BETA TEST PLAN – GLU

1. Core Functionalities for Beta Version

Feature Name	Description	Priority (High/Medium/Low)	Changes Since Tech3
Parsing of Glu code	The system should be able to parse Glu code and provide error messages for invalid programs.	High	*N/A*
Type Checking	The system should be able to perform type checking on the parsed AST.	Low	*N/A*
Code Generation	The system should be able to generate LLVM IR code from the Glu code.	High	*N/A*

2. Beta Testing Scenarios

2.1 User Roles

Role Name	Description
Glu User	A developer who uses Glu code in their codebase
Glu Wizard	A developer who works on the Glu compiler

2.2 Test Scenarios

****Scenario 1: Parsing of Glu code ****

- **Role Involved:** Glu User
- **Objective:** The Glu compiler should be able to parse Glu code and provide error messages for invalid programs.
- **Preconditions:** A set of Glu codebases to test the parser (valid and invalid).
- **Test Steps:**
    1. Run the Glu compiler on valid Glu code.
    2. Ensure that the compiler successfully parses the code.
    3. Run the Glu compiler on invalid Glu code.
    4. Ensure that the compiler provides meaningful error messages.
- **Expected Outcome:** The Glu compiler should parse valid Glu code without errors and provide meaningful error messages for invalid code.

Scenario 2: Code Generation

- **Role Involved:** Glu User
- **Objective:** The Glu compiler should be able to generate machine code from Glu code using LLVM.
- **Preconditions:** A set of Glu code files to test the code generator, that use functions, loops, conditionals, structs, enums, etc.
- **Test Steps:**
    1. Run the Glu compiler on Glu code.
    2. Ensure that the compiler generates a binary executable.
    3. Run the binary executable.
    4. Ensure that the executable produces the expected output.
- **Expected Outcome:** The Glu compiler should generate a binary executable from Glu code that produces the expected output.

3. Success Criteria

The following criteria will be used to determine the success of the beta version.

Criterion	Description	Threshold for Success
Stability	No compiler crashes	No crash reported
Usability	Glu Users can develop effectively in Glu	60% of Glu Users are satisfied with the development experience
Compiler		Compile time is less than 5 seconds for a medium-sized program (≈4k

Criterion	Description	Threshold for Success
Performance	The Glu compiler is fast to compile	LOC)
Correctness	The Glu compiler generates correct code	100% of correctness test cases pass
Program Performance	The generated code is performant	The generated code is on par with C/C++ code (within 20%)

4. Known Issues & Limitations

Issue	Description	Impact	Planned Fix? (Yes/No)
Incomplete Parser	The parser does not generate an AST yet	High	Yes
Type Checking	The type checker is not implemented yet	Medium	Yes
GILGen	The Glu Intermediate Language Generator is not implemented yet	High	Yes
IRGen	The LLVM IR generator is not implemented yet	High	Yes

5. Conclusion

The beta version of the Glu compiler is expected to provide basic functionality for parsing Glu code and generating LLVM IR code. The beta version of the Glu compiler will be tested by Glu Users and Glu Wizards to ensure that it meets the core requirements and is stable for further development. It lays the foundation for future enhancements and improvements to the Glu compiler, while it will lack the groundbreaking features of the final version, namely the decompiler and the language integration.