Mona First programing language.

Ala Al Din Afana (16395986)
Shanan Lynch(14723071)
Date finished (21 November 2019)

Introduction

1.1 overview

For our fourth year project we will develop a First programming Language using Java cc. Our programming Language will be accessible through a web application. We plan on developing a debugger for our language that will allow the user to set breakpoints , resume step-over , step-into and step-out of different sections of their code.

The design of our web-app is influenced by https://repl.it/languages/python3 and we aim to have a similar platform as our end product.

The goal of our project is not only to provide a Novice with a “first” programming language that will allow him/her to learn the foundational programming concepts but also enable them to transition smoothly to more advanced programming language.

Our project will break down the syntax and semantics of python, java and c++ and combine them into an easy to access and easy to use programming language. In 1999 Guido van Rossum defined his goals for python as:

- an **easy and intuitive** language just as **powerful** as those of the major competitors;
- **open source**, so anyone can contribute to its development;
- code that is as understandable as **plain English**;
- suitable **for everyday tasks**, allowing for short development times.

He was able to accomplish these goals and take python to the top of the rankings. Python is a great programming languages but its **simplicity** and **plain english** style hinders the ability of a programmer to transition to other languages as it eliminated many programming concepts that are used in desirable programming such as java and c++.

In java you must declare the type of data which is known as static typing (e.g int x=1) on the other hand Python code is dynamically typed, meaning that you don't need to declare the variable type, also known as duck typing. "If it walks like a duck and it quacks like a duck, then it must be a duck" (e.g x = 1).

This would not be an issue if a software engineer sticks to python for the rest of his career but as we learnt from our time at DCU a good software engineer is flexible and must have a love for learning and curiosity. Another factor to take into consideration is that the market is moving towards full stack development. Developers must now be more flexible than ever before and have a deep understanding of several programming languages.

Principles of a programing language

According to (C.A.R.Hoare paper “HINTS ON PROGRAMMING LANGUAGE DESIGN” from Stanford University)
A programming language is regarded as a tool to aid the programmer , Meaning it should give assistance in the most difficult aspects of his Art , Namely program design,

documentation and debugging. These three factors will be taken massively into our first programing language implementation and will be discussed in more details.

Program design

The first and very difficult aspect of design is to decide what the program is doing and format this to a clear and precise and acceptable specification. Often just as difficult as deciding on how to do it, and how to divide tasks into subtasks and to specify the purpose of each part and define clearly, precise, and efficient interface between them .
A good programming language should assist in establishing and enforcing the programing conventions and disciplines which will ensure harmonious co-operation of the parts of a large program when they are developed separately and finally put together. The programming language Java is a very good example. It assists the programmer in the design process by enforcing strict programming conventions. Java provides classes and functions to separate blocks of code. It has strict syntax and allows an object oriented design approach. Object Oriented Programing application is much easier and it also helps keep the system modular, flexible and extensible.

Program documentation

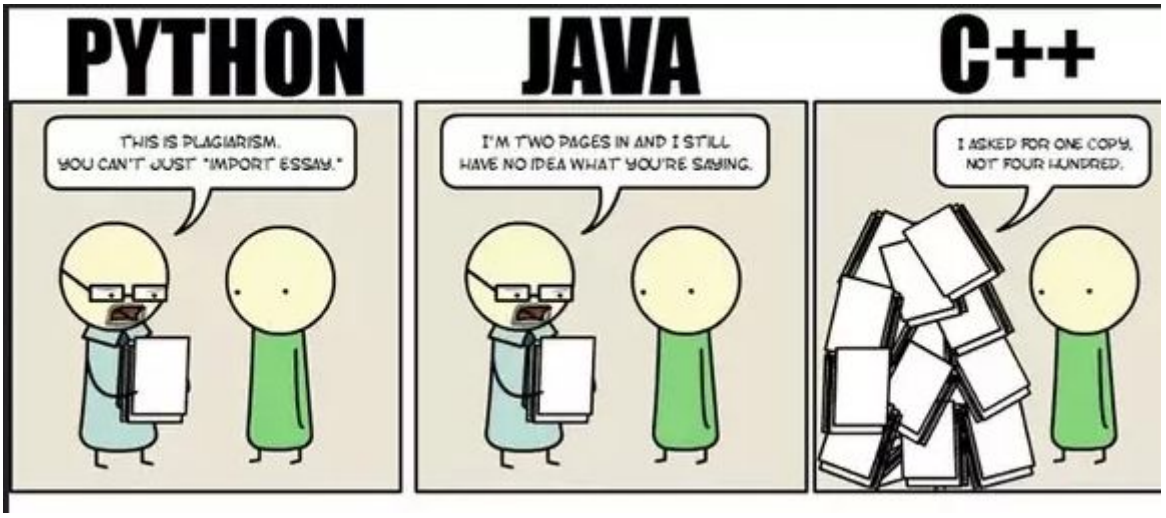
Documentation is a very important factor in developing a good programing language. The purpose of documentation is to explain to a human reader how a program works, so it can be successfully adapted after it goes into service. Often perceived that documentation is a process that is added to a program after its developed seems to be wrong in principle and counter productive in practice. Instead documentation should be regarded as a necessary part of the process of design and coding.
Our programming language will encourage and assist the programmer to write clear defined easy to understand code. In a sense our programing languages aims to be self documenting code. It will have a pleasant display and style of writing.
We aim for readability as much as writability.

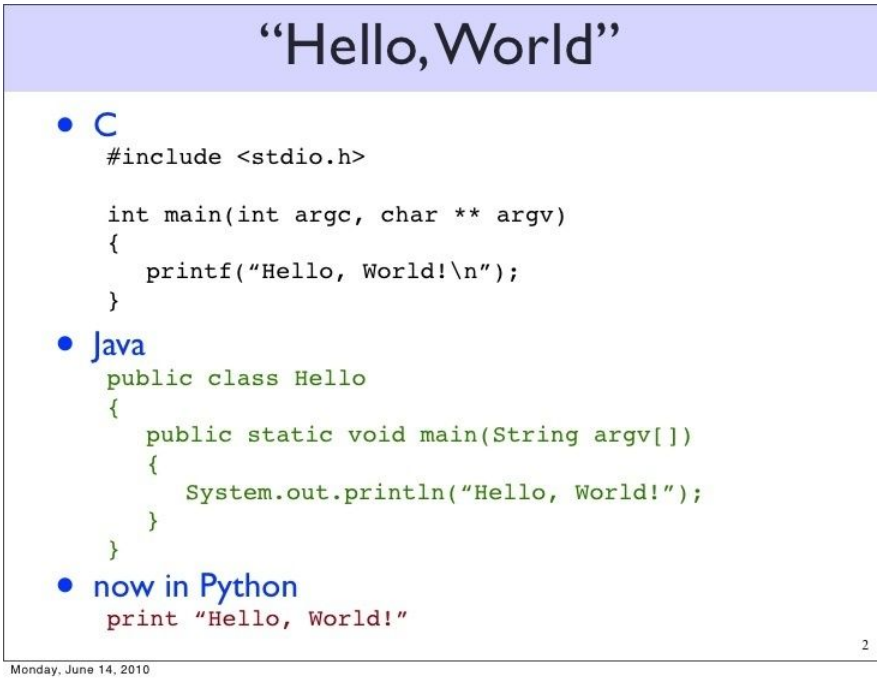
Program Debugging

Program debugging can often be the most tiresome, expensive and unpredictable phase of a program development. Particularly when adding subsections of a program together. The best way to reduce these problems is by successful initial design of the program, and by careful documentation. Even the best designed and documented programs run into issues and error. A good programming language will assist a programmer in debugging their code. We aim to develop a system that will reduce as far as possible the scope for coding errors. We will also develop a debugger that will allow the programmer to run their code step by step. This will be discussed in more detail later on.

The target of Our programming language is simplicity in the design of the language. A complicated language design has many side effects , as we complicate the design we cannot evaluate the consequences of our design decisions. But the main beneficiary of simplicity is the target user of the language.
While our goal is simplicity we have to take into account that our intentions are to provide a programming platform that will introduce a novice to programming.

Designing a language that has simple syntax for example python can negatively impact the programmer transitioning into a more complex designed language. It can be argued that a novice who learns to program in c++ will have little difficulty coming into java or any other language.It is also unreasonable for a novice programmer to learn languages that are designed for commercial use by experienced commercial programmers.The development environments that they use often designed for commercial programming and contain features beyond a novices comprehension.





Our programming design will be highly influenced by the design of python, java, c and c++ , by taking features from each of them we can design a first programming language that can be easily transitioned to and from any language.(The First Language - A Case for Python? Tony Jenkins)

Accessibility

It is important to remember the capabilities of our target audience, and how our platform can be accessed. Taking this into account we decided to tackle the compilation of the language through a web application. With this approach the novice programmer (the target audience) will be able to start programming out of the box. This means the user will not be required to follow a step by step guide on how to install the language on their computer. As long as the user has access to a browser they will be able to program. Since the user will be programming on this web application we will have a debugger to accompany the compiler. This debugger will allow to programmer to step through their code and see how the code works. We feel that this will be a critical part of the project. As our target audience are novice programmers, giving them the ability to see the flow of how the program works will be vital to speed of which the programmers will learn.

1.3 Glossary

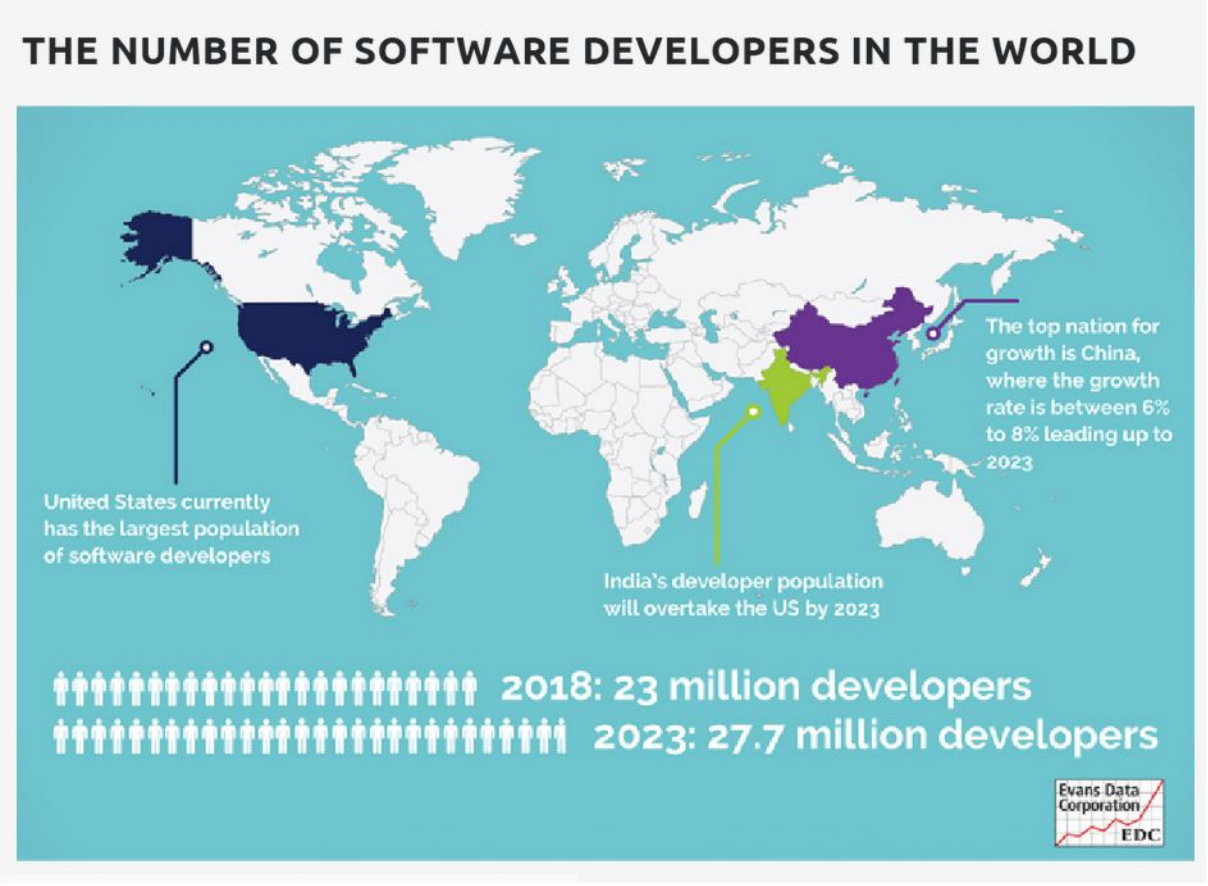
Term	Meaning
JAVACC	JAVA Compiler Compiler (JAVACC) is a parser generator, JAVACC could be downloaded from the official Javacc website.
AWS	Amazon Web Services is a subsidiary of Amazon that provides on-demand cloud computing platforms and APIs to individuals
Mona	The name of our programing language inspired by the painting Mona Lisa by Leonardo Di Vinci, We decided on this name because Leonardo Di Vinci was a polymath with work that will last decades,The name is also short and unique, this is similar to how Guido van Rossum derived the name python from “Monty Python's Flying Circus ".
AMI	Amazon Machine Images

2.General description

2.1 Product / System Functions

The functionality of this product will be to give an alternative programming language to novice programmers. The way in which our product does this is by providing the user with an online editor and debugger to use Mona programming language. The user will be able to create programmes in this web application interface to learn the various basic and foundational concepts that are included in a programming language. The functions this web application will be able to do is compile the Mona programming language, and be able to debug the code using the debugger on this application.

2.2 User Characteristics and Objectives



The user community consists of a wide demographic audience. The United States has the largest population of software developers in the world it is estimated that india will overtake the US in 2023. China also has the highest growth rate at 6 %. This illustrates the importance of programing languages. It is safe to say that the user community will have a basic understanding in software technology, users must know how to use a computer and surf the web.

Our product will provide the users an easy to access programming environment which will allow them to learn and develop code, It will also allow them to transition into commercial based programming languages such as c++ and Java.

The Objective of the product is to introduce the user to, programming , debugging, and IDEs.

2.3 Use Case Diagram

2.4 Operational Scenarios

Use Case	1
Description	The user adds a file in the editor
Primary Actor	The User
Pre-conditions	There are no pre-condition needed for the user to perform this
Post-conditions	The user will now have an additional file alongside their original file.

Use Case	2
Description	The user adds a folder
Primary Actor	The User
Pre-conditions	There are no pre-condition needed for the user to perform this
Post-conditions	The user will now have an additional folder to use in the editor

Use Case	3
Description	The user can click on the documentation which has the manual for the web application and which also has the documentation on the Mona Programming language
Primary Actor	The User
Pre-conditions	There are no pre-condition needed for the user to perform this
Post-conditions	The user will either have the documentation on the Mona language downloaded in PDF format or the user manual for this web application in PDF format

Use Case	4
Description	After the user has typed something into the editor they can click run. This will either correctly compile or not
Primary Actor	The User
Pre-conditions	The user will have to have something typed in the editor if the program can be compiled.
Post-conditions	The code will either compile and output what it is instructed to output or the code cannot be compiled. In this case the error will be indicated on the terminal interface on the web application.

Use Case	5
Description	The user can save their file by clicking the save file button of the current file they are on
Primary Actor	The User
Pre-conditions	The has to have typed something into the file before it can be saved. And the user has to be on the file which wants to be saved.
Post-conditions	The file is now saved

Use Case	6
Description	The user will be using the step over feature of the debugger which will go to the next line. If the line contains a function then it will execute that function.
Primary Actor	The User
Pre-conditions	The user will have to have clicked on the debugger option in the web application to start the debugger. The program will have

	to be compilable up until the current line in the code that the user is on
Post-conditions	The user will have either received an error on the terminal/ executed a function and arrived on the next line/ or simply arrived onto the next line

Use Case	7
Description	The user will be using the step out feature of the debugger which will go back to the line in which the current function the user is in was called
Primary Actor	The User
Pre-conditions	The user will have to have clicked on the debugger option in the web application to start the debugger. The user will have to currently be in a function that has been previously called to use this
Post-conditions	The user will be back to where the current function was called from

Use Case	8
Description	The user will be using the step into feature of the debugger which will go into the function if the current line the user is on a function
Primary Actor	The User
Pre-conditions	The user will have to have clicked on the debugger option in the web application to start the debugger. The line the current user is on in the editor will have to contain a function.
Post-conditions	The user will now be inside the function

Use Case	9
Description	The user will be using the breakpoint feature of the debugger. The user will add

	a breakpoint on the line they desire by typing in the line number in the breakpoint section
Primary Actor	The User
Pre-conditions	The user will have to have clicked on the debugger option in the web application to start the debugger.
Post-conditions	The program will now be stopped on the line that the breakpoint was placed, providing that the program up to the point of the breakpoint is compilable

2.5 Constraints

1. Developing a programming grammar which is correct and intuitive for the users.
2. Developing a web app with a ubuntu container where the code will be compiled and executed in real time is a challenge.
3. Developing a Debugger using ptrace where the step in, step out and step over are set in the front end editor will be challenging.
4. Designing a straightforward, use-able and accessible UI that meets user requirements.
5. Testing the project will also be a challenging task as there is a lot of emphasis on user testing and as such both the product and its documentation needs to be done ahead of the deadline.
6. Meeting the goal of the project where we provide the users with something new, unique and effective, and not reproducing what is already been done.

3. Functional Requirements

3.1 Compiler

Description - Develop the compiler using javacc where the code will be compiled and executed

Criticality - This functionality has the ***highest criticality*** as it contains the most difficult aspects of our project and is its ground bass.

Technical issues - Difficulties will be mainly in research and design. Developing a grammar for any programming language is a difficult task and mistakes in the grammar can have dire consequences.

Dependencies with other requirements - This requirement is the core of our project and is not dependent on any other requirements.

3.2 Debugger

Description - Develop the debugger using ptrace and incorporate it in compiler.

Criticality - This requirement is very critical as it is a key tool in any programming language.

Technical issues - incorporating ptrace command within javacc to enable the user to step in, step out and step over different sections of the code.

Dependencies with other requirements - this requirement is highly dependent on requirement 3.1 where we will be essentially be manipulating the behavior of the compiler at run time.

3.3 UI / Web application

Description - A spring-boot web-application will be developed to retrieve user input/source code and pass this source code to the ubuntu instance via api calls where the code is compiled.

Criticality - This requirement is critical as it is the means by which the user will be communicating with the compiler.

Technical issues - How step out and step over will be represented when passed to backend.

Dependencies with other requirements - This requirement doesn't depend on any other requirement here we are only focusing on the UI and parsing values from the front end and sending them as json objects to backend.

3.4 Programming Language Run Environment

Description - EC2 instance where the programming language will be compiled and executed.

Criticality - This requirement is critical as it is the environment that the code will be the runtime environment where the .mona programs will be executed.

Technical issues - This aspect of the project should be fairly straight forward.

Dependencies with other requirements - This requirement is dependent on the compiler and debugger.

3.5 Executing Code via webapplication

Description - Displaying Ubuntu Terminal via web app, retrieving code on ubuntu machine and executing it

Criticality - This requirement is very critical as it is basically connecting every aspect of the webapp together where we can evaluate and test if our product meets its goals via user testing.

Technical issues - Displaying a linux terminal that is running elsewhere within a web application is a fairly uncommon task and as such resources are limited online. We are still not 100% clear on the best way to retrieve source code and execute it from terminal. Another issue is that we need to pass the debugger parameters from json object and execute them from the terminal.

Dependencies with other requirements - This requirement is and dependant on every single requirement referenced above.

4. System Architecture

4.1 Software components of architecture

Spring-boot: we will be using spring-boot framework java to develop our web application in our project.

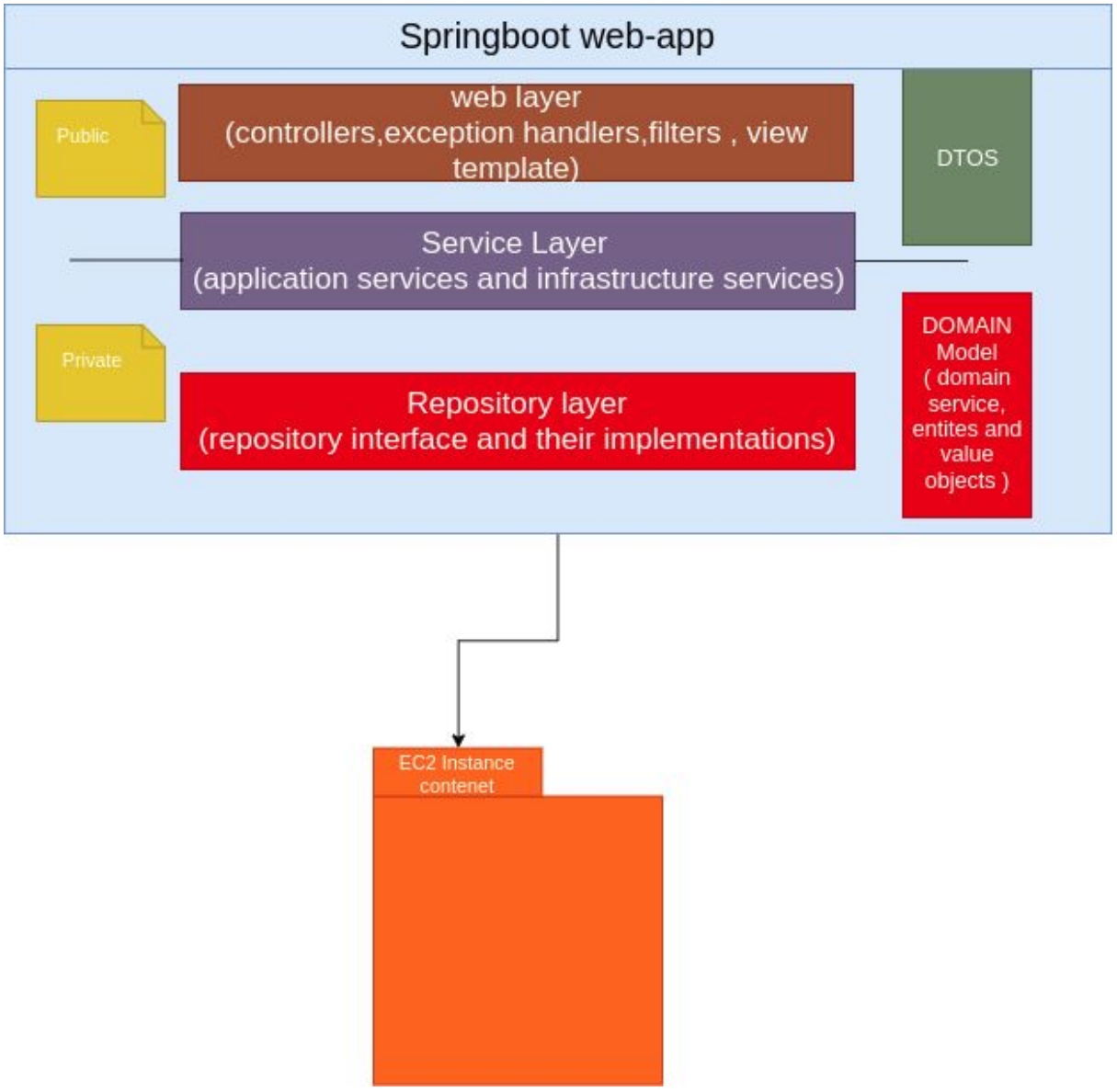
Amazon EC2 Instances : the ec2 instances boot up a custom AMI (Amazon machine instance image) which will have a base ubuntu.

Javacc: This language will be used to create our compiler The language will be compiled in the Ubuntu container.

Golang: we will use golang to build our debugger this will run on the ubuntu instance we will be using ptrace to build our debugger.

Docker : We will use a docker container to run a linux image (LXC) or a docker image of either alpine linux or ubuntu.

4.2 Spring boot web application architecture



Web layer: Upper most layer of our web application and will be responsible for processing user's input and returning the correct response. It also handles the exception thrown by other layers,It is the entry point of our application.

Service layer: Based under the web layer and acts as a transaction boundary and contains our application and infrastructure services. The service layer will have our Api calls it will be the layer where our application will communicate with outside resources.

Repository layer: this is the lowest layer in the web application architecture. Its responsible to communicate with used data.

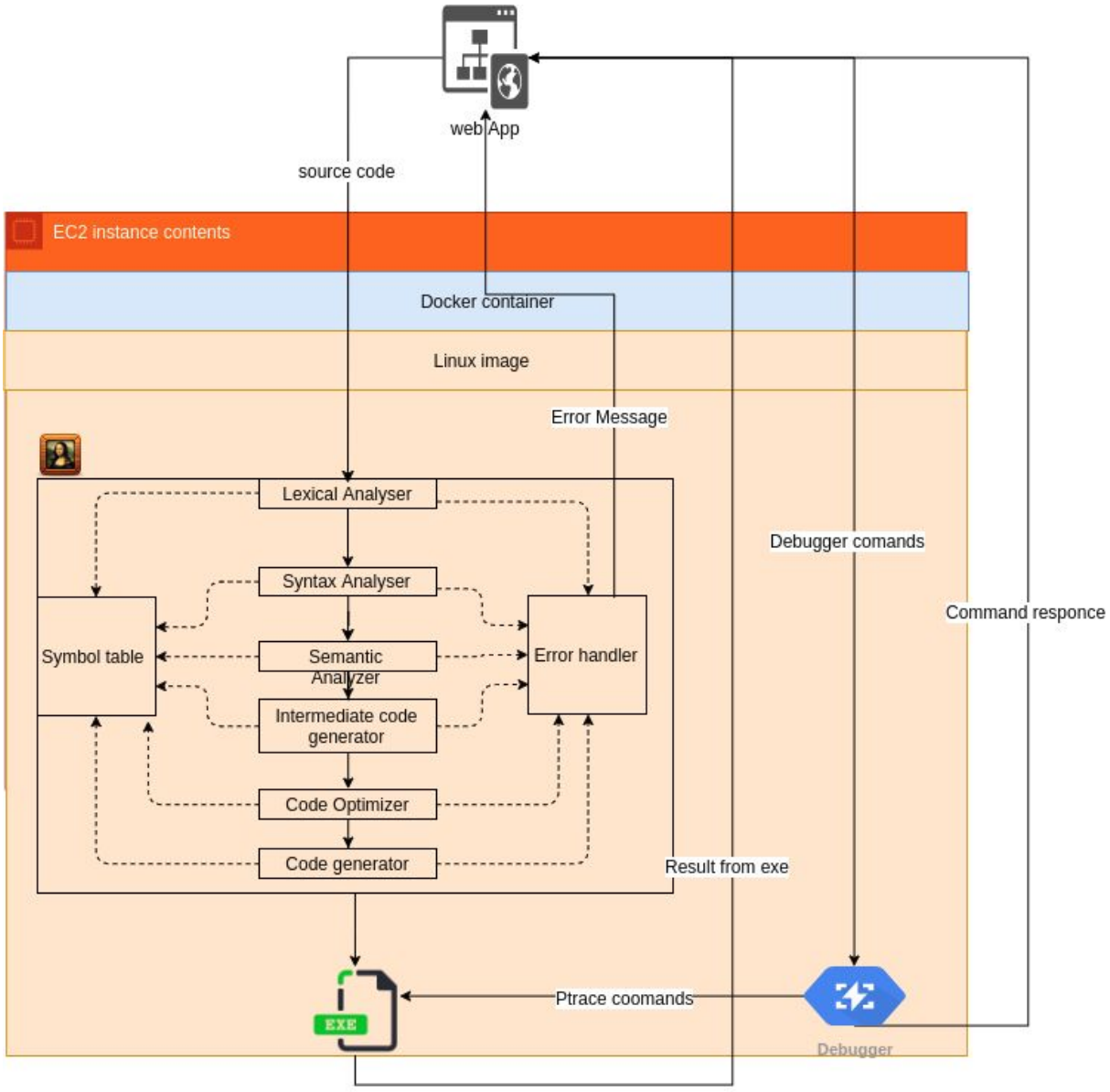
EC2 Instance: This will contain our dockerized linux image.

DTOS: Data transfer object (DTOS) is an object that is a simple container that is used to carry data between different processes and different layers of our application.

Domain model: consists of three different objects:

- A **domain service** is a stateless class that provides operations which are related to a domain concept but aren't a "natural" part of an entity or a value object.
- An **entity** is an object that is defined by its identity which stays unchanged through its entire lifecycle.
- A **value object** describes a property or a thing, and these objects don't have their own identity or lifecycle. The lifecycle of a value object is bound to the lifecycle of an entity.

4.3 Architecture overview



Web app: This will be the top layer in our system architecture of the web app is discussed above.

EC2 Instance: This will reside under the web app , an EC2 Instance is a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications on the Amazon Web Services (AWS). We will use this to host our dockerized linux image.

Docker: Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called containers. We will use a docker to contain the linux image.

Linux image: This will be the image we will use to compile run and debug our code. We will either use alpine linux or ubuntu linux. This decision will be based on which platform will meet our requirements without severe memory costs.

Mona programming language : This will be based on the linux image the architecture of our compiler will include.

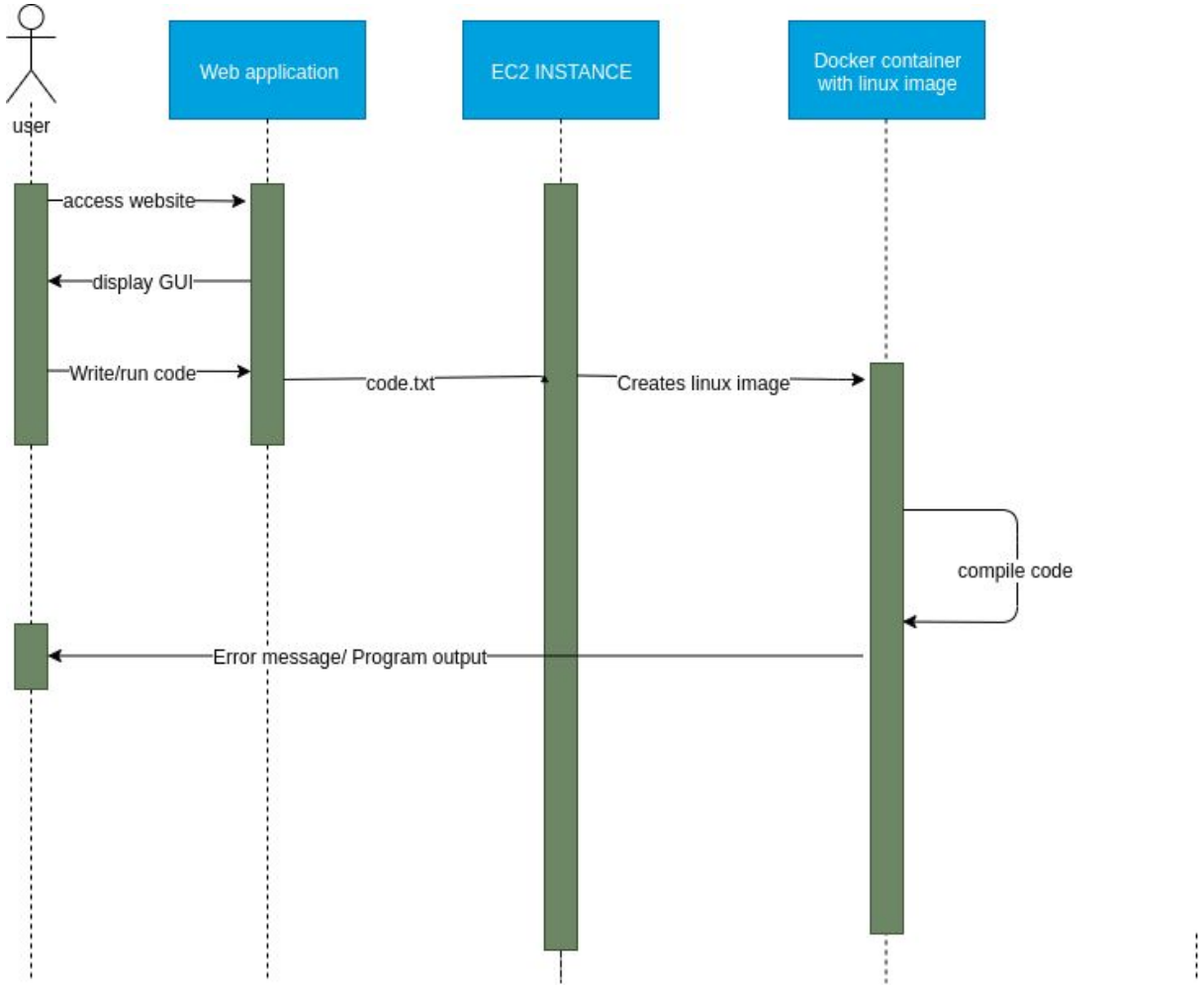
- **Lexical analyzer :** The lexical analyzer will break down the input source code into a sequence of Tokens and Lexemes.
- **Syntax analyzer:** Combines the tokens from lexical analysis into valid sentences based on the grammar. A Grammar is a set of rules that determines what is a valid combination of Tokens.
- **Semantic analyzer :** This stage will check for semantic errors and gather type information for the intermediate code generation.
- **Intermediate code generator:** Intermediate code generator will receive a syntax tree from the semantic analyzer. This can be converted to intermediate code. Intermediate code is abstract machine code which does not rely on target machine.
- **Code optimizer:** This phase is optional and can improve intermediate code to run faster.
- **Code generator:** This phase will translate intermediate code into object code allocating memory locations for data, and selecting registers.

EXE: after compilation is the code can be executed.

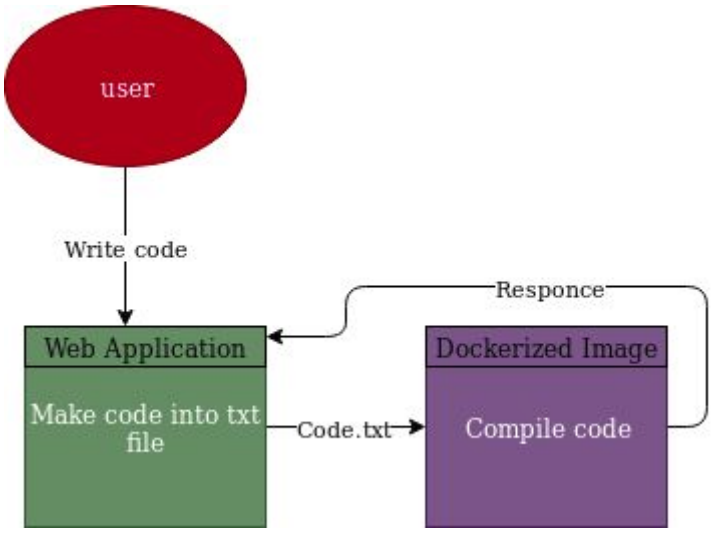
Debugger: The debugger will receive commands from the user and will use ptrace to debug the source code.

5. High level Design

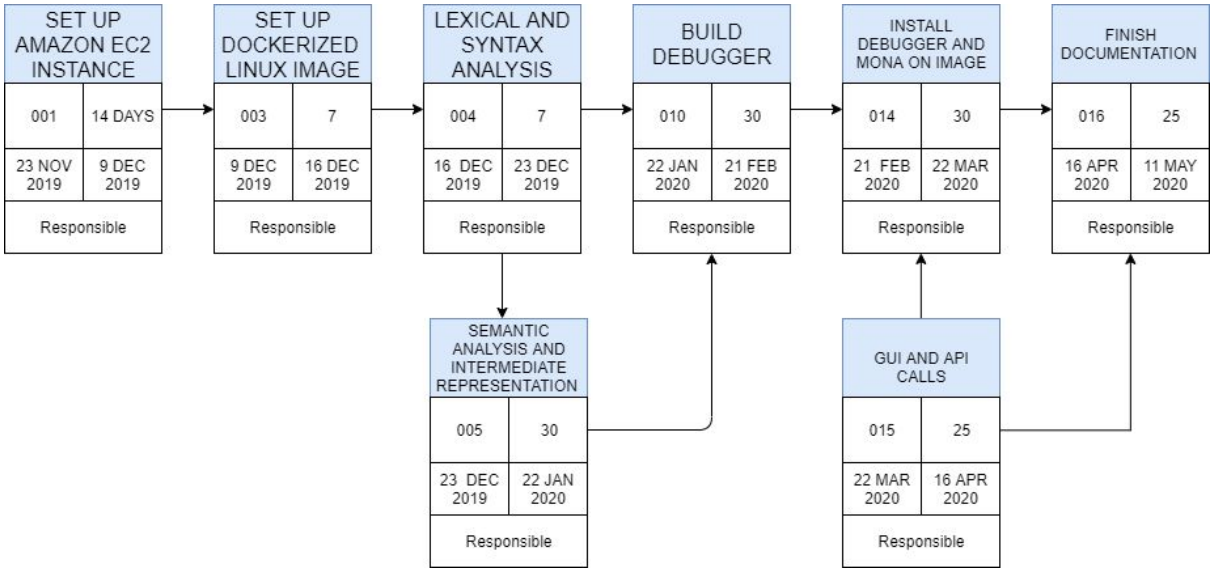
5.1 Simple transaction Sequence diagram



5.2 High Level Data Flow Diagram



6.Preliminary schedule



7.Appendices

(1) https://crossbrowsertesting.com/blog/development/best-programming-language-to-learn-first/
(2) https://www.daxx.com/blog/development-trends/number-software-developers-world
(3) https://apps.dtic.mil/dtic/tr/fulltext/u2/773391.pdf
(4) https://medium.com/@trungluongquang/java-and-python-which-is-better-to-learn-first-and-why-7f4cf5618a8e
(5) https://pythoninstitute.org/what-is-python/

(6) https://www.tandfonline.com/doi/full/10.11120/ital.2004.03020004