International Journal of Computational Science and Engineering Research
ISSN: XXXX- XXXX(Online) , http://www.ijcser.com/
Regular Issue , Vol. 2, Issue. 2 (April - June), 2025 , Pages: 1 - 7
Received: 15 January 2025 ; Accepted: 29 March 2025 ; Published: 12 April 2025.
Original Paper : Applied Research - Experimental Research Paper
https://doi.org/10.63328/IJCSER-V2I2P1

Patient Management System

Ravichander Venkatareddygiri [1*] , S. Dilli Babu [2]

[1*] Department of Computer Applications(MCA) , Mohan Babu University, Tirupathi, AP, India ; ravichander0999@gmail.com
[2] Department of Computer Science and Engineering , Mohan Babu University, Tirupathi, AP, India ; dillibabusalvakkam@gmail.com

** Corresponding Author:* ravichander0999@gmail.com

Abstract: Even if technology has been rapidly adopted in healthcare, a large number of small clinics still use outdated Patient Management Systems that do not offer much cloud integration, remote access, or communication tools. However, they hinder efficient patient data management and compliance in terms of security standards or quality care delivery. In this case, the project proposes a modern patient management system meant specifically for small clinics. It is a complete but light solution that will help in streamlining operations, improving patient care, and bettering the effectiveness of administration. The system uses cutting-edge technologies to buttress the gap and conversion from legacy systems in meeting modern healthcare demands. The patient management system will provide patient registration, schedule appointments, and hold video conferencing such that a clinic could handle interactions with patients while being able to assess timely online consultations. The backend ensures user authentication, file management, and data storage with secure and efficient Appwrite. Next.js, TypeScript, and TailwindCSS make up the frontend which responds in an intuitive interface across devices, improving the experience for healthcare providers and patients alike. Central to the system is secure data and compliance. It has met HIPAA standards, thus ensuring proper handling and storage of sensitive patient data. It also integrates with Sentry for real-time monitoring and error detection, enabling fast resolution of problems and continuous improvements in performance. The two major factors for this solution are cost and scalability. Unlike the over-burdened, complex traditional high-priced systems, this new PMS provides all essential features in a straightforward and affordable package. Hence, no small clinic would be turned away due to financial or operational encumbrance.

Keywords: Liver, Patient, TailwindCSS, Cancer , Machine Learning.

1. Introduction

The The healthcare improvement process is gaining more and more momentum. Such changes are primarily steered by technology and a host of issues that provide a transformational approach to how care is offered and provided. With all such advancements, a majority of small clinics are still using the much older Patient Management Systems that fall far short in meeting the demands of the modern healthcare industry. Because of their rigid infrastructure, these legacy systems do not have the expertise to perform some sophisticated operations that modern healthcare operations require, such as remote consultations, real-time data access, and compliance with robust data security provisions. Such limitations impact small clinics disproportionately given that they provide a significant role in the delivery of affordable healthcare services. These small operators, oftentimes starved of resources, simply cannot afford the expensive large-scale health management solutions. Hence, a handful still rely on these antiquated systems that lack key secret observe functionalities, thereby extending inefficiencies into care delivery and administrative operations. The increasing demand for digitization in healthcare makes developing robust, economical systems for small clinics very pertinent. The project creates a new-age Patient Management System to ease those pains through the application of modern techniques for a more streamlined, effective, and safe solution [1]. Customized for the clinical processes of small clinics, features include patient registration, appointment booking, and telemedicine options, while always pivoting toward data privacy compliance.

1.1. Significance of the Project:

This a huge step toward digitalizing a small clinic. This will cover certain pain points like administrative workflow inefficiency, poor accessibility,

Vol. 2, Issue 2, 2025, https://doi.org/10.63328/IJCSER-V2I2P1

data security loopholes, and relieve clinics from administrative stress and hassle, thus allowing clinics to focus more on caring for their patients [2]. The telemedicine functionalities line up the clinics with new trends in healthcare, such as the rise of remote consultations. Emphasis on security and compliance, i.e., this system safely keeps patient information while ensuring various digital healthcare compliances. Especially important now more than ever in building trust with patients who expect their healthcare provider to take data security and privacy seriously.

2. Overview of System Design

In system design. The Patient Management System (PMS) is targeted toward smaller clinics while encompassing patient registration, appointment scheduling, and video conferencing. It is covered by a three-layer architecture-consisting of presentation, business logic, and data. Next.js, TypeScript, and Tailwind are being used at the presentation end to provide responsiveness and mobile compatibility. The Workstay solution enforces the security policy for authentication, storage of data, and file management [3], with the Appwrite server for the Workstay all Appwrite handles all in-house user and file management and service integration for SMS and Tele conferencing. That database storing patient and system data implements various security capabilities to maintain HIPAA compliance via encryption and query efficiency. It will ensure reliability with real-time error logging and performance tracking using Sentry. The modular design allows scaling with less cost and great flexibility for enhancement in the future.

2.1. System Architecture Description

Frontend uses Next.js for server-side rendering (SSR) to speed things up and improve SEO. TypeScript boosts code quality through type-checking; on the other hand, TailwindCSS allows rapid responsiveness and extensive customization for UI. Frontend and backend are communicating and exchanging data through secure REST APIs for things like patient management, appointments, and video consultations [4].

2.1.1 Frontend layer: Next.js does server-side rendering to allow fast loading times and to be on the good side of the SEO. TypeScript implements type safety, whereas TailwindCSS allows rapid customization of UI along with responsiveness. Patient, doctor, and admin interactions of frontend are handled through secure APIs.

***2.1.2 Backend Layer*:** Appwrite is being used in the backend, which is an open-source BaaS that gives user authentication, data storage, and business logic. Role-based access control is provided to distinguish privileges. File management and integration of external services for SMS and video conferencing are handled internally by Appwrite APIs.

2.1.3. Database Layer: The database is securely keeping patient records, doctors' profiles, and appointment arrangements. Encryption at rest and in transit provide for full HIPAA compliance. Efficient querying allows for instantaneous retrieval of data. Backup and disaster recovery ensure the integrity of data.

2.2. External Services and Monitoring: Sentry for real-time error monitoring, a third-party API for SMS notifications and video conferencing in support of telemedicine and patient engagement.

2.3.Communication Flow: Users interact with the frontend, issuing requests to the back end-the back end authenticates users, legislates requests, and interacts with the database, from which it retrieves data. The interaction between the back end and external services for monitoring services and video conferencing adds another dimension to the whole system [5]. The modular architecture incorporates strong security facilities, performance, and scalability thus making PMS a strong and flexible yet cost-effective tool for small clinics.

3. Proposed System Model

Patient Portal: Where patients can register, schedule an appointment, and even view their own information.

Doctor Portal: Managed by Doctors to maintain their calendar, view patient data, and manage appointments. 3. Admin Dashboard: Managed by the admin to validate the doctor's profile before managing the appointments and overseeing complete system management.

Controller Layer:

- ***Patient Controller:*** Responds to actions on the patient portal including registering, scheduling an appointment, or viewing patient history. Collects or updates data through communication with Patient Model and Appointment Model.
- ***Doctor Controller:*** Processes all the requests from the doctor portal like schedule viewing and managing patient consultation. This model interacts with the Doctor Model for data retrieval and availability updates of the doctor.
- ***Admin Controller:*** Involves all administrative affairs which consist doctor registration as well as appointment system checks. It interacts with the Doctor Model and Appointment Model for the management of the system.

Vol. 2, Issue 2, 2025, https://doi.org/10.63328/IJCSER-V2I2P1

Model Layer

- *Patient Model:* It contains and retains all personal as well as medical history data concerning every patient.
- *Doctor Model:* It keeps the records of doctor data such as profiles, schedules, and areas of specialization [6].
- *Appointment Model:* Holds pieces of information regarding appointments such as the time slots, patient-doctor mapping, and status (**Figure.1**).

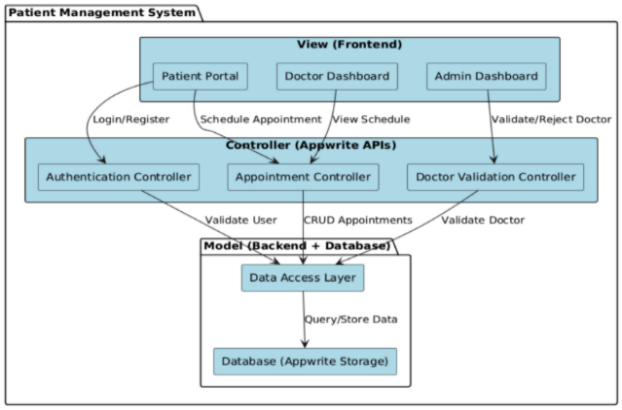

Figure.1: Explanation of the MVC Controller for PMS View Layer

4. Experimental Results

4.1. Implementation of the Patient Management System (PMS): The PMS is a contemporary web app that exploits the latest technologies to overcome the limitations ingrained in traditional systems. This section will present the key implementation facets of the PMS, namely the front end, the back end, video conferencing, and performance monitoring, with snippets highlighting the important parts of the implementation [7].

4.2 Frontend Implementation: Front-end PMS was built in Next.js, TypeScript, and TailwindCSS, such that it would be responsive and user-oriented. Most pertinent attributes and their respective implementation are discussed in the section.

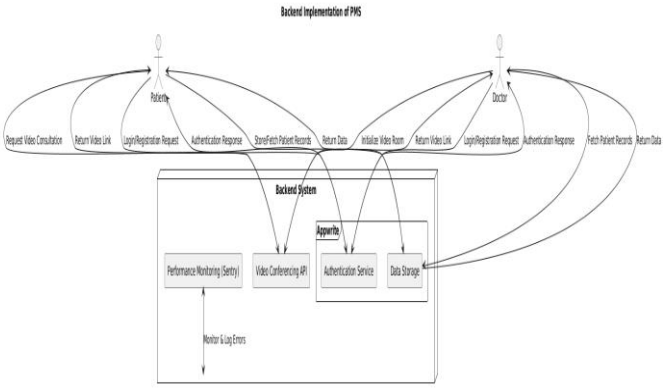

Figure.2: Controller for Frontend view PMS

4.2.1 Patient Registration: Patient registration is a primary feature that allows users build accounts by entering their names, ages, contact numbers [8], and then proceeds with medical history(**Figure.2**). The form validates the client's entry for accuracy and completeness at the client-end.

Code Snippet: Patient Registration Form

```
"use client";
import { zodResolver } from
"@hookform/resolvers/zod";
import { useRouter } from "next/navigation";
import { useState } from "react";
import { useForm } from "react-hook-form";
import { z } from "zod";
import { Form } from "@/components/ui/form";
import { createUser } from
"@/lib/actions/patient.actions";
import { UserFormValidation } from
"@/lib/validation";
import "react-phone-number-input/style.css";
import CustomFormField, { FormFieldType }
from "../CustomFormField";
import SubmitButton from "../SubmitButton";
export const PatientForm = () => {
  const router = useRouter();
  const [isLoading, setIsLoading] =
useState(false);
  const form = useForm<z.infer<typeof
UserFormValidation>>({
    resolver:
zodResolver(UserFormValidation),
    defaultValues: { name: "", email: "",
phone: "" },
  });
  const onSubmit = async (values:
z.infer<typeof UserFormValidation>) => {
    setIsLoading(true);
    try {
      const newUser = await
createUser(values);
      if(newUser)
router.push(`/patients/${newUser.$id}/registe
r`);
    } catch (error) {
      console.log(error);}
    setIsLoading(false); };
  return (
    <Form {...form}>
      <form
onSubmit={form.handleSubmit(onSubmit)}
className="space-y-6">
        <CustomFormField
fieldType={FormFieldType.INPUT}
control={form.control} name="name"
label="Full name" />
        <CustomFormField
fieldType={FormFieldType.INPUT}
control={form.control} name="email"
label="Email" />
        <CustomFormField
fieldType={FormFieldType.PHONE_INPUT}
control={form.control} name="phone"
label="Phone" />
        <SubmitButton
isLoading={isLoading}>Get
Started</SubmitButton>
      </form>
    </Form>
  );};
```

4.2.2 Appointment Scheduling: Patients have the opportunity to check time slots available for their appointments with their chosen physician [9]. The

Vol. 2, Issue 2, 2025, https://doi.org/10.63328/IJCSER-V2I2P1

interface with real-time feedback ensures that the patient is satisfied.

Code Snippet: Appointment Scheduling UI

```
"use client";
import { useState } from "react";
import { Button } from
"@/components/ui/button";
import { Dialog, DialogContent,
DialogHeader, DialogTitle, DialogTrigger
} from "@/components/ui/dialog";
import { AppointmentForm } from
"./forms/AppointmentForm";
import "react-datepicker/dist/react-
datepicker.css";
export const AppointmentModal = ({
patientId, userId, type, appointment })
=> {
  const [open, setOpen] =
useState(false);
  return (
    <Dialog open={open}
onOpenChange={setOpen}>
      <DialogTrigger asChild>
        <Button variant="ghost"
className={`capitalize ${type ===
"schedule" && "text-green-500"}`}>
          {type}
        </Button>
      </DialogTrigger>
      <DialogContent className="shad-
dialog sm:max-w-md">
        <DialogHeader>
          <DialogTitle
className="capitalize">{type}
Appointment</DialogTitle>
        </DialogHeader>
        <AppointmentForm userId={userId}
patientId={patientId} type={type}
appointment={appointment}
setOpen={setOpen} />
      </DialogContent>
    </Dialog>
  );};
```

4.3 Backend Implementation

The PMS backend was built using Appwrite, a secure and scalable backend platform. Key features include user authentication [10], data management, and integration with external APIs.

4.3.1 User Authentication and Data Storage

That offers reliable user authentication and means for storing patient and doctor data. Patient records, appointments, and medical files are stored in structured collections.

Code Registration API Snippet: Patient

```
const sdk = require("node-appwrite");
module.exports = async (req, res) => {
 const client = new sdk.Client();
 const database = new sdk.Database(client);
 client
  .setEndpoint("http://localhost/v1")
  .setProject("YOUR_PROJECT_ID")
  .setKey("YOUR_API_KEY");
 const { name, age, email, phone, medicalHistory } =
req.body;
 try {
  const response = await
database.createDocument("patients", "unique()", {
   name, age, email, phone, medicalHistory
  });
  res.status(200).json(response);
 } catch (error) {
  res.status(500).json({ error: error.message });
 } };
```

4.3.2 Integration of Video Conferencing

Telemedicine functionalities are set using third-party video conferencing APIs that allow consultations between doctors and patients online.

Code Snippet: Video Consultation Initialization

```
const axios = require("axios");
module.exports = async (req, res) => {
 const { doctorId, patientId } = req.body;
 try {
  const response = await axios.post("https://video-
api.com/create-room", {
   doctorId, patientId,
  });
  res.status(200).json({roomLink:
response.data.roomLink });
 } catch (error) {
  res.status(500).json({ error: error.message });
 }};
```

4.3.3 Performance Monitoring with Sentry

Performance monitoring and error monitoring are achieved by Sentry in the interest of providing end-users with a hitch-free experience and in instantly fixing issues.

Code Snippet: Sentry Setup

```
const Sentry = require("@sentry/node");
Sentry.init({
dsn: "YOUR_SENTRY_DSN",
tracesSampleRate: 1.0,
});
try {
} catch (error) {
 Sentry.captureException(error);}
```

The PMS implementation shows that modern technologies could achieve integration for a scalable, secure, and efficient system. Thanks to Next.js for developing a responsive UI, Appwrite for handling backend management, and Sentry for monitoring, the PMS boasts heavy-lifting capabilities targeted at small and local healthcare service providers [11].

Vol. 2, Issue 2, 2025, https://doi.org/10.63328/IJCSER-V2I2P1

5. Related Work

With these features, PMS collects, stores, and processes patient data including their appointments and consultations with doctors*:*

5.1.Patient Registration and Profile Management: Patients will create and maintain their own profiles, which will contain personal information, contact details, and medical history. The system provides simplicity in examination and management of patient data for the timely and applicable decisions by the healthcare provider. It allows profile updates while ensuring secure access, and authentication protects sensitive information[12]**.**

5.2. Management of Appointment Scheduling : Patients can proceed to schedule, change, and even postpone or cancel appointment bookings. Medical appointment reminders are automated for less missed consultations. Hence, doctors are maintaining their availability, and all appointments are recorded for future reference.

5.3. Registration of doctors and verification of doctors: There are several profiles registered by doctors with qualifications, specializations, and certifications. They are verified by the admins for allowing a professional to consult patients. A doctor can update his profile for new certification [13].

5.4. Telemedical Video Consultations: Video consultations are done in real-time. They provide tele-face consultations using the video conferencing software like Zoom and Jitsi. Once the doctor agrees to the request for consultation, the patient is forwarded a link to the video. Other modes of communication that can be used are screen sharing, document exchange, and chat.

5.5. SMS Notifications on Appointment Confirmation: These are automated SMS reminders which will confirm appointments, thus helping in reducing no-shows. Cancellation or rescheduling of appointments would send updates to patients and doctors, while follow-up reminders will encourage adherence to prescriptions and visits.

5.6. File Upload via Appwrite Storage: Patients and doctors can upload, as well as retrieve, medical documents including test reports in an authorized manner. Appwrite encrypts data, thus restricting access transferred confidentiality, while compliant with the required regulations [14].

5.7. Performance Monitoring and Error Handling using Sentry : Sentry tracks bugs and validates them due to real-time error tracking exceptions with performance issues. Therefore, it raises alerts to the developers regarding the errors to effect immediate fixes to ensure the PMS stable and reliable operations with less downtime [15].

5.8. Responsive Design for platform optimization: Designed and built in TailwindCSS and Next.js, it adapts across desktops, tables, and smartphones. The mobile-first design transforms usability for booking, profile editing, and teleconsultations across a variety of screen real estate

6. Results and Evaluation

The Patient Management System (PMS) testifies significant results that prove it has merit and is usable in healthcare management. One of the major achievements is the improved efficiency in the appointment scheduling and patient registration process. The system reduces the time for performing these tasks since the whole manual processes are automated and an easy user interface is provided. Patients get appointments within seconds, whereas doctors and administrators can manage schedules effortlessly through the backend, hence improving the overall operational productivity. The PMS was practically tested with changing loads scalably and performed well in terms of appointment booking alone because different situations were simulated-from small clinics to larger setups.

7. Restriction and Challenge

7.1. Implementation of Compliance with HIPAA: The safeguarding of patient data through such measures as encrypting the data, secure means of transferring the data, and making it accessible to authorized personnel is very challenging as it mainly concerns HIPAA compliance. Security measures make it difficult to use the entire system easily. Multiple security audits were performed on the project to check compliance.

7.2. Integrating Telemedicine: Difficulty came in including telemedicine, especially video consultations, because it required extensive network optimization, video compression, as well as integration with other third-party platforms to have high-quality audio and video at a low latency, even in areas that are not well connected.

7.3. Inadequate Testing in the Real World: Most testing done because little access to real clinics was mostly done in a simulated environment. This made it difficult to assess how theoretically the system would fair with several other factors in real-world scenarios such as overlapping schedules and workflows of clinics, thus increasing the chances of unrealized bugs post-deployment or performance issues.

7.4 Improvements to Follow: Future versions of this system should also add modules for

patient health records functionalities, which will store all information pertaining to a patient's medical history, prescription drugs and treatment plans, thus eventually further improving the quality of care. Diagnosis based on AI may help with real-time insights during consultations, and AI-based scheduling may be used for optimal resource allocation.

7.5. Balancing Security and Usability: Security versus usability is a very hot topic. The system would have to have very strong encryption as well as HIPAA compliance while remaining intuitive to non-technical staff in smaller clinics. That would always remain a pledge.

8. Conclusion and Future Scope

The Patient Management System (PMS) provides a scalable setup for clinics to safely keep and manage patient data, appointments, and telemedicine. This has been built using Appwrite, Next.js, TypeScript, and TailwindCSS to ensure user-friendliness without compromising security or performance. Core features include SMS notifications, video consults, and real-time monitoring via Sentry, making the PMS a very flexible instrument in healthcare management.

The challenges posed by HIPAA compliance, telemedicine optimization, and insufficient world testing were handled adequately through the route of planning and testing in a secure cloud infrastructure. Future improvements could include patient health records management and AI-based diagnostic assistance, enhancing clinical effectiveness. Unlike traditional systems, this system differs because it uses modern technology coupled with a user-oriented interface to meet the demand for telemedicine and secure healthcare services. This takes a step towards the cloud for patient care, empowering clinics with such technology while ensuring data privacy. With further real-time testing, it can be a stronger instrument for healthcare providers and patients.

Declaration

Conflicts of Interest: The authors declare no conflict of interest.
Author contribution: All authors wrote the main manuscript text and also consent to the submission
Ethical approval: Not applicable.
Consent to Participate: All authors consent to participate.
Funding: Not applicable, and No funding was received
Institutional Review Board Statement: Not applicable.
Informed Consent Statement: Not applicable.
Personal Statement: We Declare with our best of Knowledge that this research work is purely Original Work and No third party material Not used in this article drafting. If any such kind material found in further online publication, we are responsible only for any judicial and copyright issues.

References

[1]. A Web-based Patient Record and Appointment Management System IEEE. (n.d.). A Web-based Patient Record and Appointment Management System. Retrieved from https://ieeexplore.ieee.org/document/804398
[2]. Designing a Flow-based Mechanism for Accessing Electronic Health Records on a Cloud Environment, IEEE. (2022). Designing a Flow-based Mechanism for Accessing Electronic Health Records on a Cloud Environment. Retrieved from https://ieeexplore.ieee.org/abstract/document/10246933
[3]. PRMS: Design and Development of Patients E-Healthcare Records Management System for Privacy Preservation in Third Party Cloud Platforms, IEEE. (2022). PRMS: Design and Development of Patients E-Healthcare Records Management System for Privacy Preservation in Third Party Cloud Platforms. Retrieved from https://ieeexplore.ieee.org/abstract/document/9854886
[4]. Security-Aware Department Matching and Doctor Searching for Online Appointment Registration System, IEEE. (2019). Security-Aware Department Matching and Doctor Searching for Online Appointment Registration System. Retrieved from https://ieeexplore.ieee.org/abstract/document/8666707
[5]. REMOTE PATIENT MONITORING SYSTEM, International Journal of Distributed and Parallel Systems. (2012). REMOTE PATIENT MONITORING SYSTEM. Retrieved from https://www.academia.edu/download/37863484/3512ijdps09.pdf
[6]. A Secured Cloud based Health Care Data Management System International Journal of Computer Applications. (2012). A Secured Cloud based Health Care Data Management System. Retrieved from https://www.researchgate.net/profile/Nafiul-Rashid/publication/314879090
[7]. A Conceptual Framework to Ensure Privacy in Patient Record Management System IEEE. (2021). A Conceptual Framework to Ensure Privacy in Patient Record Management System. Retrieved from https://ieeexplore.ieee.org/abstract/document/9646903
[8]. Patient Health Record Protection Beyond the Health Insurance Portability and Accountability Act: Mixed Methods Study, Journal of Medical Internet Research. (2024). Patient Health Record Protection Beyond the Health Insurance Portability and

Accountability Act: Mixed Methods Study. Retrieved from https://www.jmir.org/2024/1/e59674/

[9]. BAMHealthCloud: A Biometric Authentication System for Healthcare Data in the Cloud Journal of King Saud University - Computer and Information Sciences. (2017). BAMHealthCloud: A Biometric Authentication System for Healthcare Data in the Cloud. Retrieved from https://www.sciencedirect.com/science/article/pii/S1319157817301143

[10]. Cloud-based Healthcare Data Management Framework KSII Transactions on Internet and Information Systems. (2020). Cloud-based Healthcare Data Management Framework. Retrieved from https://koreascience.kr/article/JAKO202011161035789.page

[11]. A Secure Framework For Enhancing Data Privacy And Access Control In Healthcare Cloud Management Systems Educational Administration: Theory and Practice. (2024). A Secure Framework for Enhancing Data Privacy and Access Control in Healthcare Cloud Management Systems. Retrieved from https://kuey.net/index.php/kuey/article/view/5783

[12]. Security and Privacy in Cloud-Based E-Health System, MDPI. (2021). Security and Privacy in Cloud-Based E-Health System. Retrieved from https://doi.org/10.3390/sym13050742

[13]. A reliable authentication scheme of personal health records in cloud computing Wireless Networks. (2021). A reliable authentication scheme of personal health records in cloud computing. Retrieved from https://link.springer.com/article/10.1007/s11276-021-02743-7

[14]. Cybersecurity In Healthcare: Securing Patient Health Information (Phi), Hippa Compliance Framework And The Responsibilities Of Healthcare Providers

[15]. Journal of Knowledge Learning and Science Technology. (2024), https://jklst.org/index.php/home/article/view/259/233