IMPORTANT NOTE: Unless mentioned below, all assumptions are open to your interpretation. Please make your own assumptions and build this system. We do not entertain any queries or clarifications associated with the problem statement below. As a team, you are open to drawing your own boundaries. Your grade has no impact on incompleteness of your assumptions.

Problem Code: YantraBhashi

Problem Statement: **Yantrabhasha** is a fictional programming language designed for students to build a syntax and semantic validator using Native JavaScript. The language includes only variable declarations with `PADAM` and `ANKHE`, conditionals using `ELAITHE` and `ALAITHE`, and loops with `MALLI-MALLI`, all enclosed in square brackets. Students must ensure correct syntax, variable declarations before use, and enforce integer-only rules. The validator should output errors and recommendations whenever code does not comply with Yantrabhasha's simple grammar.

Skills: Your team will be assessed on Git proficiency, database design (SQL & NoSQL), full-stack development with MERN, and problem-solving in handling complex data relationships and workflows.

Deliverables:

Part 1: Yanthrabhashi Validator (40%)
For the **Yantrabhasha Validator**, the deliverable is a **single-page web application** built with Native JavaScript. It should take Yantrabhasha code as input, validate it against the defined rules, and display clear **error messages and recommendations** for any violations. The tool must handle both **syntax and semantic checks** and show results in an easy-to-read format for students.

Part 2: Yanthrabhashi Validation results – MERN (60%)
Extend the validator into a full-stack MERN app where React serves the frontend input/output, Node.js + Express handle the validation logic, and MongoDB stores validation results (errors and successes). This allows instructors to track submissions, analyze patterns, and provide feedback at scale.

Here are the **documentation rules for Yantrabhasha**:

1. **Code Blocks** must be enclosed in square brackets `[ ... ]` for all conditional and loop bodies.

2. **Variables**: Declared with `PADAM name:ANKHE = <integer>;`. Every variable must be declared before use. ANKHE is for integer, VARTTAI is a string like example PADAM msg:VARTTAI = "Hello World";
3. **Data Types**: Only `ANKHE` (integer) is supported; no strings, floats, or booleans.
4. **Conditionals**: Use `ELAITHE (condition) [ ... ] ALAITHE [ ... ]`. Conditions must use integers and operators `==, !=, <, >, <=, >=`.
5. **Loops**: Use `MALLI-MALLI (PADAM i:ANKHE = 0; i < N; i = i + 1) [ ... ]`. Update step must be explicit (`i = i + 1`).
6. **Statements**: All must end with a semicolon `;` except block headers (`ELAITHE`, `ALAITHE`, `MALLI-MALLI`).
7. **Identifiers**: Must start with a letter, can include letters, digits, underscores; reserved words (`PADAM, ANKHE, VARTTAI ELAITHE, ALAITHE, MALLI-MALLI`) cannot be used as names.
8. **Print and Scan**: CHATIMPU is for printing and CHEPPU is for scaning the input
9. **Errors**: Validator should flag undeclared use, missing semicolons, type mismatches, or malformed conditions/loops.

Examples of Yathra Bhashi

Hello World program

1. PADAM message:VARTTAI = "Hello World";
2. CHATIMPU(message);

#Addition program

1. PADAM a:ANKHE;
2. PADAM b:ANKHE;
3. PADAM sum:ANKHE = 0;
4. CHEPPU(a);
5. CHEPPU(b);
6. sum = a + b;
7. CHATIMPU("The Sum is:");
8. CHATIMPU(sum);

Conditional Statements

1. PADAM username:VARTTAI;
2. CHEPPU(username);
3. ELAITHE (username == "Anirudh") [

```
    CHATIMPU("Welcome Anirudh!");
] ALAITHE [
    CHATIMPU("Access Denied!");
]
```

Printing Sum of First 10 numbers

```
PADAM i:ANKHE;
PADAM sum:ANKHE = 0;
 MALLI-MALLI (PADAM i:ANKHE = 1; i <= 10; i = i + 1) [
    sum = sum + i;
]
 CHATIMPU("Sum of first 10 numbers is:");
CHATIMPU(sum);
```

Submission Criteria:

- **Source Code:** Git repo link with all frontend and backend code (commit history reviewed).
- **README.md** with MERN implementation details and set-up steps along with short details on design decisions, solution diagram.
- **Submit Peer Review Form**: https://forms.office.com/Pages/ResponsePage.aspx?id=vDsaA3zPK06W7IZ1VVQKHFzW4INMf2JMjyL9qPnlPbNURjRUSDA0QURKWFBCTzVCNFdDU1pDVTJZSy4u
- **LLM Usage Form**: https://forms.office.com/Pages/ResponsePage.aspx?id=vDsaA3zPK06W7IZ1VVQKHFzW4INMf2JMjyL9qPnlPbNUNFFSOTM3MDE4T0NPWUpNOTZKSFFQSUJTRC4u