International Journal of Engineering and Management Research e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume-12, Issue-5 (October 2022)
www.ijemr.net https://doi.org/10.31033/ijemr.12.5.24

Design and Implementation of an Automated Hospital Management System with MERN Stack

Jayasiri K.C.N[1], Thathsarani W.R.V.K.[2], De Silva D.I.[3] and Vidhanaarachchi S.[4]
[1]3rd Year Undergraduate, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, SRI LANKA
[2]3rd Year Undergraduate, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, SRI LANKA
[3]Senior Lecturer, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, SRI LANKA
[4]Academic Instructor, Department of Computer Science and Software Engineering, Sri Lanka Institute of Information Technology, Malabe, SRI LANKA

[1]Corresponding Author: nirashajayasiri@gmail.com

ABSTRACT

A hospital is a place that needs more effective and efficient management of information, people, and assets. This paper demonstrates the design and implementation of an autonomous system with mern stack that can manage doctor information, patient information, inventory information, and administrative functionalities in a hospital environment. This was written with the intention of eliminating the problems of manual hospital management systems such as data redundancy, data inaccuracy, poor accesibility and lack of data security. The paper addressed the problems of time consumption of storing, retrieving, updating, and processing hospital data, generating ambiguous and inaccurate reports, and poor access control to sensitive information in manual paper-based hospital management systems and provides an intuitive and modern approach to solving those problems using an autonomous web application to improve the overall efficiency of a hospital environment. The tools used to implement the proposed system are reactjs library with redux, expressjs, nodejs, and mongodb as the online cloud database. Google oauth 2.0 is used as the authorization protocol. This proposed solution provides an excellent way of authentication and authorization, generating reports for statistical and information gathering purposes, managing administrative tasks, and improving information storing, manipulating and retrieving infrastructure.

Keywords— Hospital, Hospital Management System, Web Application

I. INTRODUCTION

Due to the increase in world population and an increasing number of infectious and non-infectious diseases, hospitals have become very needed places in the world. A hospital is comprised of a widely contrasting group of professionals such as doctors, nurses, laborers, security, medications, and equipment to provide curative and personalized service to patients [1]. According to World Health Organization, a hospital is one of the most integral organizations which provides various treatments and health care to people, education for medical professionals, and a place where the clinical research is done for the advancement of medicines and operations [2]. Therefore, hospitals can be considered one of the most complex and crowded organizations that require proper, secure, and efficient management [3]. Different operational tasks such as keeping records of patients, their diagnosis, laboratory test results and bills, keeping records of staff members, inventory, and physical assets, keeping records of various diseases and medicines for those diseases, and creating different types of statistical reports are done in a hospital environment [2]. In most hospitals, all of the above-mentioned tasks are carried out in a paper-based system manually. Thus, leading to inaccurate data, data redundancy, less efficiency, low security of information, low recovery, and backup of critical data [3].

A digital approach can be used to overcome the above-mentioned problems. Information technology enables keeping electronic records of data, querying data to get the information required fast and accurately, generating statistical reports with a single click, and managing inter-department data in a centralized location [4]. A hospital management system should be able to collect, insert, and manipulate data quickly and get the work done more efficiently [5]. The purpose of this paper is to design and implement a hospital management system that stores data in a database instead of paper, eliminate redundancy of data, fasten the process of retrieving information and creating reports, and increases the security and efficiency of the management of a hospital. This paper demonstrates solutions for problems in existing systems by proposing a web-based application that provides timely and accurate access to data, inserting new data into the system, and managing authentication and simultaneous multiple-user access.

In upcoming sections of this paper describes the related work that has been done to improve the efficiency of a hospital, the methodology used to implement the system, detailed information of the proposed solution, key findings and test results, conclusion, and demonstrates the areas which need to be further improved.

II. RELATED WORK

Charles [5] has proposed a design for a hospital management system that facilitates user authentication and access control mechanisms to prevent unauthorized users from accessing sensitive information. It has also included a way of retrieving information quickly from the database. But [5] design has not included any proper mechanism to generate reports of data in the database in any format.

F.Droma [6] has proposed a design with different modules for a hospital management system, but it has only included the implementation of maintaining patient records. It has not mentioned any other hospital affairs that need to be managed appropriately.

Adebisi et al. [3] have proposed a design for a hospital management system that reduces data redundancy and provides fast access to data. The design includes three main modules in a hospital management system; doctor, receptionist, and pharmacist, but it has not considered the physical assets and staff management components in a hospital.

Considering the current situation of public hospitals, Renata Brager and Andrzej Weindlocha [7] have proposed a sanitarium operation with a spare operation concept. They have stated the operation of modern associations should be focused on patient satisfaction, financial liquidity, and a high-quality sanitarium, regardless of the type of operations. The T backup surgery concept has been proven to work in hospitals.

Saravanan Raju et al. [8] have enforced a responsive web application using the M.E.R.N. stack that can be penetrated on both desktops as well as mobile phone cyber surfers. The web operation includes views, and they're erected as different factors and are rendered in the HTML element. Every element has its own set of J.S.X. elements. Additionally, stoner authentication styles and filed attestations have been developed.

Samikshya Aryal [9] has applied current web practices to the MERN stack. Her thesis explained the process of developing a full-stack web operation with stylish practices of the MERN stack similar to the reusability of JavaScript factors, operation of React hooks, MVC pattern, and numerous further from scratch.

III. METHODOLOGY

The architecture of the proposed system is based on an MVC model. Client tier of the proposed system written in JavaScript, HTML, and CSS by using ReactJS with Redux as the framework. This is the level that users interact with to access the functions of the web application. A controller is written using NodeJS and ExpressJs. This tier represents the application server that acts as a bridge between the client tier and the database tier. The controller serves HTML pages to the user's device and accepts HTTP requests from the user and follows with the appropriate response. At last, the database tier is hosted on MongoDB. Google OAuth is used for authentication and authorization. The main functionalities of the hospital management system are identified. These were Doctor Management, patient management, inventory management, and salary management. After that, those four functionalities are investigated, and planned logic and interfaces.

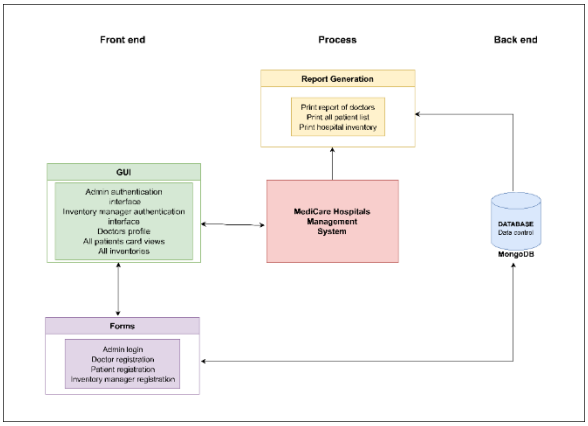

Figure 1: System Architecture

When developing patient management functionality, is divided into four phases as below figure.

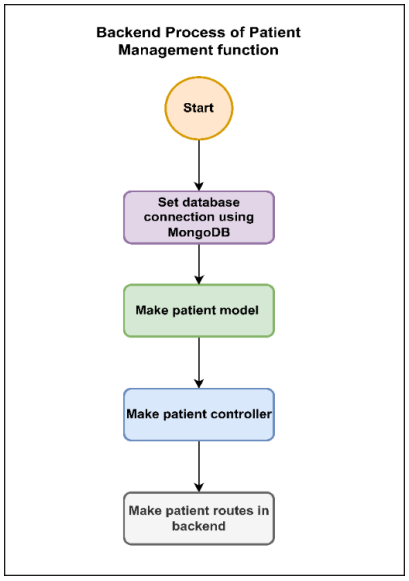

Figure 2: Backend process of patient management functionality

When implementing the backend in patient management functionality of the hospital management system, first, the database connection is set to the implementing environment. A MongoDB database is created in a cloud environment and added collections to it. For the patient management function, a collection named "patients" is in the MongoDB database. According to the MVC architecture, a data model named "patient" is created for patients and patient schema. In the patient schema, stored patient variables with their data types.

Then a controller named "patient-controller.js" is implemented and implemented all functions related to a patient in it, such as get all patients, add patients, delete patients, and get patients by id.

A JavaScript file called "patient-routes.js" is created to implement all routes in patient management functionality. After completing the backend process of patient management functionality, the front end is created using ReactJS. The main color theme for the hospital management web application is chosen to match the brandings, and a header and footer are created for that. GitHub is used as the version control tool as well as the integrating tool. For testing purposes of the proposed system, SonarQube is used to inspect the code quality of the system and the Selenium test automation tool is used for testing.

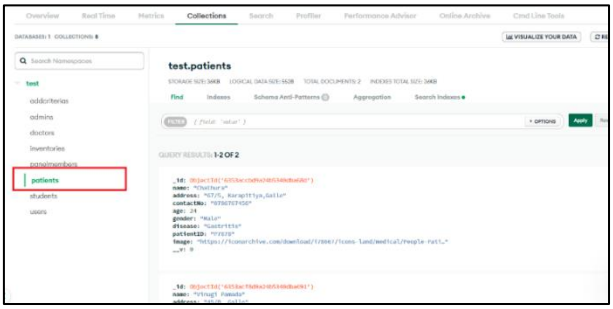

Figure 3: Patient Record in MongoDB

The backend process of the doctor management functionality is divided into four main sections, which are setting database connection, creating a database model for doctors, creating controller methods to perform different database tasks, and creating routes for the doctor model. Figure no: 4 demonstrates the backend process of the doctor management functionality.

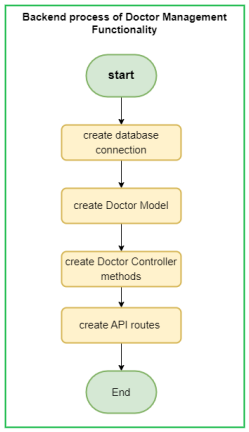

Figure 4: Backend process of doctor management functionality

The 'Doctor' model contains all the attributes, data types of the attributes, and some constraints on the attributes which is used to store doctors' data in the database. The main attributes used to create the doctor model were first name, last name, N.I.C., email address, home address, phone number, highest education qualification, doctor id, ward no, and password. A doctor schema is created using this model and saved in the MongoDB database as a 'Doctors' collection. A controller named 'DoctorCtrl.js' is implemented to store all the methods related to database functions that need to be performed on the doctor schema. This controller contained methods to insert a doctor to the database, doctor login, send an email to verify the email address and create an account, update a doctor's details, delete a doctor by id, and get all details of all doctors, get details of a doctor by id, reset the password of a doctor account methods. Furthermore, this controller contains all the validations of the doctor's attributes when creating a new doctor account. Those validations include empty field validations, email address validation, N.I.C. number validation, phone number validation, and password validation. Aside from the above-mentioned validations, before inserting a doctor record into the database, the inputted email address is checked with the existing doctors' email addresses to find matching emails. If only the email is unique, a new doctor record is inserted into the database. Before inserting a record in the database password is converted into a hashed password using the 'bcrypt' Node module.

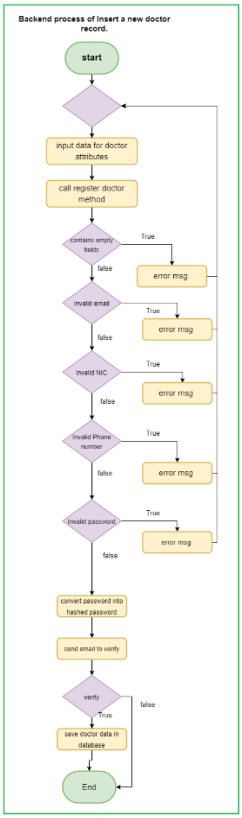

Figure 5: Inserting a new doctor record flow diagram

API routes related to doctor management functionality are stored in a file. Routes for creating a doctor account, sending verification emails, doctor login, viewing a doctor profile, updating doctor details, deleting a doctor profile, and resetting passwords are implemented in this file.

```
_id: ObjectId('6354cf786eae921ad4742bf8')
thumbnail: "https://res.cloudinary.com/dhdg4uhps/image/upload/v1631079559/samples/…"
firstName: "Chamal"
lastName: "Jayasuriya"
email: "supunsg@gmail.com"
nic: "787898778V"
address: "Colombo"
phone: "0723431234"
qualification: "Bachelors Degree"
doctorid: "DT24"
wardno: "006"
password: "$2b$12$6ewN8T3TN6ozQ7iBheZ53OfEXbQaXg5dkxYPzo.Z9nrRdPTeZ0q4W"
createdAt: 2022-10-23T05:22:00.024+00:00
updatedAt: 2022-10-23T05:22:00.024+00:00
__v: 0
```


Figure 6: Doctor Record in the MongoDB

Inventory management functionality mainly focused on managing medicine inventory in the hospital. Medicine names, available stocks, categories, minimum stock, and vendor details are stored in the MongoDB database. New medicines can be added or deleted, stocks can be updated, and all available medicine stocks can be viewed as a list using backend controller methods. Depending on the criticality of the medicine, three labels are introduced: 'critical,' 'mandatory,' and 'general.' An email-sending method is implemented to alert the hospital as well as vendors to notify the availability of the medicines based on their category labels.

Apart from these functionalities, an admin user management functionality is implemented to control admin user tasks and privileges such as viewing all doctor details and viewing all patients.

IV. PROPOSED SYSTEM

The proposed system is divided into four main modules, which are the 'Doctor module,' 'Admin Module,' 'Pharmacist Module,' and 'Common module.' Doctor management functionality is included in the 'Doctor module' and the 'Admin module.' Patient management functionality is included under the 'Admin Module.' The home interface, 'about us' interface, and 'Contact Us' interface are included in the 'Common module.' Employees' salary management functionality is included under the 'Admin Module.' Inventory management functionality is included under the 'Pharmacist Module.'

'Google OAuth 2.0' is used as the authorization protocol to grant permissions to access remote APIs using an 'Access Token.' 'JSON Web Tokens' (JWT) format is used for the 'Access Token' in this web-based system.

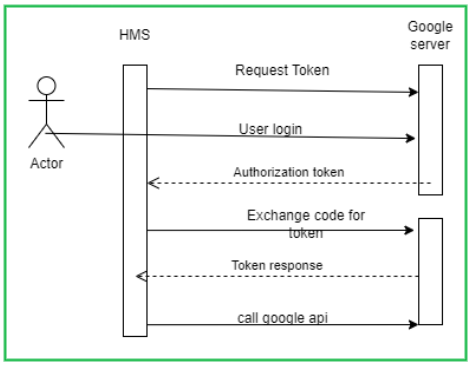

Figure 7: OAuth 2.0 mechanism

A. Patient Management

Patient management functionality is one of the main functionalities of the proposed system. As an in-hospital activity, the hospital needs to manage all patient details in a good manner. For that, patient management functionality is proposed as one of the main features of the web application.

Admin can log in to the system and manage patient records. When a new patient comes to the hospital, the admin can register that patient by giving name, address, age, contact number, gender, any major diseases that they suffered from earlier, patient id, and patient's image. Without filling in one of the fields in that form, the admin cannot register the new patient for the system. It is the validation of the patient management function.

After that, that patient is added to the "all patient" interface. It is maintained as card views of all patients that register to the hospital. In that card view, two buttons are maintained to update and delete. If the admin wants to update any fields that enter as patient details before, the admin can change those details by clicking the update button. If the admin wants to delete a patient record permanently from the system, the admin can click the delete button and delete that related record. Admin can generate a report with all patient details. When a hospital administrator needs a monthly report about all patients, it can be generated with the patient's name, address, gender, contact number, patient I.D., and list of major diseases that the patient suffered earlier.

B. Doctor Management

Under the 'Doctor module,' the 'Doctor Register' interface is implemented so that a doctor who is assigned to the hospital with a valid 'Doctor id' can self-register to the system. A registration success message and a verification email are sent to the user's email address to verify his email address. After the verification process is a success, a new user is able to log in to the system.

The 'Doctor Login' interface is proposed for doctors to log in to the system with their credentials. A registered doctor has the facility to log in to the system using his 'Doctor Id' and 'password' as credentials. If the inputted credentials are invalid, the system notifies the user by displaying appropriate error messages. When a doctor is logged in to the system, a user session is created for that user id and manages all privileges and actions of that user. If a doctor user forgets his password, a 'reset password' interface is included in the system. An email is sent to the user with the 'reset password' access link.

The 'doctor profile' interface is created to display doctor details. This interface is only accessible by the logged-in doctor user. User has the facility to update personal details and add a profile picture using this interface. Logged in user has the privilege to delete his account by clicking the 'delete button' in this interface, but further confirmation is needed to perform the action, which is clicking 'ok' in the pop-up window.

The 'View all doctors' interface is included under the 'Admin module' since it can only be accessed by an

admin user. Using this interface, an admin user has the facility to view all the doctors and generate a report of all the registered doctors. Furthermore, this interface has a search component that provides the ability to filter doctors based on 'first name,' 'last name,' 'email,' 'N.I.C.,' 'doctor id,' or 'ward number.'

C. Employee Salary Management

Salary management functionality is another main functionality in the proposed system. When developing a hospital management system, there are lots of employees working on it. Hence, there should be a functionality to manage their salaries when it comes to a hospital; most of the time, they have to work overtime. Hence, when managing salaries, management has to calculate their overtime separately. The below equation shows how the system calculates the salary for each employee that worked in the hospital.

S (1)
= a
+ (b
∗ c)

In this equation (1), "S" is denoted as Employee salary, "a" is defined as the basic salary of an employee who works in the hospital, "b" is defined as Overtime (O.T.) hours, and "c" is defined as Overtime rate for a specific employee. When it comes to a specific employee in the hospital, the Overtime rate should be different. A Doctor's Overtime rate is higher than a nurse's. As well, a surgeon's Overtime rate is higher than a normal doctor's. Those rates are stored in the database to calculate the wages according to the equation.

D. Inventory Management

Medicines are crucial property in a hospital environment. This system uses an effective method of managing medicine inventory. The 'Medicines' interface provides an overview of all available medicines in the hospital. The Medicine list is displayed in alphabetical order. 'Bubble Sort Algorithm' is used to sort medicine names in alphabetical order. The 'bubble sort algorithm' uses two string arrays containing medicines' names, and when sorting happens, larger values that are the name of the medicine starting from letters from the later part of the alphabet are drawn to the bottom, and names that are starting from the first letters of the alphabet come upward to the top. The medicine list can be further filtered and summarized using different filter criteria in the interface. Access to this interface is limited to pharmacists and administrators.

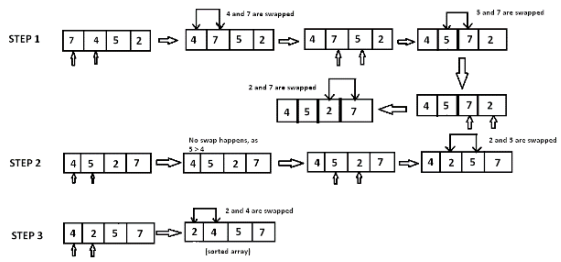

Figure 8: Bubble Sort Algorithm [10]

The 'Add Medicine' interface is added to the system for the purpose of adding new medicines to the system, and the 'update medicine' interface is implemented to update details of available medicines in the system. This system is developed to manage medicine stock properly. When the stock of a medicine is reached the minimum stock limit, if it is a medicine that is categorized under 'critical,' an email is sent to the vendor of that medicine with previously ordered stock. An email is also sent to the chief pharmacist notifying him that the medicine is back-ordered. If the minimum stock limit teched medicine belongs to the 'mandatory' category, an email is sent to the pharmacist notifying him the medicine needs to be re-ordered.

The report component of this inventory management functionality has been developed to get the information and statistics required more easily and accurately. Under this functionality, a user could generate several reports. One is the 'all available medicine' report. When generating that report, the system provides the opportunity to filter the medicines based on the disease it cures, category, availability, and vendors. Total stock used per day, per week, or per month report can be generated to analyze the usage of medicines. Minimum stock limit reached medicine lists based on category types can also be generated for the current date using this proposed system.

V. DISCUSSION

According to the 'Doctor Registration' interface of the 'Doctor module,' when submitting the form, user inputs are validated, and appropriate validation messages are displayed to the user interactively and descriptively so that the user will be able to fix the issues and submit the details correctly. When form fields are filled successfully, an email is sent to the user to verify the email address for account creation purposes. Depending on the login credentials, user sessions are created, and appropriate user privileges are offered to the users with the help of the 'Google OAuth 2.0' authorization protocol. Doctor profile deletion functionality is privileged only to the owner of that profile and the administrator. Administrators can view all the patients and view all the doctors and delete or update doctors or patients. The 'reset password' interface can only be accessed via the email which is sent to the user email address with an access token. The system manages to generate different reports related to various functionalities with different access privileges. The salary calculation facility is provided by the system using a formula to generate the salary of each employee based on their wage descriptions. Automation of sending emails and notifications to appropriate users in the inventory management system has made the re-stocking process easier.

Testing was done on individual interfaces, using the Selenium test automation tool to ensure that all the functionalities and navigations were working properly

and gave the expected results. Testing was done on authenticating users, navigation across interfaces, doctor registration and doctor login, doctor profile update, generating reports, and inserting a new patient. Integration and system test was conducted to verify the integrated system modules were working as expected.

VI. CONCLUSION

A computerized hospital management system that solved the problems of the existing manual hospital management systems has been developed, including user authentication and authorization. This system can solve many manual system problems, such as assessing old data, filtering data, updating, and removing records, and managing access to sensitive information. However, the system does not include other staff members' management functionalities, physical assets management functionality, and laboratory management functionality. Therefore, a computerized hospital management system that includes other staff member management functions and laboratory and physical asset management functions will be engaging research in future research paths.

REFERENCES

[1] A.K malhotra. (2009). *Hospital management system: An Evaluation, global Indian Publication*.
[2] Mansi Chitkara, Namita Khandelwal & Avinash Chaporkar. (2010). *Project report on hospital management system*.
[4] Ayodele Cole Benson. (2011). *Hospital information systems in Nigeria: Review of literature*.
[3] Adebisi O.A, Oladosu D.A, Busari O.A & Oyewola Y.V. (2015). Design and implementation of hospital management system. *International Journal of Engineering and Innovative Technology (IJEIT), 5*(1).
[5] Waban Charles. (2007). *Project report on computerized health records management system*.
[6] F.Droma. (2009). Project report on an automated system for patient record system. *Department of Information Technology, Maker ere University*.
[7] Renata Brajer-Marczak & Andrzej Wiendlocha. (2018). *Project report on lean management concept in hospital management – possibilities and limitations*.
[8] Saravanan Raju, S.Soundararajan & V.Loganathan. (2021). *Project report on MERN stack web application*.
[9] Samikshya Aryal. (2020). *Bachelor's thesis of MERN stack with modern web practices*.
[10] Bubble Sort Algorithm. (2022). Available at: https://www.hackerearth.com/practice/algorithms/sorting/bubble. Accessed on: 27/10/2022.