A blue and yellow crest with a bird

AI-generated content may be incorrect.

**The University of Technology, Jamaica**

**The School of Computing and Information Technology**

**CIT4004 – Analysis of Programming Languages**

**Lecturer: Dr. David White**

**Kemar Christie 2005904**

**Roberto Davis 2100101**

**Dwayne Gibbs 2007512**

**Tyoni Davis 1701860**

**Danielle Jones 1900398**

**April 5, 2025**

**Group Project**

**Project Report**

* **Paradigm the language you developed belongs to.**The programming language developed in this project adheres to the procedural programming paradigm. This paradigm is centered around the concept of procedure calls, where the program is structured into procedures or routines that operate on data. It emphasizes a step-by-step sequence of instructions, making it suitable for tasks that require a clear and logical flow of control.
* **Explaining whether your language is general purpose or domain specific.**Our booking programming language is domain-specific. It is designed solely to facilitate booking operations such as scheduling, availability tracking, reservation processing, and user management. Because it focuses on this specialized area and lacks the broad capabilities of a general-purpose language, it cannot be used effectively outside the booking domain.
* **Explaining whether your language is low level or high level.**

The programming language developed in this project is classified as a high-level language. This classification is based on the fact that it was implemented using Python, a widely recognized high-level programming language, along with parsing tools such as PLY (Python Lex-Yacc). As such, our language prioritizes readability, maintainability, and ease of use , hallmarks of high-level language design.

* **Correct grammar for the language you developed.**

**// Main command types**

START → COMMAND SYMBOL

COMMAND → BOOKING\_COMMAND

| LIST\_COMMAND

| PAYMENT\_COMMAND

| INQUIRY\_COMMAND

| RENT\_COMMAND

| CONFIRMATION\_COMMAND  
 | CANCELLATION\_COMMAND

LIST\_COMMAND → LIST\_KEYWORD RESOURCE LOCATION\_MARKER DEPARTURE LOCATION\_MARKER ARRIVAL SYMBOL

| LIST\_KEYWORD CONTEXT\_KEYWORD RENT\_KEYWORD RESOURCE LOCATION\_MARKER LOCATION SYMBOL

| LIST\_KEYWORD SERVICE CONTEXT\_KEYWORD SYMBOL

| LIST\_KEYWORD SERVICE CONTEXT\_KEYWORD LOCATION\_MARKER DEPARTURE LOCATION\_MARKER ARRIVAL SYMBOL

| LIST\_KEYWORD CONTEXT\_KEYWORD USERNAME SYMBOL

BOOKING\_COMMAND → ACTION\_KEYWORD RESOURCE LOCATION\_MARKER DEPARTURE LOCATION\_MARKER ARRIVAL SYMBOL

| ACTION\_KEYWORD RESOURCE LOCATION\_MARKER ARRIVAL LOCATION\_MARKER DEPARTURE SYMBOL

| ACTION\_KEYWORD RESOURCE LOCATION\_MARKER ARRIVAL LOCATION\_MARKER DEPARTURE CONNECTIVE\_WORD CONTEXT\_KEYWORD CONDITIONS MONEY SYMBOL

| ACTION\_KEYWORD RESOURCE LOCATION\_MARKER ARRIVAL LOCATION\_MARKER DEPARTURE ARTICLE\_CONJUNCTION ARTICLE\_CONJUNCTION RESOURCE LOCATION\_MARKER START\_DATE LOCATION\_MARKER END\_DATE SYMBOL

| ACTION\_KEYWORD SERVICE RESOURCE LOCATION\_MARKER DEPARTURE LOCATION\_MARKER ARRIVAL CONTEXT\_KEYWORD START\_DATE LOCATION\_MARKER TIME CONTEXT\_KEYWORD USERNAME SYMBOL

| ACTION\_KEYWORD RESOURCE LOCATION\_MARKER DEPARTURE LOCATION\_MARKER ARRIVAL CONTEXT\_KEYWORD START\_DATE LOCATION\_MARKER TIME CONTEXT\_KEYWORD CONTEXT\_KEYWORD END\_DATE LOCATION\_MARKER TIME SYMBOL

| ACTION\_KEYWORD RESOURCE LOCATION\_MARKER SERVICE LOCATION\_MARKER START\_DATE LOCATION\_MARKER END\_DATE CONTEXT\_KEYWORD USERNAME SYMBOL

| ACTION\_KEYWORD RESOURCE LOCATION\_MARKER DEPARTURE CONTEXT\_KEYWORD DATE ARRIVAL CONTEXT\_KEYWORD DATE SYMBOL

| ACTION\_KEYWORD RESOURCE LOCATION\_MARKER LOCATION LOCATION\_MARKER START\_DATE LOCATION\_MARKER END\_DATE CONTEXT\_KEYWORD USERNAME SYMBOL

                    | ACTION\_KEYWORD SERVICE RESOURCE LOCATION\_MARKER DEPARTURE LOCATION\_MARKER ARRIVAL CONTEXT\_KEYWORD START\_DATE LOCATION\_MARKER TIME CONTEXT\_KEYWORD NUMBER PASSENGER\_TYPE SYMBOL

                    | ACTION\_KEYWORD NUMBER RESOURCE CONTEXT\_KEYWORD RESOURCE CONTEXT\_KEYWORD DATE SYMBOL

                    | ACTION\_KEYWORD NUMBER TICKET\_TYPE RESOURCE CONTEXT\_KEYWORD RESOURCE CONTEXT\_KEYWORD DATE SYMBOL

LIST\_KEYWORD → ‘List all’ | ‘List’

PAYMENT\_COMMAND → ACTION\_KEYWORD RESOURCE CONTEXT\_KEYWORD SERVICE CONTEXT\_KEYWORD USERNAME SYMBOL

| PAYMENT\_TYPE SYMBOL

CANCELLATION\_COMMAND → ACTION\_KEYWORD RESOURCE LOCATION\_MARKER SERVICE LOCATION\_MARKER START\_DATE LOCATION\_MARKER END\_DATE CONTEXT\_KEYWORD USERNAME SYMBOL

RENT\_COMMAND → RENT\_KEYWORD RESOURCE LOCATION\_MARKER LOCATION LOCATION\_MARKER START\_DATE LOCATION\_MARKER END\_DATE CONTEXT\_KEYWORD USERNAME SYMBOL'''

INQUIRY\_COMMAND → ACTION\_KEYWORD RESOURCE CONTEXT\_KEYWORD LOCATION\_MARKER DEPARTURE LOCATION\_MARKER ARRIVAL SYMBOL

ACTION\_KEYWORD → 'Book a'| 'Confirm a'| 'Pay'| 'Cancel a'| 'Reserve a'| 'How many'| 'Duration of'

CONTEXT\_KEYWORD → 'on'| 'For'| 'Schedule'| 'are there'| 'Returning'| 'cost' | 'available'

RENT\_KEYWORD → 'Rent a' | 'Rent' | 'Rental'

LOCATION\_MARKER → 'in'| 'at'| 'from'| 'to'

CONNECTIVE\_WORD → 'that'

ARTICLE\_CONJUNCTION → 'a' | 'and'

PAYMENT\_TYPE → 'credit card' |'debit card' | 'bank transfer'

RESOURCE → 'Reservations' | 'Reservation' | 'Tickets' | 'Ticket' | 'tickets' | 'Flights' | 'Flight' | 'Rooms' | 'Room' | 'Hotels' | 'Hotel'

CONDITIONS → 'less than' | 'more than' | 'equal to' | 'greater than' | 'if' | 'then'

DATE → 'Jan' NUMBER, NUMBER | 'Feb' NUMBER, NUMBER | 'Mar' NUMBER, NUMBER | 'Apr' NUMBER, NUMBER | 'May' NUMBER, NUMBER | 'Jun' NUMBER, NUMBER | 'Jul' NUMBER, NUMBER | 'Aug' NUMBER, NUMBER | 'Sep' NUMBER, NUMBER | 'Oct' NUMBER, NUMBER | 'Nov' NUMBER, NUMBER | 'Dec' NUMBER, NUMBER

START\_DATE → DATE

PAYMENT → ‘credit card’ | ‘debit card’ | ‘bank transfer’

CONFIRM\_KEYWORD → ‘Confirm a’ | ‘Confirm the’ | ‘Confirm’

INQUIRY\_KEYWORD → ‘How many’ | ‘What is the’

PASSENGER\_TYPE → ‘adults’ | ‘children’ | ‘seniors’ | ‘students’ | ‘adult’ | ‘child’ | ‘senior’ | ‘student’

END\_DATE → DATE

TIME → NUMBER:NUMBER 'AM' | NUMBER:NUMBER 'PM' | NUMBER:NUMBER

NUMBER → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

MONEY → '$' NUMBER | '$' NUMBER.NUMBER

USERNAME → CHAR CHAR CHAR'\_'CHAR CHAR CHAR

CHAR → 'a' | 'b' | 'c' | 'd' | 'e' | 'f' | 'g' | 'h' | 'i' | 'j' | 'k' | 'l' | 'm' | 'n' | 'o' | 'p' | 'q' | 'r' | 's' | 't' | 'u' | 'v' | 'w' | 'x' | 'y' | 'z'

SYMBOL → '.' | '?'

* **Complete parse tree/AST for a sample program in your language.**sample\_input\_data = “Book a Ticket from Montego Bay to Miami on February 17, 2025 at 8:30 AM returning on March 17, 2025 at 8:30 AM.

**Parsed Result:**

('COMMAND', ('BOOKING\_COMMAND', ('ACTION\_KEYWORD', 'Book a'), ('RESOURCE', 'Ticket'), ('LOCATION\_MARKER', 'from'), ('DEPARTURE', 'Montego Bay'), ('LOCATION\_MARKER', 'to'), ('ARRIVAL', 'Miami'), ('CONTEXT\_KEYWORD', 'on'), ('START\_DATE', 'Feb 17, 2025'), ('LOCATION\_MARKER', 'at'), ('TIME', '8:30 AM'), ('CONTEXT\_KEYWORD', 'returning'), ('CONTEXT\_KEYWORD', 'on'), ('END\_DATE', 'Mar 17, 2025'), ('LOCATION\_MARKER', 'at'), ('TIME', '8:30 AM'), ('SYMBOL', '.')))

* **Full list of tokens for the language you developed.**
  + **Tokens:**
* ACTION\_KEYWORD
* CONTEXT\_KEYWORD
* LOCATION\_MARKER
* CONNECTIVE\_WORD
* DATE
* START\_DATE
* END\_DATE
* NUMBER
* SYMBOL
* MONEY
* RESOURCE
* CONDITIONS
* TIME
* USERNAME
* DEPARTURE
* ARRIVAL
* LOCATION
* SERVICE
* ARTICLE\_CONJUNCTION
* PAYMENT\_TYPE
* INQUIRY\_KEYWORD
* PASSENGER\_TYPE
* CONFIRM\_KEYWORD
* TICKET\_TYPE
* **Regular expressions you used to recognize all the tokens for the language you developed.**

**Regular Expressions Used:**

* ACTION\_KEYWORD
  + Array specified in reg ex: action\_keywords = [ 'Book a', 'Book', 'Pay', 'Cancel a', 'Reserve a', 'How many', 'Duration of', 'Booking']
  + t\_ACTION\_KEYWORD = r'\b(?:' + r'|'.join(action\_keywords) + r')\b'
* CONTEXT\_KEYWORD
  + Array specified in reg ex: context\_keywords = ['on', 'For', 'Schedule', 'are there', 'Returning', 'cost', 'available']
  + t\_CONTEXT\_KEYWORD = r'\b(?:' + r'|'.join(context\_keywords) + r')\b'
* LOCATION\_MARKER
  + Array specified in reg ex: location\_markers = ['in', 'at', 'from', 'to']
  + t\_LOCATION\_MARKER = r'\b(?:' + r'|'.join(location\_markers) + r')\b'
* CONNECTIVE\_WORD
  + Array specified in reg ex : connective\_words = ['that']
  + t\_CONNECTIVE\_WORD = r'\b(?:' + r'|'.join(connective\_words) + r')\b'
* DATE
  + t\_DATE = r'(?<=\bon\s)((?!\b(?:in|at|from|to)\b).)+?(?=\.)'
* START\_DATE
  + t\_START\_DATE = r'(?<=\bfrom\b\s).+?(?=\s\bto\b)|(?<=\bon\b\s).+?(?=\s\bat\b)'
* END\_DATE
  + t\_END\_DATE = r'(?<=\breturning on\s).+?(?=\s(?:at)\b)|' \

r'(?<=\bto\s).+?(?=\s(?:for)\b)|' \

r'(?<=\bto\s).+?(?=\.)'

* NUMBER
  + t\_NUMBER = r'\b\d+\b'
* SYMBOL
  + t\_SYMBOL = r'\.+(?=[ \t]\*$)|,|:'
* MONEY
  + t\_MONEY = r'\$\d+(\.\d+)?'
* RESOURCE
  + t\_RESOURCE = (

r'(?<=\bRent a\s|Rental\s|Book a\s)([A-Za-z]+)(?=\sin)|'

r'Reservations|Reservation|Tickets|Ticket|tickets|Flights|Flight|Rooms|Room|Hotels|Hotel|'

r'(?<=\bfor\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s\bon\b)'  
)

* CONDITION
  + t\_CONDITIONS = r'\b(?:less than|more than|equal to|greater than|if|then)\b'
* TIME
  + t\_TIME = r'\b(?:([0-9])?[0-9]):[0-9][0-9]\s\*(?:AM|PM)\b'
* USERNAME
  + t\_USERNAME = r'(?<=\bfor\b\s)[A-Za-z0-9\_]+(?=\.)'
* DEPARTURE
  + t\_DEPARTURE = r'(?<=\bfrom\b\s)([a-zA-Z\s]+?)(?=\s\band\b)|(?<=\bFrom\b\s)([a-zA-Z\s]+)(?=\s\bTo\b)|(?<=\bFrom\b\s)([a-zA-Z\s]+)(?=\s\bThat\b)|(?<=\bFrom\b\s)([a-zA-Z\s]+)(?=\s\*\.)|(?<=\bFrom\b\s)([a-zA-Z\s]+)(?=\s\b(?:' + r'|'.join(all\_keywords) + r')\b)'
* ARRIVAL
  + t\_ARRIVAL = r'(?<=\bTo\b\s)([a-zA-Z\s]+?)(?=\s\bFrom\b)|'\

r'(?<=\bTo\b\s)([a-zA-Z\s]+?)(?=\s\*\.)|'\

r'(?<=\bTo\b\s)([a-zA-Z\s]+?)(?=\s\b(?:' + r'|'.join(all\_keywords) + r')\b)'

* LOCATION
  + t\_LOCATION = r'(?<=\bin\b\s)([a-zA-Z\s]+?)(?=\s\b(?:' + r'|'.join(all\_keywords) + r')\b)|' \

r'(?<=\bin\b\s)([a-zA-Z\s]+?)(?=\s\*\.)|' \

r'(?<=\bin\s)([a-zA-Z\s]+?)(?=\s\bfrom\b)'

* SERVICE
  + t\_SERVICE = r'(?<=\ba\s)(?!(?:' + r'|'.join(all\_keywords) + r')\b)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s(?:' + t\_RESOURCE + r')\b)|' \

r'(?<=\bList\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s\bSchedule\b)|' \

r'(?<=\bList all\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s\bSchedule\b)|' \

r'(?<=\bfor\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s\bfor\b)|' \

r'(?<=\bat\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s(?:From|from)\b)|'\

r'(?<=\bfor\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s(?:From|from)\b)|'\

r'(?<=\bConfirm the\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s(?:' + t\_ACTION\_KEYWORD + r')\b)|'\

r'(?<=\bConfirm a\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s(?:' + t\_ACTION\_KEYWORD + r')\b)|'\

r'(?<=\bConfirm\s)([A-Za-z]+(?:\s[A-Za-z]+)?)(?=\s(?:' + t\_ACTION\_KEYWORD + r')\b)'

* ARTICLE\_CONJUNCTION
  + t\_ARTICLE\_CONJUNCTION = r'\b(a|and)\b'
* PAYMENT
  + t\_PAYMENT\_TYPE = r'\b(credit card|debit card|bank transfer)\b'
* CONFIRM\_KEYWORD
  + t\_CONFIRM\_KEYWORD = r'\b(Confirm a|Confirm the|Confirm)\b'
* INQUIRY\_KEYWORD
  + t\_INQUIRY\_KEYWORD = r'\b(How many|What is the)\b'
* PASSENGER\_TYPE
  + t\_PASSENGER\_TYPE=r'\b(adults|children|seniors|students|adult|child|senior|student)\b'
* TICKET\_TYPE
  + t\_TICKET\_TYPE = r'(?<=\d\s)(.\*?)(?=\s(?:ticket|tickets))'
* **Demonstration of scope and binding in sample code written in your programming language.**

**User Scope and Binding**

Our booking language implements a user context to maintain information associated with the current user throughout a session. When a user is identified by providing a username, this information is bound to a user context.

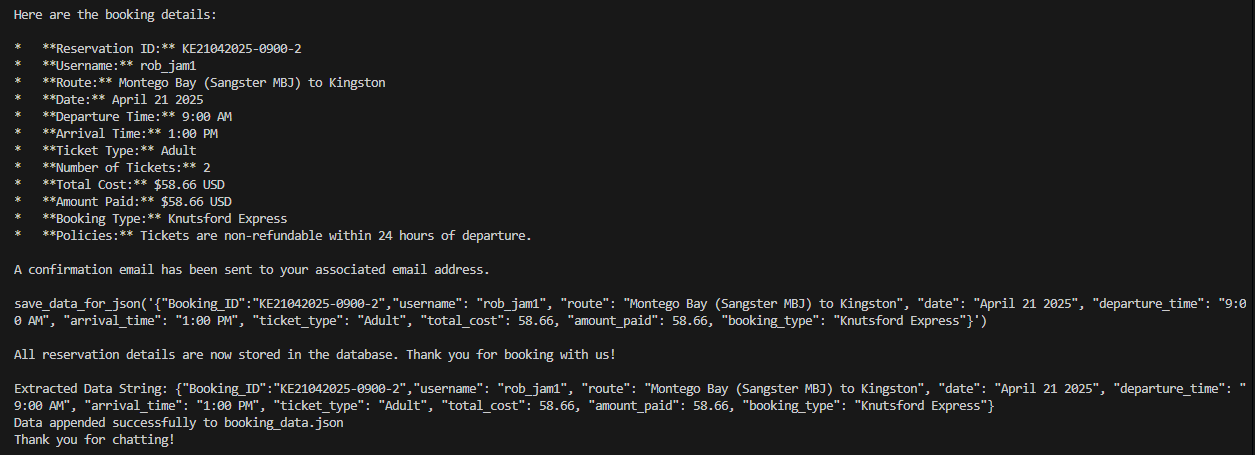
This binding ensures that subsequent actions are correctly performed on behalf of that specific user.

For instance:

* Scenario:  
  + The user initiates a booking by providing their username: Confirm the Knutsford Express booking for rob\_jam1.
  + The system establishes a binding between the user and 'rob\_jam1'.
  + Later, the user issues another confirmation: Confirm the Knutsford Express booking.
  + In this subsequent prompt, the username 'rob\_jam1' does not need to be explicitly repeated because it remains bound to the user context. The system implicitly operates within the scope of that user's context.

This approach demonstrates how our language manages scope by maintaining the user context. The user identifier retains its binding within a defined scope, allowing actions to correctly reference the associated user.





**Reservation Scope and Data Binding**

Each booking within our language is associated with a unique reservation context. This context is established when a booking is confirmed and is crucial for maintaining the integrity of booking-specific data. A key aspect of this context is the binding of a unique reservation ID to a collection of data representing the booking details.

The save\_data\_for\_json() function exemplifies this binding. When the AI want to store information it outputs save\_data\_for\_json(…..). There is some code that identifies when the AI responds says the trigger word save\_data\_for\_json and it takes the data in the parameter and creates a structured data representation (a JSON object) where specific booking details (e.g., Booking\_ID, username, route, etc.) are bound to corresponding keys. This JSON object then effectively acts as the scope for all information pertaining to that particular reservation.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.  
A black background with white text

AI-generated content may be incorrect.

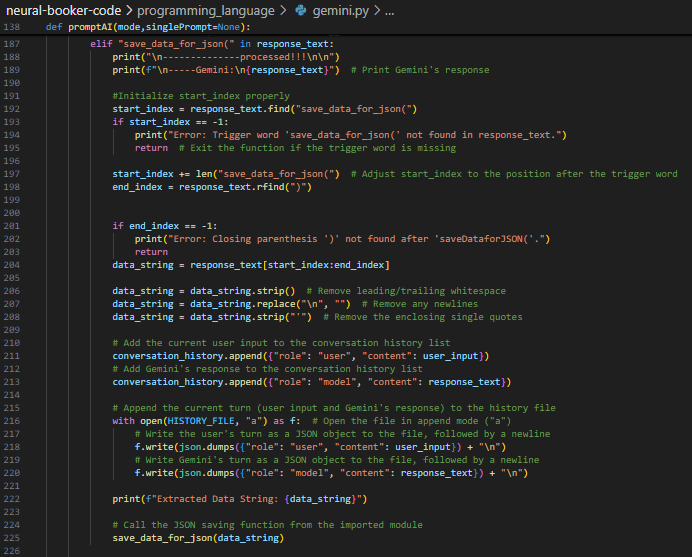
**Data saved in a JSON file:**

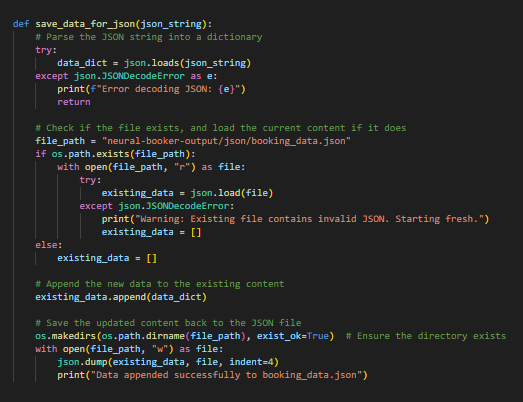
A screen shot of a computer

AI-generated content may be incorrect.

* + - save\_data\_for\_json('{"Booking\_ID":"KE21042025-0900-2","username": "rob\_jam1", ...}')
    - Here, keys such as Booking\_ID, username, etc., are bound to their respective values. This JSON object becomes the scope for all information related to reservation 'KE21042025-0900-2'.
    - This data can further be used to retrieving, modifying, or canceling the booking.

**Code that tracks AI response checking for trigger word**





* **Details on the programming language you used to develop your compiler.**The compiler for the developed programming language was implemented using Python, a high-level, general-purpose programming language known for its readability, simplicity, and extensive standard library. Python was chosen due to its strong support for rapid development, ease of syntax, and availability of robust third-party tools for language processing. To handle the lexical analysis and parsing phases of the compiler, the PLY (Python Lex-Yacc) library was utilized. PLY provides implementations of lexers (lex) and parsers (yacc) similar to the traditional Lex and Yacc tools found in C, but entirely written in Python. This allowed for efficient tokenization and syntax analysis while maintaining full integration within the Python ecosystem. The combination of Python and PLY enabled a clean, modular design for the compiler and significantly accelerated development by abstracting away many of the lower-level details typically associated with compiler construction.
* **Two characteristics of a good programming language (from those you studied in class) that are evident in your designed programming language, and examples of how do these characteristics affect the readability, writability and reliability of your designed programming language.**

**Syntax Design**

Natural language-like syntax allows users to interact with a programming language in a way that closely resembles everyday human communication. This enhances accessibility and

makes the language intuitive, especially for users without extensive programming experience.

**Example in Our Design**

Our language is designed to enable users to issue commands naturally, as if they were speaking or writing in regular conversation. This is achieved by supporting flexible, human-readable

constructs, such as:

* + - Flexible Phrasing:

Users can express similar commands in multiple natural ways without losing functionality:

* List all Knutsford Express schedules from Kingston to Montego Bay.
* What is the schedule for Knutsford Express from Kingston to Montego Bay on Apr 21, 2025?"`

Both commands are tokenized and parsed correctly, allowing seamless interaction.

**Impact on Readability, Writability, and Reliability**

* Readability:
* Commands mimic natural conversation, making it simple for users to understand the intent of a command at a glance.
* For example, What is the schedule for Knutsford Express? is inherently more intuitive than a highly technical query format.
* Writability:
* Users don’t need to memorize complex syntax rules or specific conventions. They can write commands naturally, reducing the learning curve and improving productivity.
* Reliability:
* By supporting natural syntax and case insensitivity, the language reduces the likelihood of syntax errors, ensuring commands are processed accurately even if users deviate slightly from standard phrasing.

**Simplicity**

Simplicity in a programming language refers to the ease with which users can learn, write, and understand commands. A simple language minimizes unnecessary complexity, allowing users to focus on their tasks without distraction.

**Example in Our Design**

Our language is case insensitive, meaning users can write commands in any case fully lowercase, uppercase, or a mix and the lexer will tokenize them correctly. This reduces the cognitive load on users, making the language easier to work with.  
Command in Mixed Case: “List all Knutsford Express schedule”

Command in Mixed Case: “list all knutsford express schedule”

Command in Mixed Case: “LIST ALL KNUTSFORD EXPRESS SCHEDULE”

All these variations produce the same tokenized output, ensuring that users can work naturally without worrying about letter casing. **Impact on Readability, Writability, and Reliability**

* Readability:
  + Commands are easy to read, regardless of how the user chooses to capitalize text.
* Writability:
  + Users are not burdened with strict capitalization rules, allowing for flexible and intuitive input
* Reliability:
  + By normalizing case sensitivity, the language avoids errors caused by mismatched casing, ensuring consistent interpretation of commands.

**Contributions**

|  |  |
| --- | --- |
| **Name** | **Contribution** |
| Kemar Christie | Creation of Lexer & Parser, Database & Documentation |
| Roberto James | Creation of Lexer & Parser, Training of AI & Documentation |
| Dwayne Gibbs | Creation of Lexer & Parser, Scraping Tools Implementation & Documentation |
| Tyoni Davis | Creation of Lexer & Parser, Database & Documentation |
| Danielle Jones | Creation of Lexer & Parser, Training of AI & Documentation |