# Goh Rui Zhi - Project Portfolio

## PROJECT: Scheduler

---

# Overview

*Scheduler* automates the tedious process of interview scheduling for organisations. It is designed with convenience in mind, and apart from the efficient scheduling of interviews, it also empowers user with the ability to import/export timetables and even mass email the interviewees all on the same platform. The algorithm behind *Scheduler* ensures the efficient and fair allocation interview slots, effectively minimising human-error. The user interacts with it using a CLI, and it has a GUI created with JavaFX. It is written in Java, and has about 20 kLoC.

# Summary of contributions

- **Major enhancements**:
  - **Added the ability to import both interviewer's data and interviewee's data.**
    - What it does: Allows the user to conveniently import all interviewers and interviewees data, consisting of their personal details, as well as, their availabilities.
    - Justification: It is incredibly tedious for the user to make use of the 'add' command to manually add each interviewer and interviewee into Scheduler. Hence, the automation of this process, which is to import data, is absolutely necessary. It also allows the user to conveniently make changes in the raw data (CSV file) which allows the flexibility of using formatting tools or automation (VBA) in Microsoft Excel, rather than manually edit the entries in the application.
    - Highlights: Implementing this feature requires a good understanding of the model and the 'add' feature as it builds upon it. String processing of the imported data was also tedious and very prone to bugs due to many formatting issues that could happen.
  - **Added the ability to export schedules.**
    - What it does: Allows the user to conveniently export the scheduled interview timetables.
    - Justification: Exporting these schedules to an external file allows the flexibility of sharing these timetables with colleagues/interviewers. It also allows the user to use third party applications (Excel) to touch-up/re-format the data. Most importantly, it frees Scheduler of the need to hold the current schedules so that it can be used again to schedule other interviews.
- **Minor enhancement**: Implemented responsive UI for `add`, `delete`, `edit`, `import` and `clear` commands, together with team mate Yau Dong.
- **Code contributed**: [Functional code] [Test code] *{give links to collated code files}*
- **Other contributions**:

- Enhancements to existing features:
  - Designed a way to update Schedules by re-generating the list of schedules solely from a given list of Interviewers. This is the backbone for the integration between UI and Model after the execution commands that modifies the model e.g. `add`, `delete`, `edit`, `import` and `clear` commands. The subsequent schedule command's integration with UI also builds upon this method. Look at section "Updating Schedule List" in Developer's Guide for a better idea of the above.
- Documentation:
  - Did cosmetic tweaks to existing contents of the Developers Guide:
  - Added User Preferences section into User Guide: 164
  - Added User Stories 35 and Use Cases 186
- Community:
  - PRs reviewed (with non-trivial review comments): #161
  - Handled merge conflicts for team: #80

# Contributions to the User Guide

*Given below are sections I contributed to the User Guide. They showcase my ability to write documentation targeting end-users.*

## Importing existing availabilities `import`

2 types of data can be imported - interviewers' availabilities and interviewees' details. Both of these details needs to be in CSV format and must follow the format given in the templates below. These templates were adapted versions of NUSync's interview sign up excel sheets. The templates for the import commands can be found here:

- Interviewers' availability: http://bit.ly/interviewerTemplate
- Interviewees' details: http://bit.ly/intervieweeDetails

**TIP** If you wish to import a new set of data of the same type, we strongly recommend that you use the Clear command to clear pre-existing data before re-importing newer data. This is because Scheduler does not allow duplicate persons, for both interviewer and interviewee. Unless you are very certain that the the second import does not contain any duplicate data, we would suggest that you run the 'clear' command beforehand.

Constraints:

- Data in .csv file must follow the settings defined in the user preference file. View User Preferences if you wish to change the default settings.
- Data to be imported should only be in English and should strictly follow the format of the templates given above, e.g. format of headers and timeslots.

## Importing interviewers' availability

Imports interviewers' availability from a comma separated values (CSV) file. After import command is ran, the GUI should display the imported data.

> **NOTE** The schedule displayed will only show timeslots within the Working Hours and Duration that the user has specified.

Key formatting information for Interviewer's Data:

[InterviewerData] | *InterviewerData.png*

- 1. The first column of each day's schedule will consist of the timeslots for that day.
- 2. Each timeslot should also be in the format `HH:mm - HH:mm`.
- 3. The very first header at the top left-hand corner should indicate the day of the schedule. The subsequent headers should indicate the interviewers in the format `DEPARTMENT - INTERVIEWER␣S_NAME`.
- 4. A "1" in a timeslot represents that the interviewer in that column is **available** and "0" represents that he/she is **unavailable** for that particular timeslot.
- 5. Separation of different days of schedules is made by leaving 2 empty lines after the end of the first table.

Format of command: `import interviewer fp/FILE_PATH`
- FILE_PATH needs to be of .csv extension
- E.g. C:\\Users\Bob\file.csv

Example:

```
import interviewer fp/C:\Users\johndoe\Interviewers.csv
```

Constraints:

- Timeslots must obey the allowed values that are specified in the User Preferences.
- Duplicate entries of the same interviewer (same name) cannot be presented in the CSV file to be imported.
- If none of the interviewers are available for a particular day, e.g. all "0" for all timeslots on that day, the GUI will not display the schedule for that day.

## Importing interviewees' availability

Imports interviewees' availability from a comma separated values (CSV) file.

> **NOTE** There will be no changes in the schedules tab after the import of interviewee's data. The changes will be made after Schedule command has been ran.

Key formatting information for Interviewee's Data:

- Headers specify the details of the Interviewee to be imported. The order of the headers must be strictly followed as shown in the above template.
- Timeslots: Each timeslot should be in the format `dd/MM/yyy HH:mm - HH:mm`. Use commas to separate the timeslots if there are more than 1.
- There can only be 1 unique value for all other fields. Commas should not be used in these fields.

Format: `import interviewee fp/FILE_PATH`
- FILE_PATH needs to be of .csv extension.
- E.g C:\\Users\Bob\file.csv

Example:

```
import interviewer fp/C:\Users\johndoe\Interviewees.csv
```

Constraints:

- Timeslots must obey the allowed values that are specified in the User Preferences.
- Duplicate entries of the same interviewee cannot be presented in the CSV file to be imported.

## Exporting of interview schedule `export`

Exports the allocated interview schedule timetable to the specified comma separated values (CSV) file.

Format: `export fp/FILE_PATH`

- FILE_PATH is the path to the file.
- E.g C:\\Users\Bob\file.csv

| | |
|---|---|
| **NOTE** | If `FILE_PATH` is a valid .csv file but does not exist, it will be created in the specified path. |

Example:

```
export fp/C:\Users\johndoe\schedules.csv
```

Exported data format:

- The exported data will follow the format of the schedules in 'schedules' tab.
- Each schedule (table) will be separated from the previous by 2 empty lines.
- Timeslots with "0" indicates that the interviewer is not available for the timeslot. Timeslots with "1" indicates that it is an available slot that has not been filled up by an interviewee.
- Csv file can be converted into an excel file by following this guide - https://www.ablebits.com/office-addins-blog/2014/05/01/convert-csv-excel/.

Constraints:

- This command requires the interview schedule to be generated first.
- Although the specified filepath does not need to exist, it needs to be of .csv extension.

| WARNING | The exported data will erase and replace all pre-existing data in the specified file. |
|---|---|

# Contributions to the Developer Guide

*Given below are sections I contributed to the Developer Guide. They showcase my ability to write technical documentation and the technical depth of my contributions to the project.*

## Import

### Implementation

The import feature uses `CsvReader` in the Model to read the given .csv file and stores the data into the model.

- `import interviewer fp/FILE_PATH` stores the read data as a list of `Interviewer` objects in the model.
- `import interviewee fp/FILE_PATH` stores the read data as a list of `Interviewee` objects in the model.

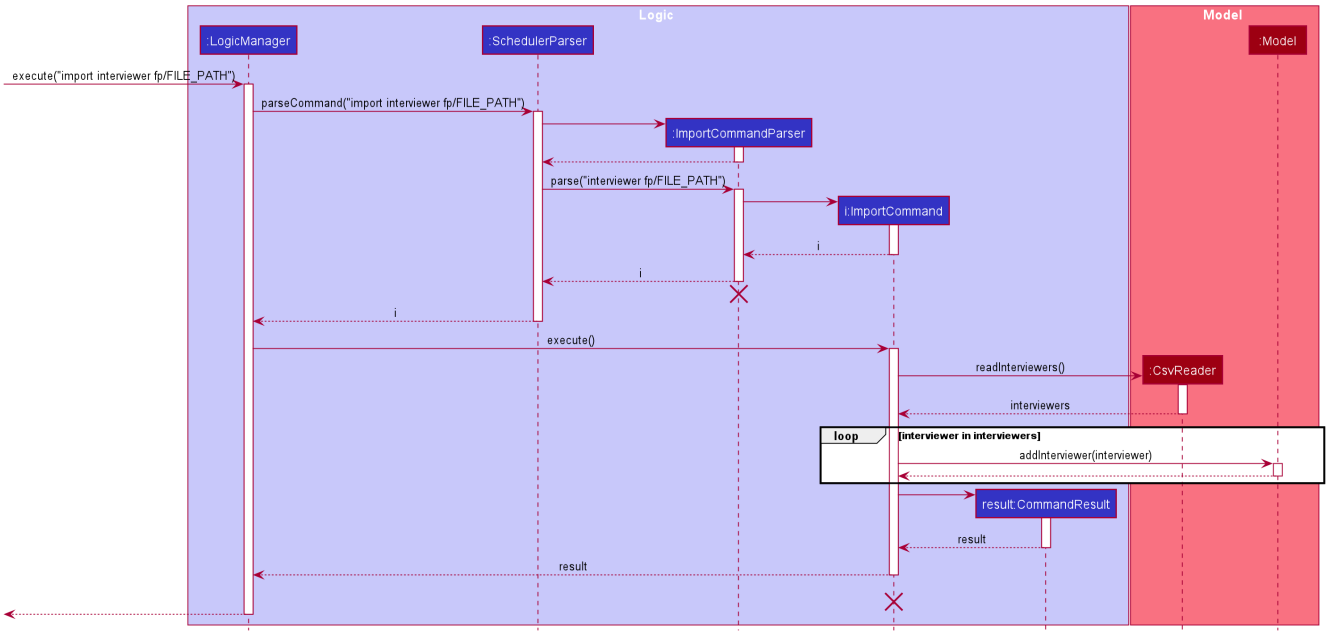


*Figure 1. Import Interviewer's Schedules Sequence Diagram*

Given above is an example of a sequence diagram for importing interviewer's schedules. It applies to both importing interviewee's and interviewer's data, with the only difference being in the string processing methods in the `CsvReader` class.

The following activity diagram summarizes what happens when a user executes a new command:

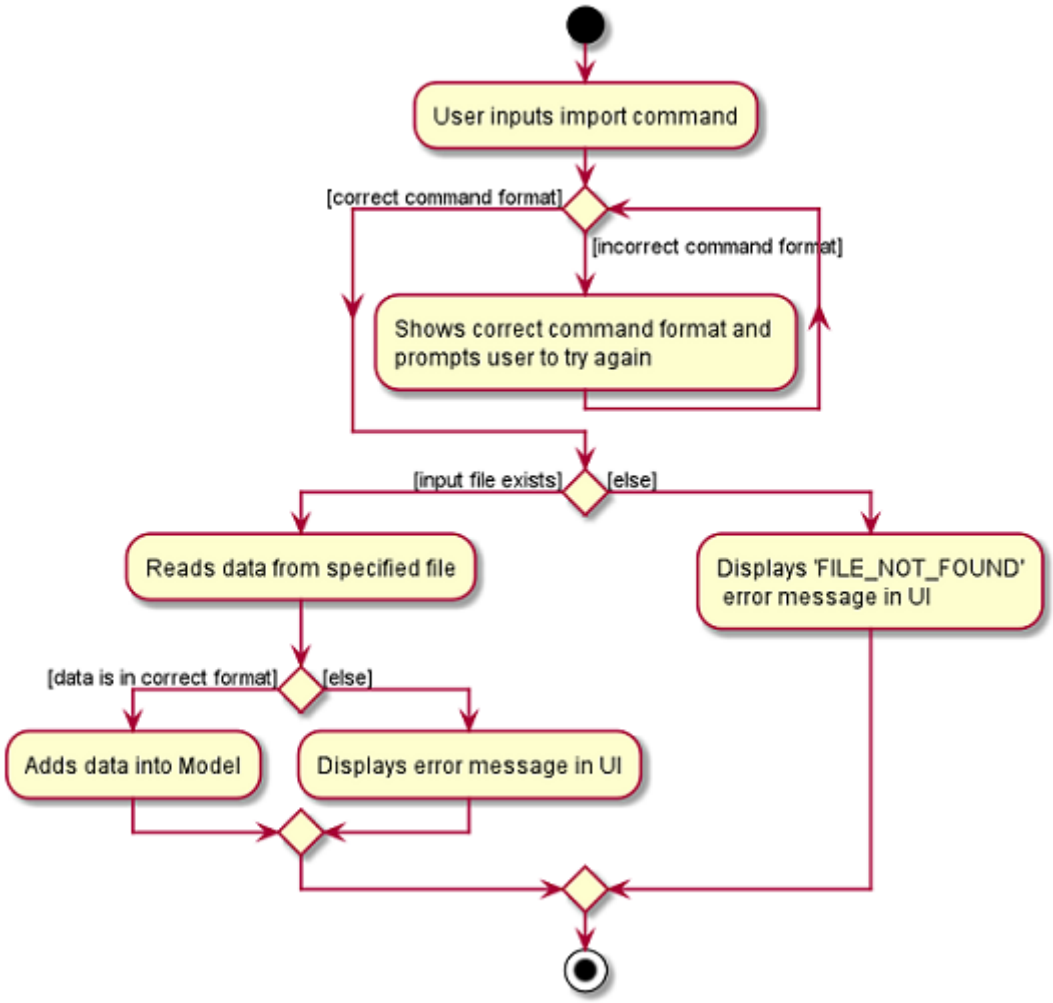


*Figure 2. Import Activity Diagram*

### CsvReader

`CsvReader` class encapsulates the string processing process that happens in an import command. It makes use of `BufferedReader` to access and read from the specified CSV file. It has 2 key methods - `readInterviewers()` and `readInterviewees()`.

The following are some of the notable areas of implementation of these 2 methods:

`readInterviewers()`:

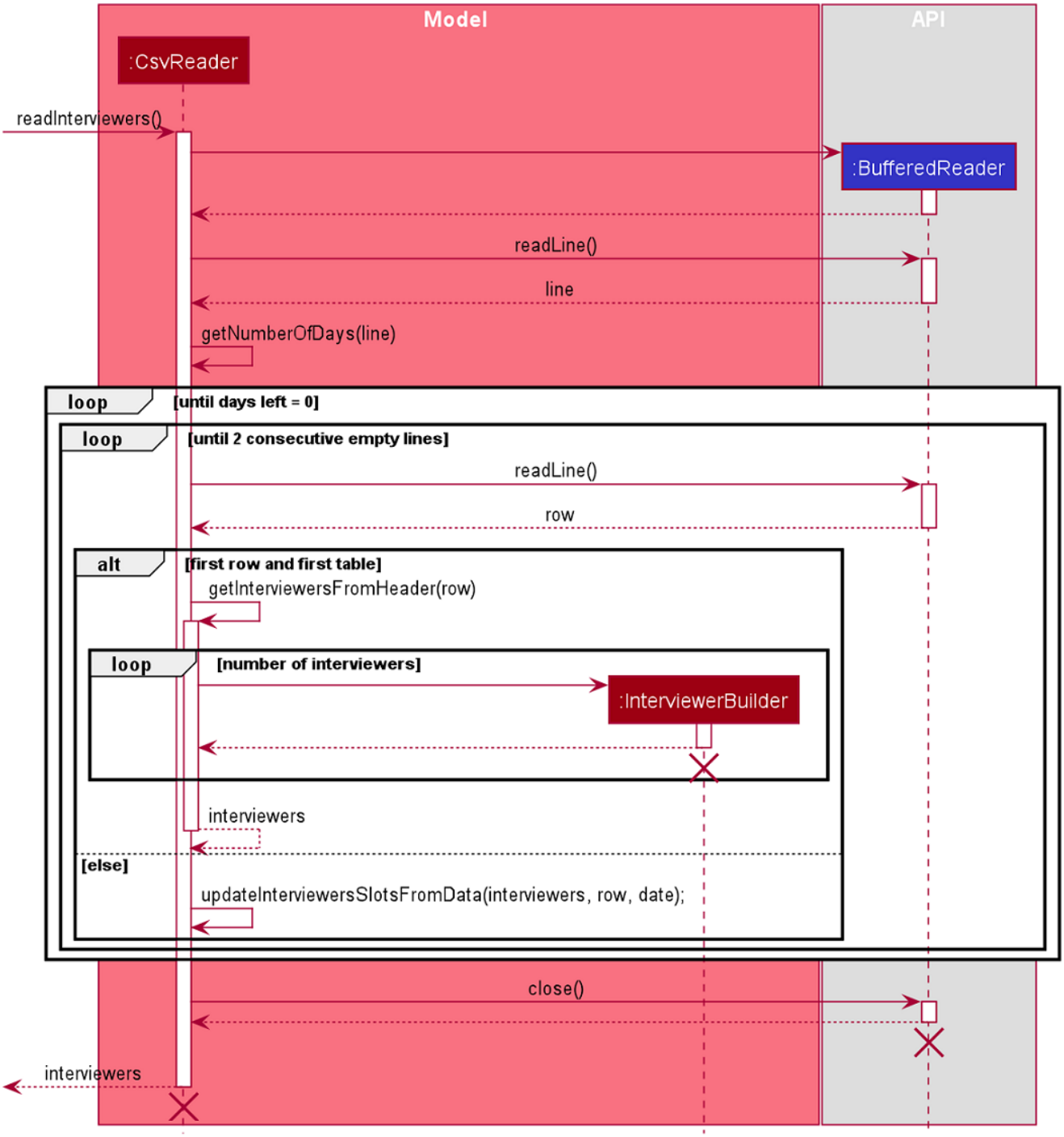


*Figure 3. Reading interviewers Sequence Diagram*

- This method reads the imported file line-by-line. Number of days (each day is represented by 1 table) and the number of interviewers has to be specified at the beginning of the CSV file.
- It generates a list of interviewers (with no availabilities yet) from the headers of each table.
- Then, it will iterate through every line of each table and adds the availabilities to the respective interviewers.
- The complexity of this method is O(N x D), where N is the number of interviewers and D is the number of days, assuming there are a constant number of timeslots per day.

`readInterviewees()`:

- This method is quite straightforward, reading each attribute from the given table into an

interviewee object.

- There can also be more than 1 preferred timeslot, which are separated by commas in the imported file.
- The complexity of this method is O(N), where N is the number of interviewees.

| | |
|---|---|
| **NOTE** | The imported data must obey the pre-existing conditions for each property of both interviewees and interviewers. E.g. `name` still cannot contain any characters other than alphabetical letters. No duplicate person is allowed as well. Exceptions will be thrown and the relevant error messages will be displayed in the UI if such error occurs. |

# Updating Schedules in the Model

The list of schedules in the model should be updated every time a command modifies the IntervieweeList or InterviewerList in the model. This is because Schedules are dependent on both Interviewers and Interviews as shown in the class diagram below.

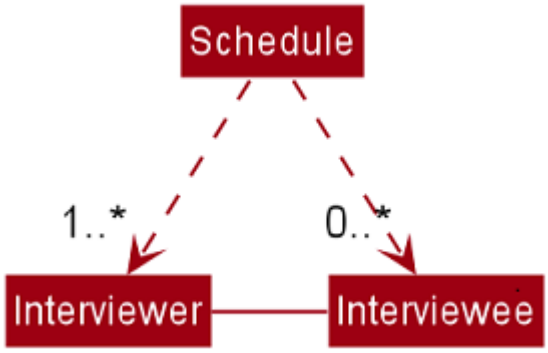


*Figure 4. Schedule Class Diagram*

These are the commands that requires the update of the list of schedules - `add`, `delete`, `edit`, `import`, `clear` and `schedule`.

## Before Scheduling of Interviews

Before the 'schedule' command is ran, the list of schedules is **solely dependent on the InterviewerList**, since it is only through the scheduling of interviews that the data in the IntervieweeList is integrated with the data of the InterviewerList. Therefore, modification of IntervieweeList will not require changes in the list of schedules at this point of time.

To update the list of schedules in the model every time the InterviewerList is modified, the method `updateScheduleList()` is called in the model. The notable implementations of this method are discussed below.

- `updateScheduleList()` basically re-generates the list of schedules in the model from the updated InterviewerList.
- It will get `startTime`, `endTime` and `duration` of a timeslot from `UserPref` and generates the allowed timeslots for daily schedules.

- Then, it will get a list of unique dates from the `availabilities` from each of the interviewers in the updated InterviewerList.
- It will then loop through the each date, generating a table of the Interviewers' availabilities based on the allowed timeslots for each day. Each unique date should generate a Schedule object. The result is a list of Schedules.
- Lastly, it will set the `scheduleList` in the model with the newly generated list of schedules.

By re-generating the schedule list every time InterviewerList is modified, it does not matter whether an Interviewer is added or deleted, or if an attribute of an interviewer is modified, the schedule list will be responsive to these changes. The only trade-off is performance, due to the re-generation of the schedules every time a command is ran. However, this will not be a big issue if the number of interviewers is < 100.

# Export

## Implementation

The Export command gets the scheduled time slots from the Model and writes them in the specified .csv file. CsvWriter facilitates the writing to the specified file.

- CsvWriter makes use of `BufferedWriter` to write data into the specified file.

Below shows the sequence diagram of an example export command.

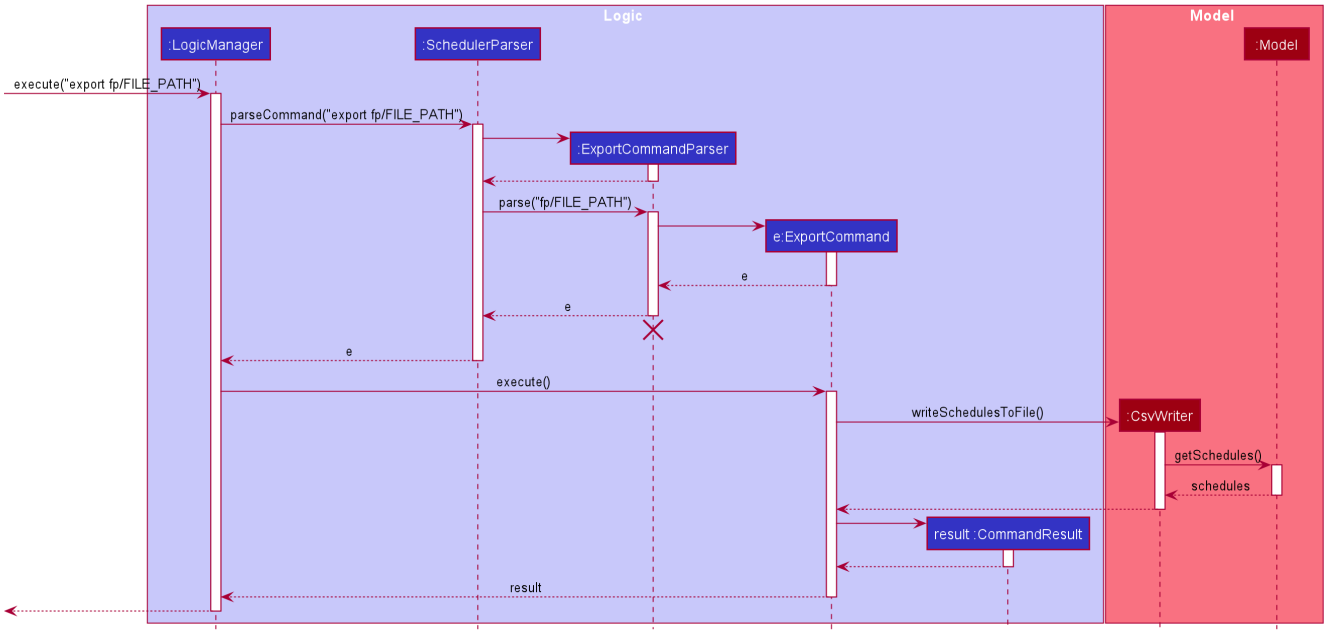


*Figure 5. Export Sequence Diagram*

The implementation is similar to the Import feature. The only differences are in the Model where CsvWriter gets the scheduled time slots from the Model and proceeds to write it into the specified file using a `BufferedWriter`.

The Activity Diagram below summarises the execution of the export command.

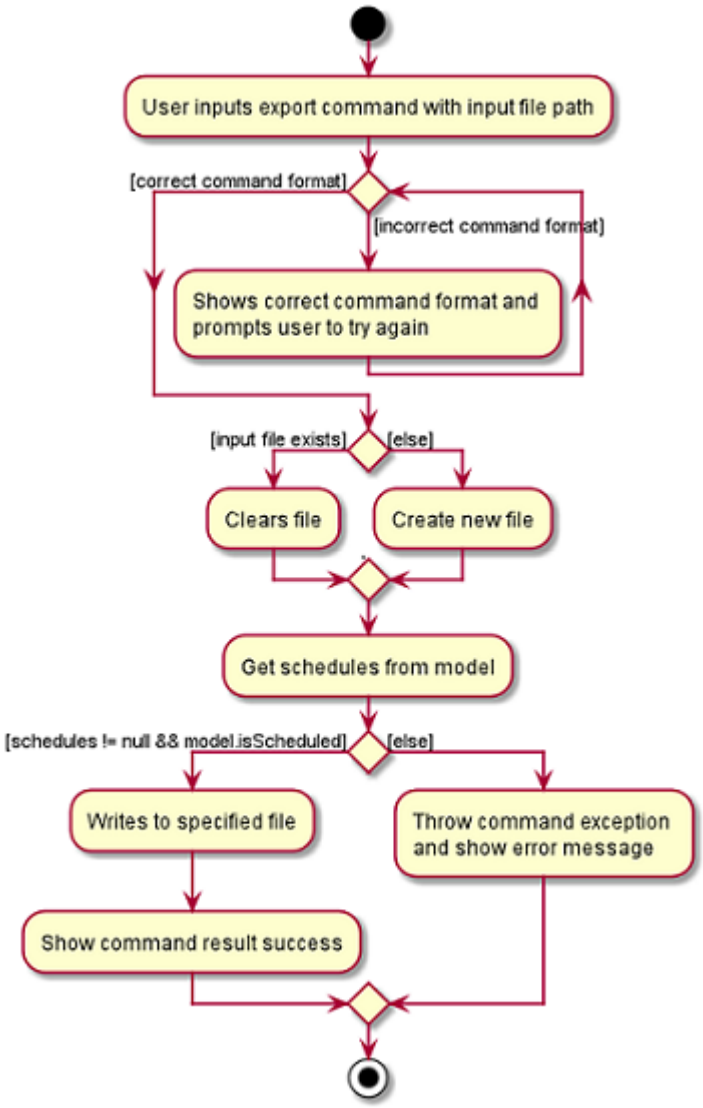


*Figure 6. Export Activity Diagram*