**Prolog – Introduction**

Prolog as the name itself suggests, is the short form of LOGical PROgramming. It is a logical and declarative programming language. Before diving deep into the concepts of Prolog, let us first understand what exactly logical programming is.

Logic Programming is one of the Computer Programming Paradigm, in which the program statements express the facts and rules about different problems within a system of formal logic. Here, the rules are written in the form of logical clauses, where head and body are present. For example, H is head and B1, B2, B3 are the elements of the body. Now if we state that “H is true, when B1, B2, B3 all are true”, this is a rule. On the other hand, facts are like the rules, but without any body. So, an example of fact is “H is true”.

Some logic programming languages like Datalog or ASP (Answer Set Programming) are known as purely declarative languages. These languages allow statements about what the program should accomplish. There is no such step-by-step instruction on how to perform the task. However, other languages like Prolog, have declarative and also imperative properties. This may also include procedural statements like “To solve the problem H, perform B1, B2 and B3”.

Some logic programming languages are given below −

* ALF (algebraic logic functional programming language).
* ASP (Answer Set Programming)
* CycL
* Datalog
* FuzzyCLIPS
* Janus
* Parlog
* Prolog
* Prolog++
* ROOP

## **Logic and Functional Programming**

We will discuss about the differences between Logic programming and the traditional functional programming languages. We can illustrate these two using the below diagram −

Diagram

Description automatically generated

From this illustration, we can see that in Functional Programming, we have to define the procedures, and the rule how the procedures work. These procedures work step by step to solve one specific problem based on the algorithm. On the other hand, for the Logic Programming, we will provide knowledge base. Using this knowledge base, the machine can find answers to the given questions, which is totally different from functional programming.

In functional programming, we have to mention how one problem can be solved, but in logic programming we have to specify for which problem we actually want the solution. Then the logic programming automatically finds a suitable solution that will help us solve that specific problem.

Now let us see some more differences below −

|  |  |
| --- | --- |
| **Functional Programming** | **Logic Programming** |
| Functional Programming follows the Von-Neumann Architecture, or uses the sequential steps. | Logic Programming uses abstract model, or deals with objects and their relationships. |
| The syntax is actually the sequence of statements like (a, s, I). | The syntax is basically the logic formulae (Horn Clauses). |
| The computation takes part by executing the statements sequentially. | It computes by deducting the clauses. |
| Logic and controls are mixed together. | Logics and controls can be separated. |

## **What is Prolog?**

Prolog or **PRO**gramming in **LOG**ics is a logical and declarative programming language. It is one major example of the fourth generation language that supports the declarative programming paradigm. This is particularly suitable for programs that involve **symbolic** or **non-numeric computation**. This is the main reason to use Prolog as the programming language in **Artificial Intelligence**, where **symbol manipulation** and **inference manipulation** are the fundamental tasks.

In Prolog, we need not mention the way how one problem can be solved, we just need to mention what the problem is, so that Prolog automatically solves it. However, in Prolog we are supposed to give clues as the **solution method**.

Prolog language basically has three different elements −

**Facts** − The fact is predicate that is true, for example, if we say, “Tom is the son of Jack”, then this is a fact.

**Rules** − Rules are extinctions of facts that contain conditional clauses. To satisfy a rule these conditions should be met. For example, if we define a rule as −

grandfather(X, Y) :- father(X, Z), parent(Z, Y)

This implies that for X to be the grandfather of Y, Z should be a parent of Y and X should be father of Z.

**Questions** − And to run a prolog program, we need some questions, and those questions can be answered by the given facts and rules.

## **History of Prolog**

The heritage of prolog includes the research on theorem provers and some other automated deduction system that were developed in 1960s and 1970s. The Inference mechanism of the Prolog is based on Robinson’s Resolution Principle, that was proposed in 1965, and Answer extracting mechanism by Green (1968). These ideas came together forcefully with the advent of linear resolution procedures.

The explicit goal-directed linear resolution procedures, gave impetus to the development of a general purpose logic programming system. The **first** Prolog was the **Marseille Prolog** based on the work by **Colmerauer** in the year 1970. The manual of this Marseille Prolog interpreter (Roussel, 1975) was the first detailed description of the Prolog language.

Prolog is also considered as a fourth generation programming language supporting the declarative programming paradigm. The well-known Japanese Fifth-Generation Computer Project, that was announced in 1981, adopted Prolog as a development language, and thereby grabbed considerable attention on the language and its capabilities.

## **Some Applications of Prolog**

Prolog is used in various domains. It plays a vital role in automation system. Following are some other important fields where Prolog is used −

* Intelligent Database Retrieval
* Natural Language Understanding
* Specification Language
* Machine Learning
* Robot Planning
* Automation System
* Problem Solving

**GNU Prolog – Introduction**

GNU Prolog is a free Prolog compiler with constraint solving over finite domains.

GNU Prolog is a Prolog compiler based on the Warren Abstract Machine (WAM). It first compiles a Prolog program to a WAM file which is then translated to a low-level machine independent language called mini-assembly specifically designed for GNU Prolog. The resulting file is then translated to the assembly language of the target machine (from which an object is obtained). This allows GNU Prolog to produce a native stand alone executable from a Prolog source (similarly to what does a C compiler from a C program). The main advantage of this compilation scheme is to produce native code and to be fast. Another interesting feature is that executables are small. Indeed, the code of most unused built-in predicates is not included in the executables at link-time.

A lot of work has been devoted to the ISO compatibility. Indeed, GNU Prolog is very close to the ISO standard for Prolog.

GNU Prolog also offers various extensions very useful in practice (global variables, OS interface, sockets,...). In particular, GNU Prolog contains an efficient constraint solver over Finite Domains (FD). This opens constraint logic programming to the user combining the power of constraint programming to the declarativity of logic programming. The key feature of the GNU Prolog solver is the use of a single (low-level) primitive to define all (high-level) FD constraints. There are many advantages of this approach: constraints can be compiled, the user can define his own constraints (in terms of the primitive), the solver is open and extensible (as opposed to black-box solvers like CHIP),…Moreover, the GNU Prolog solver is rather efficient, often more than commercial solvers.

GNU Prolog is inspired from two systems:

* wamcc: a Prolog to C compiler. The key point of wamcc was its ability to produce stand alone executables using an original compilation scheme: the translation of Prolog to C via the WAM. Its drawback was the time needed by gcc to compile the produced sources. GNU Prolog can also produce stand alone executables but using a faster compilation scheme.
* clp(FD): a constraint programming language over FD. Its key feature was the use of a single primitive to define FD constraints. GNU Prolog is based on the same idea but offers an extended constraint definition language. In comparison to clp(FD), GNU Prolog offers new predefined constraints, new predefined heuristics, reified constraints,…

Here are some features of GNU Prolog:

* Prolog system:
  + conforms to the ISO standard for Prolog (floating point numbers, streams, dynamic code,…).
  + a lot of extensions: global variables, definite clause grammars (DCG), sockets interface, operating system interface,…
  + more than 300 Prolog built-in predicates.
  + Prolog debugger and a low-level WAM debugger.
  + line editing facility under the interactive interpreter with completion on atoms.
  + powerful bidirectional interface between Prolog and C.
* Compiler:
  + native-code compiler producing stand alone executables.
  + simple command-line compiler accepting a wide variety of files: Prolog files, C files, WAM files,…
  + direct generation of assembly code 15 times faster than wamcc + gcc.
  + most of unused built-in predicates are not linked (to reduce the size of the executables).
  + compiled predicates (native-code) as fast as wamcc on average.
  + consulted predicates (byte-code) 5 times faster than wamcc.
* Constraint solver:
  + FD variables well integrated into the Prolog environment (full compatibility with Prolog variables and integers). No need for explicit FD declarations.
  + very efficient FD solver (comparable to commercial solvers).
  + high-level constraints can be described in terms of simple primitives.
  + a lot of predefined constraints: arithmetic constraints, boolean constraints, symbolic constraints, reified constraints,…
  + several predefined enumeration heuristics.
  + the user can define his own new constraints.
  + more than 50 FD built-in constraints/predicates.

**Lab 1: Prolog - Environment Setup**

We will look into how to install Prolog in our system.

**Prolog Version**

In this tutorial, we are using GNU Prolog, Version: 1.5.0

**Official Website**

This is the official GNU Prolog website where we can see all the necessary details about GNU Prolog, and also get the download link.

<http://www.gprolog.org/>

**Direct Download Link**

Given below are the direct download links of GNU Prolog for Windows. For other operating systems like Mac or Linux, you can get the download links by visiting the official website (Link is given above) –

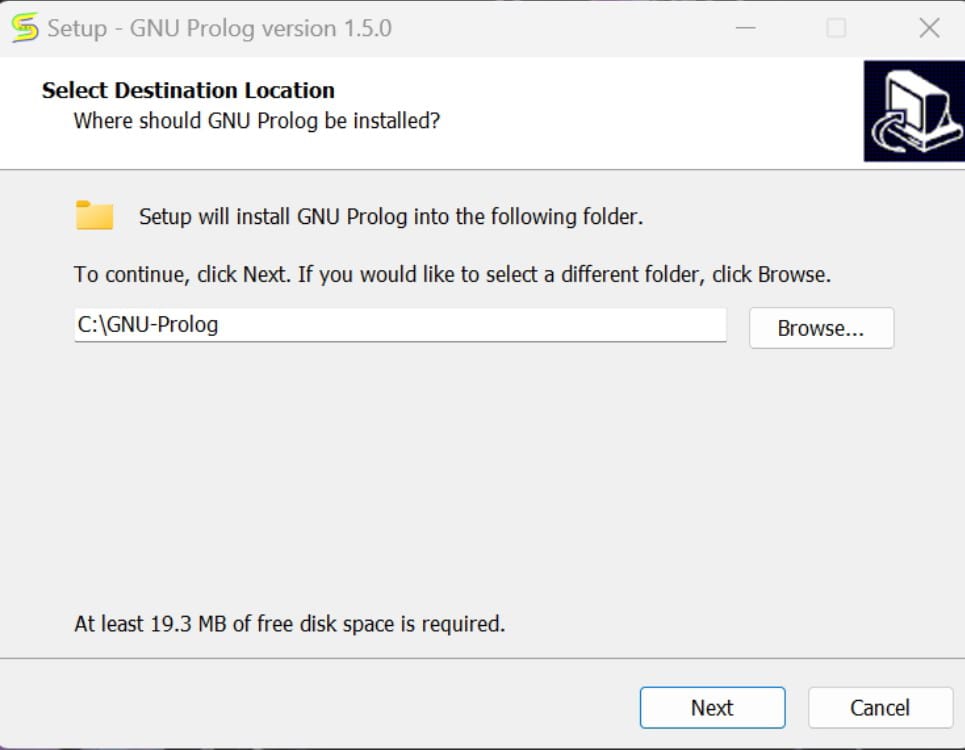
* [Windows intel 32 bits auto-install setup](http://www.gprolog.org/setup-gprolog-1.5.0-msvc-x86.exe) (compiled under ix86 / Windows 10 with MSVC++).
* [Windows intel 64 bits auto-install setup](http://www.gprolog.org/setup-gprolog-1.5.0-msvc-x64.exe) (compiled under x86\_64 / Windows 10 with MSVC++).

**Installation Guide**

* Download the exe file and run it.
* You will see the window as shown below, then click on **next** −



Select proper **directory** where you want to install the software, otherwise let it be installed on the default directory. Then click on **next**.



You will get the below screen, simply go to **next**.

Graphical user interface, text, application, email

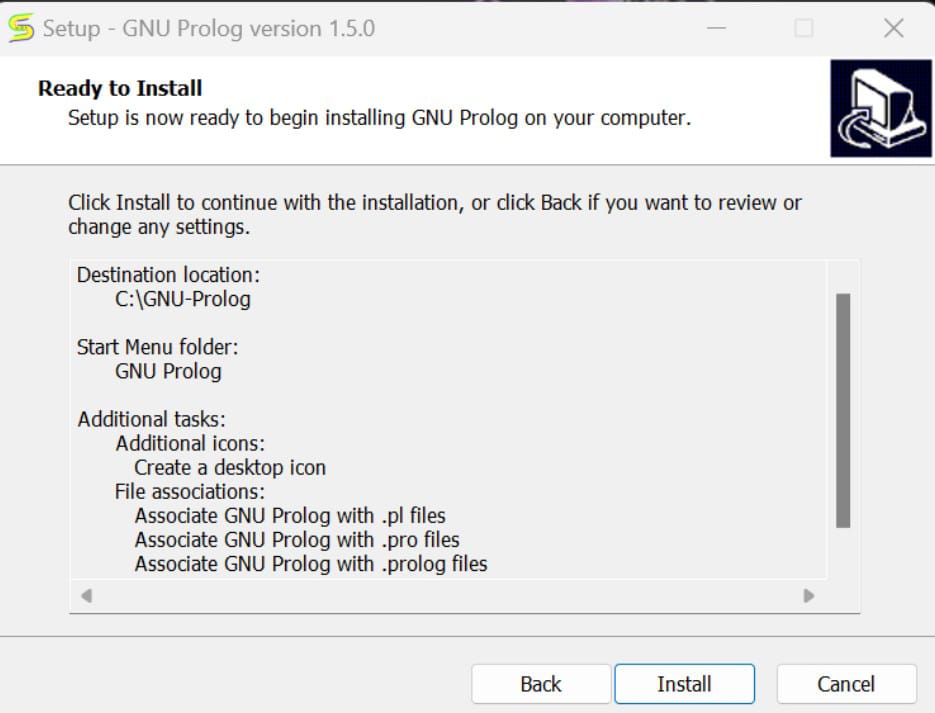
Description automatically generated

You can verify the below screen, and **check/uncheck**appropriate boxes, otherwise you can leave it as default. Then click on **next**.

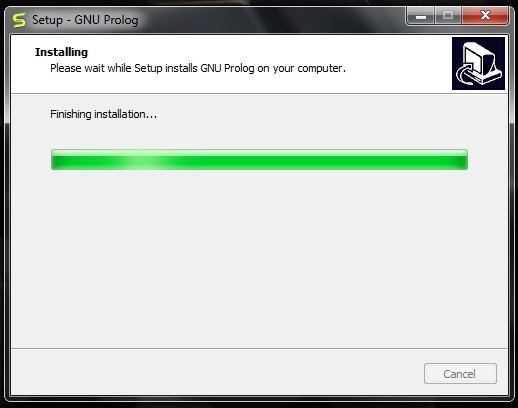
Graphical user interface, text, application, email

Description automatically generated

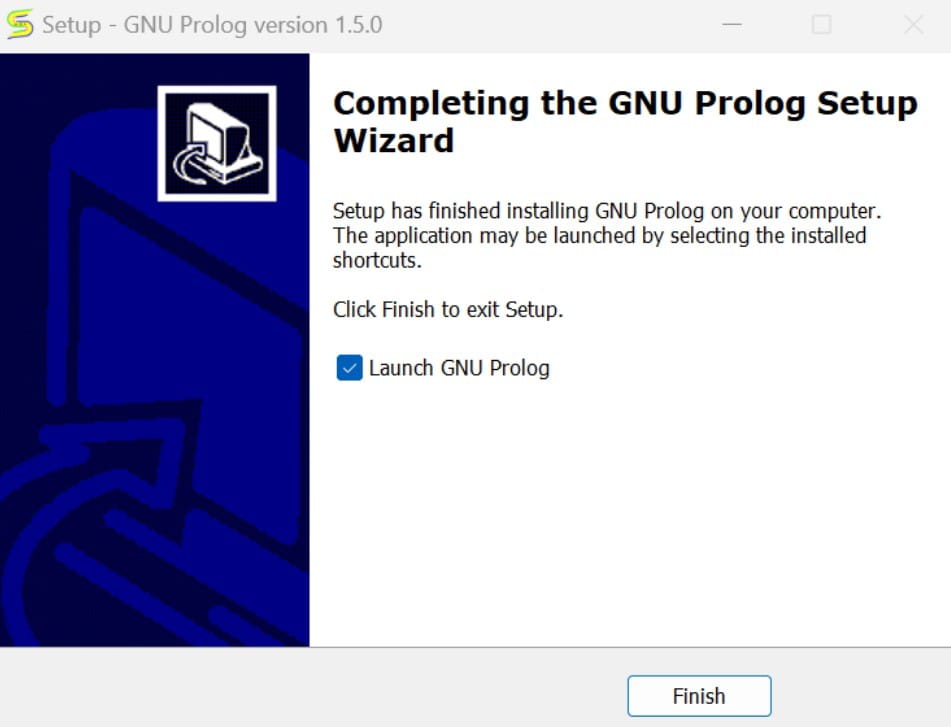
In the next step, you will see the below screen, then click on **Install**.



Then wait for the installation process to finish.



Finally click on **Finish** to start GNU Prolog.



The GNU prolog is installed successfully as shown below –

Graphical user interface, text, application, Word

Description automatically generated

**Creating a Prolog program.**

Here is how you create a Prolog program.

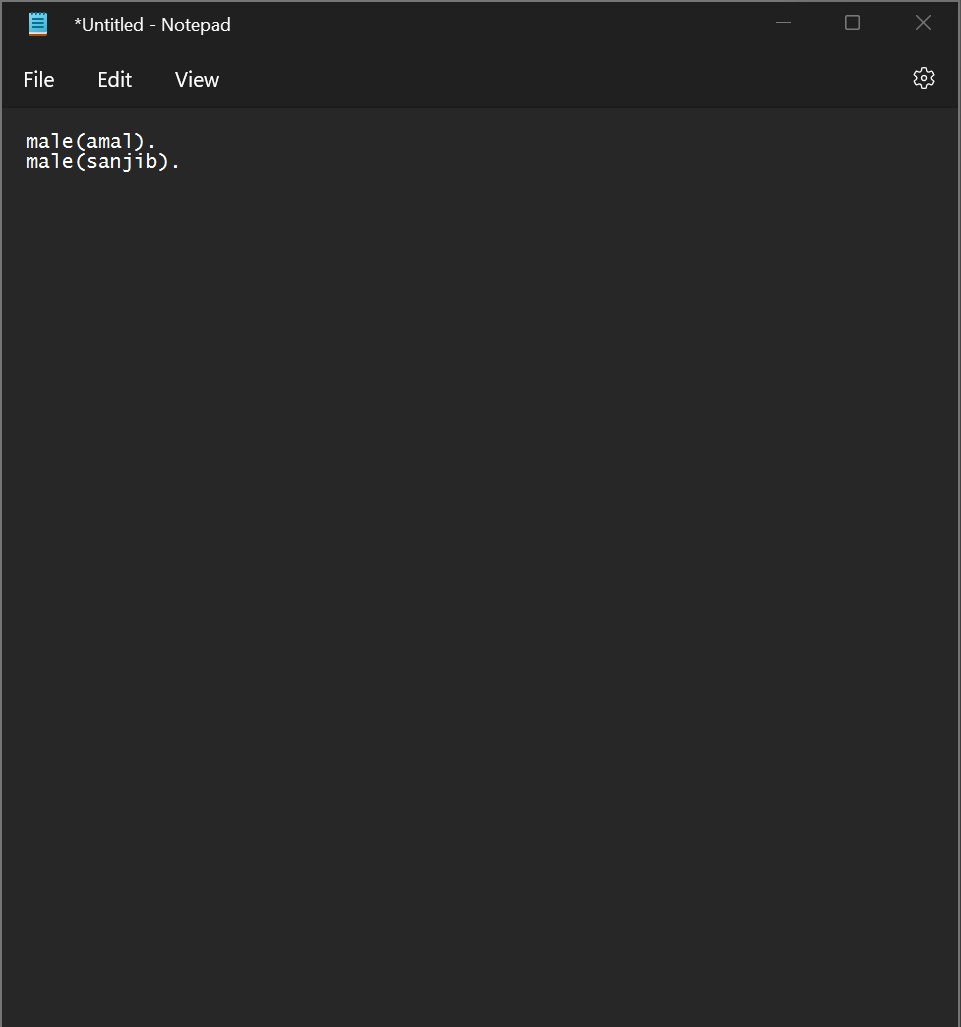
Graphical user interface, application, Word

Description automatically generated

Create a folder as a new folder on our desktop. **Graphical user interface, text, application, email

Description automatically generated**

Double-click the *Programmer's File Editor* icon.

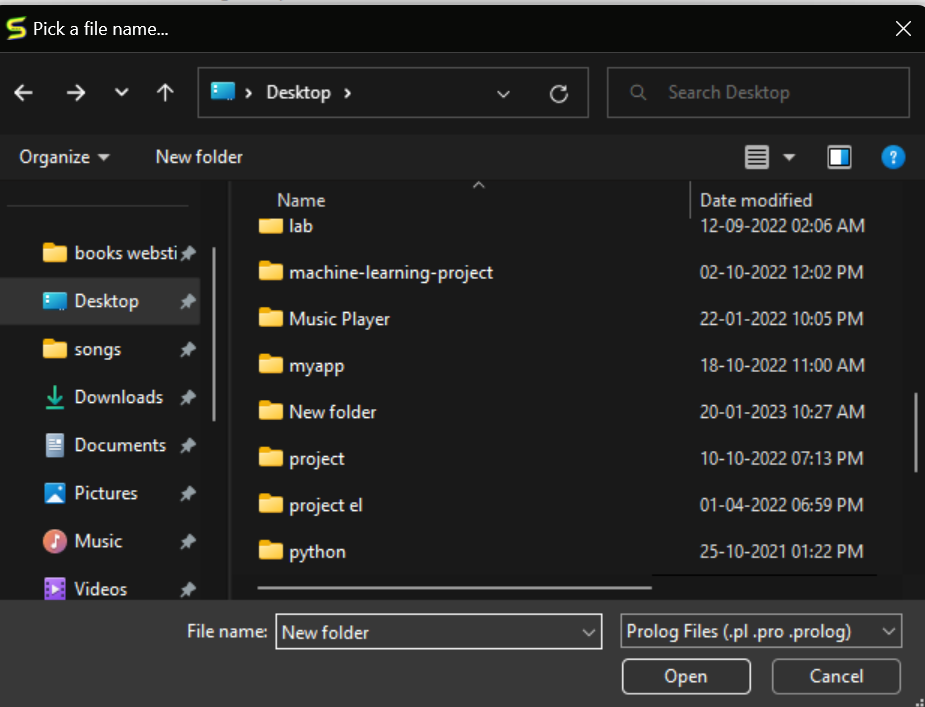


Make sure that you give your Prolog files the extension .pl .

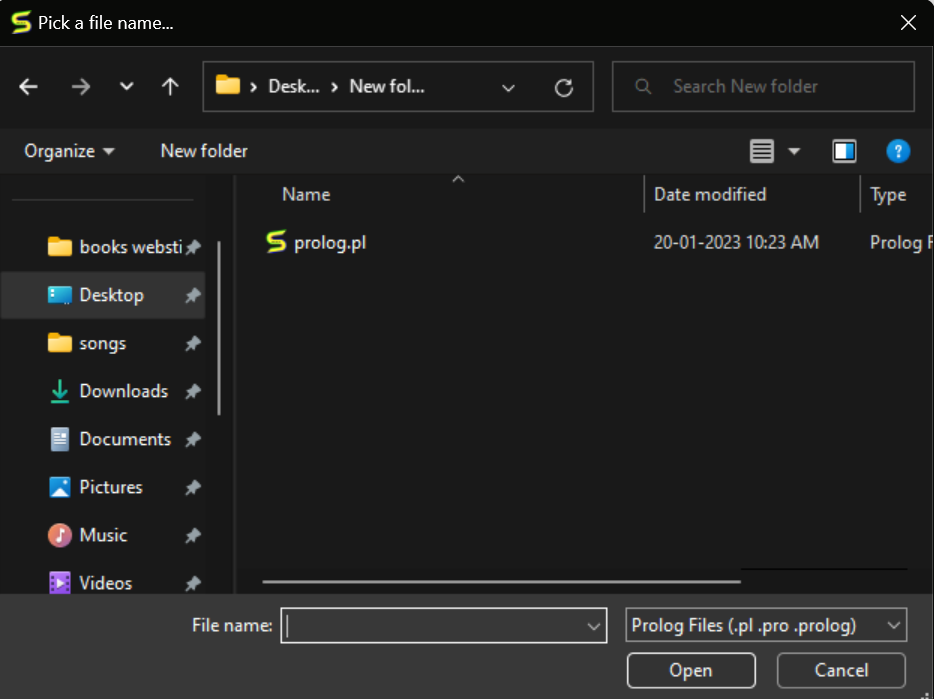
A screenshot of a computer

Description automatically generated with medium confidence

Insert the prolog file into the new folder.



The prolog file can be seen.



Use the file menu and change the directory.



For changing the directory, select the new folder which you have created.

Graphical user interface, text, application

Description automatically generated

The changed directory can be seen.

Graphical user interface, text, application, email

Description automatically generated

Write down the prolog file name which is present in the new folder for the execution.

Graphical user interface, text, application, email

Description automatically generated