Formal Modelling using UML, Automata, and TLA+: Smart Hospital Management System

Pratham Nayak, Aprameya Dash, Suyash Chintawar, Biju R. Mohan, Madhusmita Das
Department of Information Technology
National Institute of Technology Karnataka, Surathkal, India 575025
Email: {pratham.191it241, aprameyadash.191it209, suyash.191it109, biju, madhusmitadas.197it004}@nitk.edu.in

***Abstract*—Smart Hospitals are gradually becoming necessities in the present world due to the massive advancements in technology happening all over the world. Such systems not only provide better clinical care to each and every patient but also ensure a smooth and systematic interaction between the doctors and patients. Further, existing hospital management systems sometimes have serious limitations as they are often operated manually without proper testing and can have undetected loopholes, which may pose serious threats later on. These threats are handled in Smart Hospital Management System. In this paper, we propose a Smart Hospital Management System which utilizes RFID tags as the core of the system to ensure seamless communication between different entities. This system has been developed using UML. NFA has been used to understand the working of the system. The formal specification of the system has been done using TLA+ language and then verification is done using TLC model checker to ensure that there are no loopholes present in the system.**

Keywords— Formal Modelling, UML, Automata, TLA+, Smart Hospital Management System

I. Introduction

The advancement of technology and the integration of various applications of technology in day-to-day life is one of the greatest achievements of the 21st century. The integration of technology in various fields has not only increased efficiency by automating a lot of tasks and eliminating manual work but also has enhanced safety by removing errors and ensuring that all systems work perfectly. Integration of technology with hospitals has also led to the creation of Smart Hospital Management Systems [1]. They provide a lot of benefits, due to which these systems are immensely popular. These systems ensure proper and smooth interaction between various entities and provide improved treatment and diagnosis. Further, these systems also provide real-time data access, data protection, seamless communication and lower the need for extra manpower. These systems also provide extra safety, security, and convenience to patients, which helps in enhancing patient satisfaction and experience.

However, a large majority of hospitals are still using traditional and manual methods for functioning, which is quite unfortunate. Most of the systems in existing hospitals are still manual and untested for errors which may cause huge losses upon failure. A Hospital Management System is a Safety Critical System (SCS) [2, 3] as the failure of such a system can result in irreparable damage in the form of loss of life or serious injuries.

In this paper, we propose a Smart Hospital Management System which uses the concept of RFID to ensure the safety of the patients. RFID [4] is a form of wireless communication that uses radio waves to uniquely identify something. For Smart Hospital Management System, RFID tags can be inserted in wristbands, equipped with various other sensors, and then, these wristbands can be used to identify and track patients, monitor the vitals of patients, and send data by periodically communicating with the RFID reader. This real-time data provided by the RFID wristbands of the patients may help doctors to diagnose and treat patients in a much better way. RFID tags also help in uniquely identifying patients.

The Smart Hospital Management System is modelled using UML by visually drawing the different subsystems of the complete system. NFA is then used to describe the working and flow of the different subsystems [5]. Formal specification of the system is done using TLA+ specification language [6, 7], which is widely used during requirement analysis, system analysis, and system design. Finally, the model is verified using TLC model checker.

The paper consists of 5 sections. Section 2 includes the literature survey of some related works. Section 3 contains the methodology, section 4 comprises the results drawn after implementation of the proposed methodology and Section 5 contains the conclusion.

II. Literature Survey

A model of a smart parking system is proposed by the authors in this paper [8]. RFID has been used to serve the purpose. The availability of parking space in the parking lot is monitored with the help of RFID cards. Every vehicle has an RFID card which helps to keep track of the incoming and outgoing vehicles. A Browser-Server architecture is used, which helps users query the information stored in the database easily through the internet. There are many advantages of this proposed model over the traditional parking system. Some of them are eliminating the manual search for empty parking spaces, reducing the traffic flow of vehicles, etc.

In this paper [9], a smart school building system has been proposed. The modelling and formal specification of the system has been done using UML and TLA+ respectively. The concept of sensors has been used to detect smoke, fire in the school building. Sensors to measure the temperature, to sense the light in the environment to decrease power consumption,

etc., have also been used. The safety aspect is achieved with the help of credentials. The system will open the door only for students and employees who have a valid username and password. Other people such as guests and visitors need to take special permission to enter the school building. This system, however, could have integrated more security features.

The paper proposes a Smart Office system using activity diagrams and NFA [10]. The specification has been done using VDM-SL. It is an alternative of TLA+ for specifying models. The model covers security aspects such as smoke detection, alarm systems, etc. User authentication has also been inculcated in the system. If the user is successfully authenticated, then the office lights will automatically get switched on, and the system will also set the air conditioning according to the temperature of the room. Whenever the user leaves the office, the system toggles from active to sleep mode and switches off all the lights and air conditioning.

III. Methodology

A. Requirement Analysis

Requirement Analysis is the phase in which the functions, goals, and tasks of the Smart Hospital Management System are defined. The system should have proper functionalities for the initial check-up of patients, recommendation of medical tests, diagnosis of disease, prescribing medication, providing medication, monitoring the health and vitals of patients, and tracking them in real-time. Further, the system should not have any loopholes, ambiguities, and inconsistencies and should ensure the safety of patients.

B. Creating UML Diagrams

The Unified Modeling Language is a developmental modeling language in the field of software engineering which provides a standard way to visualize the designs of a system. An activity diagram represents the working of a system by portraying the flow of control from the initial start state to the end state. The proposed system consists of five important subsystems. Hence, activity diagrams have been used to represent the working of the five subsystems.

- Fig. 1 is the activity diagram of the Initial Check-up and Test Recommendation system which depicts various processes such as login of doctor, check-up of patient, and suggestion of medical tests on the basis of the check-up.
- Fig. 2 is the activity diagram of the Disease Diagnosis and Medicine Prescription system. The RFID Wristband is scanned to get the test results from the database, these test results are analyzed, and the prescribed medicines are added into the database.
- Fig. 3 is the activity diagram of the Patient Monitoring system. RFID wristband tracks and measures the vitals of the patients. All information is then transmitted to the RFID reader and abnormalities are immediately reported.
- Fig. 4 is the activity diagram of the Medicine Dispensing system which depicts how the patient can buy the prescribed medicines from the dispensary by scanning the RFID wristband.

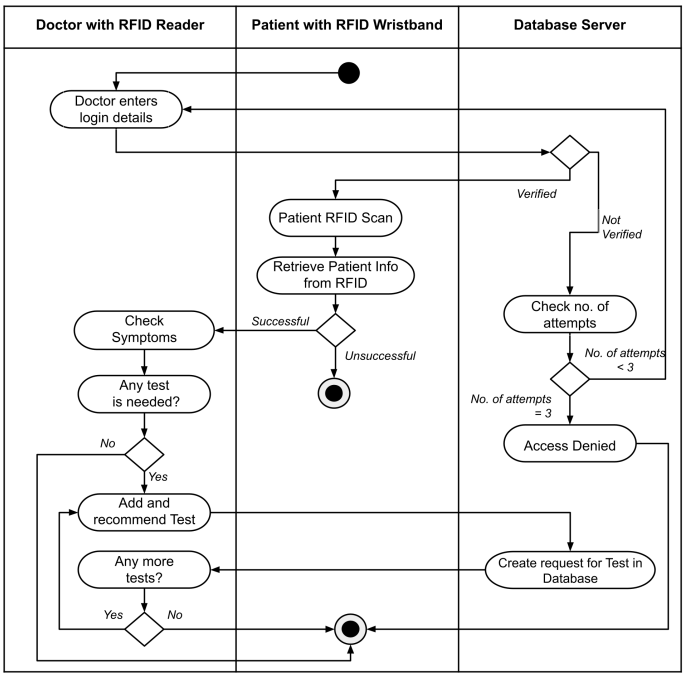

Fig. 1: Activity diagram of the Initial Check-up and Test Recommendation system

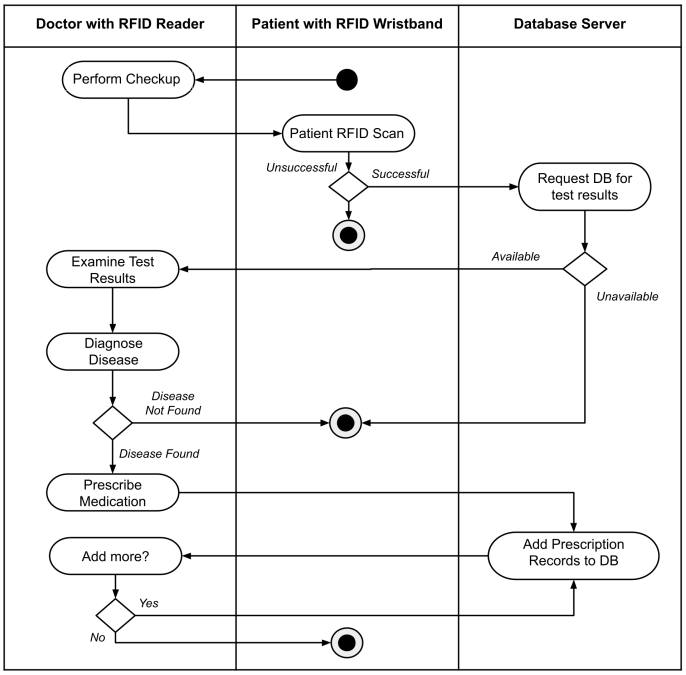

Fig. 2: Activity diagram of the Disease Diagnosis and Medicine Prescription system

- Fig. 5 is the activity diagram of the RFID Monitoring system which regularly checks the last access time of all the RFID tags and readers to ensure that they are working correctly. It monitors the low battery levels RFID tags and alerts the maintenance department in case of suspicion.

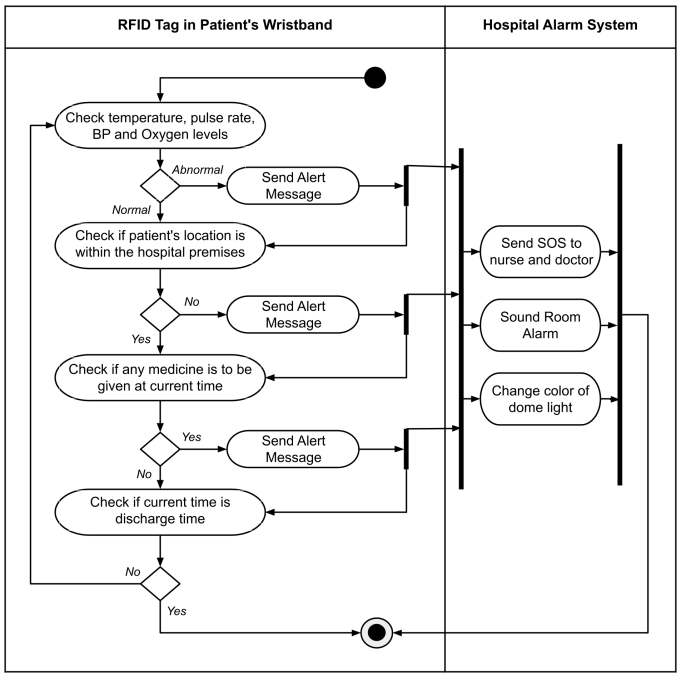

Fig. 3: Activity diagram of the Patient Health Monitoring system

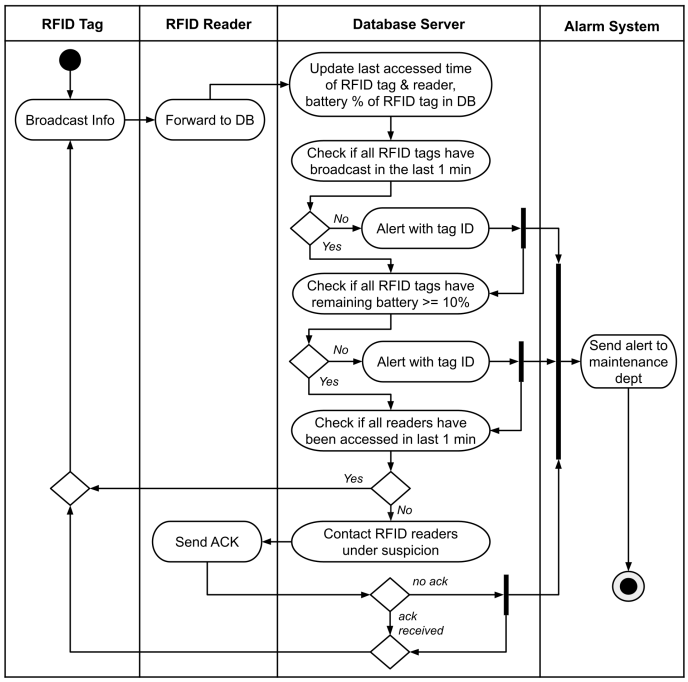

Fig. 5: Activity diagram of the RFID Monitoring System

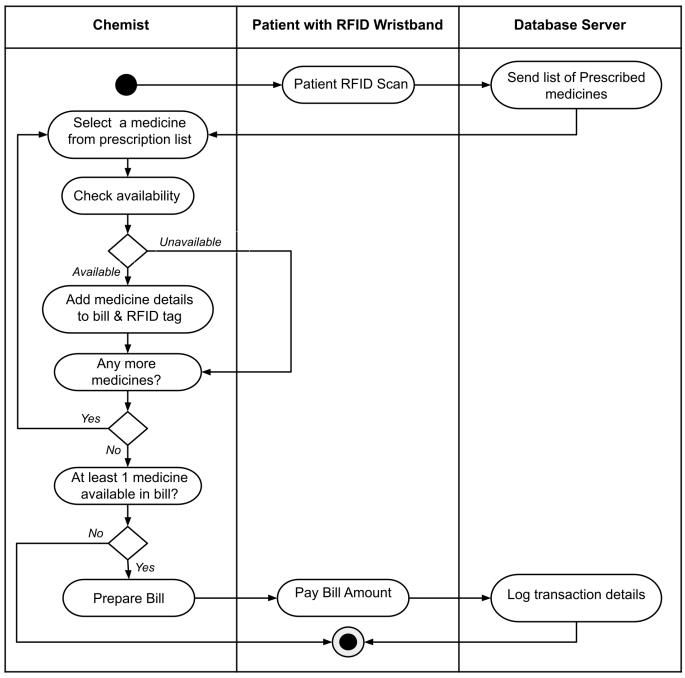

Fig. 4: Activity diagram of the Medicine Dispensing and Billing system

C. Fault Tree Analysis and modifying UML diagrams

Fault Tree Analysis is a graphical and top-down analysis technique that is used to analyze the potential for failure within a system or model. It is used to explore and investigate the causes of system-wide failures using Boolean logic and identifies all possible ways in which the system may reach an undesirable state. Fig. 6 shows the Fault Tree created for the Smart Hospital Management System. The main complications in patient care are faults in diagnosis, medication, or monitoring. All possible causes that may result in such faults were identified and added to the Fault Tree in appropriate positions. Most of the faults that may arise are due to technical errors such as faulty RFID tags, low battery of RFID tags, faults in RFID reader, etc. To eliminate all such errors, the created UML diagrams are modified.

D. Analysing state-transitions using NFA

Non-Deterministic Automata (NFA) is used for describing the working of the system in the form of states and transitions. It helps us to analyze the actions that can be taken from one state to another in detail. The NFA and the transition function for the first subsystem are displayed as an example.

Different symbols are used to represent different states of NFA: *IS* for initial state, *LC* for login credentials, *AN* for attempt number, *RS* for RFID scan, *RI* for retrieve info, *CS* for check symptoms, *TN* for tests needed, *AT* for add tests, *DB* for database request, *MT* for more tests, *TS* for termination state and ϕ for an invalid state. Similarly, the transitions of NFA are also denoted by different symbols: *v* for verified, *nv* for not verified, *s* for successful, *us* for unsuccessful, *y* for yes, *n* for no, *l3* for lesser than 3, *g3* for greater than 3 and ϵ for epsilon transition.

Fig. 7 and Fig. 8 show the NFA and the transition function of the first subsystem, 'Initial Check-up and test recommendation', respectively. The NFA and transition function display how the flow moves from one state to another state based on every possible transition. For example, when the

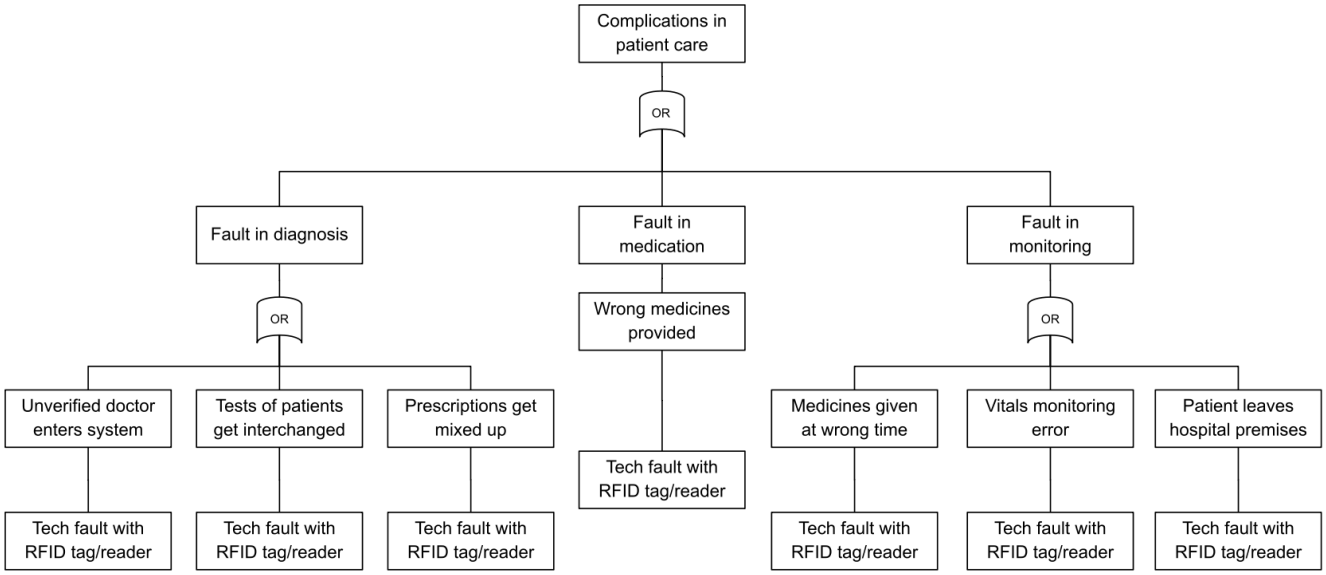

Fig. 6: Fault Tree

login credentials (LC) are verified (v), then the next state is RFID scan (RS). The flow begins at the Initial state (IS), passes through multiple intermediary states, and ends at the Terminating state (TS). ϕ in the transition function indicates invalid states.

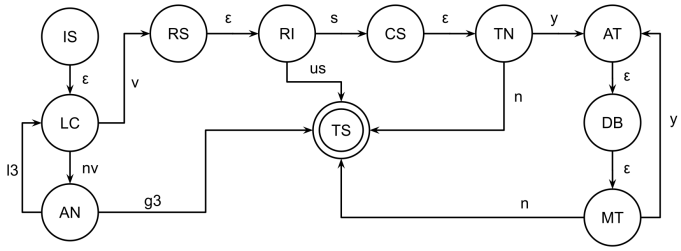

Fig. 7: NFA of Checkup & Test Recommendation System

	v	nv	s	us	y	n	l3	g3	ε
IS	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	LC
LC	RS	AN	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ
AN	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	LC	TS	ϕ
RS	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	RI
RI	ϕ	ϕ	CS	TS	ϕ	ϕ	ϕ	ϕ	ϕ
CS	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	TN
TN	ϕ	ϕ	ϕ	ϕ	AT	TS	ϕ	ϕ	ϕ
AT	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	DB
DB	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	MT
MT	ϕ	ϕ	ϕ	ϕ	AT	TS	ϕ	ϕ	ϕ
TS	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ	ϕ

Fig. 8: State transitions of NFA

E. Formal Specification using TLA+

TLA+ is a formal specification language used to design, model, document, and verify programs. It is a very expressive language that makes it easy to use and understand and is also highly flexible when needed to edit details to the specifications. TLA+ also has the Trace Explore feature that allows users to trace every single state to obtain results for each one. This makes finding, tracing, and fixing bugs much easier.

The TLA+ formal specification of the UML diagram in Fig. 1 (Initial Check-up and Test Recommendation system) has been explained below in detail as an example.

$$
\begin{aligned}
&\mathit{TypeOK} \triangleq \\
&\quad \land \quad \mathit{loggedIn} \in \{\text{“yes”}, \text{“no”}\} \\
&\quad \land \quad \mathit{credentials} \in \{\text{“correct”}, \text{“incorrect”}\} \\
&\quad \land \quad \mathit{attempt} \in (1 \,..\, 3) \\
&\quad \land \quad \mathit{rfidScan} \in \{\text{“done”}, \text{“notDone”}\} \\
&\quad \land \quad \mathit{patientInfo} \in \{\text{“retrieved”}, \text{“notRetrieved”}, \text{“error”}\} \\
&\quad \land \quad \mathit{symptomsChecked} \in \{\text{“yes”}, \text{“no”}\} \\
&\quad \land \quad \mathit{testNeeded} \in \{\text{“undetermined”}, \text{“yes”}, \text{“no”}\} \\
&\quad \land \quad \mathit{addedToDB} \in \{\text{“yes”}, \text{“no”}\}
\end{aligned}
$$

$$
\begin{aligned}
&\mathit{Init} \triangleq \\
&\quad \land \mathit{loggedIn} = \text{“no”} \\
&\quad \land \mathit{credentials} \in \{\text{“correct”}, \text{“incorrect”}\} \\
&\quad \land \mathit{attempt} = 1 \\
&\quad \land \mathit{rfidScan} = \text{“notDone”} \\
&\quad \land \mathit{patientInfo} = \text{“notRetrieved”} \\
&\quad \land \mathit{symptomsChecked} = \text{“no”} \\
&\quad \land \mathit{testNeeded} = \text{“undetermined”} \\
&\quad \land \mathit{addedToDB} = \text{“no”}
\end{aligned}
$$

Fig. 9: TypeOK and Init action

A total of 8 variables have been used in the sub-system

'Initial Check-up and test recommendation'. The acceptable range of values of each of these variables has been stated in 'TypeOK' action. The 'Init' action initializes all of these variables with their initial states. For example, initially, the doctor would not be logged in to the system. This is represented by specifying the value of the variable 'loggedIn' as false. Similarly, all other variables have been assigned their respective possible values that they can take when the system has not been started. The total number of initial states possible in this case is 2 as every variable takes a single initial value except for one variable, which can take two different initial values. The TypeOK and Init actions are displayed in Fig. 9.

$$
\begin{array}{l}
Login \triangleq \\
\quad \lor \ \land loggedIn = \text{“yes”} \\
\qquad \land \text{UNCHANGED } \langle loggedIn,\ credentials,\ attempt \rangle \\
\quad \lor \ \land loggedIn = \text{“no”} \\
\qquad \land credentials = \text{“correct”} \\
\qquad \land loggedIn' = \text{“yes”} \\
\qquad \land \text{UNCHANGED } \langle credentials,\ attempt \rangle \\
\quad \lor \ \land loggedIn = \text{“no”} \\
\qquad \land credentials = \text{“incorrect”} \\
\qquad \land credentials' \in \{\text{“correct”},\ \text{“incorrect”}\} \\
\qquad \land attempt < 3 \\
\qquad \land attempt' = attempt + 1 \\
\qquad \land \text{UNCHANGED } \langle loggedIn \rangle
\end{array}
$$

Fig. 10: Login action

The 'Login' action in Fig. 10. handles the case when the doctor attempts to login into the system. A maximum of three invalid attempts have been allowed in this case, after which access to the system is revoked, and the flow terminates. This ensures that unverified access to the system is not allowed. If the entered credentials are correct, then the doctor successfully logs into the system and this action is completed.

$$
\begin{array}{l}
Scan \triangleq \\
\quad \lor \ \land loggedIn = \text{“yes”} \\
\qquad \land rfidScan = \text{“notDone”} \\
\qquad \land rfidScan' = \text{“done”} \\
\qquad \land \text{UNCHANGED } \langle patientInfo \rangle \\
\quad \lor \ \land loggedIn = \text{“yes”} \\
\qquad \land rfidScan = \text{“done”} \\
\qquad \land patientInfo = \text{“notRetrieved”} \\
\qquad \land patientInfo' \in \{\text{“retrieved”},\ \text{“error”}\} \\
\qquad \land \text{UNCHANGED } \langle rfidScan \rangle \\
\quad \lor \ \land loggedIn = \text{“yes”} \\
\qquad \land rfidScan = \text{“done”} \\
\qquad \land patientInfo \in \{\text{“retrieved”},\ \text{“error”}\} \\
\qquad \land \text{UNCHANGED } \langle rfidScan,\ patientInfo \rangle \\
\quad \lor \ \land loggedIn = \text{“no”} \\
\qquad \land \text{UNCHANGED } \langle rfidScan,\ patientInfo \rangle
\end{array}
$$

Fig. 11: Scan action

The 'Scan' action in Fig. 11 simulates the scanning of the patient's RFID tag. This action is performed only when the doctor is logged into the system. Otherwise, it is terminated. After the RFID tag is scanned, the patient information is requested from the database. Depending upon the response, this information can be either retrieved successfully, or it may show an error if the data is non-existent.

$$
\begin{array}{l}
Diagnosis \triangleq \\
\quad \lor \ \land patientInfo = \text{“retrieved”} \\
\qquad \land symptomsChecked = \text{“no”} \\
\qquad \land symptomsChecked' = \text{“yes”} \\
\qquad \land \text{UNCHANGED } \langle testNeeded,\ addedToDB \rangle \\
\quad \lor \ \land patientInfo = \text{“retrieved”} \\
\qquad \land symptomsChecked = \text{“yes”} \\
\qquad \land testNeeded = \text{“undetermined”} \\
\qquad \land testNeeded' \in \{\text{“yes”},\ \text{“no”}\} \\
\qquad \land \text{UNCHANGED } \langle symptomsChecked,\ addedToDB \rangle \\
\quad \lor \ \land patientInfo = \text{“retrieved”} \\
\qquad \land symptomsChecked = \text{“yes”} \\
\qquad \land testNeeded = \text{“yes”} \\
\qquad \land addedToDB = \text{“no”} \\
\qquad \land addedToDB' = \text{“yes”} \\
\qquad \land \text{UNCHANGED } \langle symptomsChecked,\ testNeeded \rangle \\
\quad \lor \ \land patientInfo = \text{“retrieved”} \\
\qquad \land symptomsChecked = \text{“yes”} \\
\qquad \land testNeeded = \text{“yes”} \\
\qquad \land addedToDB = \text{“yes”} \\
\qquad \land addedToDB' = \text{“no”} \\
\qquad \land testNeeded' \in \{\text{“yes”},\ \text{“no”}\} \\
\qquad \land \text{UNCHANGED } \langle symptomsChecked \rangle \\
\quad \lor \ \land patientInfo = \text{“retrieved”} \\
\qquad \land symptomsChecked = \text{“yes”} \\
\qquad \land testNeeded = \text{“no”} \\
\qquad \land \text{UNCHANGED } \langle symptomsChecked,\ testNeeded \rangle \\
\qquad \land \text{UNCHANGED } \langle addedToDB \rangle \\
\quad \lor \ \land patientInfo = \text{“notRetrieved”} \\
\qquad \land \text{UNCHANGED } \langle symptomsChecked,\ testNeeded \rangle \\
\qquad \land \text{UNCHANGED } \langle addedToDB \rangle
\end{array}
$$

Fig. 12: Diagnosis action

The 'Diagnosis' action in Fig. 12 simulates the diagnosis of the patient. This action is executed only when the patient information is successfully retrieved. In this step, the doctor checks for symptoms and recommends tests that the patient needs to perform. After checking the symptoms, if any tests are needed, then the recommended tests are added to the database. This continues till there are no more tests to be added.

$$
\begin{array}{l}
Next \triangleq \\
\quad \land Login \\
\quad \land Scan \\
\quad \land Diagnosis \\
\quad \land TypeOK
\end{array}
$$

Fig. 13: Next action

Finally, the 'Next' action in Fig. 13 is composed of all the above actions, namely, Login, Scan, Diagnosis. TypeOK checks for the domain of each variable.

F. Verification using TLC model checker

TLC Model Checker is a tool that is available in the TLA+ Toolbox. It is a checker and simulator for TLA+ specifications. The TLC Model checker is not limited to single memory processors like the B, VCC, and Event-B model checkers as it can utilize multiple cores efficiently and hence can handle a large number of states.

IV. Results

All sub-systems are first modelled using UML. The activity diagrams are shown in Fig. 1, 2, 3, 4, and 5. Then, NFA and transition functions are used to understand the working of each subsystem in a better way. Further, formal specification of all the sub-systems is done using TLA+ formal specification language. Next, the created formal specifications are parsed, and the syntax is checked. Then the specifications are verified and validated using the TLC Model Checker available in the TLA+ Tool Box. Finally, the statistics of all the formal specifications obtained from the TLC Model Checker are displayed in Table 1.

TABLE I: Number of states checked by TLC Model Checker

Subsystem	Init States	Next States	Total States
1	2	28	30
2	1	10	11
3	69995796	72176832	142172628
4	1	735	736
5	100	1419	1519

The TLC model checker terminates successfully after finding 30 distinct states for the 'Initial Checkup and test recommendation' subsystem, 10 distinct states for 'Disease Diagnosis and Medicine Prescription' subsystem, 142,172,628 distinct states for the 'Patient Health Monitoring' subsystem, 736 distinct states for 'Medicine Dispensing and Billing' subsystem and 1,519 distinct states for 'RFID Monitoring' subsystem. Successful termination of the TLC model checker proves the correctness of all formal specifications.

V. Conclusion

In this paper, a Smart Hospital Management System is proposed, which focuses on automating various tasks and activities which are performed manually in traditional hospital systems. The proposed system uses RFID technology as its core to automate various tasks, ensure security and consistency of information of patients and maintain seamless communication between various entities involved in the system. Wristbands equipped with RFID tags and other sensors are provided to patients. The RFID tags are used to store information related to the patients as well as to periodically transmit collected data to the RFID readers installed in the hospital. This data is then analyzed, and then alarms are activated, if necessary. The system contains 5 major subsystems: Initial Check-up and Test Recommendation system, Disease Diagnosis and Medicine Prescription system, Patient Health Monitoring system, Medicine Dispensing and Billing system, and RFID Monitoring System. These subsystems are modelled using UML and their working is analyzed using NFA. Then, TLA+ language is used to create the formal specifications, and finally, the subsystems are verified using TLC Model Checker.

References

[1] Erwin Halim et al. ""Smart Healthcare" a Medical Record System for Effective Health Services". In: *2020 International Conference on Information Management and Technology (ICIMTech)*. IEEE. 2020, pp. 806–811.

[2] Emanuel S Grant. "Requirements engineering for safety critical systems: An approach for avionic systems". In: *2016 2nd IEEE International Conference on Computer and Communications (ICCC)*. IEEE. 2016, pp. 991–995.

[3] Pooja Singh and Lalit Kumar Singh. "Reliability and Safety Engineering for Safety Critical Systems: An Interview Study With Industry Practitioners". In: *IEEE Transactions on Reliability* (2021).

[4] Paul H Frisch. "RFID in Today's intelligent hospital enhancing patient care & optimizing hospital operations". In: *2019 IEEE International Conference on RFID Technology and Applications (RFID-TA)*. IEEE. 2019, pp. 458–463.

[5] Saba Latif, Aniqa Rehman, and Nazir Ahmad Zafar. "NFA Based Formal Modeling of Smart Parking System Using TLA+". In: *2019 International Conference on Information Science and Communication Technology (ICISCT)*. IEEE. 2019, pp. 1–6.

[6] Denis Cousineau et al. "TLA+ proofs". In: *International Symposium on Formal Methods*. Springer. 2012, pp. 147–154.

[7] Young-Mi Kim and Miyoung Kang. "Formal verification of SDN-based firewalls by using TLA+". In: *IEEE Access* 8 (2020), pp. 52100–52112.

[8] Lanxin Wei et al. "Design and implementation of smart parking management system based on rfid and internet". In: *2012 International Conference on Control Engineering and Communication Technology*. IEEE. 2012, pp. 17–20.

[9] Nawar H Obeidat and Carla Purdy. "Modeling a smart school building system using UML and TLA+". In: *2020 3rd International Conference on Information and Computer Technologies (ICICT)*. IEEE. 2020, pp. 131–136.

[10] Aniqa Rehman, Saba Latif, and Nazir Ahmad Zafar. "Formal Modeling of Smart office using Activity Diagram and Non Deterministic Finite Automata". In: *2019 International Conference on Information Science and Communication Technology (ICISCT)*. IEEE. 2019, pp. 1–5.