PROJECT TARS : Final Architecture ( UPDATED)

Chris King

CST-452 Capstone Project Final Architecture & Design

Grand Canyon University

Professor Mark Reha

3.0

04/21/2023

**ABSTRACT**

The peak of cutting-edge technology can be defined by many different subjects. Embedded systems, cloud computing, API; many of these represent the true power that technology is capable of. However, there is one large, powerful area of technology that cultivates a particular interest; artificial intelligence. What makes AI so powerful is its applicability in the real world. There are almost no limits to what you can apply AI to. From making arguably self-aware robots that can be your best friend, artists, all the way to creating an intelligent, automated process behind your application’s fundamental processes. For this project, an AI model will be created and designed to be able to be mounted on different systems.

The problem with current industry-level AI is its portability; AI today feels more like a software application than an individual entity. The goal of this project is to lay a foundation to solve exactly this problem, by creating an AI model that’s ‘portable’, in that it can be transported from environment to environment with relative ease. With inspiration taken from multiple science-fiction movies, an AI model with this level of portability will kickstart the solution of creating an AI that is more entity than application. The key function to this portability is to give this AI model the ability to learn to perform tasks that it wasn’t explicitly programmed to perform. As it’s important to initialize an AI like this on a corpus for a particular subject to kickstart its intelligence with a particular field, the corpus I’ve chosen to initialize this AI on is meteorology.

|  |
| --- |
| History and Signoff Sheet |

**Change Record**

|  |  |  |
| --- | --- | --- |
| **Date** | **Author** | **Revision Notes** |
| 04/19/2023 | Chris King | Migration from anaconda to virtualenv |
| 04/15/2023 | Chris King | Wrap communication endpoint in Flask API. |
|  |  |  |

|  |
| --- |
| **Overall Instructor Feedback/Comments**  Diagrams look good, however technical frameworks should be moved from the integration layer into the software layer. |

|  |
| --- |
| **Overall Instructor Feedback/Comments** |

**Integrated Instructor Feedback into Project Documentation**

☐ Yes ☐ No

**TABLE OF CONTENTS**

**Design Overview 4**

**Detailed High-Level Solution Design 5**

**Detailed Technical Design 6**

**Appendix A – Technical Issue and Risk Log 7**

**Appendix B – References 8**

**Appendix C – External Resources 9**

**Problem**

The challenge being approached is industry AI. Today, AI is handled like a software application, used to either support another software application, or to connect a user to information that’s already available. Research in developing AI that acts as an individual entity is still relatively new in the industry, and this project will act as a contribution to that research.

**Solution**

An AI NLP Model (Artificial Intelligence, Natural Language Processing) will be developed on a personal computer, using TensorFlow and Python, and then planted onto two different Raspberry Pi environments. Three diagrams will be provided below. One depicting the physical architecture of the solution, and two for the logical solution. A process flow diagram, as well as a logical architecture will be provided below. Accompanying the diagrams will be a list of Hardware and Software technologies, and proof of concepts.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Deliverable Acceptance Log | | | | | |
| ID | Deliverable Description | Comments | Evaluator (internal or external as applicable) | Status | Date of Decision |
| 1 | API Documentation | https://openweathermap.org/api/one-call-3 | External | Up | 10/15/2022 |
| 2 | API Documentation | https://platform.openai.com/overview | External | Up | 02/15/2023 |
| 3 |  |  |  |  |  |
| 4 |  |  |  |  |  |
| 5 |  |  |  |  |  |

**Detailed High-Level Solution Design**

|  |  |  |
| --- | --- | --- |
| Proof of Concepts | |  |
| **Description** | **Rationale** | **Results** |
| 1. Data Science Bot - Trained on Wiki to answer questions related to data science | This proof of concept is integral for the foundational understanding of Natural Language Processing, as well as outlining the concept flow of the system. | Successful. |
| 2 - Intent/Response Bot with TensorFlow and Raspberry Pi | This proof of concept is vital for understanding the Intent/Response system, and Natural Language Processing with TensorFlow AI. This system will be used for development of Project TARS. | Successful. |
| 3 - Data Science Bot - Extended to Raspberry Pi | This proof of concept is required for understanding Raspberry Pi, and assessing all outlined project risks. Concepts such as deployment and device-to-pi communication are approached through this proof of concept. | Succesful. |
| 4 - |  |  |
| 5 - |  |  |

|  |
| --- |
| Hardware and Software Technologies |
| 1 - MSI GS76 Stealth Laptop Computer - Intel Core I9 Processor, GeForce RTX 3070 Graphics Card |
| 2 - TensorFlow 2.0 AI Framework |
| 3 - Python3 Programming Language |
| 4 - MySQL 8.0 Relational Database |
| 5 - Jupyter Local Development Server |
| 6 - Raspberry Pi 4B 8GB Starter Kit 8GB RAM - External 64GB MicroSD |
| 7 - FileZilla File Transfer Protocol |
| 8 - PuTTy SSH Terminal |

**Logical Solution Design:**

**The Logical Solution architecture is divided into three areas. The primary area, AI System, consists of two portions, the User Interface and the Local Invocation.**

**The presentation interface contains the SSH Terminal, as well as FTP File transfer which can be used for directly transferring files from the laptop to the embedded system. The Local Invocation consists of the AI, powered by TensorFlow and Python, and the NLP algorithms, which will connect to and leverage the Engine.**

**The engine area consists of the MySQL 8.0 database, and the Dataset API’s, as well as the Corpus. This engine will be responsible for the information the system uses in order to operate.**

**The Application Framework area consists of the technical blueprint that the system is built on. This includes the TensorFlow framework that the AI will be built on, as well as the various Python libraries that the system will use for innate operation.**

Graphical user interface, diagram, application

Description automatically generated

**Physical Solution Design:**

**Below is the physical architecture for the proposed solution. There are three system tiers to the solution.**

**End-User Tier**

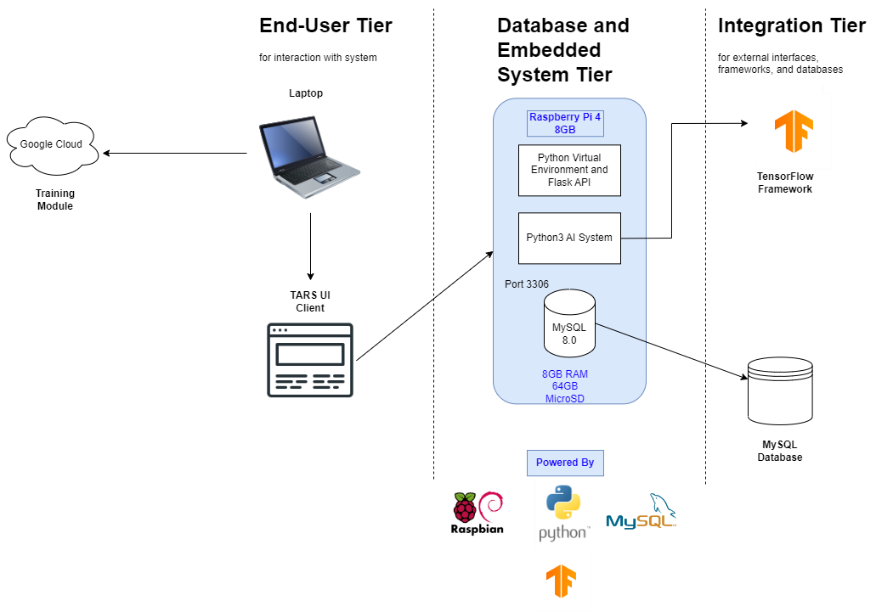
**The end-user tier is the system endpoint in which the user will communicate with the system. In this solution, the user will use a laptop, and an SSH terminal, in order to communicate with the system. Once SSH connection is established, the AI will automatically wake up and connect with the user for communication.**

**Database and Embedded System Tier**

**This tier is the primary component of the system, the Raspberry Pi. This is where the AI is located, as well as the configurations for connections to the corpus and dataset API’s, as well as the MySQL Database. This system is powered by 8GB of ram, as well as a 64GB MicroSD card, of which is upgradeable should the need arise. The technologies that power this tier are Python3, MySQL 8.0, Raspbian OS, and TensorFlow3 AI.**

**Integration Tier**

**The integration tier consists of the external interfaces that will be utilized by the Embedded System tier. These technologies exist in the cloud, and are retrieved through configurations in the Embedded System tier through the use of connections and APIs. While the database will be accessed through internal configurations on port 3306, the corpus and datasets will be retrieved as python imports via APIs.**



**Detailed Technical Design**

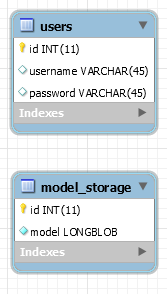
**General Technical Approach:**

The general technical approach to this project was extremely ambitious. Originally wanting to implement AI concepts such as response generation and unsupervised learning, I eventually realized that the same goals of this project can be initially accomplished with less-advanced natural language processing objectives. With this information in mind, I started to research more implementable AI concepts. After experimenting with AI through proof of concepts, I discovered that the technical approach I wanted to take would be by combining API’s and corpus files with the utilization of AI intents and responses. The way AI intents work with machine learning is less intelligent, but more implementable. It works by hard-creating a list of key-words, and a list of responses. The AI uses machine learning in order to process and understand sentences given by an end-user, and break each word down into tokens. These tokens are then cross-referenced with the list of intents, and a response is elected from the created list of responses that best suits what the user is asking for. By injecting data pulled from the data API’s into the list of responses, and corpus files to train the AI on possible input from the user, the intelligence, usability, and scalability we are looking for can be accomplished.

**Key Technical Design Decisions:**

There are a myriad of natural language processing frameworks to choose from when approaching a project like this. When considering the in-scope and out-of-scope objectives of this project, two frameworks came to mind for the respective scopes. For in-scope, TensorFlow AI was elected. Having previous experience with developing Computer Vision systems with TensorFlow AI, I have an innate familiarity with this framework when working with artificial intelligence. This framework is a lot more limited when it comes to an intelligent Natural Language Processing entity, however the implementability and modularity of this framework is unmatched. For out-of-scope, the system will likely be migrated onto the GPT-3 framework. Unlike TensorFlow AI, GPT-3 is an open-source framework dedicated to Natural Language Processing, and has ready access to training sets. This framework isn’t free, however, and requires a cost in order to access developer tokens that are used to develop with this framework. This framework is extremely powerful, and leveraging the power of GPT-3 would be overkill for initial development of TARS. From the portability perspective, the chat endpoint will be wrapped in a secure Flask API to allow for cross-platform communication.

**Database ER Diagram: three tables are involved with the Database. The modelstorage table contains the model’s memory. The users table contains the users’ information so that they can log into the system. The tokenizer storage simply contains the embedding matrix.**



Graphical user interface, text, application

Description automatically generated

**Database DDL Scripts:**

-- phpMyAdmin SQL Dump

-- version 4.9.5

-- https://www.phpmyadmin.net/

--

-- Host: localhost:8891

-- Generation Time: Apr 21, 2023 at 07:11 PM

-- Server version: 5.7.24

-- PHP Version: 7.4.1

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Database: `tensorflow\_models`

--

CREATE DATABASE IF NOT EXISTS `tensorflow\_models` DEFAULT CHARACTER SET utf8 COLLATE utf8\_general\_ci;

USE `tensorflow\_models`;

-- --------------------------------------------------------

--

-- Table structure for table `model\_storage`

--

CREATE TABLE `model\_storage` (

`id` int(11) NOT NULL,

`model` longblob NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Dumping data for table `model\_storage`

--

INSERT INTO `model\_storage` (`id`, `model`) VALUES

(1, );

-- --------------------------------------------------------

--

-- Table structure for table `tokenizer\_storage`

--

CREATE TABLE `tokenizer\_storage` (

`id` int(11) NOT NULL,

`tokenizer` blob

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Dumping data for table `tokenizer\_storage`

--

INSERT INTO `tokenizer\_storage` (`id`, `tokenizer`) VALUES

(1, );

-- --------------------------------------------------------

--

-- Table structure for table `users`

--

CREATE TABLE `users` (

`id` int(11) NOT NULL,

`username` varchar(45) DEFAULT NULL,

`password` varchar(45) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Dumping data for table `users`

--

INSERT INTO `users` (`id`, `username`, `password`) VALUES

(1, 'sasqe', 'sq');

--

-- Indexes for dumped tables

--

--

-- Indexes for table `model\_storage`

--

ALTER TABLE `model\_storage`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `tokenizer\_storage`

--

ALTER TABLE `tokenizer\_storage`

ADD PRIMARY KEY (`id`);

--

-- Indexes for table `users`

--

ALTER TABLE `users`

ADD PRIMARY KEY (`id`);

--

-- AUTO\_INCREMENT for dumped tables

--

--

-- AUTO\_INCREMENT for table `model\_storage`

--

ALTER TABLE `model\_storage`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

--

-- AUTO\_INCREMENT for table `tokenizer\_storage`

--

ALTER TABLE `tokenizer\_storage`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

--

-- AUTO\_INCREMENT for table `users`

--

ALTER TABLE `users`

MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=2;

COMMIT;

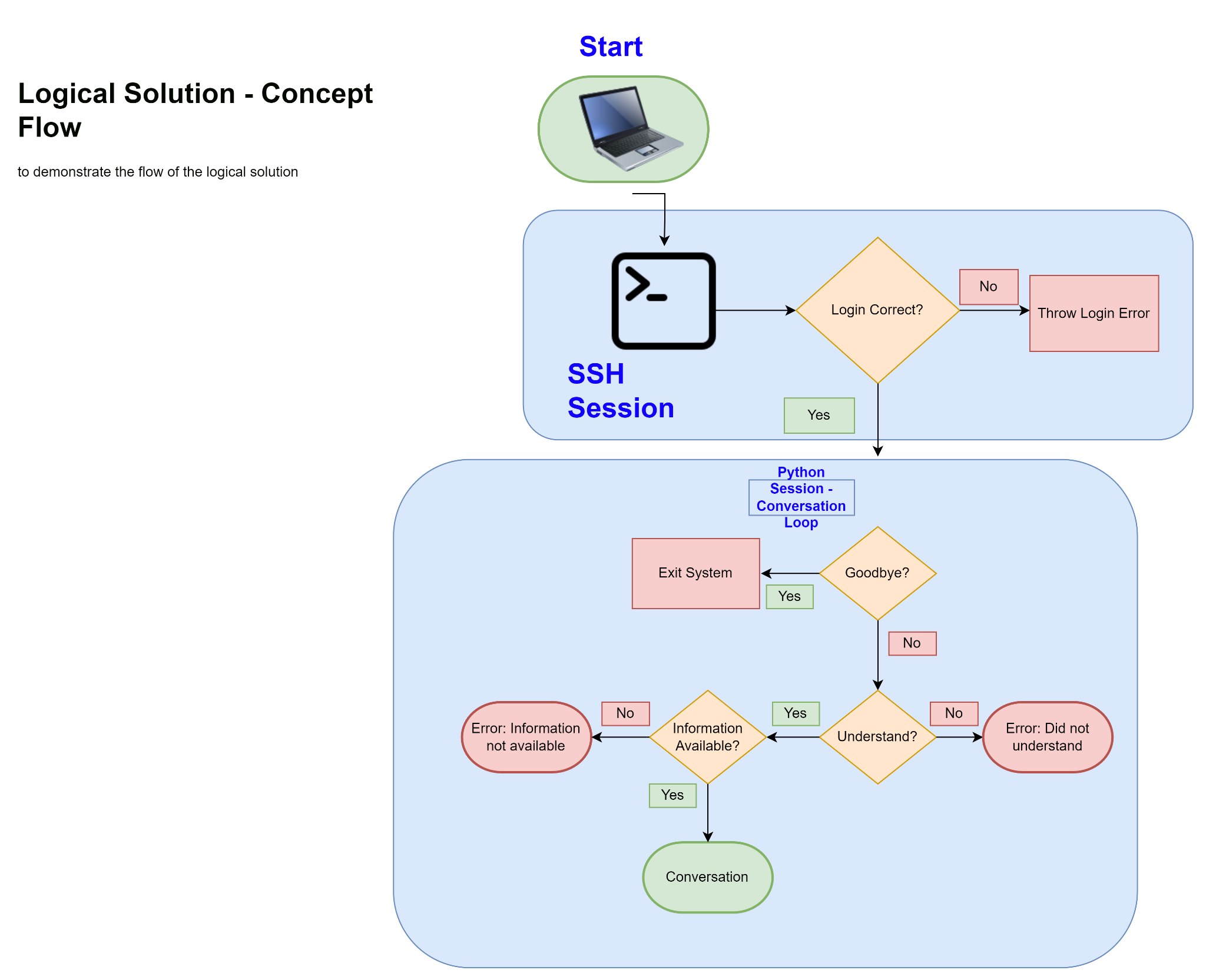
/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

**Flow Charts/Process Flows:**

**Below is the process flow of the logical solution. As depicted in the physical solution, the end-user will use SSH to log in, and connect to the system. After successfully logging in, the AI will automatically wake up and connect to the user. The system locks into a conversation loop. This loop is exited with keywords such as “Goodbye”. While this condition is not met, the user will be asking questions and receiving responses from the system. If the AI is unable to understand what the user is asking, or the information for which the user is asking is not accessible to the AI, an error will be thrown respectively and the prompt will be handed back to the user. Otherwise, a response will be delivered and the conversation will continue.**



**Sitemap Diagram:**

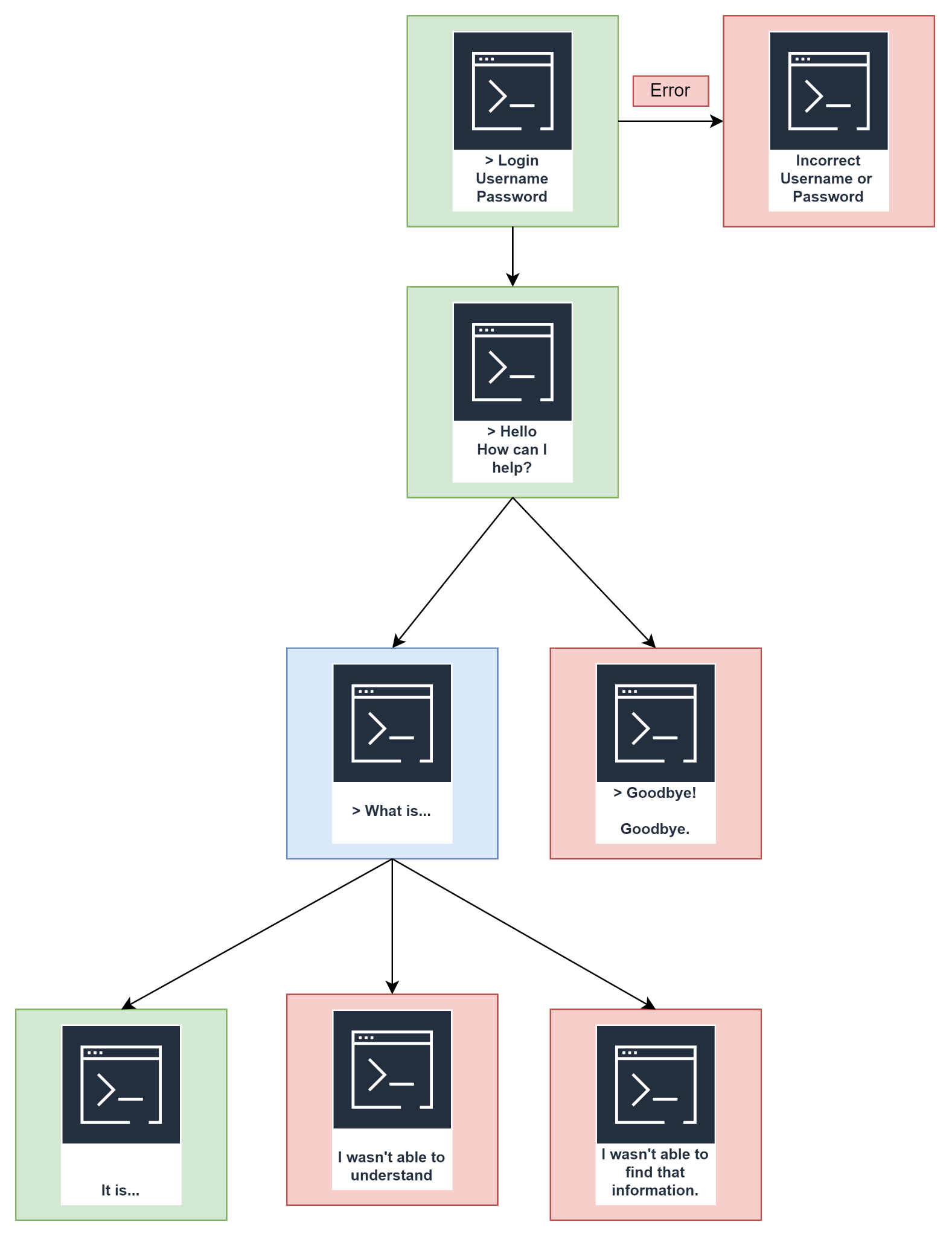
**As depicted in the logical process flow, the sitemap consists of logging into the system, exiting the system, and conversation with the system. The only two errors that will be generated by the system during conversation is an error for not understanding an input, and an error for not being able to access data for a given question. If these errors are not generated, and the AI both understands the input, and has access to the data, a response will be selected and handed with this information and handed to the user.**

Diagram

Description automatically generated

**User Interface Diagrams:**

**Wireframe**



**UML Diagrams:**

N/A.

**Service API Design:**

<https://openweathermap.org/api/one-call-3>

https://platform.openai.com/overview

**NFR’s (Security Design, etc.):**

Extensibility:

Intents will be used to select responses with data injected into them via an API. The combination of this intent system and injection from external API’s and the utilization of the TensorFlow AI framework, provides the modularity and extensibility required to offer the ability to both train the AI model on different fields in the future, and even migrate the system onto another framework such as GPT-3. Measuring extensibility is simple. The feasibility of integrating intelligence in other fields will be the metric for the system’s extensibility. For example, if I want to train the system on intelligence in another field such as mathematics, the feasibility of introducing this field into the system with datasets and corpus files is both measured and decided by the extensibility and modularity of the system itself.

**Operational Support Design:**

As this system only exists in one place on an embedded system, monitoring is not applicable. While there is also currently no intent on implementing logging into this system, thought has been given on the purpose logging would serve, and how integration would be performed. Logging may be integrated into the back-end system; processing end-user input, training and processing corpus files, and interacting with data from external API’s and databases. logging these processes may aid the developer in debugging the system and investigating any errors or discrepancies that may arrive during operation.

**Other Documentation:**

N/A. Risk tasks have been approached and completed. Technology outlining has been performed in order to develop multiple technical solutions. The foundational understanding of Natural Language Processing required to approach this project has been acquired through classes, programs, and proofs of concept.

**Appendix A – Technical Issue and Risk Log**

1. Use the template to identify and monitor project issues and risks.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Issues and Risk Log | | | | | | | | |
| **Issue or Risk** | **Description** | **Project Impact** | **Action Plan/Resolution** | **Owner** | **Importance** | **Date Entered** | **Date to Review** | **Date Resolved** |
| I/R | What is the issue or risk? | How will this impact scope, schedule, and cost? | How do you intend to deal with this issue? | Who manages this issue? |  |  |  |  |
|  | N/A | N/A | All risks have been directly addressed with proof of concepts, and external programs and classes. | Chris King | N/A | 09/23/2022 | 09/25/2022 | 11/20/2022 |
|  |  |  |  |  |  |  |  |  |

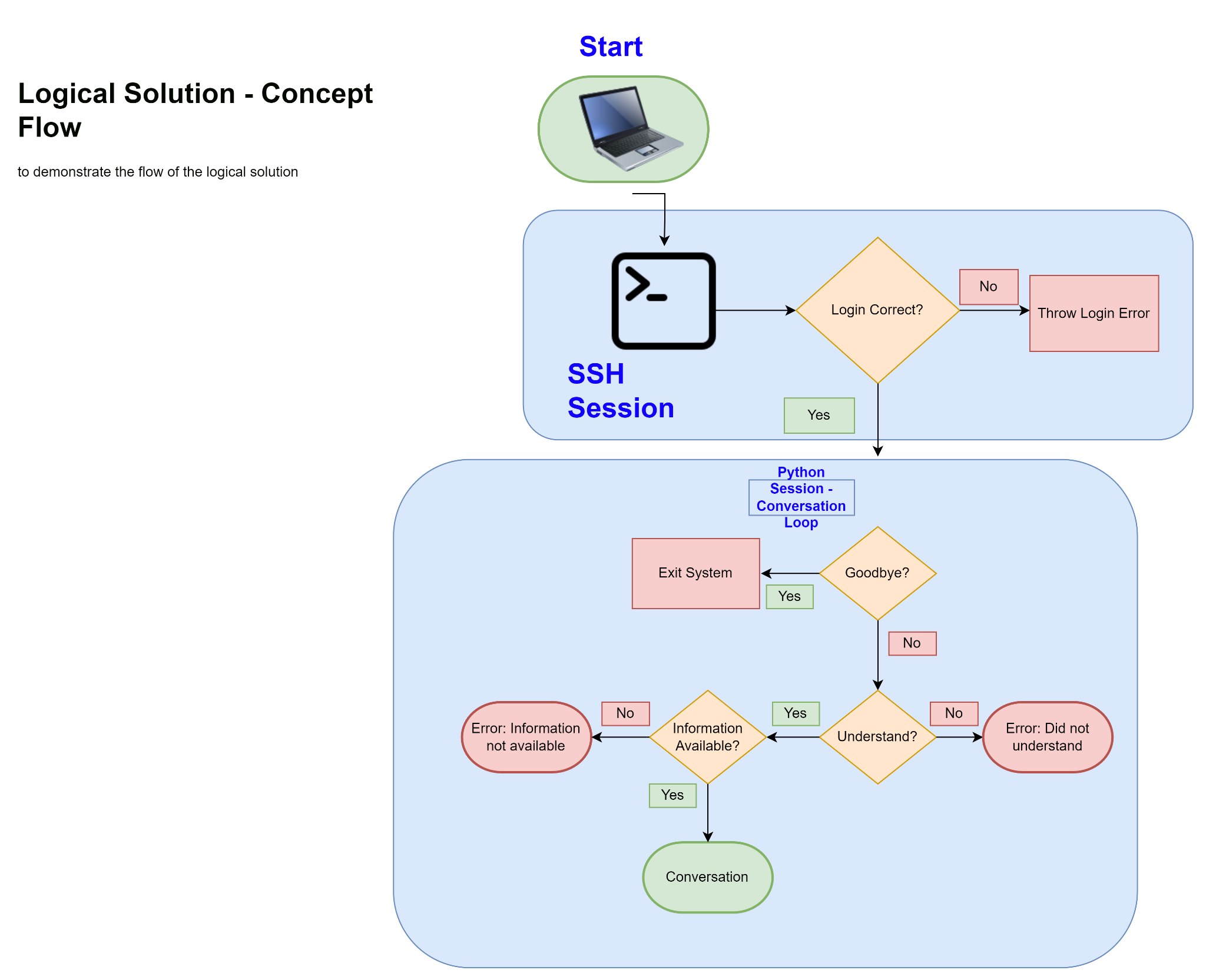
**Appendix B – References**

*Appendix B - Physical Architecture*

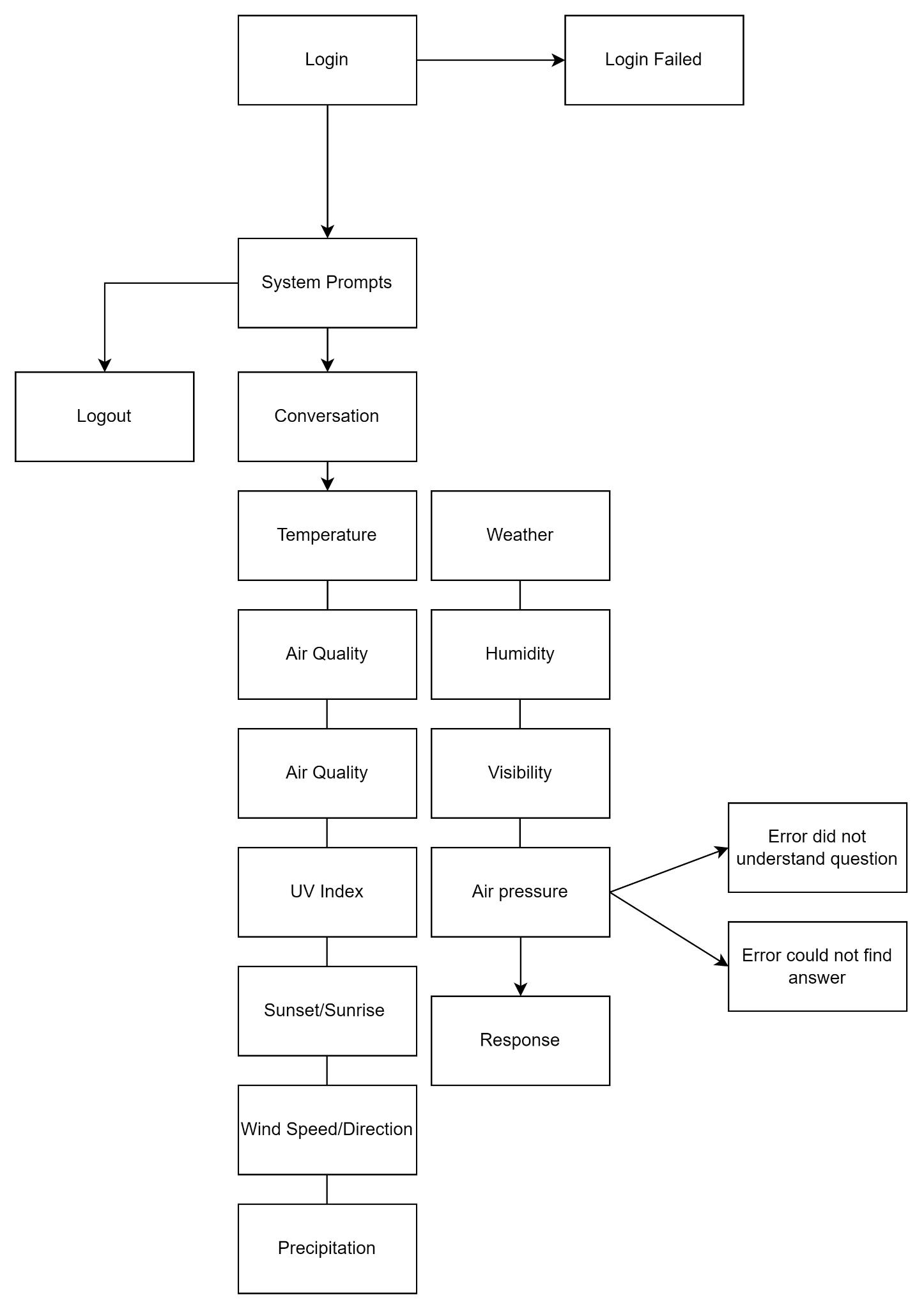
Graphical user interface

Description automatically generated

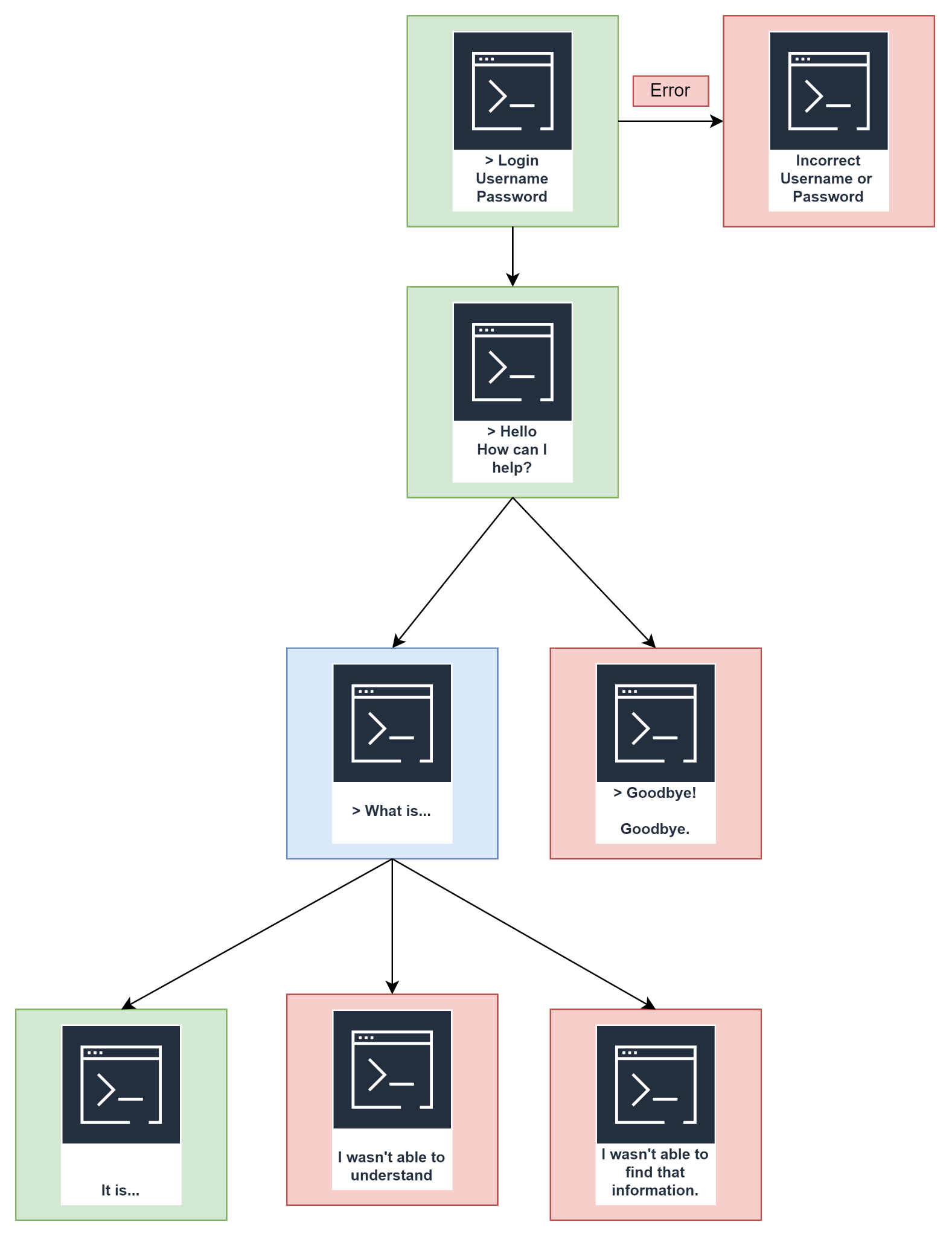
*Appendix B - Logical Process Flow*



Appendix B - Sitemap Diagram



Appendix B - Wireframe Diagram



**Appendix C – External Resources**

|  |  |
| --- | --- |
| **GIT URL:** | *https://github.com/Sasqe/TARS* |
| **Hosting URL:** | *Not Yet Applicable* |