**FireApp**

**Minizinc Artificially Intelligent Scheduler**

**Handover Documentation**

Documentation by Phillip La

**Table of Contents**

**Introduction ……………………………………………………………………………….. 3**

**Skills Required for Feature Contribution ……………………………………………. 3**

**Documentation for MiniZinc AI Modelling**

**Development Tools Needed ……………………………………………………. 4**

**Basic Workflow ……………………………………………………………..… 4 - 8**

**Scheduler Parameters …………………………………………………………... 9**

**A …………………………………………………………………………….. 9**

**P …………………………………………………………………………….. 9**

**V …………………………………………………………………………….. 9**

**Q …………………………………………………………………………….. 9**

**C …………………………………………………………………………… 10**

**shiftcoefficient ………………………………………………………….. 10**

**qualrequirements ………………………………………………………. 10**

**rolerequirements ……………………………………………………….. 10**

**posrequirements ……………………………………………………….. 10**

**qualability ………………………………………………………………… 11**

**roleablity …………………………………………………………………. 11**

**availability ………………………………………………………………... 11**

**clashes ……………………………………………………………………. 11**

**Scheduler Output - assignment ……………………………………………… 11**

**Introduction**

One of FireApp’s most essential features is the artificially intelligent scheduler that automatically assigns volunteers to shifts based on their availability, qualified roles, skills and number of previous shifts. The scheduler has been continuously improved and rebuilt over the years. The most previous change was in 2022, semester 2, where the scheduler was entirely re-implemented to consider more sophisticated requirements.

**Skills Required for Feature Contribution**

The scheduler feature requires two major technical skills; basic proficiency in MiniZinc for modelling the algorithm, and a basic proficiency in Python and REST APIs for integrating the AI model to the Python backend.

Though one person could work on the schedule individually, this is not recommended. It is recommended that there are at least two different developers contributing to the feature. One developer should work on modelling the AI using MiniZinc, and a second developer should work on integrating the model to the backend using Python.

In the FireApp team of semester 2, 2022, Phillip La specialised on the MiniZinc and Hang Su specialised on the Python backend. The rest of the documentation will provide both objective advice and tips subjective to these two team members’ personal experiences.

There is a slightly steep learning curve with MiniZinc, requiring a couple days of intense solving each iteration. However, if the developer has previously taken ANU’s Logic course (COMP2620), then constraint programming should not be an entirely new concept.

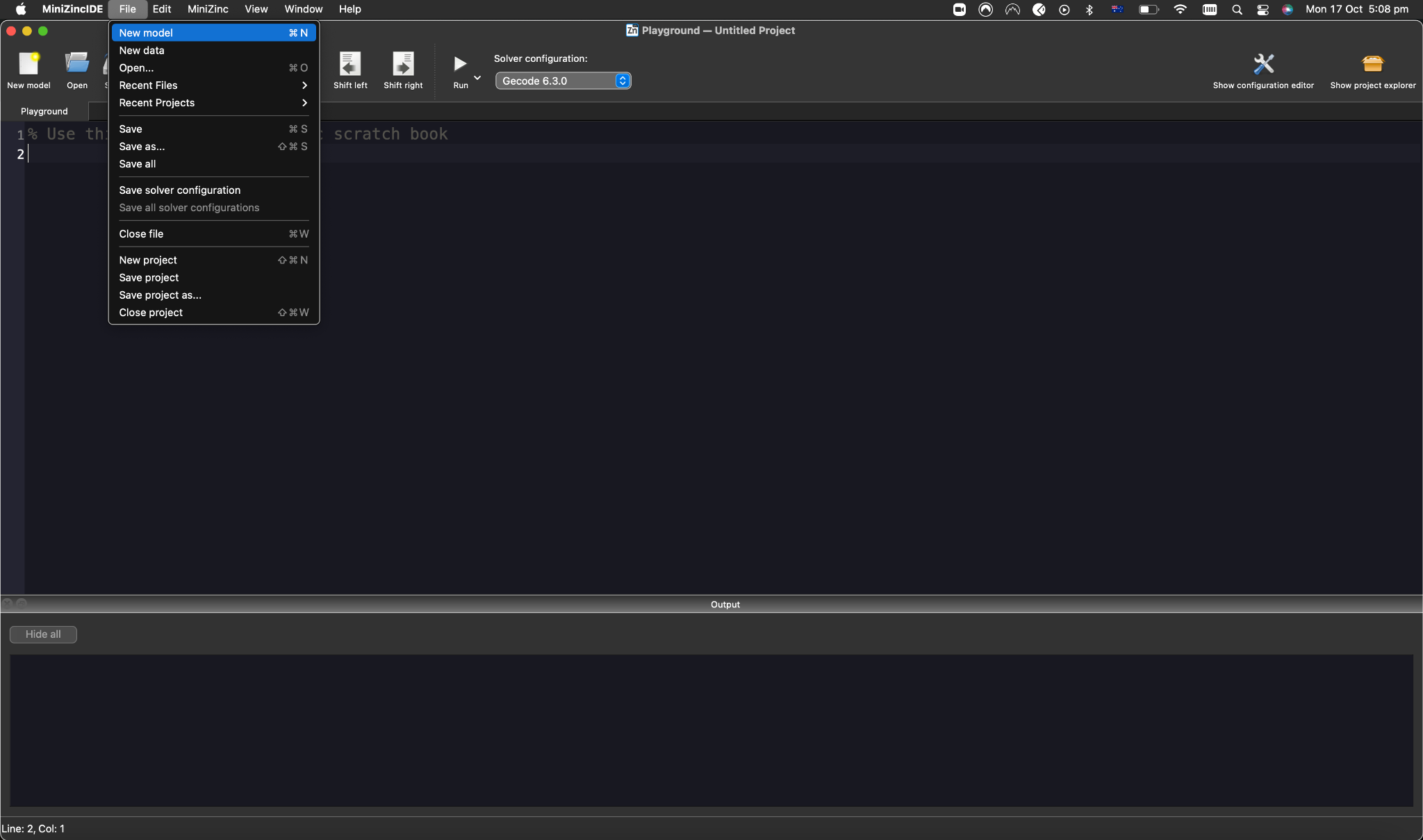
**Development Tools Needed**

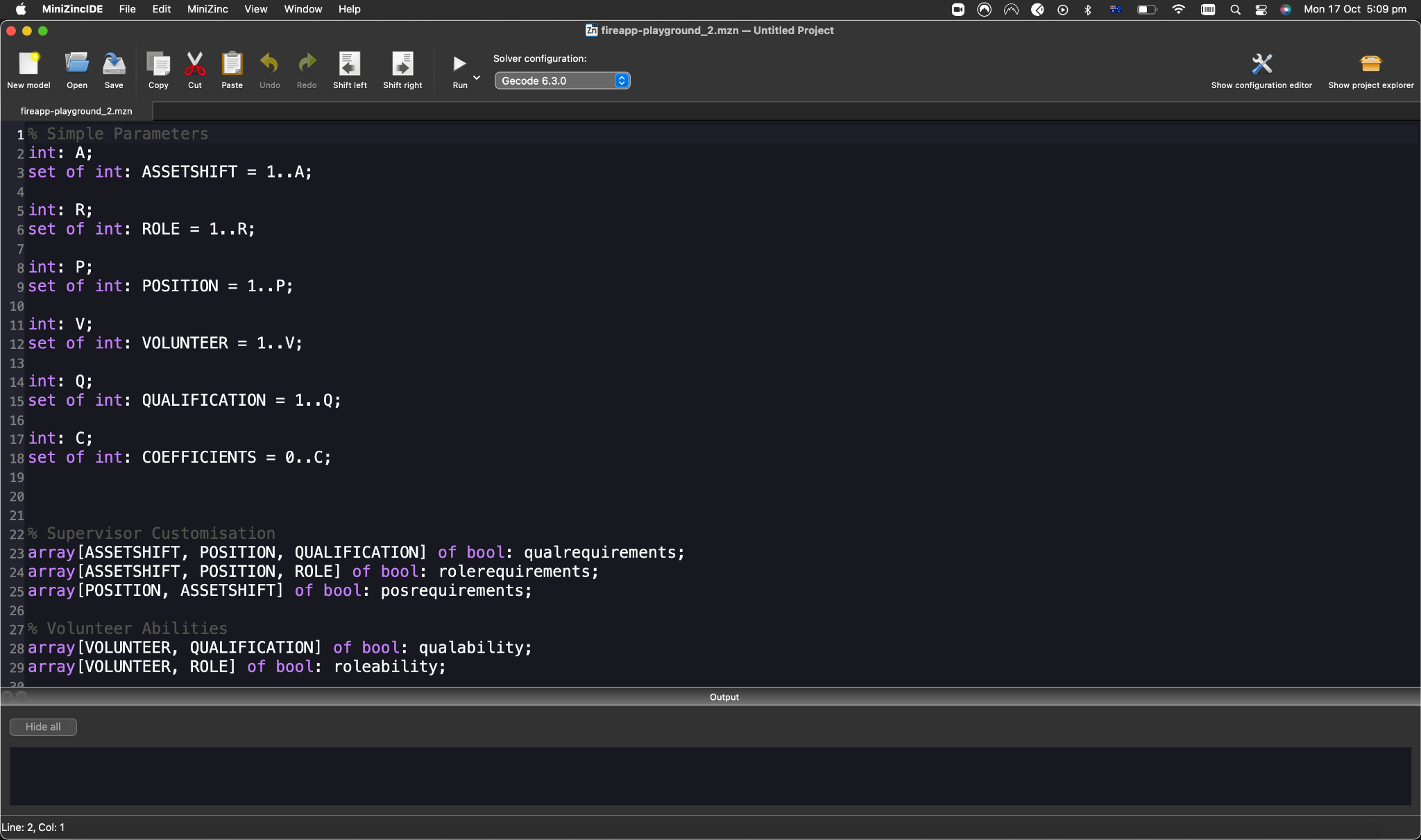
In summary, the developer will need the following to aid in AI scheduler contribution:

* MiniZinc IDE, which can be installed from here: ***https://www.minizinc.org/ide/***
* Basic dev environment (PyCharm and Python dependencies) with the correct version of MiniZinc

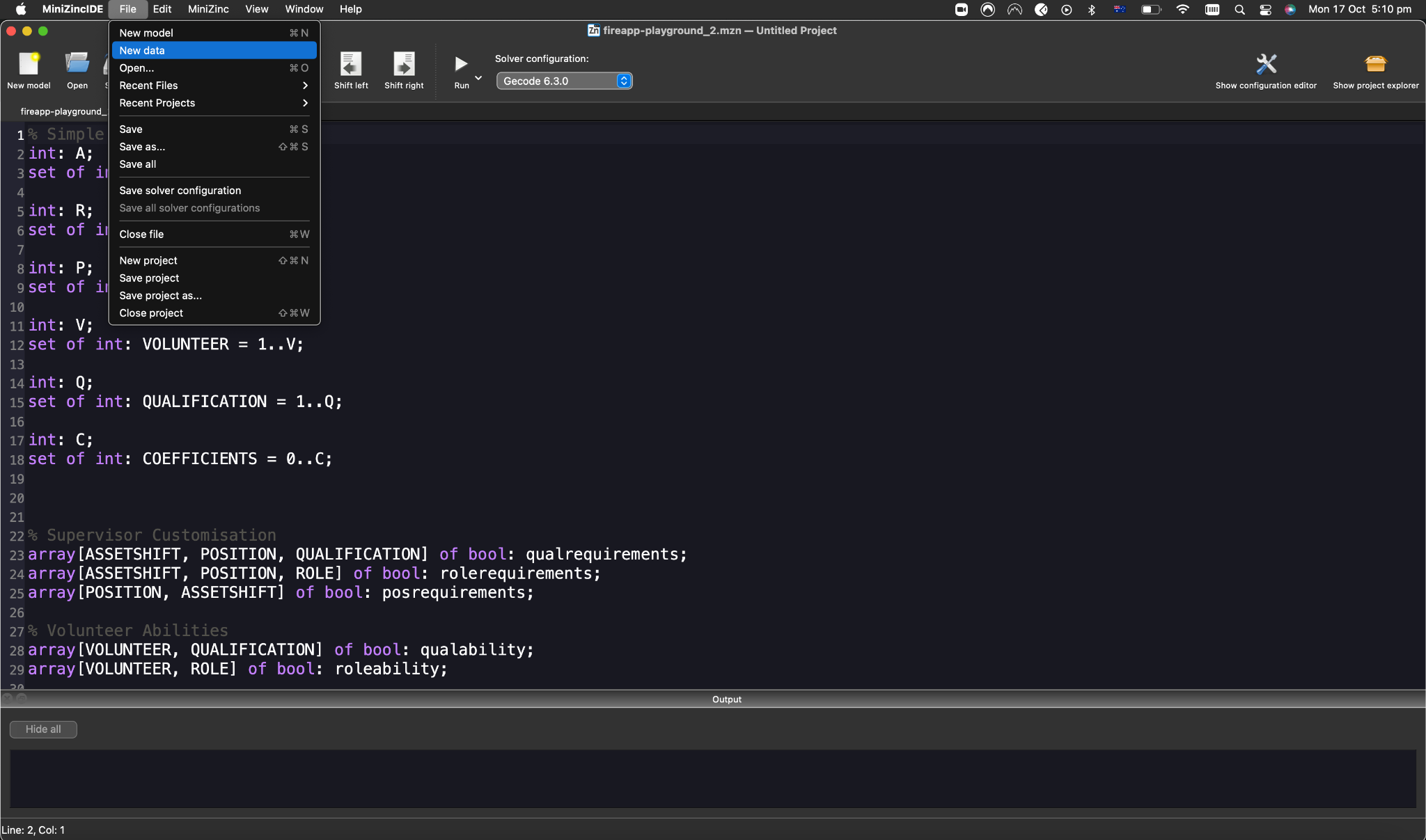
**Basic Workflow**

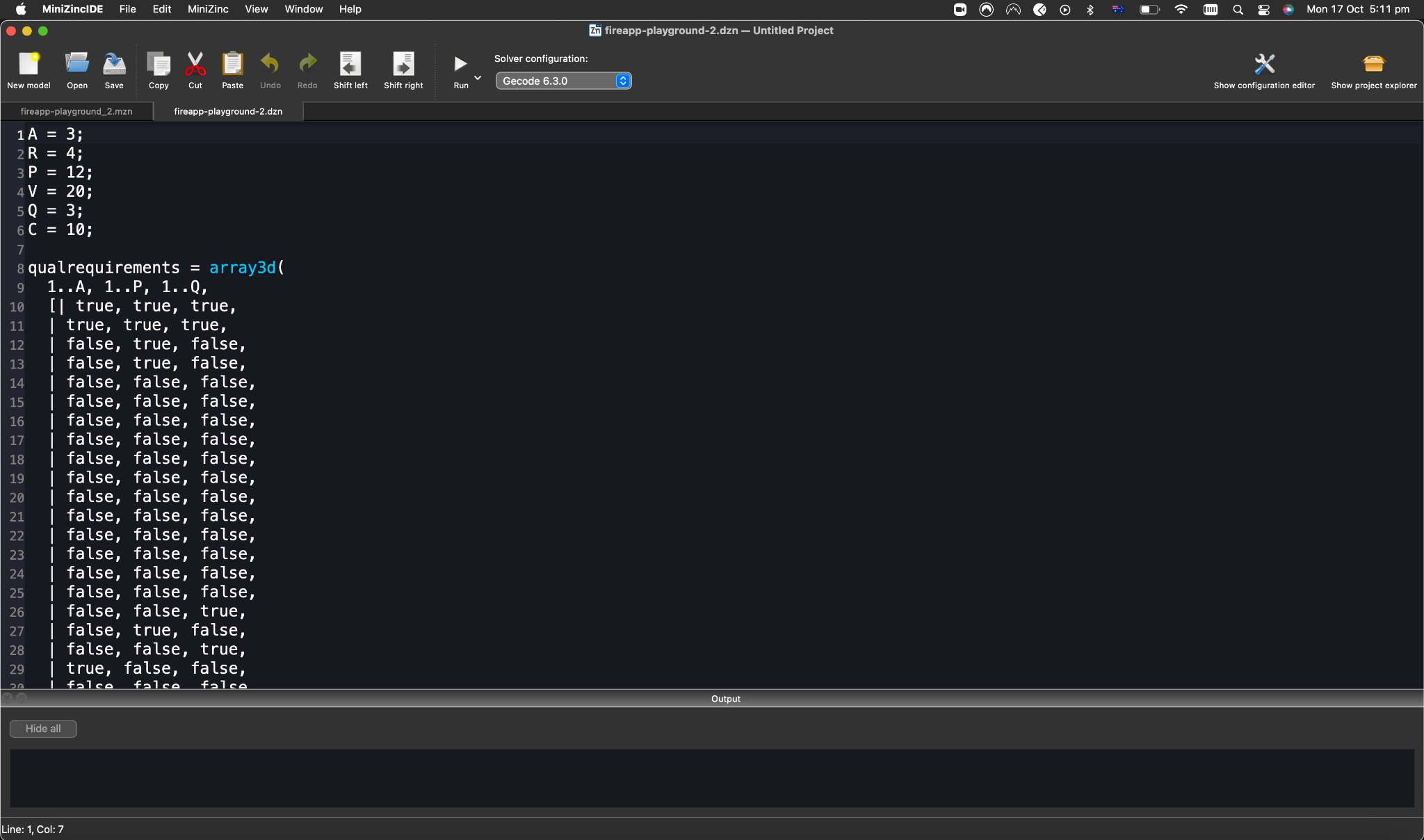
1. Once all your dev environments are set up, open up MiniZinc IDE. Create a new file with a ***.mzn*** file extension. You can give this file any appropriate name as this file will eventually contain the Minizinc code for you scheduler. Write any implementation of the scheduler’s AI model in the file.



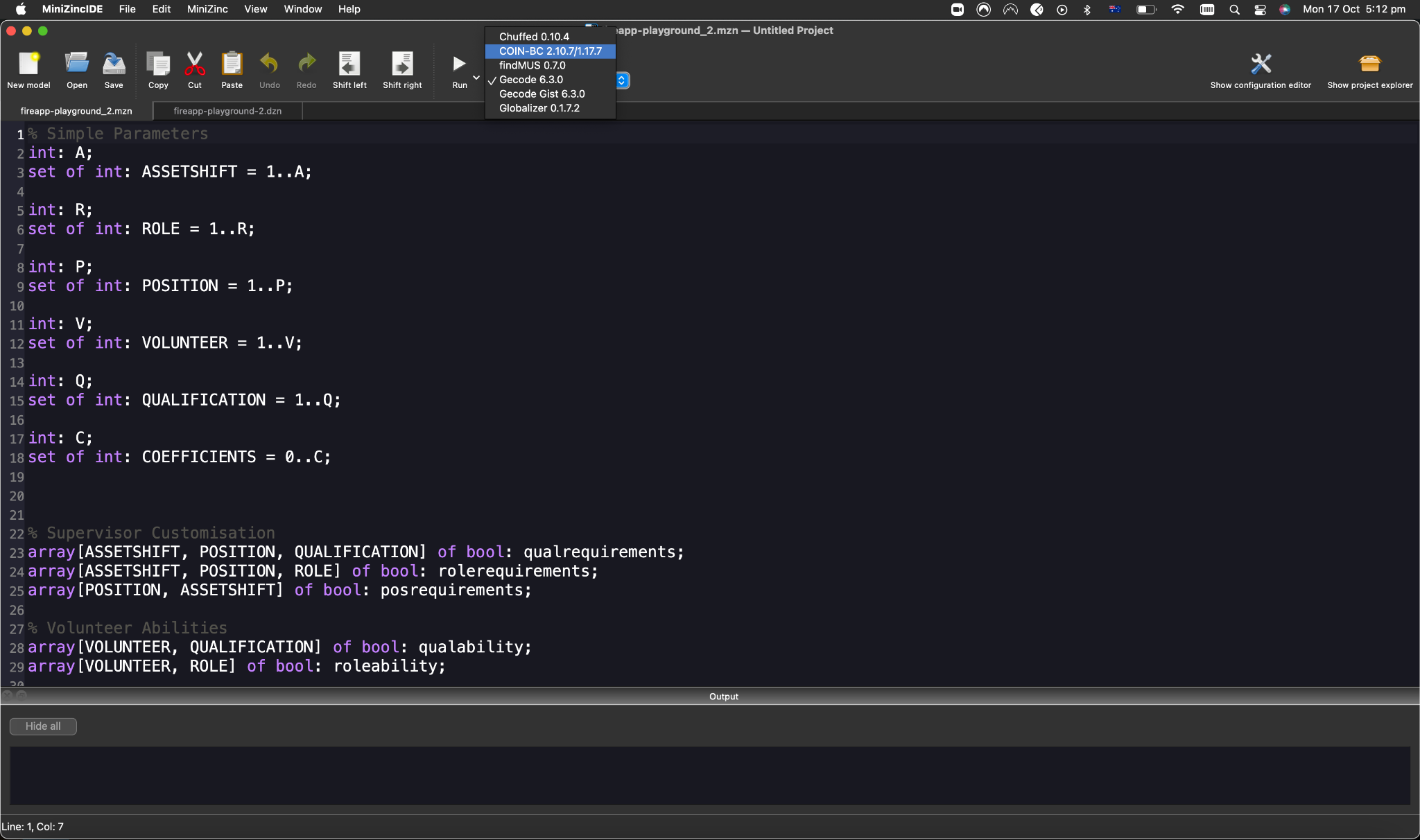


1. Once you have some syntax error-free implementation of the scheduler’s AI model in the ***mzn*** file, you need to create a new file of ***.dzn*** extension. Write data that you will input into the scheduler in this dzn file.

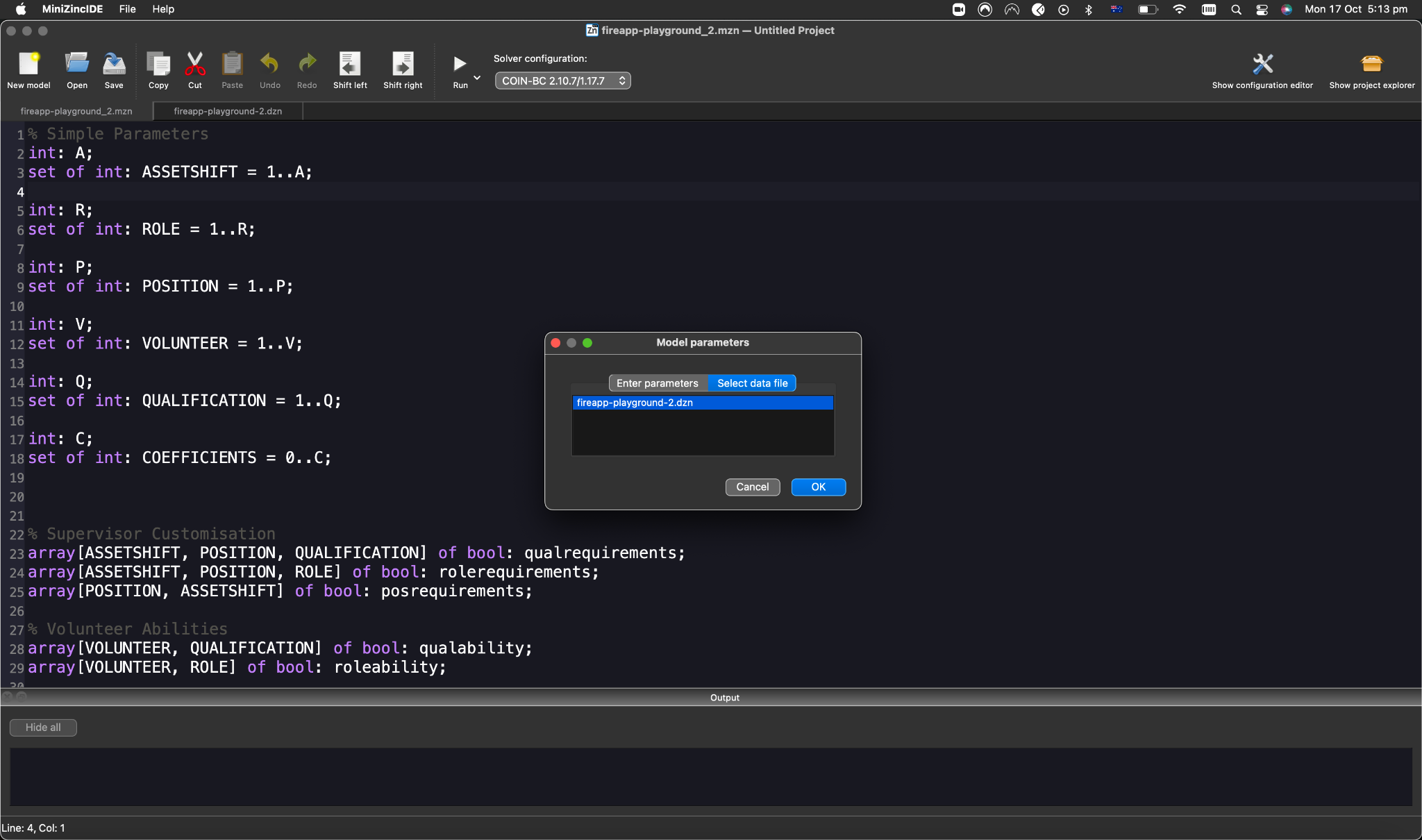




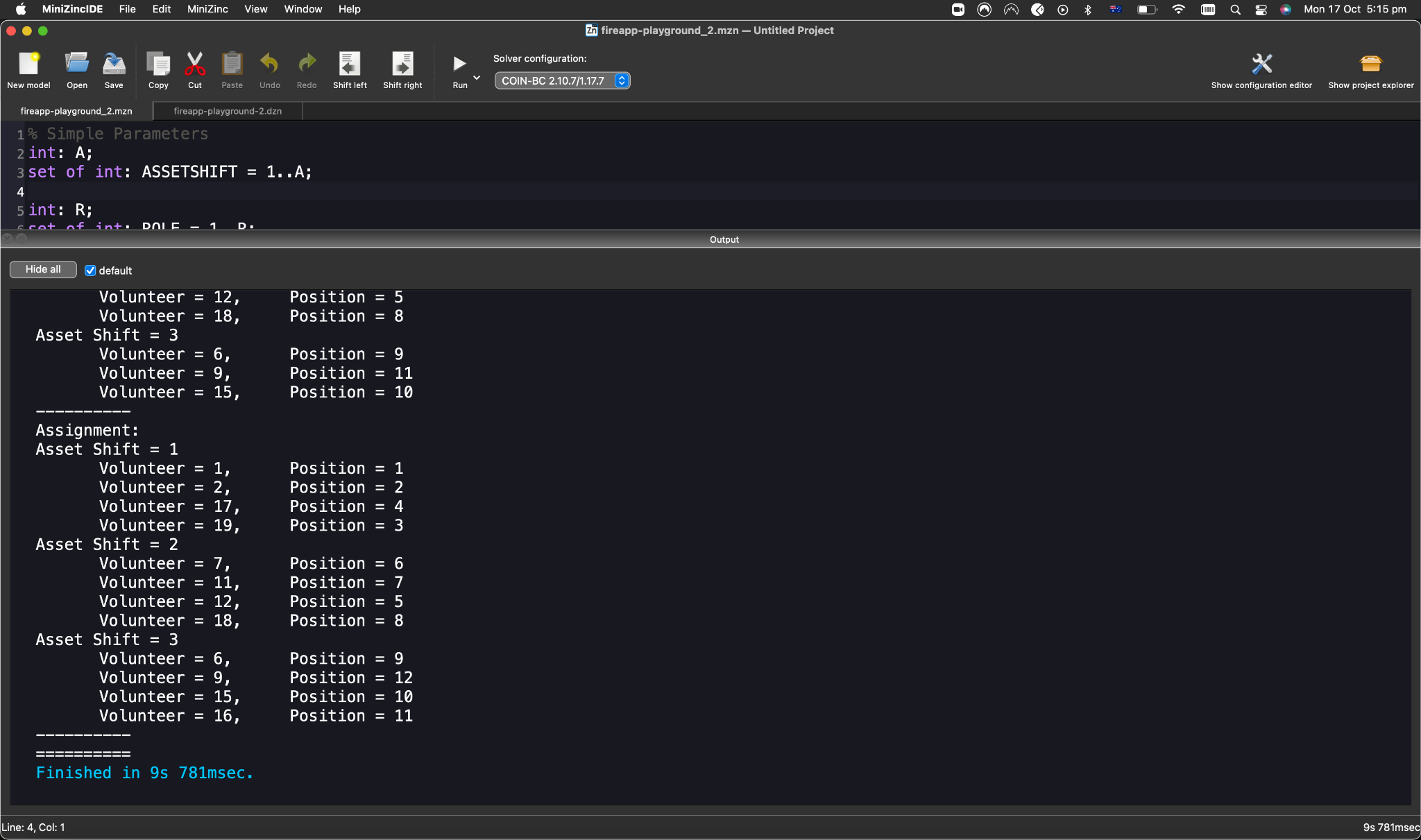
1. Set the solver to ***COIN-BC***. You may use other solvers in the drop-down menu, but previous teams have researched and discovered that this solver is the most efficient in most cases.



1. Click the “solve” button ***from the mzn file.*** It is important that you do this from the correct file.

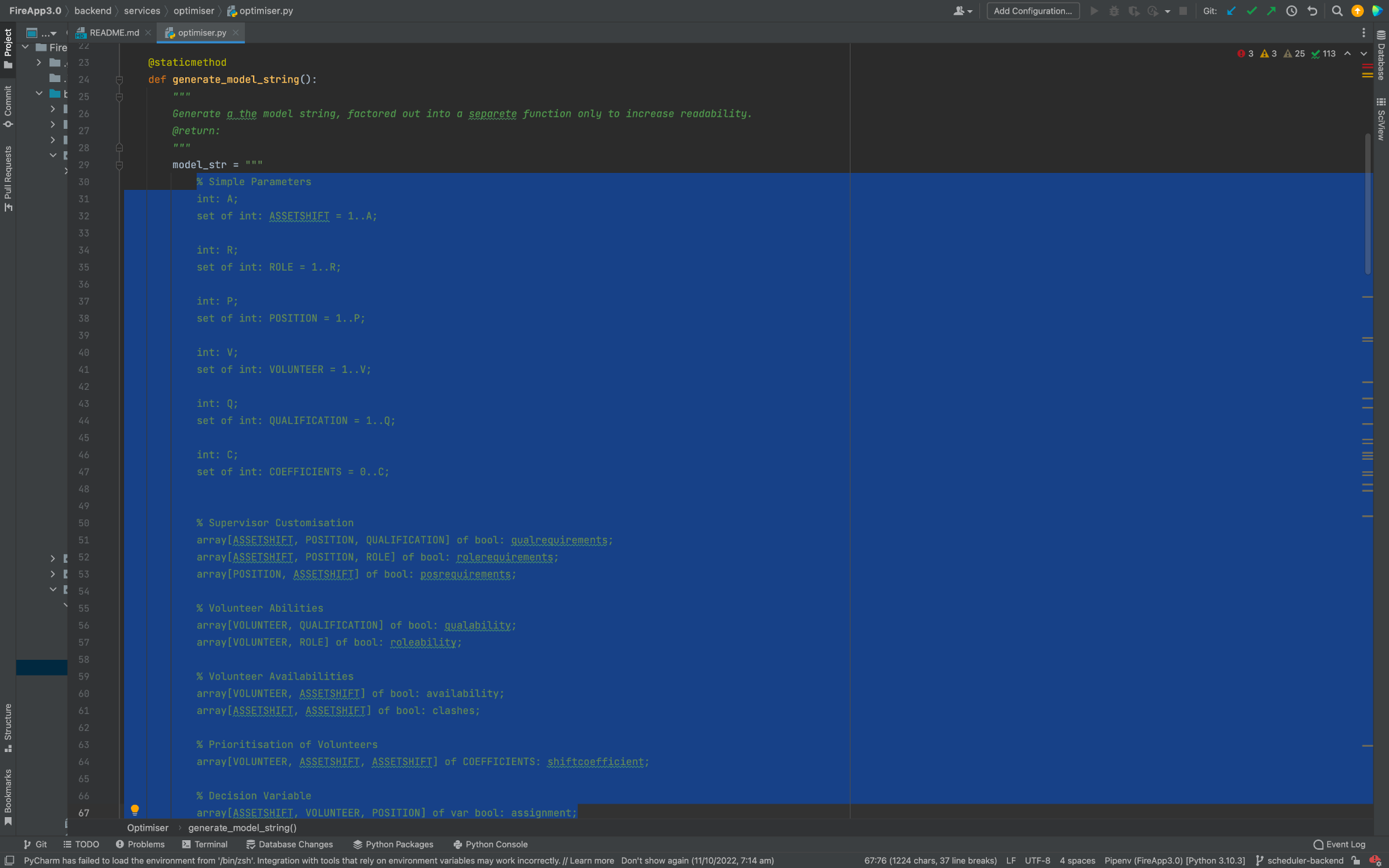


1. View the results in the console. If there are any bugs, it will appear in the console as well. A single “---------” line break means that the result is one of the valid ones calculated by the solver using the model. A double “======” line break means that the result is the final ***optimal***result. If you do not get any results, then there may be some unintended bug giving you no solution or your constraints are too strict.



1. Once you are satisfied with the scheduler implementation, copy the code in the mzn file only. Paste it in the ***./backend/services/optimiser/optimiser.py*** file within the multiline string (“”” ”””) value of ***model\_str*** in the ***generate\_model\_string()*** function.

**N.B.** You may decide to remove the comments in the MiniZinc implementation before copying and pasting. Or, for ease of understanding, keep it in and remove it after the code reviewer (usually Kenneth Young) has accepted the branch. However, note that sometimes comments in the implementation may cause a bug for the Python compiler.



1. The scheduler part of the feature is complete! The other developer, or yourself if you are willing, will now work on integrating this scheduler to the Python backend.

**Scheduler Parameters**

The MiniZinc model of the scheduler which is integrated as a string in the optimizer.py file requires various inputs from the Python backend. Please refer to the different inputs which are located on pages indicated in the table of contents. The reader should also refer to the fireapp\_playground\_2.dzn file which is one of the handovers to see concrete examples of what the following parameters would look like.

**A**

An integer representing the number of asset requests being made in a single scheduling effort from the supervisor. i.e., if the supervisor chooses to schedule three different assets/vehicles, then A=3.

**R**

An integer representing the number of roles that the supervisor can choose from. The number of roles is dynamic, so if new roles are added in the system, it would not break the scheduler. i.e., if the available role options are crew leader, driver, basic firefighter, then R=3.

**P**

An integer representing the number of positions being requested by the supervisor. Previously, role represented position, but now that roles and qualifications are dynamic, there is now a new variable position to take in role and qualification as its qualities. For example, a position on one of the assets might require the person to be a driver and also be qualified to cut down trees.

No firefighter may have two same positions as it represents their unique place in all of the asset shifts. i.e., if there are 12 assignable places in all the asset shifts, then P=12.

**V**

An integer representing the number of volunteers in that brigade station, i.e., if there are 20 registered volunteers, then V=20.

**Q**

An integer representing the number of qualification types that the supervisor can choose from, e.g. having a driver’s licence, being qualified to cut tress, being qualified to deal with asbestos. Similarly to roles, this qualification is also dynamic and will break the scheduler if more qualifications are added or removed, as long as there is more zero. If there are three different qualifications available, then Q=3.

**C**

An integer representing the maximum coefficient values. For now, leave this value to be 10.

**shiftcoefficient**

A 3-dimensional matrix of size [V, A, A] that contains integer values. These values follow the criteria outlined in this table:

| Situation | Additive Constant |
| --- | --- |
| The volunteer had two shifts already in the past 48 hour period. | +5 to all of the volunteer’s shifts |
| The volunteer had one shift already in the past 48 hour period. | +3 to all of the volunteer’s shifts |
| The asset shifts are immediately back to back. | +4 |
| The asset shifts have at least one shift of break in between. | +2 |

The bigger the number, the less likely that volunteer will be assigned by the schedule for those two shift pairs.

**qualrequirements**

A 3-dimensional matrix of size [A, P, Q], that contains boolean values. These boolean values represent the qualifications that are required for each position at each asset request.

**rolerequirements**

A 3-dimensional matrix of size [A, P, R], that contains boolean values. These boolean values represent the roles required for each position for each asset shift.

**posrequirements**

A 2-dimensional matrix of size [P, A], that contains boolean values. These boolean values represent the positions that are required for each asset shift. That is, if the supervisor has customised the scheduler to have certain positions (with certain roles and qualifications), then it should be a true value in that asset shift.

**qualability**

A 2-dimensional matrix of size [V, Q] that contains boolean values. These boolean values represent the qualifications that each volunteer registered at a brigade station is recognised as satisfactorily competent in doing.

**roleability**

A 2-dimensional matrix of size [V, R] that contains boolean values. These boolean values represent the roles that each volunteer at a brigade station is given authorisation to undertake.

**availability**

A 2-dimensional matrix of size [V, A] that contains boolean values. These boolean values represent the asset shifts that each volunteer has the “free time” to be scheduled. This information should be coming from the calendar.

**clashes**

A 2-dimensional matrix of size [A, A] that contains boolean values. The boolean values represent whether each asset shift clashes (time-wise) with other asset shifts requested by the supervisor. Because a shift cannot clash with itself, the values on the left diagonal should always be false.

**Scheduler Output - assignment**

The variable assignment is the decision variable of the model. The COIN-BC scheduler will try to find the best arrangement of volunteers that will maximise the number of true’s in this variable.

The decision variable assignment is a 3-dimensional matrix of size [A, V, P] that contains boolean values. These boolean values represent the final assignment of eligible volunteers for each shift and position.