PROBLEM STATEMENT :

Developing an book recommender (a book that the reader should read and is new) Expert system or (any other).

OBJECTIVE :

• To develop problem solving abilities for smart devices.
• To develop problem solving abilities for gamifications.
• To develop problem solving abilities of pervasiveness,embedded security and NLP.
• To study algorithmic examples in distributed, concurrent and parallel environments

THEORY :

PROLOG :

Prolog is a general purpose logic programming language associated with artificial intelligence and computational linguistics. Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is declarative: the program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a *query* over these relations.

There are only three basic constructs in Prolog: facts, rules, and queries. A collection of facts and rules is called a knowledge base (or a database) and Prolog programming is all about writing knowledge bases. That is, Prolog programs simply are knowledge bases, collections of facts and rules which describe some collection of relationships that we find interesting. So how do we use a Prolog program? By posing queries. That is, by asking questions about the information stored in the knowledge base.

There are four kinds of term in Prolog: atoms, numbers, variables, and complex terms (or structures). Atoms and numbers are lumped together under the heading constants, and constants and variables together make up the simple terms of Prolog.

Atoms

An atom is either:

1. A string of characters made up of upper-case letters, lower-case letters, digits, and the underscore character, that begins with a lower-case letter. Here are some examples: butch , big_kahuna_burger , listens2Music and playsAirGuitar .
2. An arbitrary sequence of characters enclosed in single quotes. For example ’ Vincent ’, ’ The Gimp ’, ’ Five_Dollar_Shake ’, ’ &^%&#@$ &* ’, and ’ ’. The sequence of characters between the single quotes is called the atom name. Note that we are allowed to use spaces in such atoms; in fact, a common reason for using single quotes is so we can do precisely that.
3. A string of special characters. Here are some examples: @= and ====> and ; and :- are all atoms. As we have seen, some of these atoms, such as ; and :- have a pre-

defined meaning.

Numbers

Real numbers aren't particularly important in typical Prolog applications. So although most Prolog implementations do support floating point numbers or floats (that is, representations of real numbers such as 1657.3087 or π)

Variables

A variable is a string of upper-case letters, lower-case letters, digits and underscore characters that starts either with an upper-case letter or with an underscore. For example, X , Y , Variable , _tag , X_526 , List , List24 , _head , Tail , _input and Output are all Prolog variables.

The variable _ (that is, a single underscore character) is rather special. It's called the anonymous variable .

Complex terms

Constants, numbers, and variables are the building blocks: now we need to know how to fit them together to make complex terms. Recall that complex terms are often called structures.

Complex terms are build out of a functor followed by a sequence of arguments. The arguments are put in ordinary parentheses, separated by commas, and placed after the functor. Note that the functor has to be directly followed by the parenthesis; you can't have a space between the functor and the parenthesis enclosing the arguments. The functor must be an atom. That is, variables cannot be used as functors. On the other hand, arguments can be any kind of term.

CONCEPT OF UNIFICATION :

PROLOG uses the concept of unification internally . Two terms unify if they are the same term or if they contain variables that can be uniformly instantiated with terms in such a way that the resulting terms are equal.

This means, for example, that the terms mia and mia unify, because they are the same atom. Similarly, the terms 42 and 42 unify, because they are the same number, the terms X and X unify, because they are the same variable, and the terms woman(mia) and woman(mia) unify, because they are the same complex term. The terms woman(mia) and woman(vincent) , however, do not unify, as they are not the same (and neither of them contains a variable that could be instantiated to make them the same).

Now, what about the terms mia and X ? They are not the same. However, the variable X can be instantiated to mia which makes them equal. So, by the second part of our working

definition, mia and X unify. Similarly, the terms woman(X) and woman(mia) unify, because they can be made equal by instantiating X to mia . How about loves(vincent,X) and loves(X,mia) ? No. It is impossible to find an instantiation of X that makes the two terms equal. Do you see why? Instantiating X to vincent would give us the terms loves(vincent,vincent) and loves(vincent,mia) , which are obviously not equal. However, instantiating X to mia, would yield the terms loves(vincent,mia) and loves(mia,mia) , which aren't equal either.

FIRST ORDER LOGIC :

FOL assumes that the world contains Objects , Relations and Functions .The basic syntactic elements of first-order logic are the symbols that stand for objects, relations, and functions. The symbols, therefore, come in three kinds: constant symbols, which stand for objects; predicate symbols, which stand for relations; and function symbols, which stand for functions.

Syntax of First-order Logic: Basic Elements

Symbols

Constants	$KingJohn,\ 2,\ Koblenz,\ C, \ldots$
Predicates	$Brother,\ >,\ =,\ \ldots$
Functions	$Sqrt,\ LeftLegOf,\ \ldots$
Variables	$x,\ y,\ a,\ b,\ \ldots$
Connectives	$\wedge \quad \vee \quad \neg \quad \Rightarrow \quad \Leftrightarrow$
Quantifiers	$\forall \quad \exists$

Terms :

A term is a logical expression that refers to an object. Constant symbols are therefore terms, but it is not always convenient to have a distinct symbol to name every object. For example, in English we might use the expression "King John's left leg" rather than giving a name to his leg. This is what function symbols are for: instead of using a constant symbol, we use **LeftLeg(John).** In the general case, a complex term is formed by a function symbol followed by a parenthesized list of terms as arguments to the function symbol.

Atomic sentences :

Now that we have both terms for referring to objects and predicate symbols for referring to relations, we can put them together to make atomic sentences that state facts. An atomic sentence is formed from a predicate symbol followed by a parenthesized list of terms:

Brother(Richard, John).

This states, under the intended interpretation given earlier, that Richard the Lionheart is the brother of King John .

Atomic sentences can have complex terms as arguments.

Married (Father(Richard), Mother(John))

Complex sentences :

We can use logical connectives to construct more complex sentences, just as in propositional calculus. The semantics of sentences formed with logical connectives is identical to that in the propositional case.

¬ Brother(LeftLeg(Richard), John)

Brother(Richard, John) Λ Brother(John, Richard)

Quantifiers :

Once we have a logic that allows objects, it is only natural to want to express properties of entire collections of objects, instead of enumerating the objects by name. Quantifiers let us do this. First-order logic contains two standard quantifiers, called universal and existential.

A x King (x) Λ Person (x)

з x Crown(x) → OnHead(x, John) .

ALGORITHM :

1. Define all the clauses and predicates to be used .
2. Create a customized Expert window .
3. Start a interactive session . Ask questions to the user .
4. Based on User's answer's make ask more appropriate questions .
5. Continue step 4 until we get a complete idea of user's choice .
6. Recommend appropriate book .

INPUT :

User's answers in the form of choices based on questions asked .

EXPECTED OUTPUT :

Appropriate recommendation of book .

MATHEMATICAL MODEL :

Let P be the solution perspective .

P ={ S , E , I , O, F }

S = { Initial state of the expert system consisting of a intercative session }

I = Input of the system → { I1 , I2 }
where I1 = { User's answers in the form of choices based on questions asked }
I2 = { Questions based on users choice repeatedly }

O = Output of the system → { O1 , O2 }
where O1 = { Appropriate response question }
O2 = { Appropriate recommended Book }

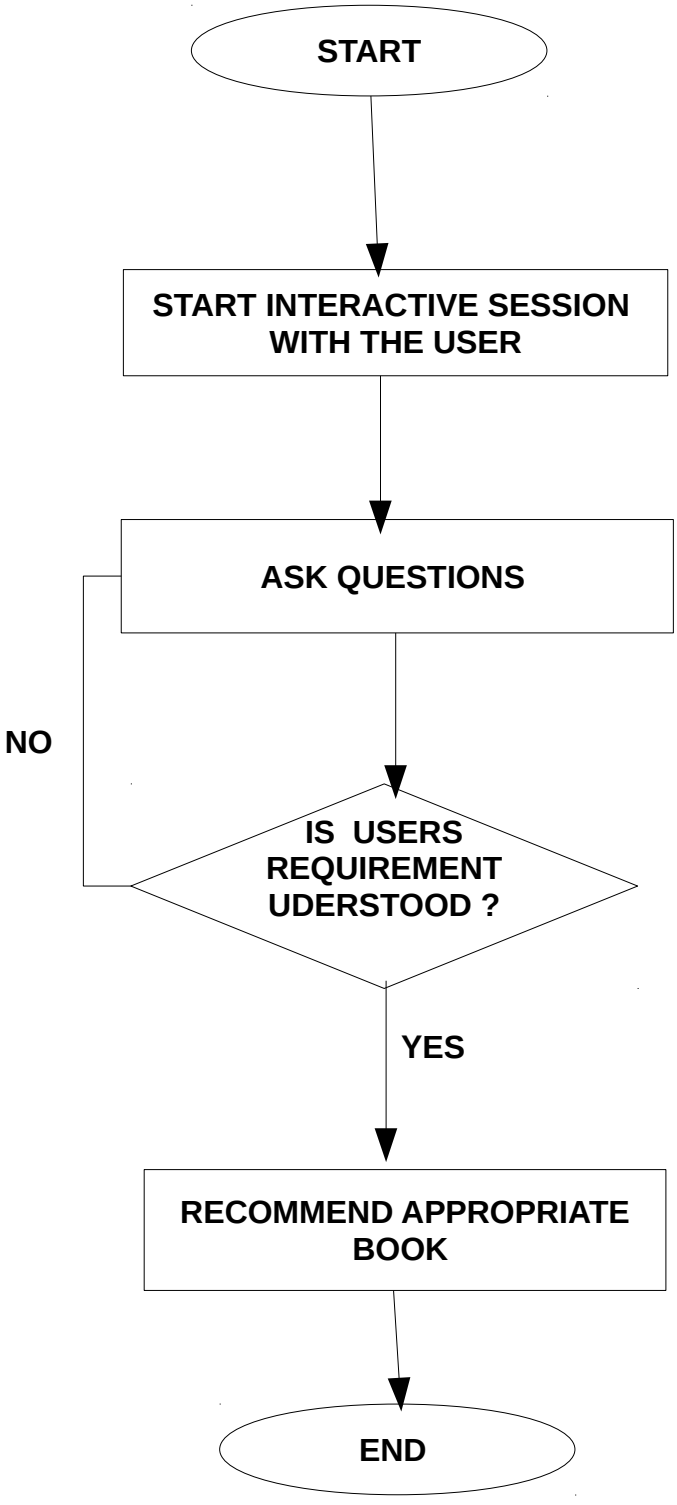
F = Functions used → { f1 , f2 , f3 }
where f1 = { readint for getting choice }
f2 = {write for writing to screen }
f3 = { predicate() for selecting appropriate question }

E = End state of the system which shows the proper recommendationn of book according to users need .

TEST CASES :

TEST CASE	INPUT	EXPECTED OUTPUT	OUTPUT ACHIEVED	REMARKS
1	CHILD	CHILDISH BOOKS	COMICS, FAIRYTALES	Correct
2	YOUTH	LATEST TREND BOOKS	NOVELS, AUTOBIOGRAPHY	Correct
3	OLD AGE	SPIRITUAL BOOKS	MYTHOLOGY, RELIGION HYMNS	Correct

FLOWCHART :



CONCLUSION :

Hence we have successfully implemented a smart expert Book Recommender system using PROLOG.

OUTCOMES ACHIEVED :

COURSE OUTCOME	**ACHIEVED(√)**
Problem solving abilities for smart devices.	√
Problem solving abilities for gamifications.	
Problem solving abilities of pervasiveness,embedded security and NLP.	
To solve problems for multicore or distributed,concurrent/Parallel environments	√