a distribution of the environment variables to their optional values. For example:

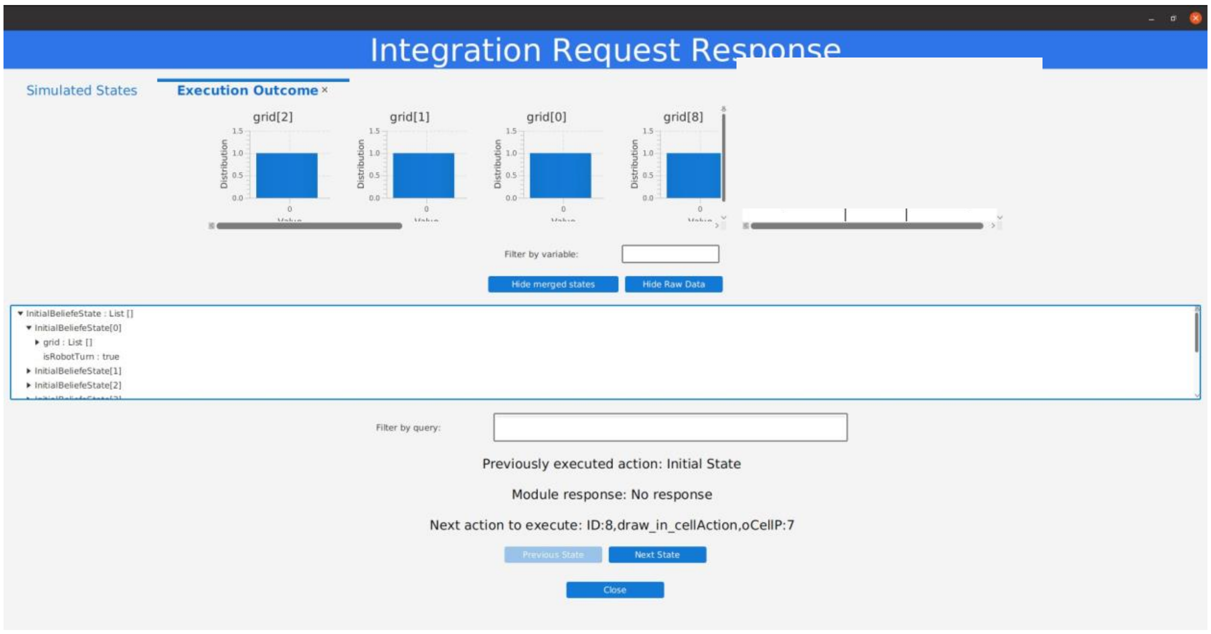


Previously executed action: ID:8,draw_in_cellAction,oCellP:7

Module response: draw_in_cell_res_success

Next action to execute: ID:1,draw_in_cellAction,oCellP:0

12. when clicking on the Show Raw Data a text box should open, presenting the raw belief state. For example:



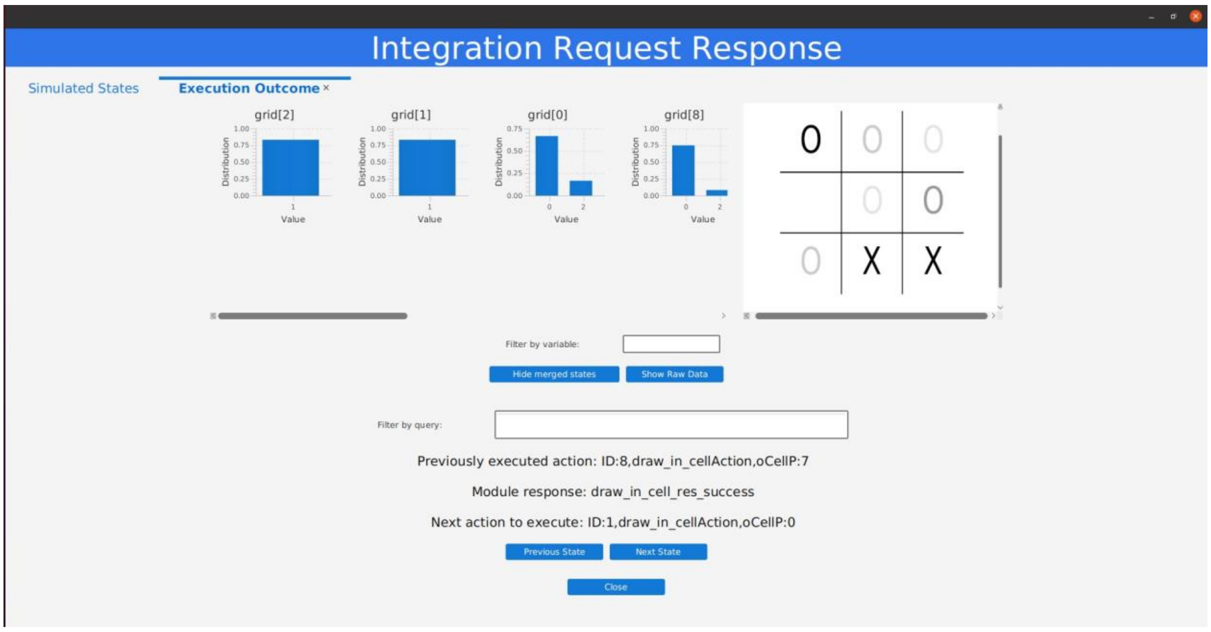
```
▾ InitialBeliefeState : List []
  ▾ InitialBeliefeState[0]
    ▸ grid : List []
      isRobotTurn : true
  ▸ InitialBeliefeState[1]
  ▸ InitialBeliefeState[2]
```

Previously executed action: Initial State

Module response: No response

Next action to execute: ID:8,draw_in_cellAction,oCellP:7

13. When pressing on Display Merged States, an image representing the indeterministic state will be shown. For example:



Previously executed action: ID:8,draw_in_cellAction,oCellP:7

Module response: draw_in_cell_res_success

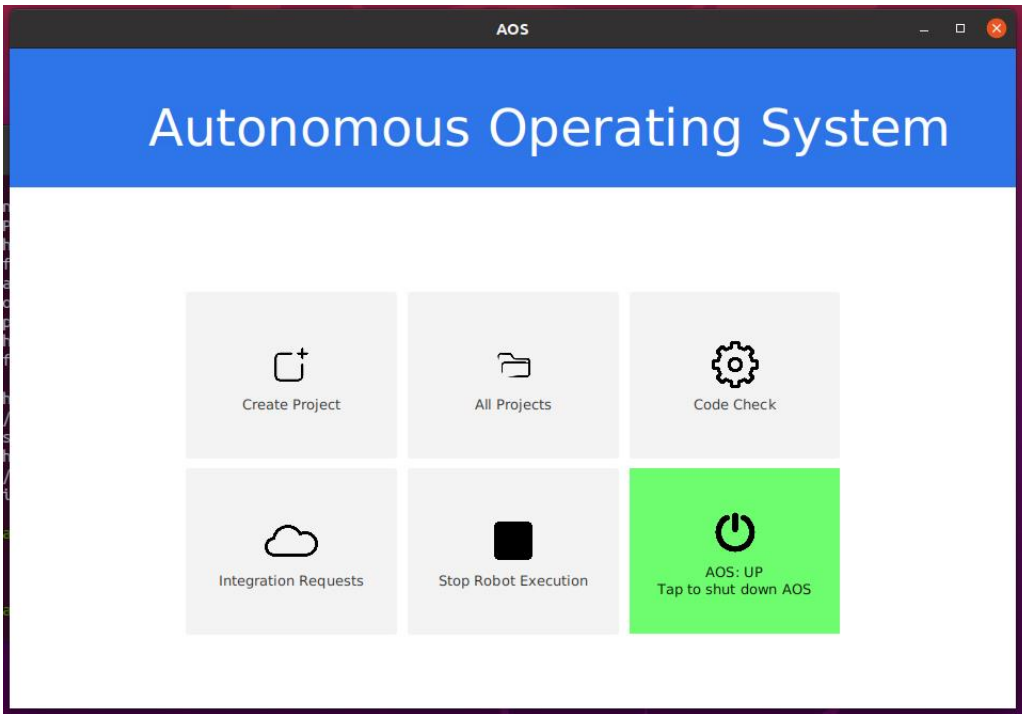
Next action to execute: ID:1,draw_in_cellAction,oCellP:0

14. You can also filter the desired environment variable in the 'filter by variable' textbox.
15. In the Get Execution request press on back button to return to the main window.

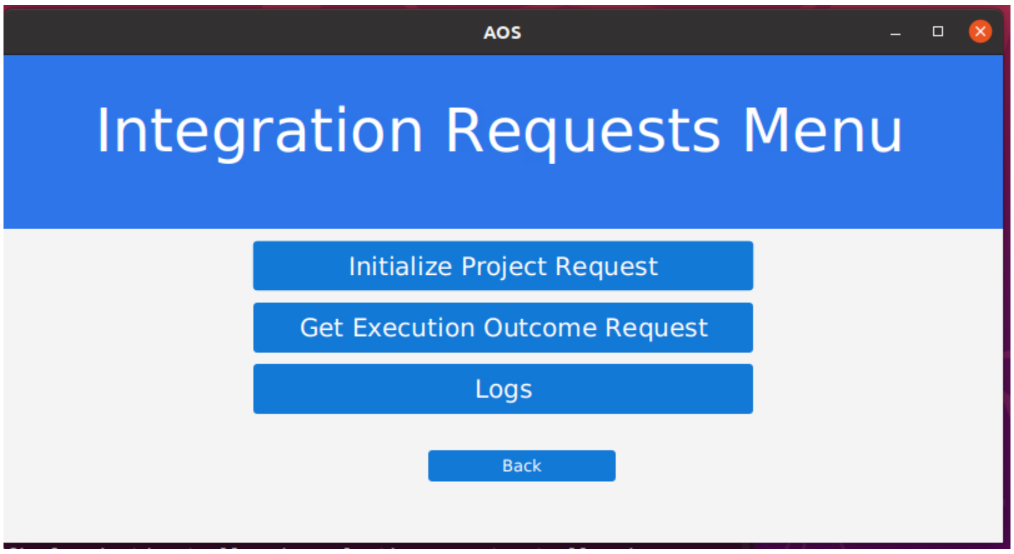
Request to show Server’s Logs

Goal: get different logs from to server to help you debug it's actions.

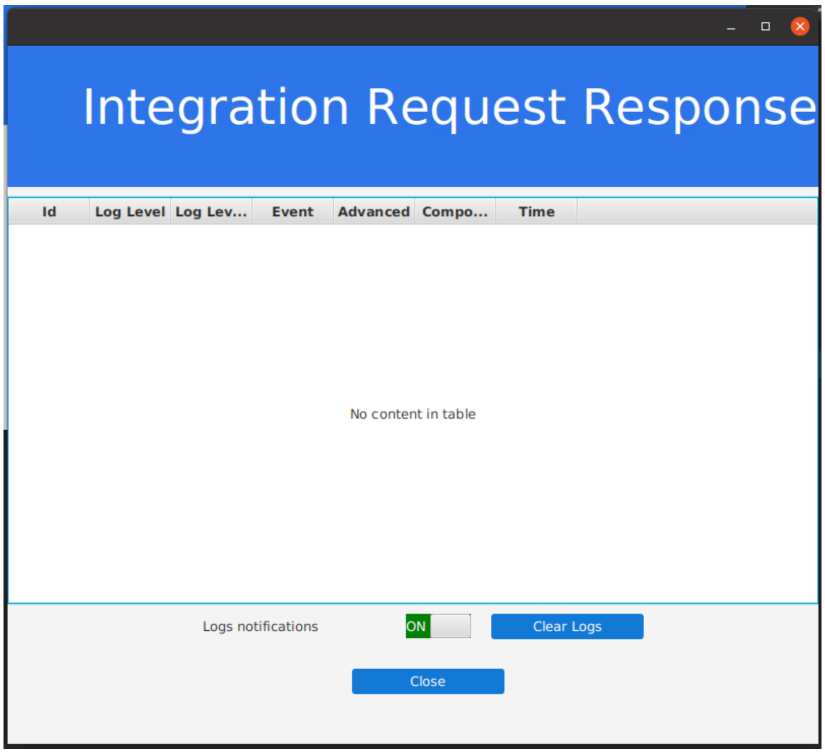
1. Make sure the server is up, and initialization project ran successfully.
2. Press on the Integration requests button in the home page.



3. In the Integration request menu choose Logs.


AOS
Integration Requests Menu
Initialize Project Request
Get Execution Outcome Request
Logs
Back

4. A new window should open up as bellow

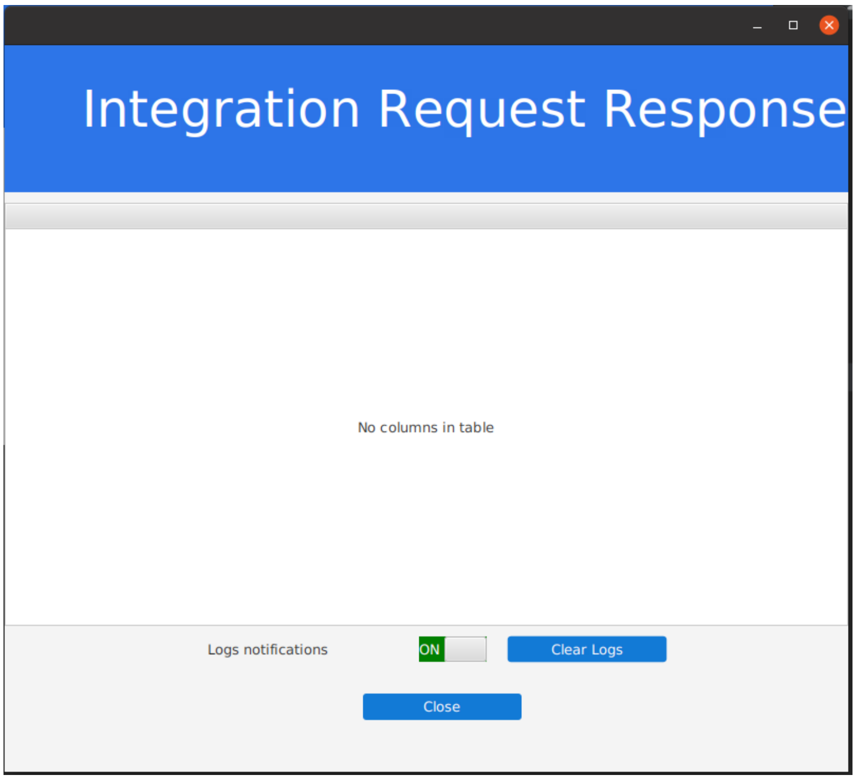


Integration Request Response

Id	Log Level	Log Lev...	Event	Advanced	Compo...	Time

No content in table

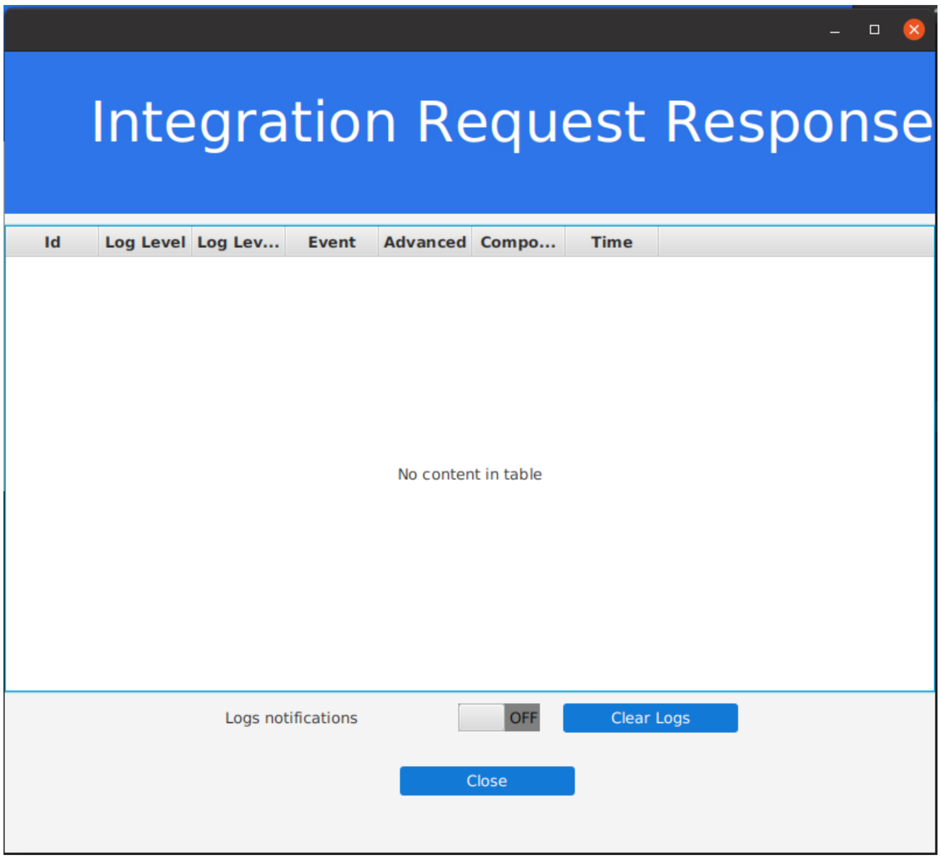
5. Press on Clear Logs button, a pop up be shown in the top corner of the screen and the window should look like



Integration Request Response

No columns in table

6. Press on the logs Notification radio button to switch logs notification off.



Integration Request Response

Id	Log Level	Log Lev...	Event	Advanced	Compo...	Time

No content in table

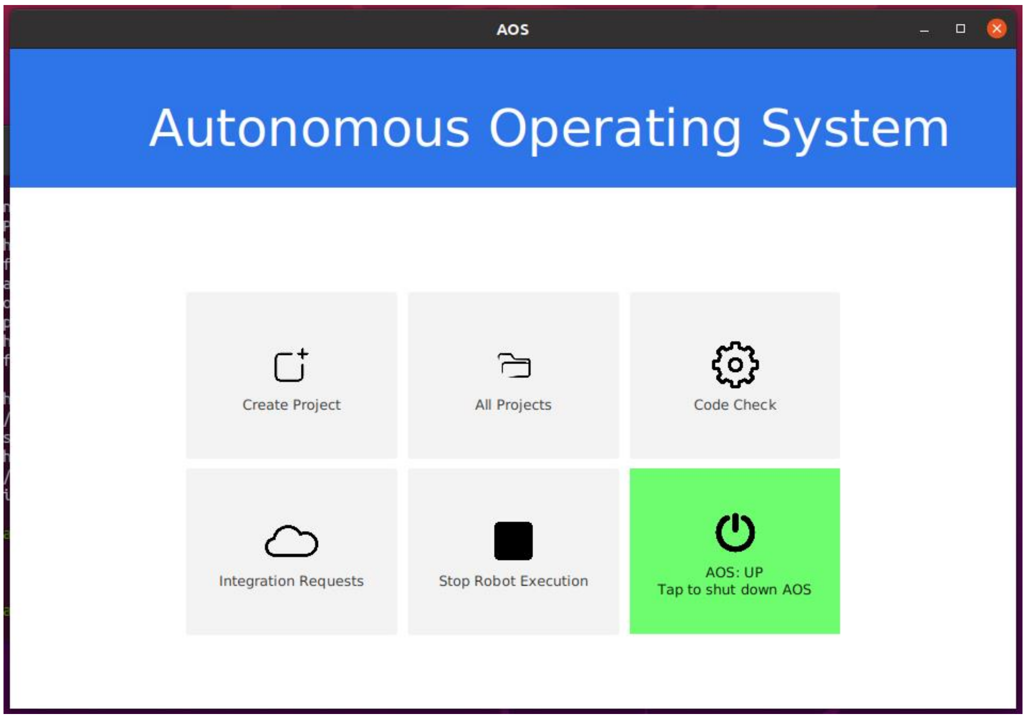
Logs notifications OFF

7. Press on the Close button to close the window.

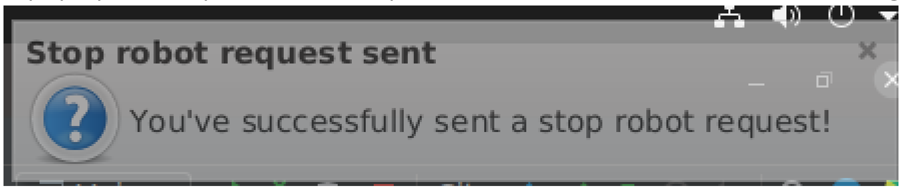
Request to Stop the Robot

Goal: stop the robot activation if needed.

1. Make sure the server is up and initialization project request ran successfully.
2. Press on Stop Robot Execution button as shown below.



3. A pop up will be placed in the top corner of the screen with success massage



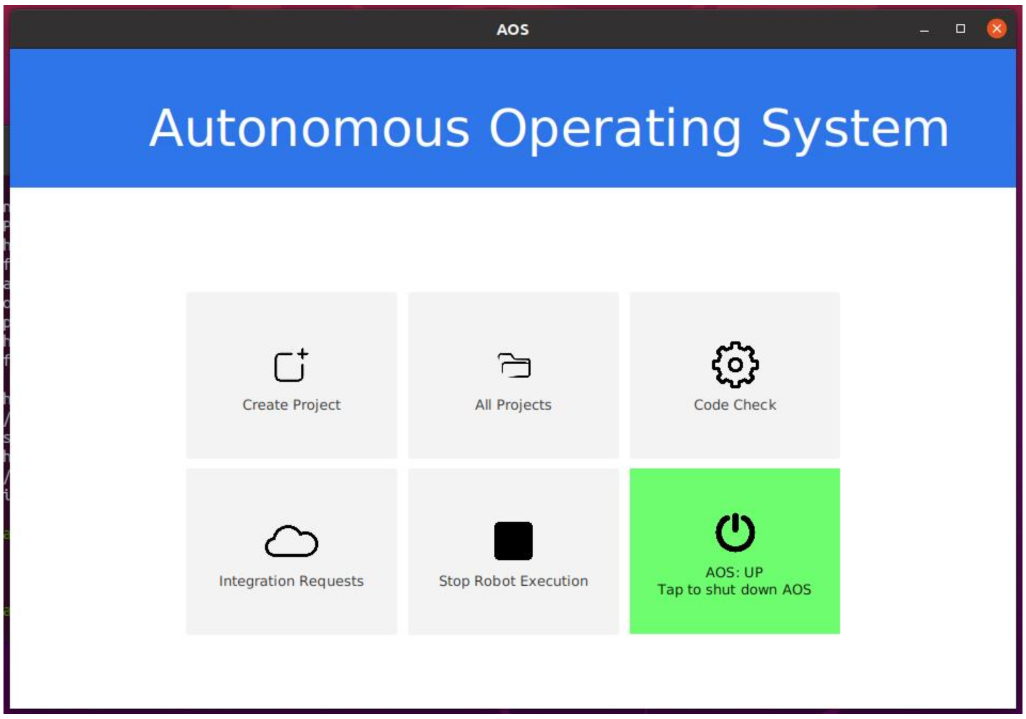
Stop robot request sent

You've successfully sent a stop robot request!

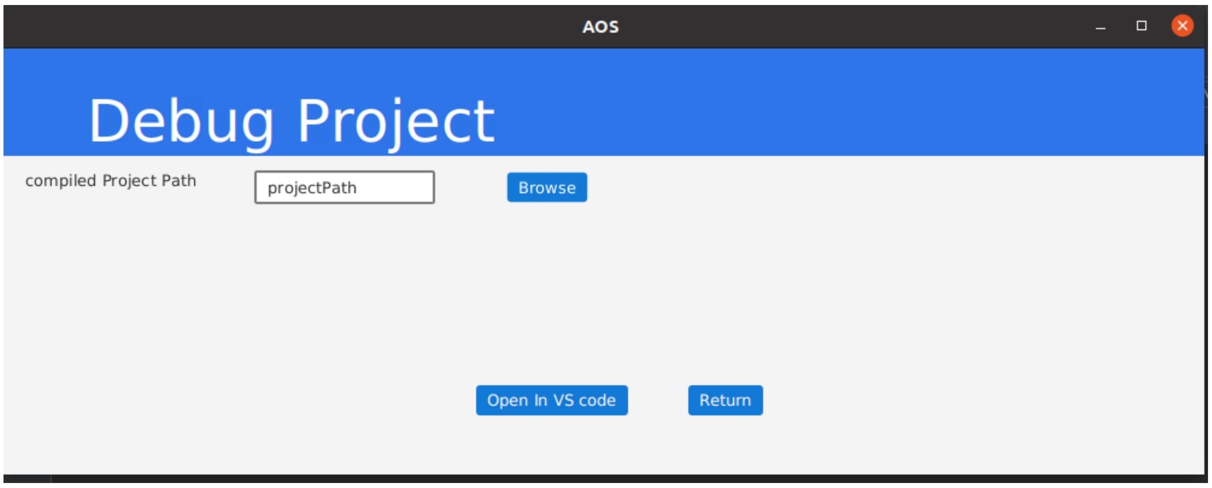
Request the Generated Code

Goal: when we send request 'initialize project', the AOS server compiles all the given project files and create a generated output code. As a developer, you can see the generated code.

1. In the home page press on the code check button

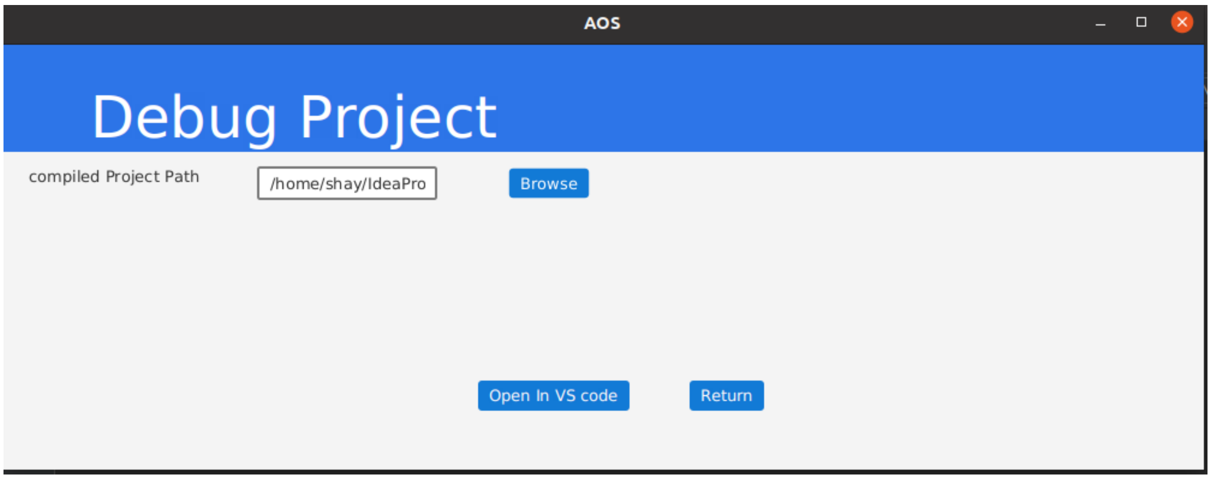


2. In the new window, press the browse button.



3. Navigate to the project compiled file and press open.

4. The textbox of the compiled project path should now contain the absolute path of the file.



compiled Project Path /home/shay/IdeaPro

5. Press on the Open in VS code button.
6. A Visual Studio Code window should open and will present the generated code.