Software Requirements Specification

AsaanLang

Version 1.0 approved

Prepared by:
Fayyez Farrukh (22L-7987)
Eehab Sadaat (22L-7944)
Zain Iqbal (22L-7897)
Ahmed Karim (22L-7857)
Uffan Mehmood Khan (22L-7947)

Dream League V

Created: 12-30-2023 submitted: 30-10-2023

Table of Contents

Name	**Date**	**Reason of change**	**version**
AsaanLang_SRS	30-11-23	First draft	1.0

1. Introduction

1.1 Purpose

AsaanLang is designed to bridge the development gap between native Urdu/Hindi speakers and the software industry. The primary purpose is to empower Urdu/Hindi speakers by providing them with a programming language featuring a simple Roman Urdu syntax. This initiative aims to promote linguistic inclusivity, create opportunities for South Asians in the field of software development, and cater to both novice and experienced programmers who may not have native-level English proficiency.

1.2 Document Conventions

The software requirements specifications for AsaanLang were created following the software and systems engineering standards of the Institute of Electrical and Electronics Engineers (IEEE). Each system feature specified has been listed in the order of the priority of implementation, such that every preceding feature possesses higher priority of implementation than, and is necessary for, the succeeding feature(s). Meanwhile the requirements tracing out from each respective feature, for that feature, possess equal priority, although inheriting priority over requirements over other features from their parent feature.

1.3 Intended Audience and Reading Suggestions

1.3.1 Intended Audience:

- **Developers**: Developers will be interested in the technical details of AsaanLang, understanding its syntax, features, and how it transpiles to Python.
- **Project Managers**:Project managers need an overview of the project scope, objectives, timelines, and resource requirements.
- **Users**:Users, especially novice and experienced programmers, will want to understand how to use AsaanLang, its syntax, and its advantages.
- **Testers**: Testers need information on testing requirements, expected behaviors, and potential areas of concern.
- **Documentation Writers:** Documentation writers require a deep understanding of the language and its features to create user-friendly guides and manuals.

1.3.2 Document Overview:

This document has been written according to IEEE standards following shows a short overview of the document:

- **Introduction:** Sets the stage for the SRS, highlighting the purpose, intended audience, and providing a brief scope of the AsaanLang project.

- **Overall description:** Expands on the context and origin of AsaanLang, elucidating its positioning as a standalone product addressing linguistic and cultural gaps in software development. Describes its functions, user classes, operating environment, and any design and implementation constraints.
- **External Interface Requirements:** Details the external interfaces that AsaanLang interacts with, including user interfaces, hardware interfaces, software interfaces, and communication interfaces.
- **System Features:** Lists specific functionalities of AsaanLang. Each feature is described in detail, including inputs, processes, and outputs. Examples may include language syntax, transpilation process, and other key aspects contributing to the overall functionality of AsaanLang.

1.4 Product Scope

The scope of AssanLang is to create a user-friendly language learning platform tailored for professionals having Urdu/Hindi as their primary language and struggling to learn, adapt and catch-up to the rapidly changing technological landscape.

1.5 References

1. Institute of Electrical and Electronics Engineers (IEEE), Inc. (1998). IEEE Standard 1016-1998, IEEE Recommended Practice for Software Requirements Specifications. New York, NY, USA: IEEE, Inc. Retrieved from http://www.math.uaa.alaska.edu/~afkjm/cs401/IEEE830.pdf
2. Florianus, C. C., & Syamsi, V. (2021). Error analysis of inflectional affixation in academic writing of freshman students. LLT Journal: A Journal on Language and Language Teaching, 24(2), 471–492. https://doi.org/10.24071/llt.v24i2.2759
3. PEP 8 – Style guide for Python code | peps.python.org. (n.d.). https://peps.python.org/pep-0008/
4. Project Proposal & Elaboration Techniques Breakdown. (n.d.). Google Docs. https://docs.google.com/document/d/1GadODY7vvYVBy-X4sizUMyfWT8Hec_URRvg__LJAQ-Q/edit
5. Asaanlang. (n.d.). Figma. https://www.figma.com/file/ewxOP27QDS1tcOXsLFUB9v/asaanlang?type=design&node-id=42%3A36&mode=design&t=LyI18GMBjjrUFGFw-1
6. https://docs.google.com/forms/d/1dY75SA5S3ZiJxFMPOSHtqQND1vknh8r7jc_XclCmkOI/edit
7. AsaanLang github repository: https://github.com/eehab-saadat/AsaanLang

2. Overall Description

2.1 Product Perspective

AsaanLang emerges as a novel, self-contained product aimed at addressing the linguistic and cultural gaps in the software development industry. This section provides insight into the context, origin, and positioning of AsaanLang within the broader software ecosystem.

2.1.1 Context and Origin

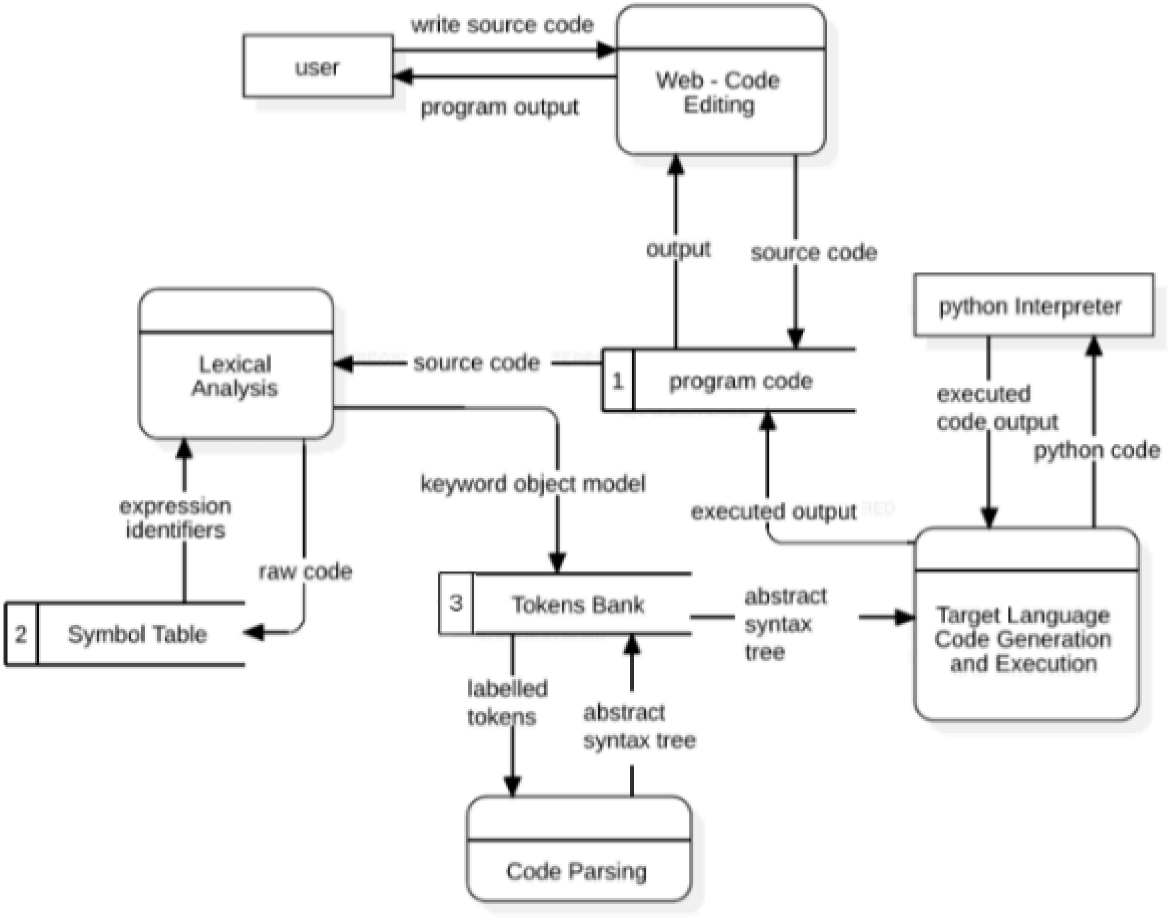
AsaanLang is conceived as a groundbreaking programming language with Roman Urdu syntax, designed to empower Urdu/Hindi speakers in the field of software development. The absence of a high-level programming language catering to the linguistic preferences of this demographic prompted the development of AsaanLang. Its origin lies in the need to bridge the gap between native Urdu/Hindi speakers and the software industry, fostering inclusivity and diversity.

2.1.2 Product Positioning

- **Self-contained Product:** AsaanLang is designed to be a standalone product, serving as a comprehensive programming language rather than a component or add-on to an existing system.
- **Cultural Empowerment:** It positions itself as a tool to empower Urdu/Hindi speakers, promoting cultural inclusivity and creating opportunities for a segment of the population that might face challenges due to language barriers in the software development domain.

2.1.3 Subsystem Interconnections: The AsaanLang transpiler interacts with the Python runtime environment to ensure that the transpiled code adheres to Python syntax and can be seamlessly executed.

The following data flow diagram describes the correlation of different components and modules of the transpiler.



2.2 Product Functions

2.2.1 Roman Urdu/Hindi Based Programming

- AsaanLang is an experimental programming language transpiled down to python, with a Roman Urdu/Hindi syntax.
- Through AsaanLang, our team aims to empower urdu/hindi speakers by developing a programming language with simple *Roman Urdu* syntax which shall promote linguistic inclusivity and open new doors of opportunities for south asians in terms of career options and global competency. It shall also offers an opportunity for reduced learning curve and improved logic building for native Urdu/Hindi speakers

2.2.2 Programming Interface

- Provision of a console or web-app based interface for language usage and execution and for input/output actions.

2.2.3 Miscellaneous Features

- Creation of comprehensive documentation in Latin Urdu script to guide new users and developers on language usage and collaboration.
- Provision of open-source license for version-controlled and tracked contribution and collaboration for future project enhancement and establishing community support.

2.3 User Classes and Characteristics

The user classes provided are based on the demographics generated from a questionnaire in the form of google form filled by a random sample of 49 people, with 56.5% CS students, 23.9% professional software engineers, 6.5% self-taught programmers, and 13% non-coders with varying degree of proficiency in English. Interviews of domain experts Hasan Nafees and Shehryar brought further clarity for the user classes.

2.3.1 Novice User

- **Characteristics:**
  Novice users of AsaanLang are native Urdu/Hindi speakers with no prior coding experience, engaging in unprofessional coding to learn programming.
- **Aim:**
  Novice users aim to explore programming and acquire coding skills without being hindered by English proficiency barriers. The focus is on creating an inclusive learning environment with AsaanLang's Roman Urdu/Hindi syntax, overcoming language challenges associated with programming education. Additionally, users are driven by the desire to experiment and familiarize themselves with programming concepts through hands-on practice, fostering curiosity and exploration
- **User Scenario:**
  The novice user opens the AsaanLang web interface. They are presented with the home page which contains description of AsaanLang, its features and link to the 'Dastawezaat' documentation page and ‘Code Medaan’ code playground. They first click on the 'dastawezaat' page and are then redirected to the documentation of AsaanLang where they can familiarize themselves with the syntax of AsaanLang. From there, they can click on the ‘Code Medaan’ link. They are then redirected to another page which has a code editor. They write the AsaanLang code in the code editor and press the ‘Chalao’ button. Suppose the programmer makes a syntactical mistake in the code and presses the ‘Chalao’ button. The system detects the mistake and displays an error message and asks the programmer to

correct the syntax and press ‘Chalao’ again. Otherwise, if there was no syntax error, the source code is passed to the transpiler and executed as python code by python interpreter. The code output is then displayed on the output console of the code editor and systems return to standby state.

2.3.2 Professional Developer

- **Characteristics:**
  They are expert coders with a minimum of 2 years experience in Professional Programming and are actively participating in Industry standard coding.
- **Aim:**
  Professional developers utilizing AsaanLang aim to leverage their extensive coding expertise and industry experience for efficient and high-quality programming. Their goal is to seamlessly integrate AsaanLang into their coding workflow, adhering to industry standards and best practices. They will code in AsaanLang directly on the transpiler or on the command line.
- **User Scenario:**
  The Professional developer opens the AsaanLang web interface. They are presented with the home page which contains a description of AsaanLang, its features and a link to the 'dastawezaat' documentation page. They will click the link and will be redirected to the AsaanLang documentation. In the documentation page, there are steps on how they can easily and quickly download AsaanLang on their local machines and switch to AsaanLang for their future programming endeavors.

2.4 Operating Environment

Version 1.1 of AsaanLang transpiler currently supports usage only through the web application. Hence, the system has been developed with the development constraints of Python 3.11 and NodeJs 20.10.0 LTS and later.

2.4.1 Hardware Platform: AsaanLang, being a transpiler, operates primarily on the hardware platform that supports the execution of Python code. Therefore, it is designed to be platform-agnostic, allowing it to run on a wide range of hardware configurations that support Python version 3.1. The system requirements for running the generated Python code would align with the requirements for running Python applications. Officially recommended hardware requirements For Python v3.11:

- x86 64-bit CPU (Intel / AMD architecture). ARM CPUs are not supported.
- 4 GB RAM
- 5 GB free disk space

And for Node.JS v20.10.0 LTS:

- 4 GB RAM
- 1 GB Free disk space
- Intel Core 2 Duo E8400 or higher

2.4.2 Operating System and Versions: AsaanLang is developed to be compatible with multiple operating systems to accommodate diverse user preferences. The primary target operating systems include:

- **Windows:** Versions compatible with Python version 3.11 and Node.JS version 20.10.0 LTS and later.(Windows 7 and later).
- **macOS:** Versions compatible with Python version 3.11 and Node.JS version 20.10.0 LTS and later. (macOS 10.10 and later).
- **Linux:** Various distributions compatible with Python version 3.11 and Node.JS version 20.10.0 LTS and later. , ensuring flexibility for users on different Linux flavors.

The compatibility with multiple operating systems aligns with Python's cross-platform nature, allowing users to choose their preferred environment.

2.4.3 Software Components and Applications: AsaanLang must peacefully coexist with various software components and applications, particularly those related to Python development. This includes:

- **Python Interpreter**: AsaanLang transpiles to Python, requiring compatibility with Python interpreters.
- **Development Environments:** AsaanLang should be compatible with popular Python development environments, whether users prefer online integrated development environments (IDEs) or local ones
- **Version Control Systems:** Integration with version control systems such as Git for tracking changes in AsaanLang codebases.
- **Online IDE:** For the use of online IDE the system should also be compatible with Node.JS version 20.10.0 LTS and later.

2.4.4 Online and Local IDEs: AsaanLang is designed to be versatile, allowing users to code in both online and local IDEs. This flexibility caters to a broad user base, accommodating those who prefer the convenience of online platforms or the control offered by local development environments.

2.5 Design and Implementation Constraints

AsaanLang as a source-to-source transpiled language with Python as its target language pre-requires the installation of Python 3.11.0 or higher for usage and setup. Both the transpilation and execution of AsaanLang scripts are solely supported and carried out through the underlying Python interpreter. The Python Enhancement Proposal (PEP-8) coding convention has been followed for the design and development of this project. Furthermore, the supportive web application being designed with ReactJs and NodeJs also pre-requires the installation of NodeJs 20.10.0 LTS or higher for usage, setup and running where standard JSX coding conventions have been followed.

Additionally, in accordance with the system's open source licensing policy, refer to *Section 5.4.2*, the user, although being free to use, modify, and profit from the project, is subjected to the terms of the MIT license and shall include it in any reproduction of AsaanLang, while any programs written through AsaanLang shall subject to the program owner's will. Hence, AsaanLang, like many open source projects, can only be installed through the official GitHub repository.

Finally, AsaanLang's true vision to provide a Latin Urdu/Hindi-based development solution with the power of Python's powerful ecosystem support has been only partially implemented due to time constraints by limiting the project's scope to implementation of core and common programming language features, refer to *Section 4.2*, as the baseline to be iteratively developed to achieve full vision in future versions. Meanwhile, the choice of using Latin Urdu/Hindi based syntax over Arabic or Sanskrit script is an implementation and cultural constraint made in lieu of the tremendous underlying complexity and to unify all the South Asian users through Latin script.

2.6 User Documentation

AsaanLang syntax details are conveniently accessible online through the dedicated 'Dastaweezat' page within the AsaanLang web application. Users can refer to this comprehensive documentation for step-by-step instructions on using AsaanLang, including its syntax conventions. Further information on supporting documentation is elaborated in *Section 4.4* of this document. For additional community support and collaborative discussions, users are directed to *Section 6.1*, which delves into online community resources associated with AsaanLang. This user documentation is designed to facilitate a smooth and user-friendly experience for individuals exploring and utilizing AsaanLang in their programming endeavors.

2.7 Assumptions and Dependencies

2.7.1 Development Environment Stability:

Assumption: The assumption is made that the stability and functionality of the development environments used (e.g., online IDEs, local IDEs) will not significantly change during the development process.

Dependency: Any changes in the selected development environments may require adjustments to the AsaanLang.

2.7.2 Linguistic Evolution:

Assumption: AsaanLang is designed with Roman Urdu syntax, assuming that the linguistic preferences and nuances in Urdu and Hindi will remain relatively stable.

Dependency: If there are significant linguistic changes or shifts, it might require adjustments to the language syntax or documentation to maintain cultural inclusivity.

2.7.3 Community Contributions:

Assumption: The assumption is that there will be a thriving open-source community to enhance and refine AsaanLang. Like those of Java and Cpp.

Dependency: If community contributions are limited, it may place more responsibility on the core development team to maintain and improve the language.

2.7.4 Relevance to novice coders:

Assumption: Research by Hammarberg, 2010, indicates that learning complex scientific ideas in a second language is cognitively demanding and hinders understanding. Alternatively, using one's native language improves comprehension of abstract concepts by a factor of 20%

Dependency: If feedback suggests a need for adjustments to better help new users, it may influence language design or documentation.

2.7.5 Python Standard Library:

Assumption: Assumes the continued availability and stability of the Python Standard Library, which AsaanLang code may leverage.

Dependency: Any significant changes to the Python Standard Library may require modifications to AsaanLang code.

3. External Interface Requirements

3.1 User Interfaces

The AsaanLang web application has three screens, and all the content in the web application is in Roman Urdu/Hindi so that native readers can understand everything and lack of proficiency in English is not a hurdle in their journey with AsaanLang or programming in general. A specific color palette is used and the colors are consistent across the web application. The pages are modern and responsive (Prototype: figma/AsaanLang).

- **Navigation Bar:** When the user loads the website, they will be directed to the homepage, which contains a permanent/sticky/consistent navigation bar allowing users to navigate to other screens with minimal effort. On the navigation bar, there is a 'Code Medaan' link which redirects the user to the code playground and the 'Dastawezaat' link which redirects the user to the AsaanLang documentation. Users can navigate to any page from any point in the web application as all the pages are hyperlinked. The interface is easy to use and easy to learn.
- **Documentation**: In the 'dastawezaat' page, there are steps that professional developers can follow to download AsaanLang on their local machines with minimal effort, and the syntax of AsaanLang, explained in detail.
- **Code Playground:** In the 'Code Medaan' page, there is a code editor and an output window. Novice programmers and users who are testing AsaanLang can just type the AsaanLang code in the code editor, press the 'Chalao' button, which is situated directly below the input and output windows, and get instant executed output of their code.

3.2 Software Interfaces

3.2.1 Internal System Interfaces

The underlying infrastructure of the transpilation process consists of four primary components/modules: lexer, tokenizer, parser and the code generator. The lexer receives the source code written in AsaanLang syntax from an external interface (as defined in *Section 3.2.2*) and generates a list of identified expressions. This list of expressions is then passed onto the tokenizer which compares the expressions from the list of reserved keywords and symbols and replaces them with their corresponding tokens, which define the attributes and equivalent representation associated with the identified expression, hence producing a list of tokens which is then fed to the parser. The parser ensures that the source code is syntactically correct and generates an Abstract Syntax Tree (AST) to determine the order of parsing and code scopes. Once the AST is generated, the code generator utilizes it to generate the equivalent Python code which is then passed to the interpreter for evaluation and execution and the output is then sent back to the external interface for display.

3.2.2 External Interfaces

The system currently uses a web playground for code input and output, prompting the source code from the user and querying execution from the AsaanLang transpiler for the output to be displayed back again. Future versions of the system are intended to include support for famous IDEs and code editors by provision of standalone AsaanLang script files and plug-ins while also incorporating support for external interaction through direct Command Line Interfaces (CLIs) for using the transpiler functionalities.

3.3 Communications Interfaces

The web application runs in the default web browser and is currently hosted by the user on their localhost machines for usage, hence the basic HTTP protocol is used without the use of any SSL certificate. The Flask web micro-framework is utilized to handle any web server requests and page routing at the backend. There is no sensitive data being transmitted or stored, hence no encryption or security schemes are utilized.

4. System Features

Domain analysis, expert interviews, brainstorming sessions and close ended surveys . Following is a comprehensive summary of all the project features organized on the basis of priority and functional hierarchy (i.e., a preceding functional requirement must be satisfied before moving to the next):

NOTE: Wiegers's Prioritization Technique was employed to calculate the priority value of each system feature. The relative factor considered and their associated weights are (Factor : Weight): Benefit:2, Cost:1, Risk:1. Formula for priority calculation used:

Value = (Benefit * Benefit Weight)
Priority = Value / [(Cost% * Cost Weight) + (Risk * Risk Weight)]

4.1 Latin Urdu/Hindi Syntax

4.1.1 Description and Priority

Implement a language syntax based on Latin or Roman Urdu/Hindi language.

- Priority: 1.4
- Benefit: 7
- Cost: 2, 6%
- Risk: 1, 4%

4.1.2 Stimulus/Response Sequences -

4.1.3 Functional Requirements:

REQ_01 *AsaanLang syntax shall be defined based on Urdu/Hindi vocabulary.*

REQ_02 *A datastore of reserved AsaanLang keywords shall be generated, called a symbol table for the tokenizer to refer to the symbol table while tokenizing source code.*

.

4.2 Basic Language Functionalities

4.2.1 Description and Priority

Implement key functionality of a high level programming language, including all basic functionality of python (detailed below).

- Priority: 0.424
- Benefit: 7
- Cost: 5, 15%
- Risk: 4, 18%

4.2.2 Stimulus/Response Sequences -

4.2.3 Functional Requirements:

REQ_03 *AsaanLang shall allow for in-line code comments.*

REQ_04 *AsaanLang v1.0 shall have implementations for primitive data types (namely; integer, boolean, string and float).*

REQ_05 *AsaanLang shall have implementation for implicit (variable type does not need to be explicitly mentioned during variable declaration) variable declaration and data assignment using the assignment operator '='.*

REQ_06 *AsaanLang shall allow for basic binary and unary arithmetic operations (namely; increment, decrement, addition, multiplication, subtraction, division and mod).*

REQ_07 *AsaanLang shall have implementation for loops (namely; an AsaanLang equivalent of the for loop and while loop in python).*

REQ_08 *AsaanLang shall have implementation for branching and conditional statements.*

4.3 Source-to-Source Transpilation

4.3.1 Description and Priority

AsaanLang is a transpiled language which is transpiled down to python.

- Priority: 0.382
- Benefit: 9
- Cost: 7, 21%
- Risk: 6, 26%

4.3.2 Stimulus/Response Sequences

(see data flow diagrams in appendix B for pictorial representation)

1. AsaanLang code is fed into the transpiler *source code).

2. Source code is tokenized and passed through lexical analysis and converted into a keyword object model (KOM).
3. Tokenized code is fed into the parser to generate an AST.
4. If syntax errors are detected in the code, then the transpilation process halts (an error message is thrown in case the transpilation request was initiated from the code medaan).
5. Otherwise, the code generator converts that AST into equivalent python code.
6. The python code can then be executed using the python interpreter.

4.3.3 Functional Requirements:

REQ_09 *The AsaanLang transpiler shall use a tokenizer that can decompose the source code on the basis of reserved keywords.*

REQ_10 *A lexer shall be created to ensure syntactical correction of the AsaanLang code.*

REQ_11 *The AsaanLang transpiler shall use a parser to generate AST from the given AsaanLang code.*

REQ_12 *The AsaanLang transpiler shall be able to identify syntactical errors in the AsaanLang code.*

REQ_13 *The AsaanLang transpiler shall use a code generator module to convert the parsed AST into equivalent executable python code.*

4.4 Supporting Documentation

4.4.1 Description and Priority

Providing detailed documentation support for AsaanLangen listing key features of the language, installation and setup process and AsaanLang syntax.

- Priority: 0.347
- Benefit: 4
- Cost: 6, 19%
- Risk: 1, 4%

4.4.2 Stimulus/Response Sequences

(sample documentation given in figma prototype; refer to point 5 in *Section 1.5*)

1. User wants to install AsaanLang transpiler or learn AsaanLang syntax.
2. The user opens the AsaanLang documentation page through a web browser
3. User navigates to the relevant section and reads the official documentation.

4.4.3 Functional Requirements:

REQ_14 A detailed supporting documentation (of installation process, AsaanLang features and syntax) of AsaanLang shall be created and available online for everyone. *The documentation shall be updated with each new release (newer version) of AsaanLang.*

4.5 Web Editor

4.5.1 Description and Priority

Implement a web-based code editing platform to enable a user to write and run AsaanLang codes online.

- Priority: 0.12
- Benefit: 3
- Cost: 4, 12%
- Risk: 3, 13%

4.5.2 Stimulus/Response Sequences

(prototype given in figma prototype; refer to point 5 in *Section 1.5*)

1. User opens the web editor interface on the browser.
2. User writes AsaanLang code in the input window.
3. Usr presses the run button.
4. Code out (or appropriate error messages) are displayed in the output window.

4.5.3 Functional Requirements

REQ_15 *The web interface shall allow users to write/edit AsaanLang code in an input window. The input window shall color code the input code being written in real time.*

REQ_16 *Users shall be able to run the AsaanLang code in the input window and view the code output in an output window on clicking the run button. In case of syntax errors in the source code, the web interface shall display error messages to the user. The web interface shall time-out in case the transpilation and execution request is not completed within 4 minutes of pressing the run button and display an appropriate error message.*

4.6 Supplementary Frameworks Mapping

4.6.1 Description and Priority

Mapping supplementary python frameworks and commonly used python libraries to AsaanLang in order to enhance the usability of AsaanLang. (susceptible to rejection for in AsaanLang v1.0)

- Priority: 0.065
- Benefit: 2
- Cost: 9, 27%
- Risk: 8, 35%

4.6.2 Stimulus/Response Sequences -

4.6.3 Functional Requirements

REQ_17 *AsaanLang shall have native equivalents of built-in functions of commonly used python frameworks and libraries in order to make AsaanLang relevant and useful for professional developers who intend to use AsaanLang.*

5. Other Nonfunctional Requirements

5.1 Performance Requirements

In addition to giving correct code output, ensuring optimal performance is crucial for the AsaanLang transpiler's usability and user satisfaction. The transpilation time of AsaanLang transpiler must not exceed (TBD) number of throughput (transpilation rate). Employing efficient parsing and abstract syntax tree generation techniques, along with optimized code generation strategies, must be ensured to provide faster and resource-efficient transpilation.

5.2 Security Requirements

No sensitive user data is stored or transmitted through either the usage of the transpiler or web application, hence no level of user authentication or transfer security protocol is required. Syntactic checking and error handling is ensured to make the system more robust and avoid any abuse of code execution privileges through the transpiler.

5.3 Software Quality Attributes

5.3.1 Maintainability

Transpiler design of AsaanLang must be modular with sequential coupling, where each step of code translation is a black-box (further elaborated in Appendix B) to allow for adaptability (changes in code base) and evolvability (easy integration of new functionality) in the transpiler design to minimize the "cost to change" .

5.3.2 Correctness

The correctness of the AsaanLang transpiler is of paramount importance. The transpiler must accurately translate code from the source language to Python, ensuring that the output is semantically equivalent to the input. Rigorous testing procedures, including unit tests and integration tests, must be implemented to validate the correctness of the transpiler's functionality under various scenarios. The AsaanLang Shura is responsible for overseeing and enforcing the correctness standards throughout the development lifecycle.

5.3.3 Robustness

The AsaanLang transpiler should exhibit robust behavior in the face of diverse inputs and potential edge cases. It should gracefully handle unexpected or malformed source code, i.e, provide meaningful error messages and avoid unintended behaviors if a user tries to run a code which is syntactically incorrect (does not conform to AsaanLang syntax).

5.3.4 Code Readability and Standards

Source code of AsaanLang shall be well-written and well-commented (provide code comments in roman Urdu/Hindi), following the *"PEP 8" (see: https://peps.python.org/pep-0008/)* coding guidelines. AsaanLang Shura shall be responsible to ensure the open source contributions to AsaanLang do not violate the PEP 8 standards of python coding before merging the contributions with base code.

5.4 Business Rules And Project Policy

5.4.1 Medium of Communication

In light of team AsaanLang's commitment to providing a palatable development environment for native speakers of Urdu/Hindi, all supporting official documentation, code comments as well as official development team to user official communications should prefer Roman Urdu/Hindi as official language.

5.4.2 Version Control and Project Ownership Policy

Official development team hierarchy/roles, version control and, usage and licensing policy of AsaanLang is as follows:

- **Open Source Licensing:** AsaanLang is an open source project licensed under the MIT license. This grants users the freedom to use, modify, and profit from the project, subject to the terms of the MIT license.
- **User Rights and Local Copies:** Users have the right to use, edit, and maintain their local copies of AsaanLang without oversight and derive benefits, including financial gains from the project.
- **Ownership Structure:** The ownership of AsaanLang is vested in the original five developers, referred to as the "**Owners"**. These owners collectively form the AsaanLang Shura, which is entrusted with the responsibility of maintaining the base version of AsaanLang at all times, approving new owners, enforcing coding standards, deciding the scope aim of the project and approving (or rejecting) open source contributions to AsaanLang. AsaanLang Shura shall hold periodic meetings, at least once every 12 months to discuss the scope and direction of the project.
- **Addition of New Owner:** A new owner may be added to the AsaanLang Shura through a unanimous vote in favor by all current Owners. This

inclusive approach ensures that any expansion of ownership is a collective decision supported by the existing founding members.

- **Contributions to the Source Code:** The AsaanLang Shura possesses the authority to edit, modify, update, and merge open contributions into the base code. This decision-making process requires a majority vote of at least 60% of the Shura members, calculated using the ceiling function. This mechanism ensures that significant changes to the project are made with a substantial level of agreement among the Shura members.

6. Other Requirements

6.1 Community Support

A crucial non-functional requirement to keep any language alive and relevant in the dynamic world of programming is the cultivation of an active and supportive developer community. Establishing a footprint on prominent platforms such as Stack Overflow and GeeksforGeeks is pivotal, emulating the success trajectories of mainstream programming languages. Recognizing the formidable challenge posed by the experimental and niche nature of AsaanLang, our team endeavors to initiate a dedicated Reddit community. This space will serve as a nexus for AsaanLang developers and contributors worldwide, fostering growth, collaboration, and mutual assistance.

6.2 Ecosystem Support

AsaanLang must be integrated with present programming language ecosystems. It includes extension packs for popular IDEs, AsaanLang versions of python libraries being used for development in various domains of software.

6.3 Future Releases (Roadmap)

While the inaugural release (v1.0) of AsaanLang encompasses rudimentary programming language functionality, the project is conceived as a long-term endeavor, with each future version providing additional functionality and building on the last. Among the prioritized features for upcoming releases (not mentioned in a particular order) are:

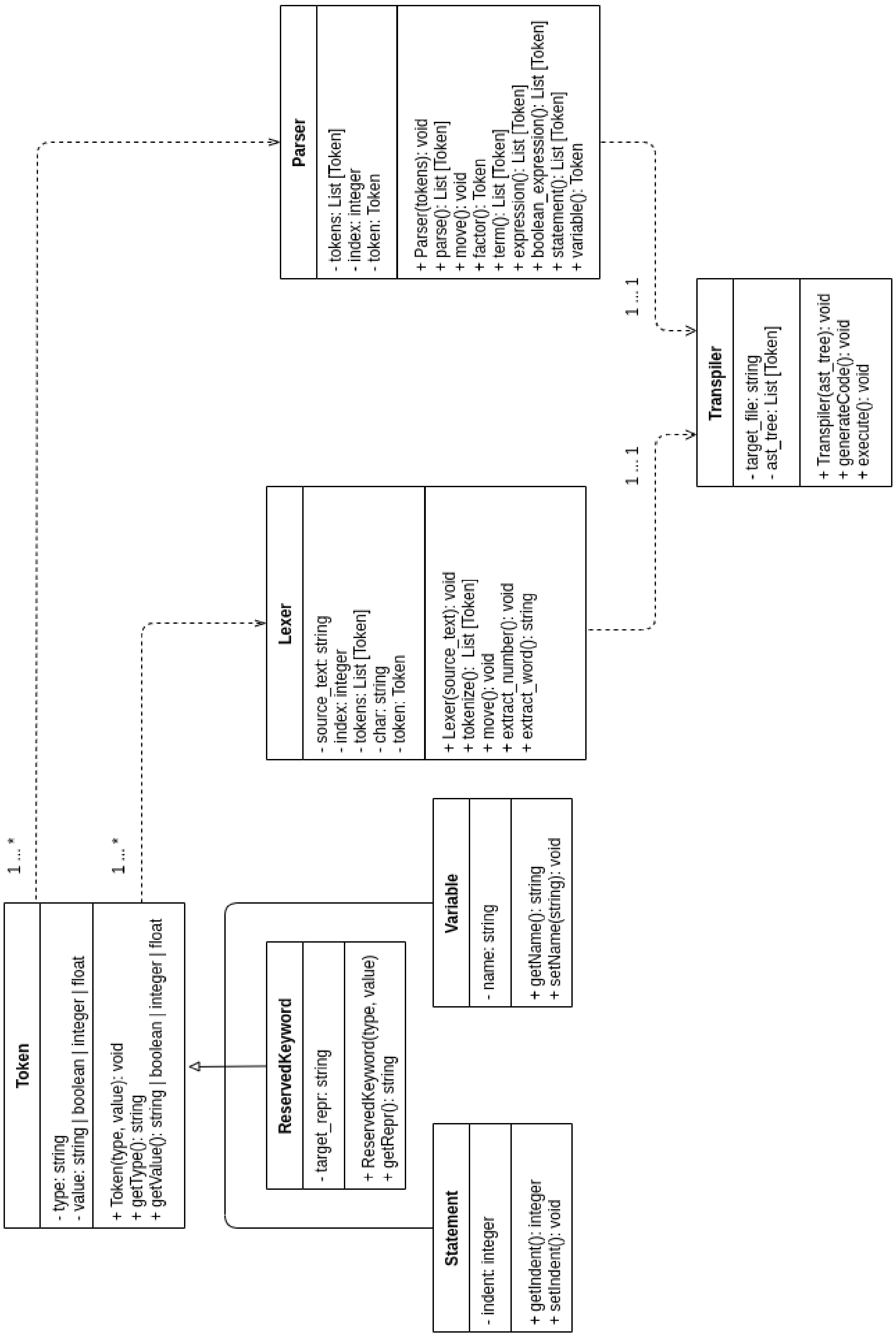
- User defined functions
- Recursive coding
- Object-oriented programming
- Exception handling
- Mapping for builtin python functions
- Higher python data types (sets, lists, tuples and dictionaries)
- Anonymous or lambda functions

Appendix A: Glossary

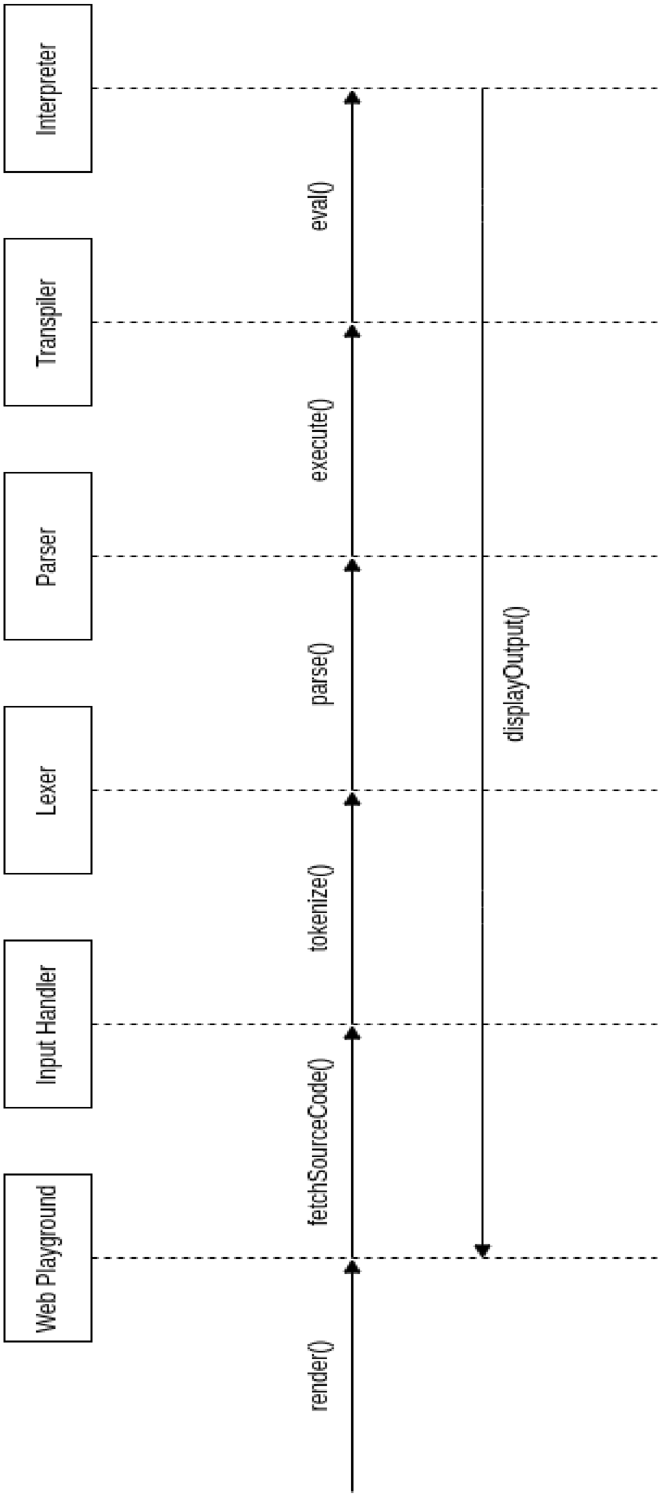
AST (Abstract Syntax Tree): hierarchical tree-like structure representing the syntactic structure of source code, used in compilers and interpreters.
AsaanLang Shura: A board of all current owners.
Blackbox: A system or component viewed solely in terms of its input and output, with its internal workings considered opaque or unknown.
Code generator: a component that converts high-level code or intermediate representations into executable code or lower-level languages.
Code Medaan: synonym for a code playground / official AsaanLang web editor interface.
Code playground: an online environment allowing users to experiment, test, and share code snippets in various programming languages.
CLI: a text-based interface where you can input commands that interact with a computer's operating system
Dastawezaat: synonym for documentation for AsaanLang.
IDEs: software applications that provide comprehensive tools and features for software development, including code editors, debuggers, and build automation.
JSON (JavaScript Object Notation): a lightweight data interchange format that uses human-readable text to represent data objects consisting of key-value pairs. Commonly used for data transmission between a server and a web application.
KOM (Keyword Object Model): structured representation and organization of keywords within a programming language
Latin or Roman Urdu/Hindi: Urdu or Hindi language using the Latin script
Lexer: breaks down source code into tokens for further processing by a compiler or interpreter.
Owner: a member of the AsaanLang Shura.
Plug-ins: modular software components that add specific functionalities to a larger software, enhancing extensibility and customization.
Source-to-source transpiler: translates source code from one programming language to another, maintaining the original code's high-level structure.
Throughput: number of lines of code transpiled per second. A Metric for measuring transpiler speed
Transpilation: process of converting source code from one programming language to another without changing the program's functionality
Tokenizer: a component that breaks down input into smaller units, often used in natural language processing or parsing.
.

Appendix B: Analysis Models

8.1 UML Diagram

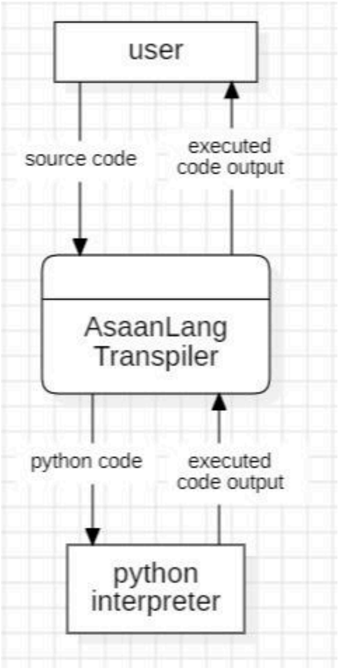


8.2 Sequence Diagram

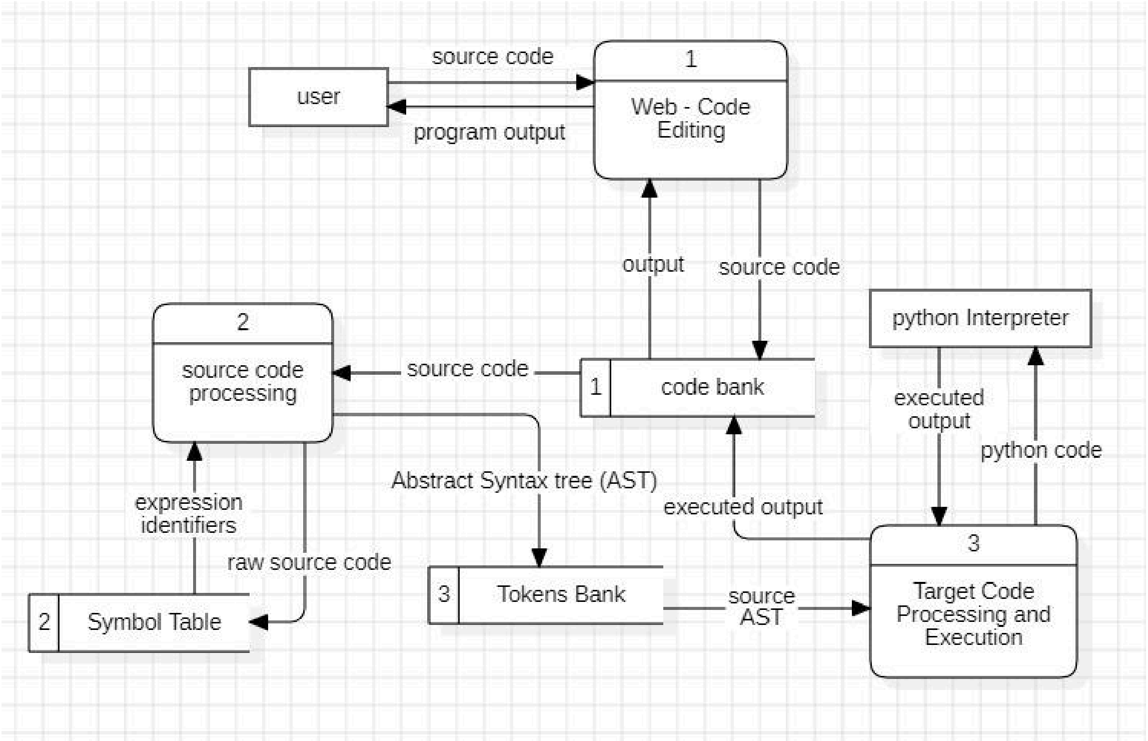


8.3 Data flow diagrams

8.3.1. Level 0 DFD (or context diagram)

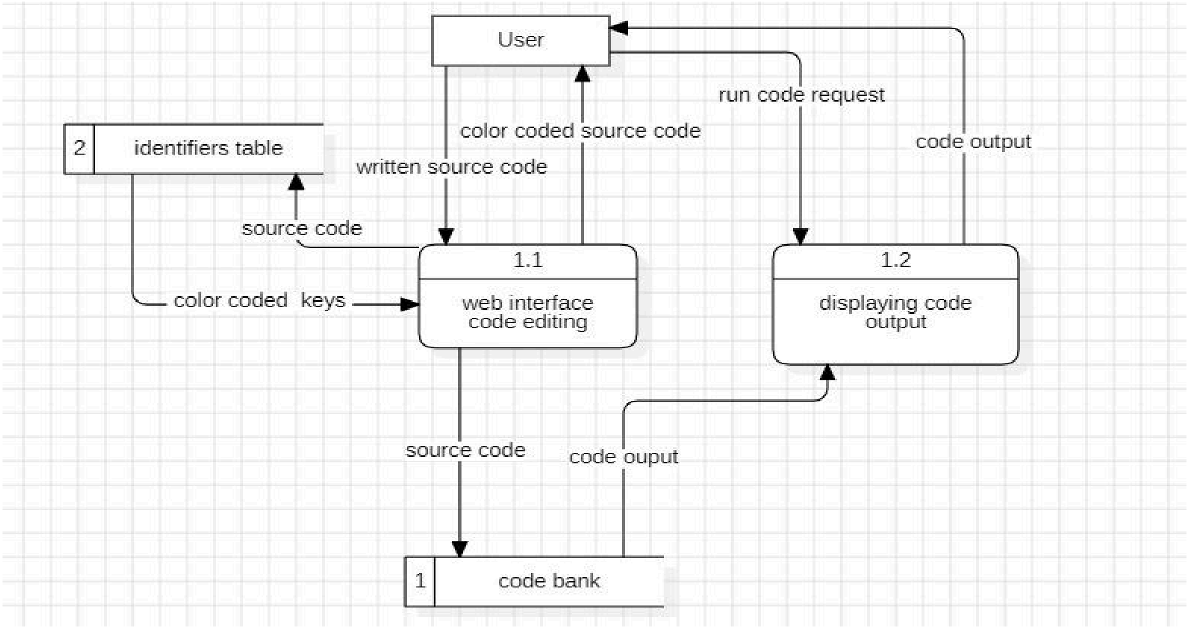


8.3.2 Level 1 DFD

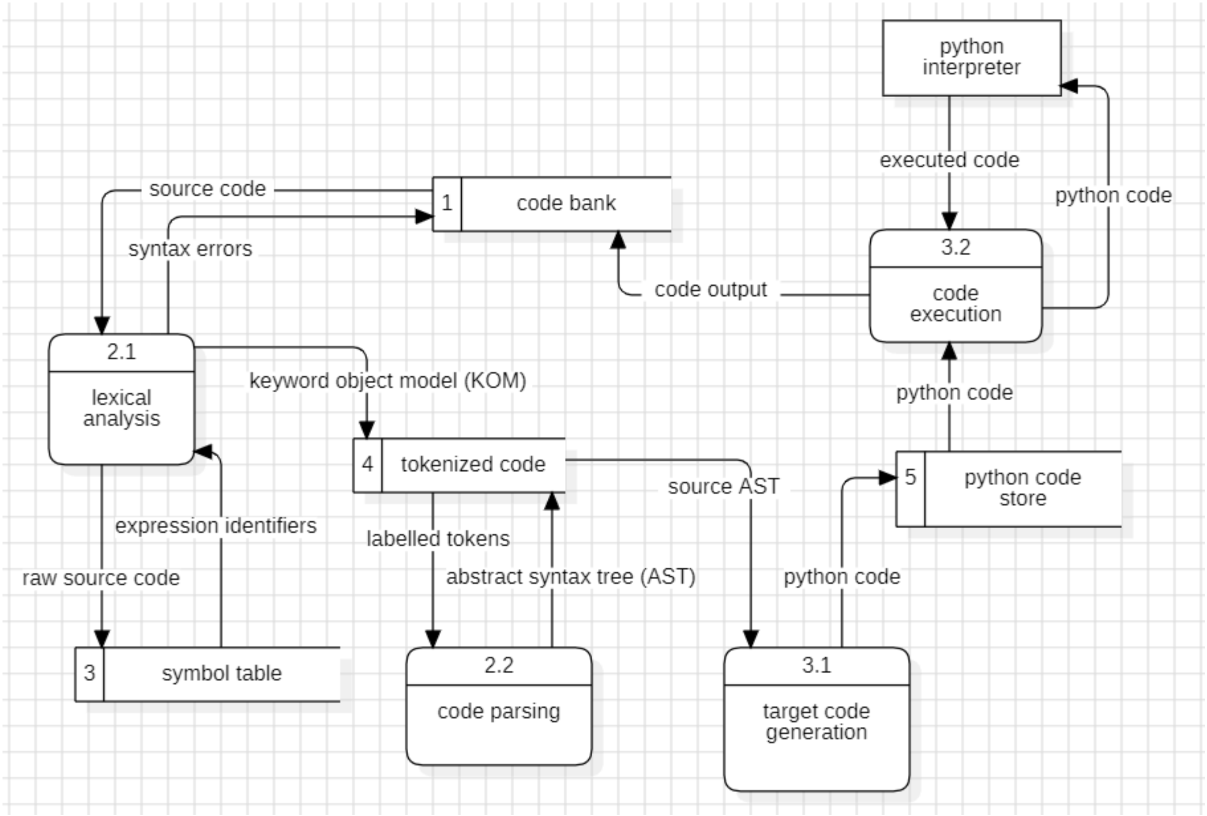


8.3.3 Level 2 DFD

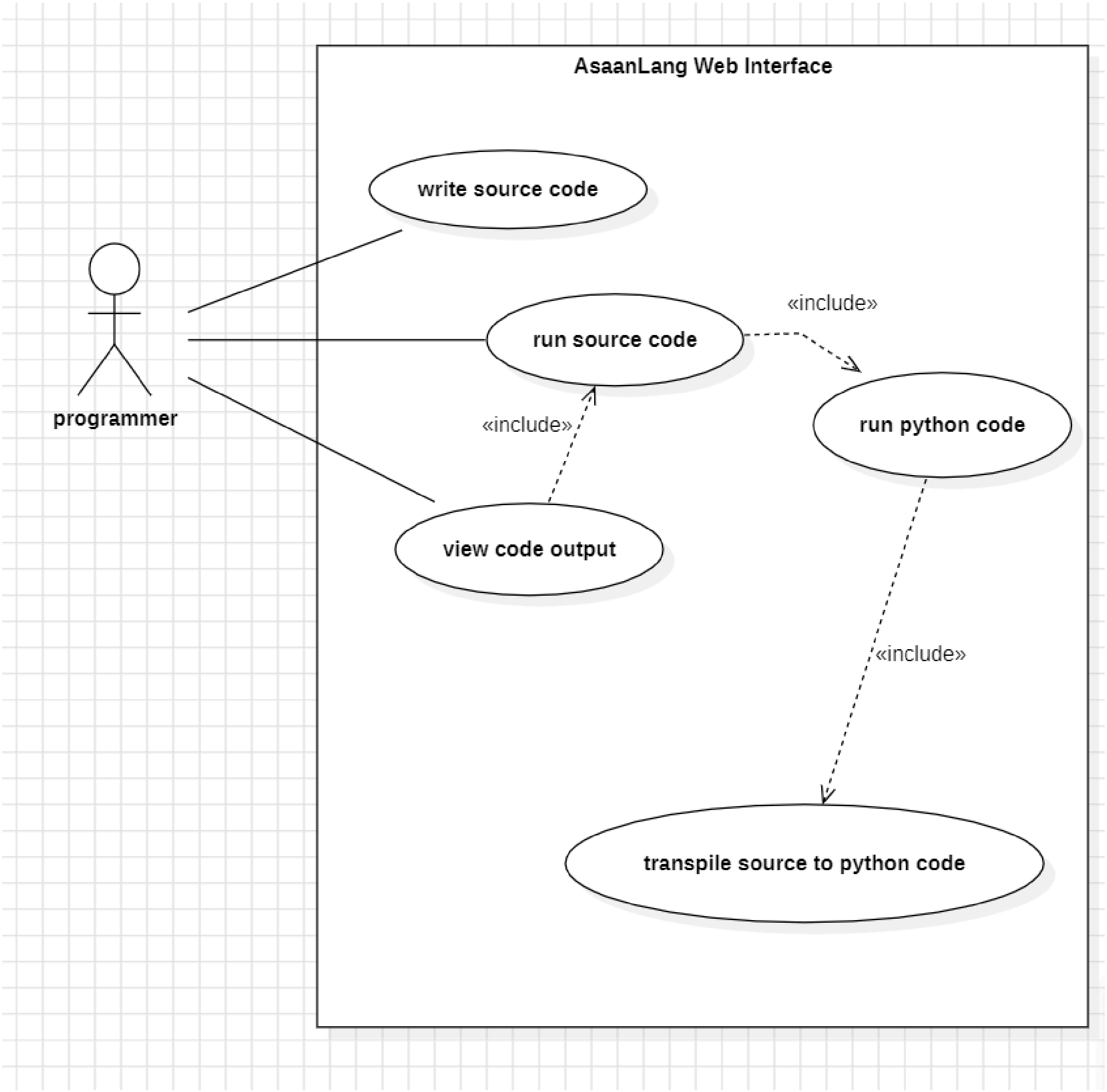
a)



b)

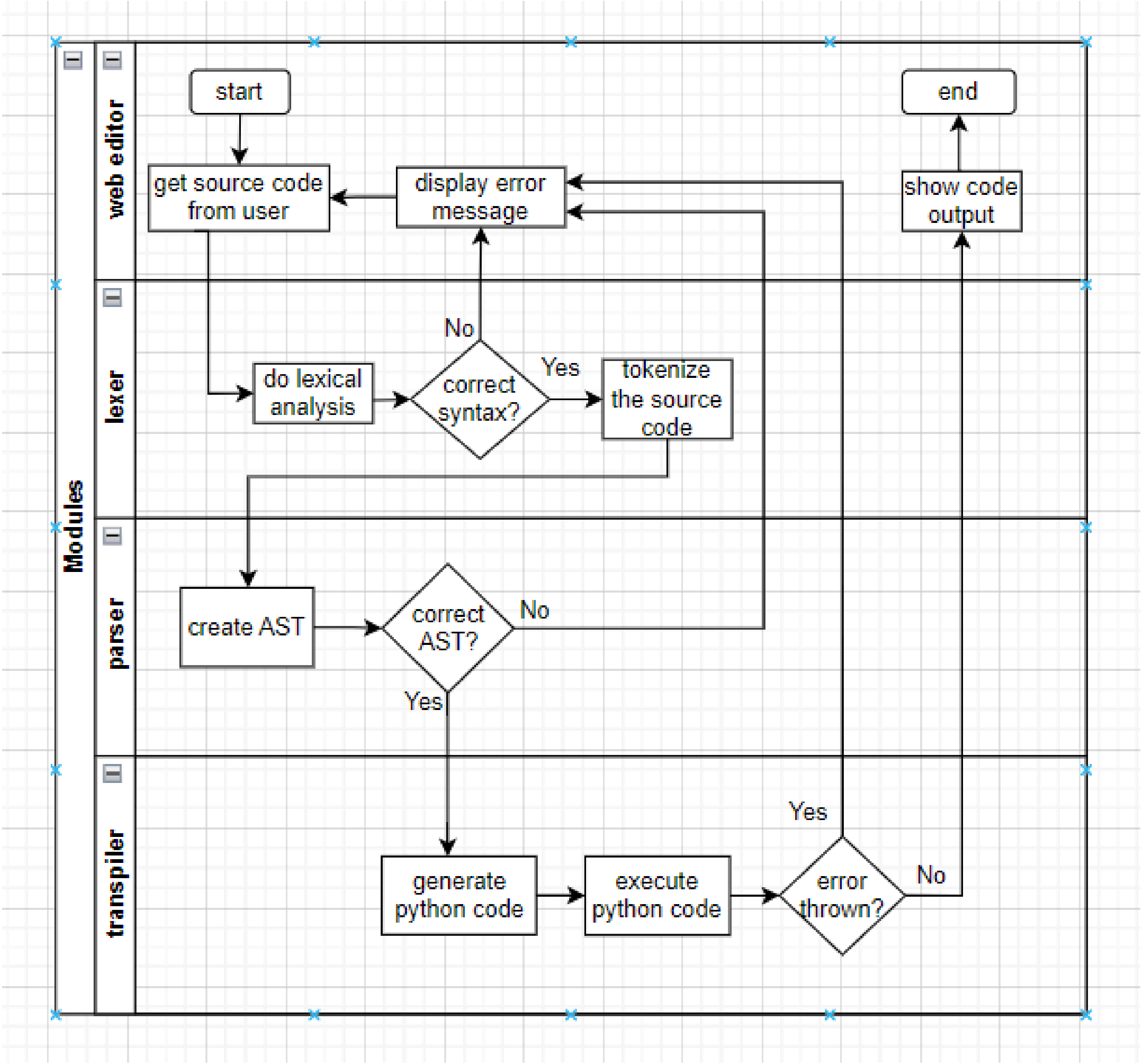


8.4 Use Case Diagram



8.5 Swimlane diagram

The following swimlane diagram show the basic process of AsaanLang code execution and output using the web code editor



Appendix C: To Be Determined List

1. Throughputs of AsaanLang transpiler are to be determined.
2. The storage scheme for reserved keywords and symbols is to be decided, whether it would be stored as a hardcoded list, in a JSON or an external python or text file.