Design Document: Lyric

Team - 6 Members:

- Abhijna Maiya (@abhijnamaiya)
- Darshan Phaldesai (@dphaldes)
- Hitha Shamasundar (@hshamasu)
- Shreyas Baburayanakoppal Sunil (@sbabura1)

About the language:

Lyric - It uses references to popular song and music for syntax. This makes obvious separation between task/code logic and programming language constructs.

Language Design:

Data Types	num (number), bool (boolean), str (string)
Binary Arithmetic Operations	+, -, *, /
Assignment	play
Comparison	==, <, >, <=, >=
Ternary Operator	(expr) ? (expr) : (expr)
Condition	check (expr) here { code_block } there { code_block }
Iteration	loop (expr) { code_block }, repeat (num) { code_block }
Reserved Keywords	num, bool, str, check, here, there, loop, repeat, yeah, nah, release, play
Symbols	{, }, (,), ;, +, -, *, /, =, ==, <, <=, >, >=, whitespace, newline
Identifiers Should begin as a non-digit character	Should begin as a non-digit character

Grammar Rules:

<digit> ::= '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
<char> ::= 'a' | 'b' | … | 'z'
<num> ::= <num> <digit> | <digit>
<cpr_sym> ::= ‘<’ | ‘<=’ | ‘>’ | ’>=’ | ‘==’
<data_type>::= num | bool | str
<bool_value>::= yeah | nah
<str> ::= <str> ::= { <char> }

Statement

<stmt> ::= <expr> ';'

```
        | <dec_stmt> ';'
        | <loop_stmt>
        | <repeat_stmt>
        | <check_stmt>
        | '{' <stmts> '}'
<stmts> ::= <stmts> <stmt> | ε
```

Expression

```
<expr> ::= <id>
        | <assign_expr>
        | <math_expr>
        | <cpr_expr>
```

Declaration

```
<dec_stmt> ::= <data_type> <id> | <data_type> <assign_expr>
```

Identifier

```
<id> ::= <char> | <id> <char> | <id> <digit>
```

Assignment expression

```
<assign_expr> ::= play <id>  <expr>
```

Mathematical expression

```
<math_expr>. ::= <math_expr> '+' <math_term>
               | <math_expr> '-' <math_term>
               | <math_term>

<math_term> ::= <math_term> '*' <math_factor>
               | <math_term> '/' <math_factor>
               | <math_factor>

<math_factor> ::= <num> | '(' <math_expr> ')'
```

while-loop statement

```
<repeat_stmt> ::= repeat '(' <expr> ')' <stmt>
```

if statement

```
<check_stmt> ::= check '(' <expr> ')' here <stmt>
               | check '(' <expr> ')' here <stmt> there <stmt>
```

Comparison expression

```
<cpr_expr> ::= <cpr_term> <cpr_sym> <cpr_term>
<cpr_term> ::= <id> | <number> | '(' <math_expr> ')'
```