Module 3

- Step by step define new features : already done in previous documentation

- Start with basic minimum programming features : already done in previous documentation

Test Case 1: Correct Variable Declarations

File: test_var_decl.rcb

Program:

```
a:number = 42
b:number = 3.14
c:text = "Hello RCBScript"
d:bool = yes
e:char = 'Z'

h = make handle<number>
h = 99

result("a = " + a)
result("c = " + c)
```

Test Case 2: Type Error

File: test_type_error.rcb

Program:

```
x:number = 10
x = "wrong"   # ERROR: expected number, got text
```

Expected Compiler Output:

```
Line 2: Type Error → expected number, got text
```

Test Case 3: Handle Error

File: test_handle_error.rcb

Program:

```
p = make handle<number>
discard q    # ERROR: undeclared handle
```

Expected Compiler Output:

```
Line 2: Reference Error → discard called on undeclared handle
```

Test Case 4: Initialization Error

File: test_init_error.rcb

Program:

```
z:number
result(z)   # ERROR: variable used before assignment
```

Expected Compiler Output:

```
Line 2: Initialization Error → variable used before assignment
```

Test Case 5: Testing Framework Example

File: test_framework.rcb

Program:

```
func sum(a:number, b:number) -> number
  reply a + b

test "sum of integers"
  expect sum(2,3) is 5

test "variable type test"
  x:number = 5
  y:text = "hello"
  expect x is 5
  expect y is "hello"
```

Work Division

Vineeth

- Created and documented Test Cases 1, 2, and 4
- Wrote example programs for correct declarations, type error, and initialization error
- Drafted expected compiler output messages for these cases

Lachiram Nayak

- Created and documented Test Cases 3 and 5
- Wrote example programs for handle error and testing framework integration
- Drafted expected compiler output messages for these cases