



PROJECT REPORT

---

# Intelligent Chrome Support : Leveraging aima3 for Resolving Technical Issues

---



*Group members :*

AYMEN DENOUB

YASSER MESSAHLI

BENSMAIL ANIS

20 April 2024

## **I Introduction**

In this report, we'll discuss our expert system app designed to help users troubleshoot technical issues in Google Chrome. We'll explain why we chose this topic, provide an overview of our solution, and describe the project in general.

## **II Chosen Topic : Why we chose it!**

We chose to focus on creating a Google Chrome Technical Support System because many people use Chrome for browsing the internet, and it's essential to help them solve any technical problems they encounter. By providing a simple interface and step-by-step solutions, we aim to make troubleshooting easier for users, even if they're not tech-savvy.

## **III General Description**

Our project focuses on developing an expert system application using `aima3` and `tkinter`. This system is designed to assist users in resolving technical problems encountered while using Google Chrome.

The application organizes Chrome issues into distinct categories such as Performance, Security, Installation, and others. Users can select the category relevant to their issue and respond to targeted questions to pinpoint the problem.

Leveraging this information, the expert system offers potential solutions to address the identified issue. Through this intuitive approach, our goal is to deliver an effective and user-friendly tool for diagnosing and resolving Chrome-related technical issues.

## **IV User Interface**

We used the popular Python library, `Tkinter`, for the user interface. It provides a set of modules and classes to build desktop applications with a rich and interactive interface. `Tkinter` is based on the `Tk` GUI toolkit, which originated as a part of the `Tcl` scripting language but has been ported to several other programming languages, including Python.

## UI

- When the desktop app is first opened, the user is presented with a list of categories : Performance, Security, Installation, Web Content Problems, Sync Issues, Appearance and Themes, and Search Problems.
- The user then selects a category that corresponds to their issue.

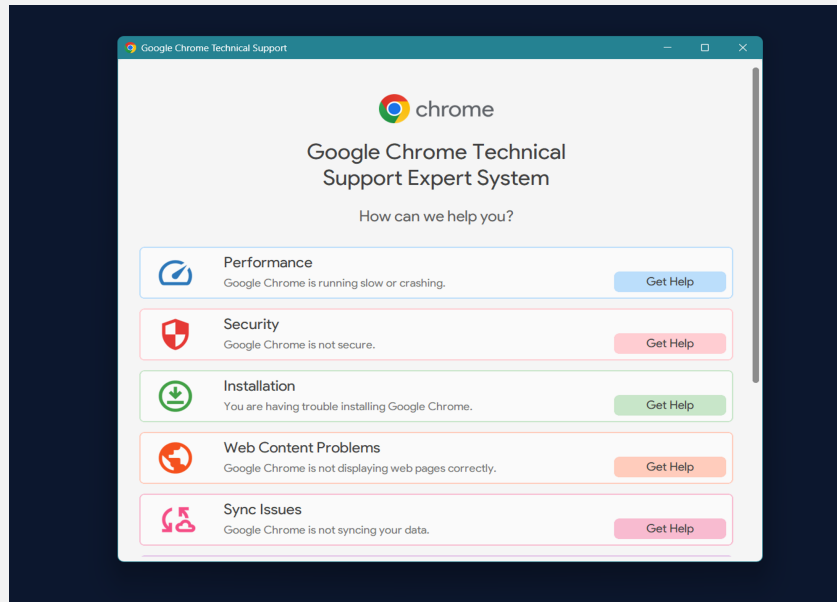


FIGURE I – Categories part

- The app displays a sequence of questions in the selected category to identify the user's problem.

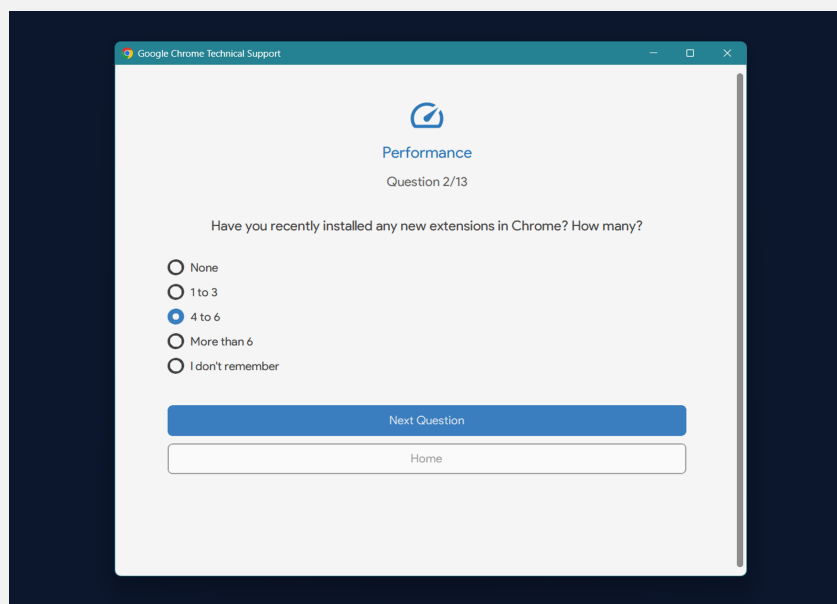


FIGURE II – Questions part

TABLE I – User Interface

## UI

- At the end, the app presents a solution based on the user's answers and our expert system engine.

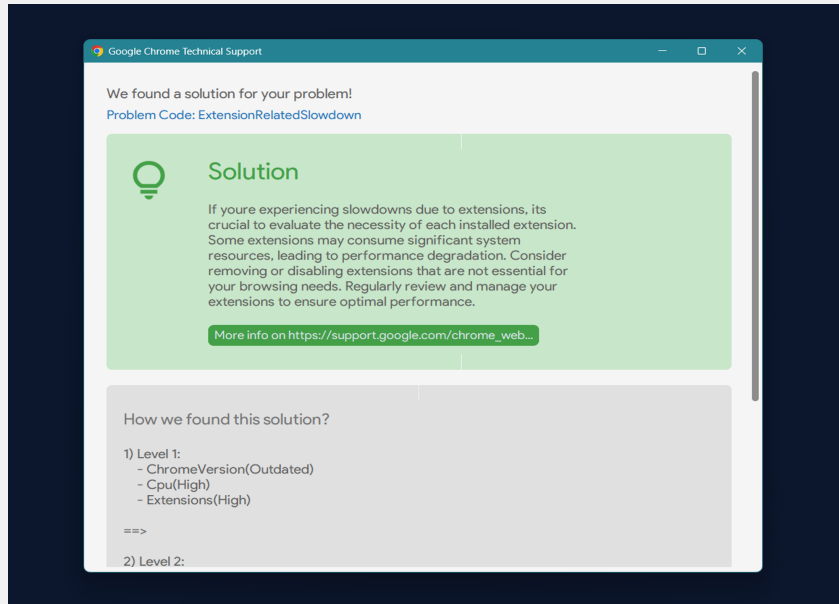


FIGURE III – Problem/Solution part

TABLE II – User Interface

## V Utilizing the aim3 Package

The aim3 package, short for "Artificial Intelligence : A Modern Approach," provides a comprehensive set of tools for building intelligent systems. It offers functionality for logic programming, knowledge representation, and inference, which are essential components for creating an expert system. By leveraging the capabilities of the aim3 package, we are able to develop a robust and efficient solution for Chrome support.

## VI Defining Rules and Knowledge Base

In our expert system application, we define rules using first-order logic expressions to represent relationships between different variables and problem types. These rules are structured in a knowledge base, which serves as the repository of domain knowledge for the system. Each rule encapsulates a specific condition or inference logic that helps identify the underlying issue based on user input. For example, a rule in the Performance category may state : "If the browsing experience is slow and a large number of tabs are open, then the potential issue is memory overload." Similarly, rules are defined for other categories such as Security, Installation, and Sync Issues, covering a wide range of potential Chrome-related problems.

```

expert_system > kb.py > ...
1  rules = {
2      # Performance issues
3      "Performance" : [
4          'BrowsingExperience(Slow)',
5          'Tabs(High) ==> Potential(MemoryOverload)',
6          'BrowsingExperience(Slow) ==> Potential(MemoryOverload)',
7          'Potential(MemoryOverload) & Extensions(High) ==> ProblemType(ExtensionRelatedSlowdown)',
8          'Potential(MemoryOverload) & Ram(Low) ==> ProblemType(ResourceLimitation)', # Alternative Inference
9          'BrowsingExperience(Slow) & Glitches(Graphical) & GraphicsCard(Dedicated) ==> Potential(HardwareAccelerationConflict)', # E2 (new)
10         'Potential(HardwareAccelerationConflict) & HardwareAcceleration(Enabled) ==> ProblemType(HardwareAcceleration)', # F2 (new)
11         'BrowsingExperience(Slow) & Content(Heavy) & Antivirus(Activated) ==> Potential(AntivirusInterference)', # E3 (new)
12         'Potential(AntivirusInterference) & ExclusionPossible(Chrome) ==> ProblemType(AntivirusInterference)', # F3 (new)
13         'BrowsingExperience(Slow) & NewTabs(Slow) & Cpu(High) ==> Potential(ResourceContention)', # E4 (new)
14         'Potential(ResourceContention) & BackgroundProcesses(High) ==> ProblemType(BackgroundProcessesImpact)'
15     ],
16
17     #Security
18     "Security" : [
19         'EmailReceived(Suspicious) & Link(Clicked) ==> Potential(PhishingAttempt)',
20         'Potential(PhishingAttempt) & Emailfiltering(Disabled) ==> ProblemType(PhishingAttempt)',
21         'BrowserBehavior(Unexpected) ==> Potential(MalwareInfection)',
22         'Popups(Unusual) & SoftwareDownloaded(Untrusted) ==> Potential(MalwareInfection)',
23         'Potential(MalwareInfection) & Antivirus(Activated) ==> ProblemType(AntivirusDetectionLimitations)',

```

FIGURE IV – Some initial rules in the system

## VII System Inference

The expert system employs forward chaining inference to deduce the root cause of the user's issue based on their responses to targeted questions. Forward chaining starts with the available information (e.g., user input) and works forward through the rules in the knowledge base to determine the eventual outcomes or conclusions. By iteratively applying rules and evaluating conditions, the system progresses step-by-step towards identifying the specific problem type affecting the user's Chrome experience. This approach allows the system to dynamically update its inference state as the user provides input, guiding them towards a resolution. The inference process is facilitated by the `aima3` package's logic programming capabilities, which enable efficient evaluation of logical expressions and rule-based reasoning. As the user provides input, the system dynamically updates its inference state, guiding the user towards a resolution.

### VII.1 Example of Inference Process

Let's consider a user who is experiencing slow browsing in Google Chrome. Upon selecting the Performance category in the application, the system prompts the user with targeted questions related to their browsing experience, such as the number of tabs open and CPU usage. Based on the user's responses, the system applies relevant rules from the knowledge base to infer potential issues.

For instance, if the user indicates slow browsing and a high number of tabs open, the system identifies a potential memory overload issue according to the rule : 'BrowsingExperience(Slow) and Tabs(High) leads to Potential(MemoryOverload)'. Similarly, if the user reports slow browsing and low RAM, the system infers a potential resource limitation problem as per the rule : 'Potential (MemoryOverload) and Ram(Low) leading to ProblemType(ResourceLimitation))'.

Subsequently, based on the identified potential issues, the system may present additional questions or recommendations tailored to address memory overload or resource limitation, guiding the user towards a solution. If memory overload is identified, the system may suggest closing unnecessary tabs or optimizing memory usage. Conversely, if resource limitation is inferred, the system may recommend upgrading RAM or closing resource-intensive applications.

## VIII Agenda

The agenda is an essential part in the full expert system, it solves the problem of conflict between the fired rules. If two or many rules are fired because of the input facts they will be putted in the agenda and its task is to define which one will be activated first based on predefined priorities. In our project the agenda is a python class :

- It will be initialized with a dictionary of fact-priority pairs.
- It has a function called performsort that sort the facts based on their priorities.

The agenda is located in the utils.py file and will be used in the engine.py file.

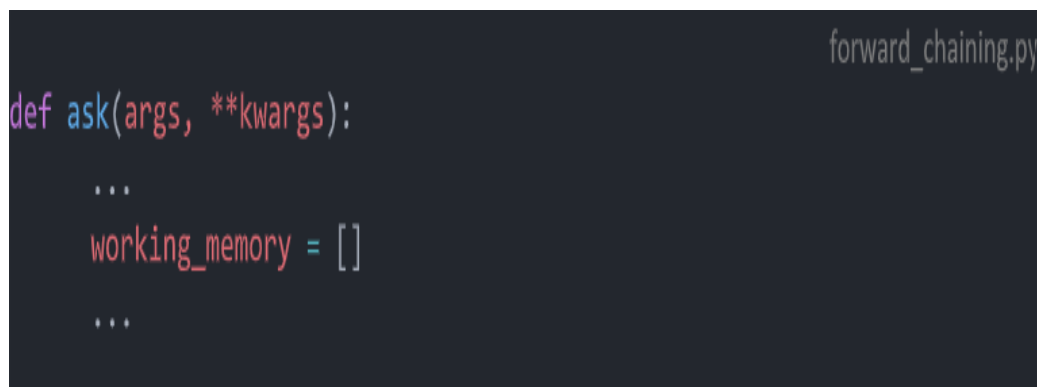
A screenshot of a code editor showing the definition of the 'Agenda' class in a file named 'utils.py'. The code is written in Python and includes a class definition with two methods: '\_\_init\_\_' and 'perform\_sort'.

```
utils.py  
  
class Agenda:  
    def __init__(self, category):  
        ...  
  
    def perform_sort(self, facts):  
        ...
```

FIGURE V – Agenda part

## IX Working Memory

The working memory is a component of the inference engine that stores the temporal fact that are produced during the inference process to use them. In our project the working memory is simply a variable (a list).

A screenshot of a code editor showing the 'ask' function in a file named 'forward\_chaining.py'. The function is defined with 'args' and '\*\*kwargs' as parameters. Inside the function, there is a line that initializes 'working\_memory' as an empty list.

```
forward_chaining.py  
  
def ask(args, **kwargs):  
    ...  
    working_memory = []  
    ...
```

FIGURE VI – Working Memory part

## **X Concluded Results**

Through the inference process, the expert system can conclude the specific problem type affecting the user's Chrome experience. These concluded results are based on the logical deductions made by the system using the defined rules and user input. Examples of concluded results may include Extension Related Slowdown, Antivirus Interference, or Sync Conflict or many others. Once the problem type is identified, the system provides targeted solutions and recommendations to help users resolve the issue effectively. These solutions may include adjusting settings, disabling conflicting extensions, or performing troubleshooting steps tailored to the identified problem type.