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 -> LIST\_COMMAND  
 | BOOKING\_COMMAND

| CONFIRMATION\_COMMAND  
 | CANCELLATION\_COMMAND

| PAYMENT\_COMMAND

| INQUIRY\_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

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

LIST\_KEYWORD -> List all | List

PAYMENT\_COMMAND -> ACTION\_KEYWORD RESOURCE CONTEXT\_KEYWORD SERVICE CONTEXT\_KEYWORD USERNAME SYMBOL

CANCELLATION\_COMMAND -> ACTION\_KEYWORD RESOURCE LOCATION\_MARKER SERVICE 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

**// Terminal Tokens**

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

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'

DEPARTURE -> departure

ARRIVAL -> arrival

LOCATION -> location

SERVICE -> service

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
* **Regular expressions you used to recognize all the tokens for the language you developed.**

**Regular Expressions Used:**

* ACTION\_KEYWORD
  + t\_ACTION\_KEYWORD = r'\b(?:' + r'|'.join(action\_keywords) + r')\b'
* CONTEXT\_KEYWORD
  + t\_CONTEXT\_KEYWORD = r'\b(?:' + r'|'.join(context\_keywords) + r')\b'
* LOCATION\_MARKER
  + t\_LOCATION\_MARKER = r'\b(?:' + r'|'.join(location\_markers) + r')\b'
* CONNECTIVE\_WORD
  + 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'Reservations|Reservation|Tickets|Ticket|tickets|Flights|Flight|Rooms|Room|Hotels|Hotel'
* CONDITIONS
  + 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\*\.)'

* 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'(?<=\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)'

* ARTICLE\_CONJUNCTION
  + t\_ARTICLE\_CONJUNCTION = r'\b(a|and)\b'
* **Demonstration of scope and binding in sample code written in your programming language.**
* **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.**

In the development of our programming language, two essential characteristics stood out: Syntax Design and Simplicity. These attributes, both emphasised in the theoretical foundation of programming language design, directly influenced the readability, writability, and reliability of our language.

Syntax Design  
A well-designed syntax is critical for enhancing both the readability and writability of a programming language. In our language, we adopted a clear and consistent syntax structure, modeled in part after widely accepted conventions in modern languages such as Python and C. For example, our language uses recognizable keywords, avoids excessive punctuation, and maintains predictable patterns in expression and statement formatting.

* + Readability: Users can easily understand the flow of code due to the familiar and logical structure.
  + Writability: Developers can express concepts with minimal confusion or ambiguity, reducing the risk of syntax errors.
  + Reliability: A consistent syntax design makes it easier to implement error handling in the compiler and reduces unintended behavior during execution.

Simplicity

Simplicity in language design helps reduce cognitive load for the programmer. In our case, we deliberately limited the number of constructs and avoided complex features that could confuse beginners or lead to misuse. For instance, instead of supporting multiple types of loops or complex object-oriented structures, we focused on basic control flow, arithmetic operations, and simple variable declarations. This makes the language more approachable for new programmers while maintaining enough power to perform practical tasks.

* Readability: Less complexity means that code can be quickly scanned and understood, especially by novices.
* Writability: Programmers can focus on solving problems without needing to memorize a large set of rules or syntax variants.
* Reliability: A simpler language reduces the likelihood of unexpected behavior and makes it easier to test and debug programs.