



INSE 6250 - Quality Methodologies for Software

ONLINE BLOOD BANK MANAGEMENT SYSTEM

PROJECT REPORT

Submitted To:

Prof. Jamal Bentahar

Submitted By:

Name	Student ID
Mohammed Faizan	40131626

TABLE OF CONTENTS

CHAPTER	PAGE
ABSTRACT.....	i
CHAPTERS	
CHAPTER 1 – Introduction.....	1
CHAPTER 2 – Model Checking.....	1
CHAPTER 3 – Implementation Model.....	3
CHAPTER 4 – Verification.....	6
CHAPTER 5 – Results.....	7
CHAPTER 6 – Discussion.....	7
CHAPTER 7 – Conclusion.....	7
CHAPTER 8 – Future Work.....	7
REFERENCES.....	7
APPENDICES	
APPENDIX A – Project Code.....	8
APPENDIX B – Execution & Counter Examples	11

ABSTRACT

With the advent of technology, there have been drastic advancements in the field of healthcare. While a doctor's skills and diagnosis are important in medicine, with modeling and verification, quality and correctness of the system play a crucial role. The donation of blood can save people, especially those who are cancer patients or people who have met with an accident. Therefore, there is a need for a system to monitor the page navigation system's correctness which in turn can make sure more people get the blood transfusions they need.

This paper proposes a model checking and verification approach for an Online Blood Bank Management System. The Blood Bank has been modeled in the form of the page navigation system and atomic proposition transitions. To represent the behavior of the system, we use the Kripke structure and to simulate the sequential logic and computation of the model we use Finite State Machine (FSM). Further to identify the correctness of the model, we use formal language specifications such as Linear Temporal Logic (LTL) and Computation Tree Logic (CTL). The complete automation for checking and verification of the model can be done using the NuSMV (New Symbolic Model Verifier) tool.

Keywords: Online Blood Bank Management System, Kripke Structure, Linear Temporal Logic, Computation Tree Logic, NuSMV.

1. INTRODUCTION

With the advent of technology, there have been drastic advancements in the field of healthcare. The data plays an important role in this advancement. The donation of blood can save people, especially those who are cancer patients or people who have met with an accident. Thus the accuracy of the webpage information is of utmost importance and plays a significant role.

In computer science, there are several ways to verify and test models concerning for to cases and specifications. Model Checking is a formal method of verifying system model properties. Several tools offer model checking such as NuSMV, SPIN, Uppaal, PRISM to mention a few. This paper aims to verify the possible properties of a web-based blood bank management system using the NuSMV tool.

1.1 Objectives

- To design a model that automates operations of an Online Blood Bank Management System.
- Verify model specifications of the system and identify areas of improvement using counterexamples.

1.2 Project Approach

- The project is modelled in the form of navigation of a blood bank website and each state represents a particular page or functionality.
- Properties are defined using Linear Temporal Logic (LTL) and Computational Tree Logic (CTL).

2. MODEL CHECKING

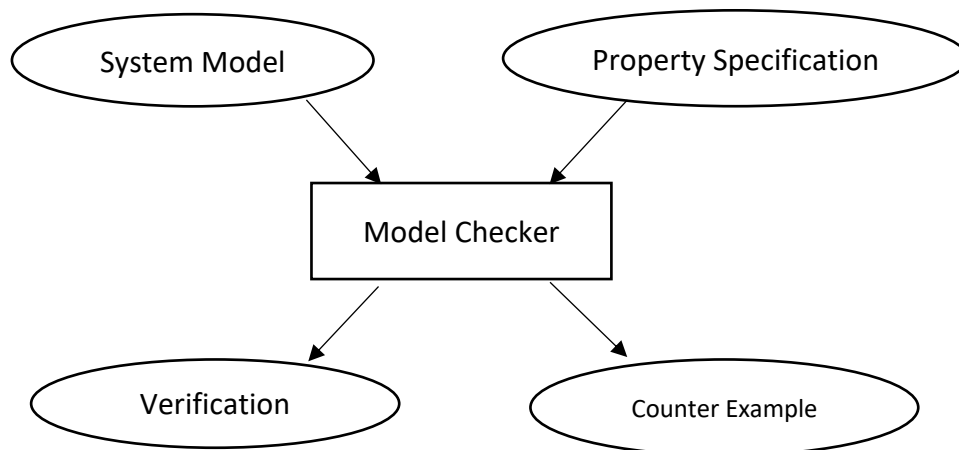


Fig. 1. General Structure of a Model Checking [3].

Model Checking is a formal technique for verifying a system model and properties. To prove the system satisfies intended properties, we use formal methods. The test cases cannot be used extensively to validate software. Thus we require the use of formal methods. The specification for software or hardware is a definition used to specify desired qualities and requirements. Model Checking allows us to identify the rightness of the developed system. It uses a brute force approach to traverse through different states of a model systematically.

Model Checking is automated, which means no expert is required to monitor and check the correctness of the property specified.

A Kripke structure can be used to represent a finite state system [2]. It is defined as follows:

$K = (S, I, R)$ Label

- K : a tuple
- S : a countable set of states
- $I \subseteq S$: a set of initial states
- $R \subseteq s \times s$: a transition relation satisfying $\forall s \in S. (\exists s' \in S. (s, s') \in R)$
- Label: $S \rightarrow 2^{AP}$: an interpretation. Function

2.1 Steps of Model Checking

1. A model specification language for the system model to be verified. Denoted by 'M'.
2. A property or requirement specification 'Φ'.
3. A verification method or logic (If $M \models \Phi$ is satisfied or not satisfied).

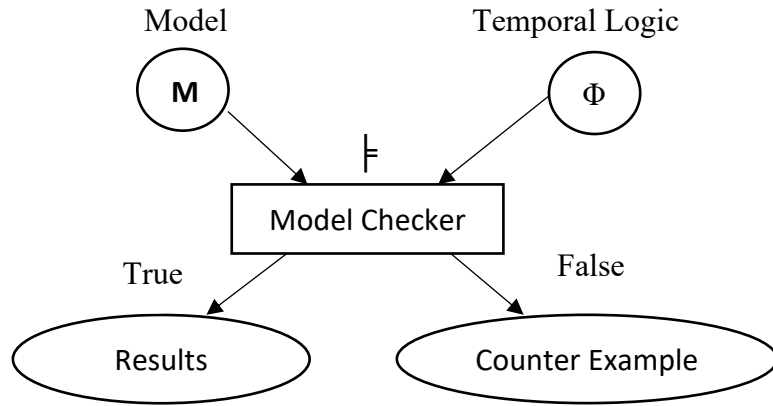


Fig. 2. The Flow of a Model Checking [3].

2.2 System Model

The system model is represented by Finite State Machines, Buchi Automata and Kripke Structure and other models. To model the behaviour of the system we used FSM, which consists of three parts; states, transitions, and actions. The Buchi Automata is an FSM algorithm rather than accepts an infinite number of words.

2.3 Property Specification

The properties to be checked for the system model is written in propositional Linear Temporal Logic (LTL) and Computational Tree Logic (CTL). Linear Temporal Logic as the name itself suggests that it is a linear-time temporal logic that describes a set of an infinite sequence of states and it has the following syntax:

$$\Phi ::= p \mid \neg\Phi \mid \Phi \vee \Phi \mid X\Phi \mid \Phi U \Phi$$

Computational Tree Logic as the name itself suggests that it is a branching-time logic that follows a tree-like structure. The CTL formulas as defined by Backus-Naur form has the following syntax:

$\phi ::= T \mid \perp \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \phi \vee \phi \mid \phi \rightarrow \phi \mid AX\phi \mid EX\phi \mid AF\phi \mid EF\phi \mid AG\phi \mid EG\phi \mid A[\phi U \phi] \mid E[\phi U \phi]$, where

- T and \perp is True and False, and p is a CTL formula.

- If φ, ϕ is CTL formula, then $\neg\varphi, \varphi \wedge \phi, \varphi \vee \phi, \varphi \rightarrow \phi, AX\varphi, EX\varphi, AF\varphi, EF\varphi, AG\varphi, EG\varphi, A[\varphi U \phi]$ and $E[\varphi U \phi]$ are CTL formulas.

3. IMPLEMENTATION MODEL

To identify the correctness of the system model, we specify some specifications that the system model must follow or satisfy. The model defined in this paper follows a Kripke structure to model an Online Blood Bank Management System. Each state represents a particular navigation step in the workflow of the online system.

The Online Blood Bank Management System webpage here is primarily used by the receptionist or hospital officials to keep and track important information on blood availability and donors and patients. To be able to navigate and have communication between the various functions of the system, we use four atomic propositions namely.

- *page*-Page: represents the specific webpage navigation step of the online system model.
- *link*-Link: represents a hyperlink, navigation, and connectivity between webpages. Based on these links, further actions will be taken by the system model.
- *call*-Link: describes the action to be performed upon linking by the page.
- *build*-Link: describes the response for the action specified.

The Hospital List/Hub is the initial state which is a commonplace hospital listing. Upon clicking one of the hospital lists, it is linked to the Homepage of that particular hospital. The further steps of the flowchart depict an online representation of the webpage navigation of the blood bank system. The user (receptionist or hospital official) can also directly exit from the system without performing any task or function as shown in the Fig. 3 (a).

The equivalent state model representation of the flowchart navigation of the Online Blood Bank Management System is shown in the Fig. 3 (b) below, where,

- S0: Hospital List/Hub; a list or database of hospitals.
- S1: Homepage; welcome page for the specific hospital.
- S2: Login; to be done by the user (administrator or health official of that hospital).
- S3: Authentication; to check to the validity of the login credentials entered.
- S4: Blood Bank System; if the login credentials are correct then the main page of the system where the user has the option of choosing two functions (to ways to navigate).
- S5: Login Failed; if the login credentials are incorrect, it prompts the user to input correct details.
- S6: Blood Group; this section specifies possible blood group related functionalities such as identifying the availability of the blood, type of blood, stockpile, etc.
- S7: Fetch Details; gets the blood group list upon user request.
- S8: Blood Group List; displays the details of blood group list to the user which has all the details related to donors/patients, blood type, blood availability, etc.
- S9: Donate Blood; means the user takes the blood from the donor or patient and proceeds further.
- S10: Donor/Patient details; the donor name, age, and other relevant information is stored by the user.
- S11: Laboratory Tests; the user then sends the blood sample for lab tests and stores in hospital records. Also, this is linked to the Blood Group (state s6) as it needs access to the details of the donor and tests.
- S12: Exit; exits the webpage either performing needed tasks or without performing any task.

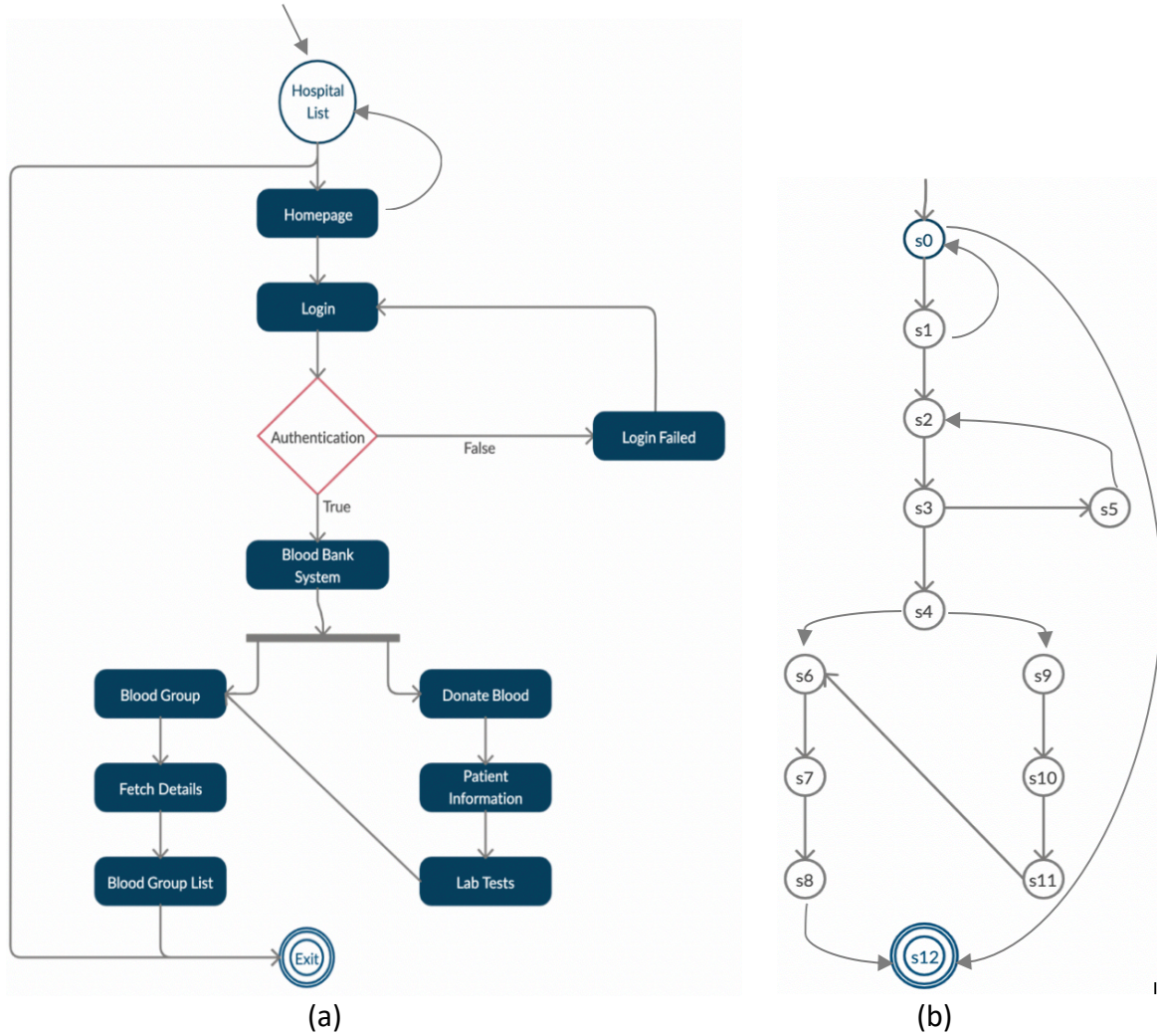


Fig. 3 (a). Flowchart representation. (b) State Model representation of Online Blood Bank Management System.

3.1 Property Specification

The property specification or requirement specification as the name itself suggests that it specifies properties for the system functionalities. The model checker uses the model in Fig. 3&4, along with the properties to verify the correctness of the system.

For the property specification, we specify Reachability and Liveness property for Computational Tree Logic, and some properties for Linear Temporal Logic as well.

1. Reachability (CTL)- the Reachability property means that all states in the blood bank system must be covered or reached at least once. Thus we utilize EF operator for defining reachability property in CTL as listed in the Table. 1.

Property No.	State No.	State Name	Property
Valid Specifications:			
1	s0	HospitalsHub	EF page-HospitalsHub
2	s1	Homepage	EF page-Homepage
3	s2	LoginPage	EF page-LoginPage
4	s3	Authentication	EF page-Authentication
5	s4	BloodBankSystem	EF page-BloodBankSystem

6	s5	LoginFailed	EF page-LoginFailed
7	s6	BloodGroup	EF page-BloodGroup
8	s7	FetchBloodGroupList	EF page-FetchBloodGroupList
9	s8	BloodGroupList	EF page-BloodGroupList
10	s9	DonateBlood	EF page-DonateBlood
11	s10	PatientDetails	EF page-PatientDetails
12	s11	LaboratoryTest	EF page-LaboratoryTest
13	s12	Exit	EF page-Exit
Some Invalid Specifications with Counter Examples:			
14	s3	Authentication	AG (page-Authentication -> AX page-BloodBankSystem)
15	s4	BloodBankSystem	AG (page-BloodBankSystem -> AX page-BloodGroup)
16	s9	DonateBlood	AG (page-DonateBlood -> AX page-LaboratoryTests)

Table 1. Reachability properties for Online Blood Bank Management System.

- The property numbers 1 to 13 means-
EF page- ϕ – “It is possible to get a state where ϕ is True”
 - The property number-14 is executed and results are attached in Appendix section- B.
AG (page-Authentication -> AX page-BloodBankSystem) – “In all states, it is true that if page-Authentication holds in a state, then the next state in all paths from that state is page-BloodBankSystem”
2. Liveness (CTL)- The Liveness property means all transitions (legal transitions) in the system model must be covered at least once. For this purpose, we use AG, EX, and EF operators of CTL to determine legal transitions in the system as shown in Table. 2.

Property Number	State Name	Property
Valid Specifications:		
17	HospitalsHub	AG EF(page-HospitalsHub -> page-Exit)
18	Homepage	AG EF(page-Homepage -> page-LoginPage)
19	LoginPage and Authentication	AG EF(page-LoginPage -> (call-Authentication & EX page-Authentication))
20	BloodBankSystem and Blood Group	AG EF(page-BloodBankSystem -> (link-BloodGroup & EX page-BloodGroup))
21	Donate Blood	AG EF(page-DonateBlood -> page-PatientDetails)
22	FetchBloodGroupList	AG EF(page-FetchBloodGroupList -> page-BloodGroupList)
23	PatientDetails	EF page-PatientDetails
Some Invalid Specifications with Counter Examples:		
24	BloodBankSystem and BloodGroup	AG(page-BloodBankSystem -> (link-BloodGroup & EX page-BloodGroupList))
25	DonateBlood & PatientDetails	AG(page-BloodBankSystem -> (link-BloodGroup & EX page-PatientDetails))

Table 2. Liveness properties for Online Blood Bank Management System.

3. LTL Specifications- we can also see that a CTL property can be converted to a correct LTL property specification by omitting operators E and A, as shown in the Table. 3. Also, we have utilized various operators in writing the LTL formula, and they are mentioned as follows:

- $\neg \text{ltl_expr}$ – logical NOT
- $\text{ltl_expr} \ \& \ \text{ltl_expr}$ – logical AND
- $\text{ltl_expr} \ | \ \text{ltl_expr}$ – logical OR
- $\text{ltl_expr} \rightarrow \text{ltl_expr}$ – logical implies
- $\text{ltl_expr} \leftrightarrow \text{ltl_expr}$ – logical equivalence
- $X \text{ ltl_expr}$ – next state
- $G \text{ ltl_expr}$ - globally
- $F \text{ ltl_expr}$ - finally
- $\text{ltl_expr} \ U \ \text{ltl_expr}$ - until

Property Number	State Name	Property
Valid Specifications:		
26	HospitalsHub	(page-Homepage page-HospitalsHub)
27	FetchBloodGroupList	$\neg G$ (page-FetchBloodGroupList)
28	Homepage	F (page-Homepage)
29	LoginPage	$G F(\text{page-LoginPage} \rightarrow X \text{page-Authentication})$
30	BloodGroupList	(page-BloodGroupList)
31	Authentication	$G F(\text{page-LoginPage} \rightarrow (\text{call-Authentication} \ \& \ X \text{page-Authentication}))$
32	Authentication	(page-Authentication \rightarrow page-PatientDetails)
33	DonateBlood	$G F(\text{page-DonateBlood} \rightarrow \text{page-PatientDetails})$
34	Exit	page-Authentication \rightarrow F page-Exit
Some Invalid Specifications with Counter Examples:		
36	HospitalsHub	(page-Homepage \leftrightarrow page-HospitalsHub)
37	Homepage	G (page-Homepage)
38	LoginPage	(page-Homepage \rightarrow page-LoginPage)
39	DonateBlood	$F(\text{page-DonateBlood} \ U \ \text{page-PatientDetails})$
40	Exit	(page-Exit)

Table 3. LTL properties for Online Blood Bank Management System.

4. VERIFICATION

To verify the system model, we use the NuSMV (New Symbolic Model Verifier) tool. NuSMV is a model checker tool based on the SMV symbolic model checker [4]. With the help of the NuSMV tool, we can represent synchronous and asynchronous finite state machines using BDD-based techniques.

After building the model and specifying the system properties for the model, the NuSMV model checker takes both of them as inputs, and using an algorithm (written in SMV code) we verify the correctness of the model.

4.1 Steps for Verification

After installation of the NuSMV model checker and other system specific requirements (BDD-based, SAT-based, etc), we write the SMV code specifying states and transitions of the

Online Blood Bank Management System we specify the logic behind the code and run the following commands:

1. Open the terminal, go to NuSMV folder, navigate to source code file, and run
‘terminal>path> NuSMV -i’
2. NuSMV > read_model -i *filename* (bloodbank.smv)
3. NuSMV > flatten_hierarchy
4. NuSMV > encode_variables
5. NuSMV > build_model
6. NuSMV > check_ctlspec “*property*” (to check the CTL specifications of Table 1&2)
7. NuSMV > check_ltlspec “*property*” (to check the CTL specifications of Table 3 only)

Optional,

8. NuSMV > show_vars (to print all states of the model)
9. NuSMV > print_bdd_stats (to print BDD related details of the model)

5. RESULTS AND DISCUSSIONS

The model checker takes the system model and properties to be verified, if the property is correct then the property is satisfied. If the input or property is incorrect, then a counterexample will be generated showing how and which state the property fails to satisfy. The results of some important property execution if satisfied and also counterexamples if not satisfied is shown as snapshots in the Appendix Section.

6. CONCLUSION

The project was aimed at modeling and verifying an Online Blood Bank Management System using the NuSMV model checker tool. The model represented the general flow or navigation webpage of the system with necessary functionalities. The properties were specified and verified using CTL and LTL formulas. The final execution of the properties hinted that most of the requirements were satisfied, and counterexamples were generated accordingly for the properties that did not satisfy.

7. FUTURE WORK

The model specified in the paper, aimed at representing only necessary functionalities of a typical representation and flow of Online Blood Bank Management System. The model further can be enhanced by including other functionalities and also by specifying several other model-specific properties to have better correctness of the model. The model can also be designed and verified in other model checker tools to get different results and improve efficiency.

REFERENCES

1. Paul and Haque, “Modeling and Verifying e-shopping System”, Concordia University, Montreal, 2013
2. Marc Frappier et al., “Comparison of Model Checking tools for Information Systems”, GRIL, Université de Sherbrooke, Québec, Canada, June 16, 2010
3. Jamal Bentahar. (2020, Winter), Available:
<https://users.ensc.concordia.ca/~bentahar/inse6250.html>
4. NuSMV Tutorials. Available:
http://nusmv.fbk.eu/NuSMV/userman/v21/nusmv_2.html

APPENDICES

APPENDIX – A PROJECT CODE

The logic behind the code is simple, we declare the states according to the flowchart, we use atomic propositions and declare pages as Boolean values, we specify the transitions, then finally we connect the pages using a simple case-esac logic [].

```

MODULE main
VAR
    state: {s0, s1, s2, s3, s4, s5, s6, s7, s8, s9, s10, s11, s12};

page-HospitalsHub: boolean;
page-Homepage: boolean;
page-LoginPage: boolean;
page-Authentication: boolean;
page-LoginFailed: boolean;
page-BloodBankSystem: boolean;
page-BloodGroup: boolean;
page-FetchBloodGroupList: boolean;
page-BloodGroupList: boolean;
page-PatientDetails: boolean;
page-LaboratoryTests: boolean;
page-DonateBlood: boolean;
page-Exit: boolean;
link-HospitalsHub: boolean;
link-LoginPage: boolean;
link-BloodGroup: boolean;
call-Authentication: boolean;
call-DonateBlood: boolean;
call-FetchBloodGroupList: boolean;
build-BloodBankSystem: boolean;
build-LoginFailed: boolean;
build-BloodGroupList: boolean;
build-PatientDetails: boolean;
build-LaboratoryTests: boolean;

ASSIGN
init(state):= s0;
init(page-Homepage):=TRUE;
init(page-LoginPage):=FALSE;
init(page-Authentication):=FALSE;
init(page-HospitalsHub):=FALSE;
init(page-LoginFailed):=FALSE;
init(page-BloodBankSystem):=FALSE;
init(page-BloodGroup):=FALSE;
init(page-FetchBloodGroupList):=FALSE;
init(page-BloodGroupList):=FALSE;
init(page-PatientDetails):=FALSE;
init(page-LaboratoryTests):=FALSE;
init(page-DonateBlood):=FALSE;
init(page-Exit):=FALSE;
init(link-HospitalsHub):=TRUE;
init(link-LoginPage):=TRUE;
init(link-BloodGroup):=FALSE;
init(call-Authentication):=FALSE;
init(call-DonateBlood):=FALSE;

```

```

init(call-FetchBloodGroupList):=FALSE;
init(build-BloodBankSystem):=FALSE;
init(build-LoginFailed):=FALSE;
init(build-BloodGroupList):=FALSE;
init(build-PatientDetails):=FALSE;
init(build-LaboratoryTests):=FALSE;

next(state):=case
    state=s0: {s1,s2};
                state=s1: s12; state=s2: s3; state=s3: {s4,s5}; state=s4: {s6,s9};
                state=s5: s2; state=s6: s7; state=s7: s8; state=s8: s12; state=s9: s10;
                state=s10: s11; state=s11: s6; state=s12: s12;
                TRUE: state;
esac;
next(page-Homepage):=case
    next(state)=s0: TRUE;
    TRUE:FALSE;
esac;
next(page-HospitalsHub):=case
    next(state)=s1: TRUE;
    TRUE:FALSE;
esac;
next(page-LoginPage):=case
    next(state)=s2: TRUE;
    TRUE:FALSE;
esac;
next(page-Authentication):=case
    next(state)= s3:TRUE;
    TRUE:FALSE;
esac;
next(page-LoginFailed):=case
    next(state)=s5:TRUE;
    TRUE:FALSE;
esac;
next(page-BloodBankSystem):=case
    next(state)=s4:TRUE;
    TRUE:FALSE;
esac;
next(page-BloodGroup):=case
    next(state)=s6:TRUE;
    TRUE:FALSE;
esac;
next(page-FetchBloodGroupList):=case
    next(state)=s7:TRUE;
    TRUE:FALSE;
esac;
next(page-BloodGroupList):=case
    next(state)=s8:TRUE;
    TRUE:FALSE;
esac;
next(page-PatientDetails):=case
    next(state)=s10:TRUE;
    TRUE:FALSE;
esac;
next(page-DonateBlood):=case
    next(state)=s9:TRUE;
    TRUE:FALSE;

```

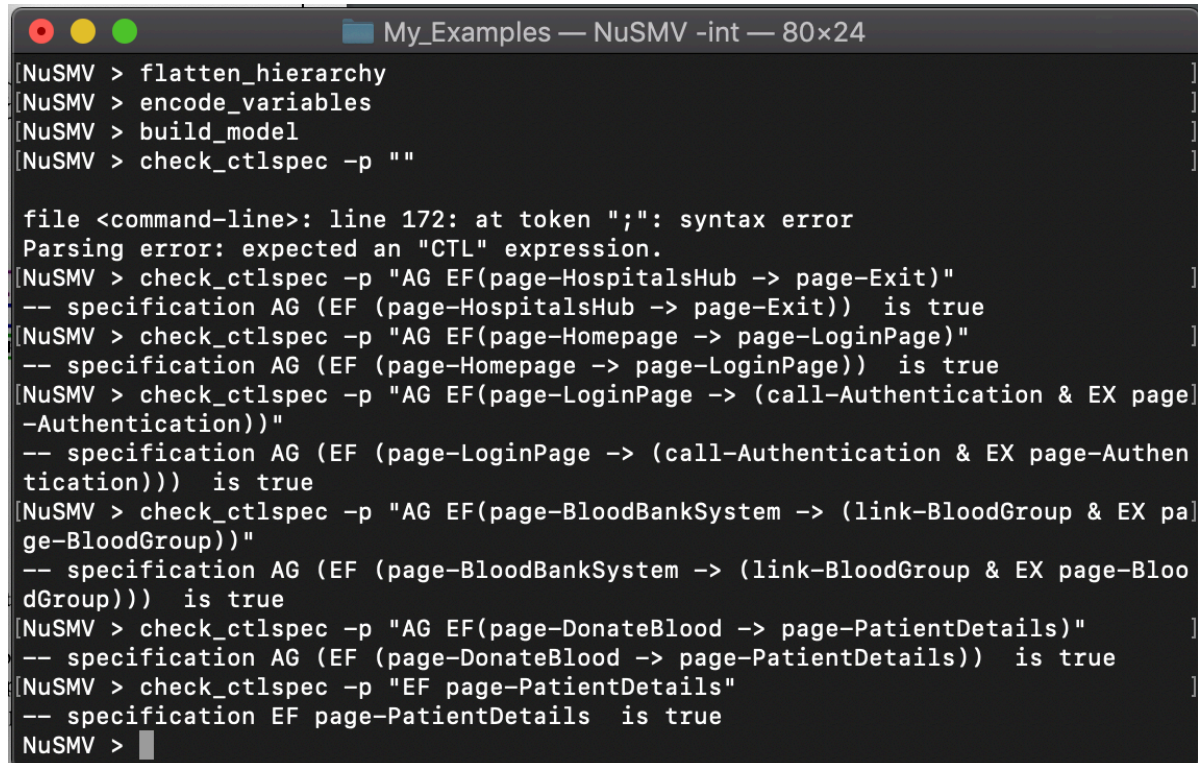
```

esac;
next(page-LaboratoryTests):=case
    next(state)=s11:TRUE;
    TRUE:FALSE;
esac;
next(page-Exit):=case
    next(state)=s12:TRUE;
    TRUE:FALSE;
esac;
next(link-HospitalsHub):=case
    next(state)=s0: TRUE;
    TRUE: FALSE;
esac;
next(link-LoginPage):=case
    next(state)=s0: TRUE;
    TRUE: FALSE;
esac;
next(link-BloodGroup):=case
    next(state)=s4|next(state)=s11: TRUE;
    TRUE: FALSE;
esac;
next(call-Authentication):=case
    next(state)=s2: TRUE;
    TRUE: FALSE;
esac;
next(call-DonateBlood):=case
    next(state)=s4: TRUE;
    TRUE: FALSE;
esac;
next(call-FetchBloodGroupList):=case
    next(state)=s6: TRUE;
    TRUE: FALSE;
esac;
next(build-BloodBankSystem):=case
    next(state)=s3: TRUE;
    TRUE: FALSE;
esac;
next(build-LoginFailed):=case
    next(state)=s3: TRUE;
    TRUE: FALSE;
esac;
next(build-BloodGroupList):=case
    next(state)=s7: TRUE;
    TRUE: FALSE;
esac;
next(build-PatientDetails):=case
    next(state)=s9: TRUE;
    TRUE: FALSE;
esac;
next(build-LaboratoryTests):=case
    next(state)=s10: TRUE;
    TRUE: FALSE;
esac;

```

APPENDIX – B EXECUTION AND COUNTER EXAMPLES

- 1) Execution of all valid specifications falling under Liveness (CTL) property: all transitions in the system model must be covered at least once. For this purpose, we use AG, EX, and EF operators of CTL to satisfy legal transitions in the system.



```

My_Examples — NuSMV -int — 80x24
[NuSMV > flatten_hierarchy
[NuSMV > encode_variables
[NuSMV > build_model
[NuSMV > check_ctlspec -p ""

file <command-line>: line 172: at token ";": syntax error
Parsing error: expected an "CTL" expression.
[NuSMV > check_ctlspec -p "AG EF(page-HospitalsHub -> page-Exit)"
-- specification AG (EF (page-HospitalsHub -> page-Exit)) is true
[NuSMV > check_ctlspec -p "AG EF(page-Homepage -> page-LoginPage)"
-- specification AG (EF (page-Homepage -> page-LoginPage)) is true
[NuSMV > check_ctlspec -p "AG EF(page-LoginPage -> (call-Authentication & EX page-
-Authentication))"
-- specification AG (EF (page-LoginPage -> (call-Authentication & EX page-Authen
tication))) is true
[NuSMV > check_ctlspec -p "AG EF(page-BloodBankSystem -> (link-BloodGroup & EX pa
ge-BloodGroup))"
-- specification AG (EF (page-BloodBankSystem -> (link-BloodGroup & EX page-Bloo
dGroup))) is true
[NuSMV > check_ctlspec -p "AG EF(page-DonateBlood -> page-PatientDetails)"
-- specification AG (EF (page-DonateBlood -> page-PatientDetails)) is true
[NuSMV > check_ctlspec -p "EF page-PatientDetails"
-- specification EF page-PatientDetails is true
NuSMV >

```

- 2) Counter Example of Reachability (CTL) property
Property No. 14; **AG (page-Authentication -> AX page-BloodBankSystem)**
Meaning- “In all states, it is true that if page-Authentication holds in a state, then the next state in all paths from that state is page-BloodBankSystem”
Reason- from Authentication, there are 2 transitions based on condition:
 - BloodBankSystem (if True)
 - LoginFailed (if False)

```

My_Examples — NuSMV -int — 80x52
[NuSMV > check_ctlspec -p "AG (page-Authentication -> AX page-BloodBankSystem)" ]
-- specification AG (page-Authentication -> AX page-BloodBankSystem) is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
-> State: 3.1 <-
  state = s0
  page-HospitalsHub = FALSE
  page-Homepage = TRUE
  page-LoginPage = FALSE
  page-Authentication = FALSE
  page-LoginFailed = FALSE
  page-BloodBankSystem = FALSE
  page-BloodGroup = FALSE
  page-FetchBloodGroupList = FALSE
  page-BloodGroupList = FALSE
  page-PatientDetails = FALSE
  page-LaboratoryTests = FALSE
  page-DonateBlood = FALSE
  page-Exit = FALSE
  link-HospitalsHub = TRUE
  link-LoginPage = TRUE
  link-BloodGroup = FALSE
  call-Authentication = FALSE
  call-DonateBlood = FALSE
  call-FetchBloodGroupList = FALSE
  build-BloodBankSystem = FALSE
  build-LoginFailed = FALSE
  build-BloodGroupList = FALSE
  build-PatientDetails = FALSE
  build-LaboratoryTests = FALSE
-> State: 3.2 <-
  state = s2
  page-Homepage = FALSE
  page-LoginPage = TRUE
  link-HospitalsHub = FALSE
  link-LoginPage = FALSE
  call-Authentication = TRUE
-> State: 3.3 <-
  state = s3
  page-LoginPage = FALSE
  page-Authentication = TRUE
  call-Authentication = FALSE
  build-BloodBankSystem = TRUE
  build-LoginFailed = TRUE
-> State: 3.4 <-
  state = s5
  page-Authentication = FALSE
  page-LoginFailed = TRUE
  build-BloodBankSystem = FALSE
  build-LoginFailed = FALSE
NuSMV >

```

3) Valid Example of an LTL property

Property No. 35; **G F(page-LoginPage -> (call-Authentication & X page-Authentication))**

Reason- It is always the case that eventually from page-LoginPage, if we call Authentication, then the next state will be page-Authentication.

```

[NuSMV > check_ltlspec -p "G F(page-LoginPage -> (call-Authentication & X page-Authen
tication))"
-- specification G ( F (page-LoginPage -> (call-Authentication & X page-Authen
tication))) is true
NuSMV >

```