

Normalized Entity Parser Usage Guide

Build-1.0.0-Alpha

1. Overview

NEP (Normalized Entity Parser) is a purpose-built software designed to assist Lead Invigilators in streamlining the coordination of daily exam accommodations. It automates the conversion of detailed PDF rosters, generated by the university's scheduling system, SARS, into a clear, structured summary of scheduled exams. By extracting and grouping key information such as course codes, exam times, and locations, NEP generates concise reports that show how many students are scheduled per course at specific times and venues. This significantly reduces the manual effort required to review rosters and enhances efficiency in exam planning and logistics.

2. System Requirements

To run this software, ensure your system meets the following minimum requirements:

Operating System: Windows, macOS, or Linux

Java Runtime Environment (JRE): Version 8 or higher

RAM: 1 GB or higher

Storage: Minimum 100 MB of free disk space

Dependencies:

1. Apache PDFBox (Bundled with the application)
2. Swing GUI support (Built into standard Java)

3. Input Format:

NEP is designed to take in a specific file format.

File Type: PDF

Document Type: Daily Report generated by SARS

Required Information:

1. Student Name
2. Student Dal ID
3. Location
4. Time
5. Course Code

4. Output Format:

NEP generates 5 distinct types of files for every full pass through the application. All files end in .txt.

1. Normalized Objects:

Generalized form of every line detected in each individual roster. Every roster generates its own unique Normalized Object. Usually, each line saved in the format of:

“[DAL ID | LAST NAME, FIRST NAME | COURSE CODE | LOCATION | TIME]”

2. Combined Objects:

All individual Normalized Object files are combined into one file for further cleaning. As part of this step, the program goes through individual entries and removes any that are missing details or contain errors.

3. Removed Objects:

Entries that are removed from the Combined Objects file are saved separately in this file. If a valid student is removed by mistake due to an error, it is possible add them back using the Removed Object Viewer Panel in the GUI. Each removed entry is usually saved in the following format: “[DalID: DAL ID | Name: LAST NAME, FIRST NAME | Code: COURSE CODE | Location: LOCATION | Time: TIME]”

4. Grouped Objects:

The final file generated from the process. It follows the same formatting used in the pen-and-paper booklet for consistency and ease of use. **Note:** Grouped Objects is created directly from Combined Objects, not from Filtered Objects. Usually in the format of:

“Course Code: COURSE CODE
STUDENT COUNT – LOCATION – TIME”

5. Filtered Objects:

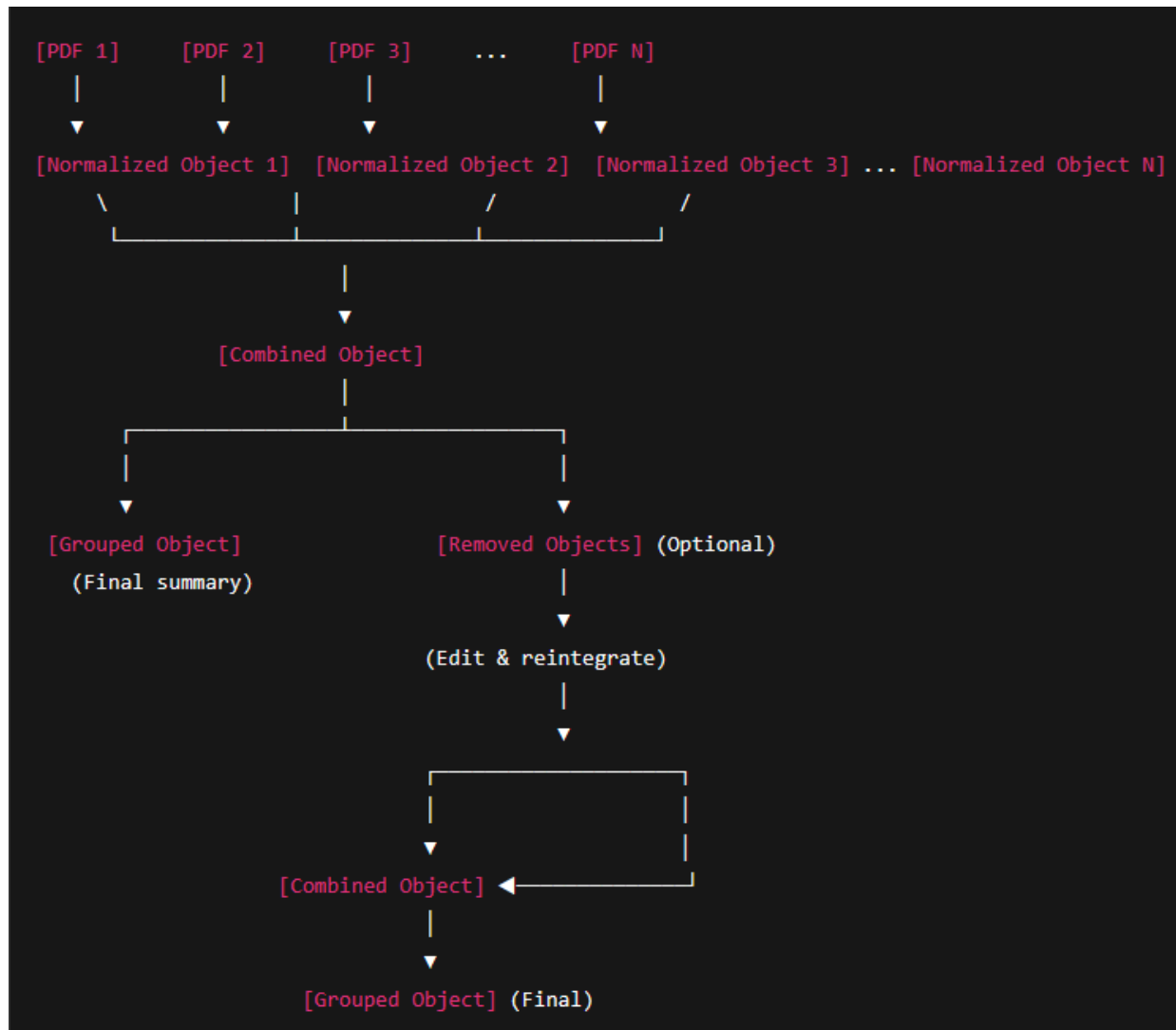
Generated from Combined Objects during the process of generating the Grouped Objects file. Mainly used for debugging through cross checking information in multiple files. Usually in the format of:

“Time: TIME, Course Code: COURSE CODE”

5. Error Handling & Validation

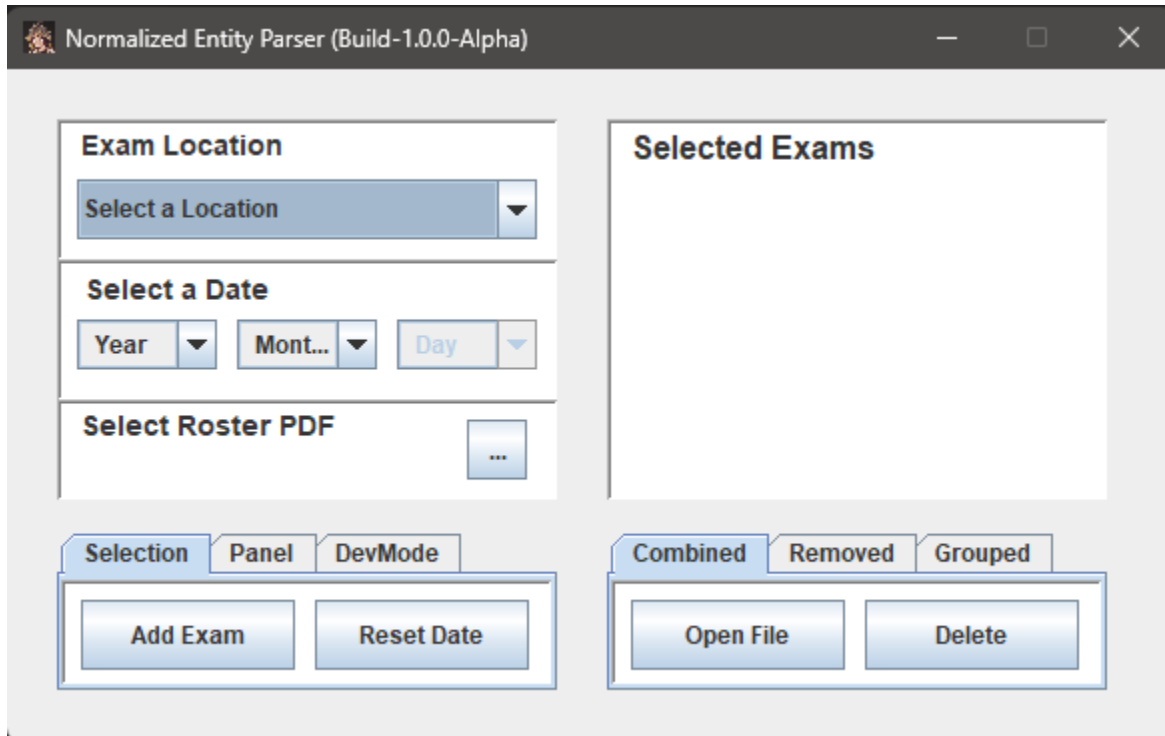
1. **Student Dal ID:** Starts with 'B' followed by 8 digits.
2. **Student Name:** Alphabetic characters, spaces, and commas only.
3. **Location:** Valid string extracted from the exam location column (i.e. 2000 MCCAIN).
4. **Time:** Match the format h:mm AM/PM (i.e., 12:00 PM).
5. **Course Code:** [DEPT] [NUMBER] or [DEPT] [NUMBER] [SECTION], with a 4-letter code, 4-digit number and optional section.

6. File Pipeline



This flow shows how multiple PDFs are turned into clean, grouped data. Each file is normalized, merged, and processed, with optional cleanup and reintegration steps to keep things accurate and flexible.

7. Step By Step Guide



The front-end has four distinct components,

1. Input Panel Section (Top Left: Exam Location, Select a Date...)
2. Output Panel Section (Top Right: Selected Exams)
3. Settings Tab Section (Bottom Left: Selection, Panel...)
4. Files Tab Section (Bottom Right: Combined, Removed...)

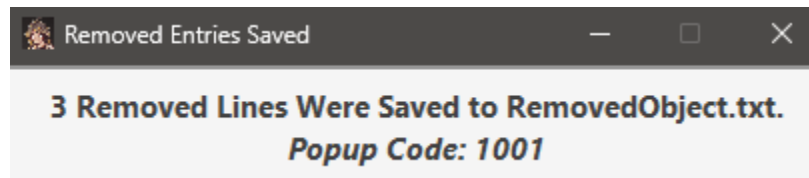
Task 1: Input Panel and Submission

1. Select the Exam Location, date, and the SARS roster for each PDF. The currently working roster locations are Mark A. Hill, G28, and Alternate Location.
2. Once all the inputs are filled, press on the “Settings Tab > Selection > Add Exam” Button to add a single roster to the Output Section.
3. Repeat step 1 and 2 for every Roster PDF with its corresponding location. If a mistake is made in the process, use “Settings Tab > Panel > Clear or Undo” to reverse it.
4. After every single Roster PDF has been added to the Output Section, press on “Settings Tab > Panel > Submit” once to submit every Selected Exams for processing.

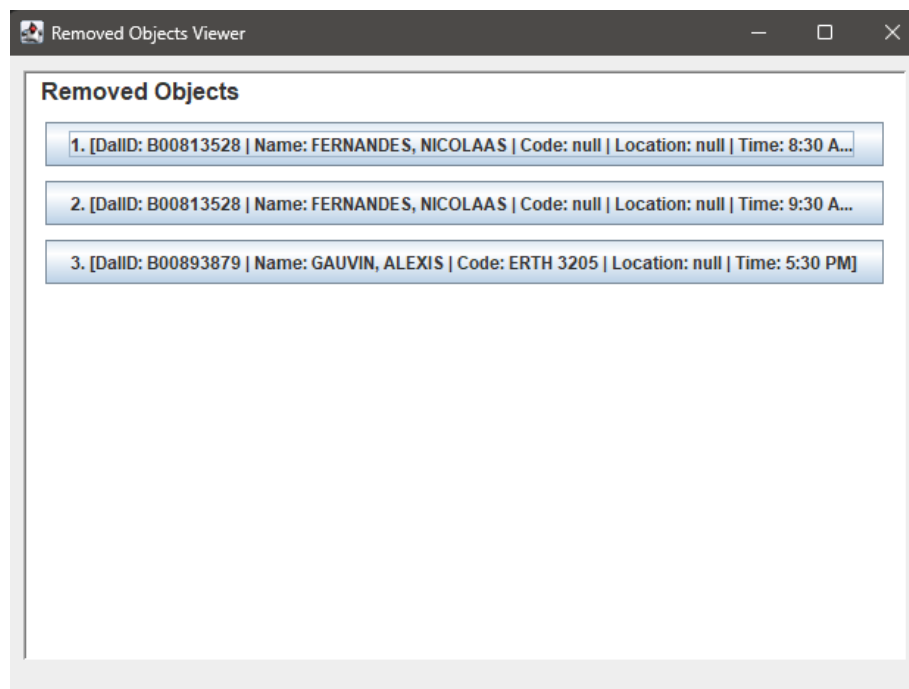
Exam Location	Selected Exams
<div>Mark A Hill</div>	<div>Mark A Hill 21-April-2025</div>
<div>Select a Date</div> <div>2025<div>▼</div> April<div>▼</div> 21<div>▼</div></div>	<div>G28 21-April-2025</div>
<div>Select Roster PDF</div> <div>rosterMAH.PDF<div>...</div></div>	<div>Alternate Location 21-April-2025</div>

Task 2: Roster Validation and Object Check

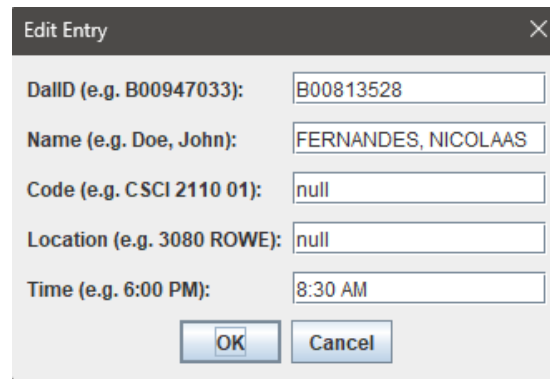
1. Step 4 of Task 1 creates multiple popups displayed by the UI, keep an eye out for “Popup Code 1001: Removed Entries Saved”. This indicates that an entry was found with an invalid or missing field(s) (Consult 5. Error Handling & Validation). This indicates that further work must be done to cross-check the removed entries. Skip Task 2 otherwise.



2. If a Popup Code 1001 appears, press on “Files Tab > Removed > Edit Entry” to open the Removed Objects Viewer frame.



3. Every entry is clickable and clicking it opens the Edit Entry frame. Invalid or missing fields are replaced with a null value. Every entry can be edited via keyboard input.

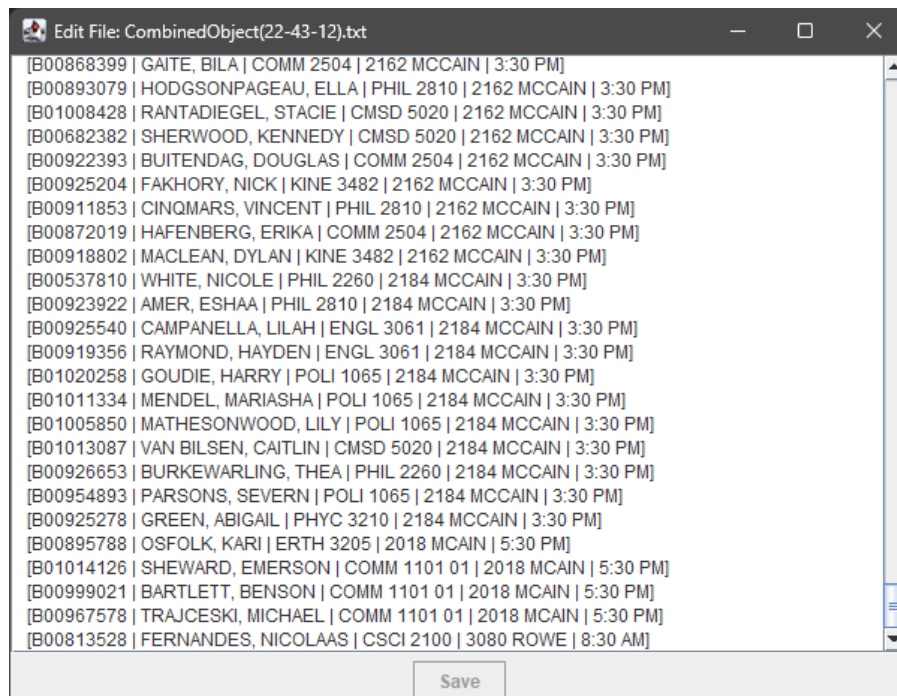


The 'Edit Entry' dialog box contains the following fields and values:

Field	Value
DalID (e.g. B00947033):	B00813528
Name (e.g. Doe, John):	FERNANDES, NICOLAAS
Code (e.g. CSCI 2110 01):	null
Location (e.g. 3080 ROWE):	null
Time (e.g. 6:00 PM):	8:30 AM

Buttons: OK, Cancel

4. Pressing OK after filling all the null values with valid fields leads to the Removed Objects entry being added back into Combined Objects at the bottom. Verify this by pressing “Files Tab > Combined > Open File” and scrolling down.



The window displays a list of objects in a text file. The last entry is the one that was just added:

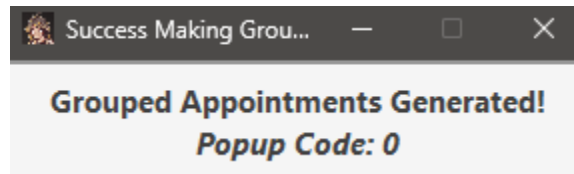
```
[B00813528 | FERNANDES, NICOLAAS | CSCI 2100 | 3080 ROWE | 8:30 AM]
```

Buttons: Save

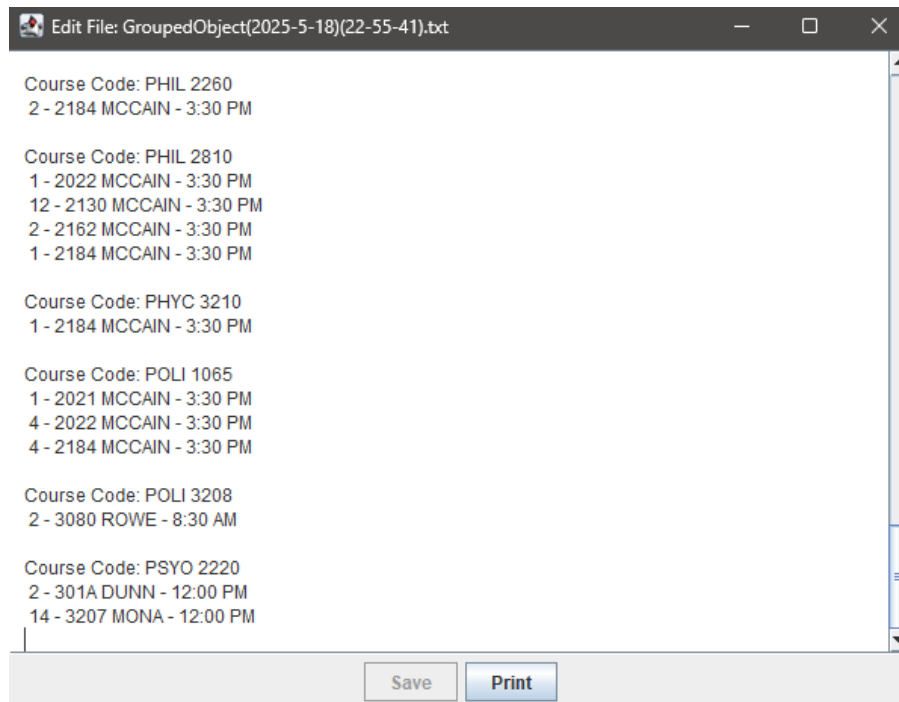
5. Repeat this step for every Removed Objects entry that we want to add back to Combined Objects. If no Removed Object file was created, then skip Task 2.

Task 3: Grouped Object and Printing

1. Once all Combined Object entries have been successfully decided on, press “Files Tab > Grouped > Create” for the final step of the process. This format is meant to imitate the booklet.

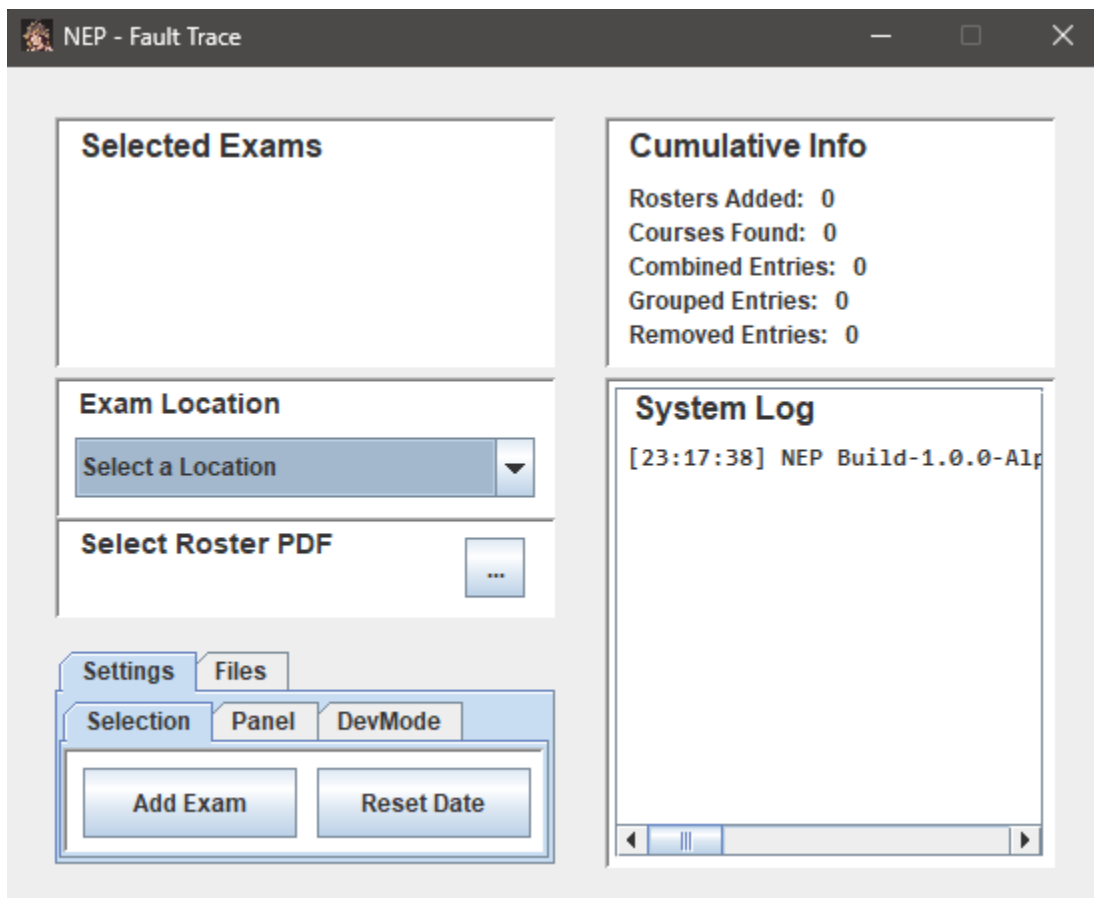


2. Press “Files Tab > Grouped > Open” to open the final file and for printing. Pressing Print opens the print tab. Before printing, please follow Task 4 for book validation.



Task 4: Fault Trace and Validity

1. Press “Settings Tab > DevMode > Trace” to open Fault Trace mode. This panel contains a Cumulative Information panel that shows the number of entries between different parsing stages. Inconsistency is a cause of concern, and the faulty Grouped Objects files must be verified heavily by a Lead Invigilator before being used for logistics.



Note: The difference between the Base NEP UI and Fault Trace is the lack of Date Selection. Date selection is considered an arbitrary input and only used to minimize human errors when selecting PDF Rosters in this application.

2. Follow previous Tasks 1, 2, and 3 on Fault Trace and generate a Grouped Object file. Pressing “Files Tab > Grouped > Create” updates the Cumulative Info panel, which refreshes every 2 seconds while the program is running to improve accuracy.



Cumulative Info	
Rosters Added:	1
Courses Found:	36
Combined Entries:	296
Grouped Entries:	296
Removed Entries:	0

Rosters Added: The total number of Rosters selected for parsing.

Courses Found: The total number of courses found in Grouped Entries.

Combined Entries: Total entries found in Combined Objects.

Grouped Entries: Total entries found in Grouped Objects.

Removed Entries: Total entries found in Removed Objects.

3. The main numbers to look out for is that Combined Entries should always be equal to Grouped Entries regardless of the number of Removed Entries or Courses found. A lack of equality means that something went wrong during the Grouping process. Few diagnostics tips:

- i. Open and analyze the Combined Object file for overlooked entries with mistakes and manually fix the mistakes through the save tool.
- ii. Open the Grouped Object file and manually count and verify the number of entries.
- iii. Open the Combined Object file to verify the total number of entries.

Finally, if nothing works,

Email the Roster PDF to Zawad.Atif@dal.ca stating the issue.

8. Notes on Reliability

Also, it's worth remembering: no software is perfect. There will always be edge cases, unexpected formats, or rare inputs that might throw the parser off. While the tool is built to handle most cases reliably, occasional hiccups are part of the process. That's why having human oversight, especially from someone familiar with the system, is still a key part of ensuring accuracy. Think of the software as a helpful assistant: fast, efficient, and mostly accurate, but still needing a second pair of eyes when something looks odd.

This program spans over 5100 lines of code, covering a wide range of logic paths, data structures, and parsing conditions. It has been tested against numerous real-world datasets and is designed to handle a variety of formatting styles and edge cases. However, given the complexity and the variability of input streams, certain unexpected scenarios may still arise. These may include formatting inconsistencies, unanticipated data patterns, or edge cases that haven't yet been encountered in testing. While the system is built to be robust and fault-tolerant, no software can account for every possible input with perfect accuracy. As such, occasional issues should be expected, and human review remains an essential part of the verification process, especially when preparing outputs for logistics or official use.

9. Source & Ownership

This software was developed and is maintained by Zawad Atif and Nafisah Nubah. It was built specifically for internal use at the Dalhousie Student Accessibility Center.

You can find the full source code, report issues, or contribute via the official GitHub repository: <https://github.com/NepSauce/Normalized-Entity-Parser>

For inquiries and to report bug related issues, send an email to Zawad.Atif@dal.ca.

10. Development Overview

Zawad Atif assumed technical leadership for the project, architecting the overall software design and strategically selecting the technology stack, libraries, and dependencies to optimize performance and maintainability. Responsibilities included front-end development, system integration across multiple modules, and the implementation of a Developer Mode diagnostics toolkit to facilitate advanced debugging and streamline feedback collection workflows.

Nafisah Nubah was responsible for the backend engineering, designing, and implementing the core parsing algorithms that enable accurate extraction and normalization of data from exam scheduling PDFs, ensuring robust data processing and validation.

Both contributors engaged collaboratively in comprehensive quality assurance and testing processes to validate software accuracy, enhance reliability, and ensure an intuitive user experience.

Technical Stack and Development Environment

Programming Language: Java

UI Framework: Swing

PDF Parsing Library: Apache PDFBox

IDE: IntelliJ IDEA

Packaging Tool: Launch4j (for creating Windows executables)

Version Control: Git

Repository Hosting: GitHub