REST Backend System for Electronic Health Record

Abhilash Nambiar[1], Sandeep Konar[2], Soham Kulthe[3], Varun Vankudre[4], Mohini Misale[5]

[1,2,3,4]*Student of Computer Engineering, Terna Engineering College, Nerul, Navi Mumbai*
[5]*Professor of Computer Engineering, Terna Engineering College, Nerul, Navi Mumbai*

Abstract: *The electronic health record system manages and stores the patient record digitally. In our system admin adds the hospital in our system then the particular hospital adds new patient and doctor in the system. The patient can view their own personal record and the doctor can manage the patient's record whom they are treating. The patient cannot view the other patient record in the system. Data is managed properly because of role based authorization and bearer token are used for authentication. With the help of REST API developers can develop management systems for different purpose.*

Keywords: REST API, Electronic Health record, Role-based Authorization, Token-based Authentication.

1. Introduction:

The IT field is developing their technologies and their methods everyday. Hence the things which are developed using this technologies are also improved and their methods are also change but they are more efficient than previous one.

Previously for developing softwares soap api is used for backend purpose but it provides data only in the format of XML. But Now REST architecture comes in the picture Most of the MNC's and developers moved to the REST backend because it gives response in the form of JSON and it is lightweight. Also REST backend is faster than SOAP. We can develop any frontend on any platform if we used REST APIs.

Maintaining patient records and information is critical in determining the quality of care provided by a healthcare organization. Therefore, medical records management has emerged as a high priority among healthcare units for solutions that integrate data accessibility with day-to-day workflows to manage the total revenue cycle.

In our system, we manage health records in the digital format. An electronic health record (EHR) is the systematized collection of patient and population health information stored electronically in a digital format. These records can be shared across different healthcare settings. EHR systems eliminate the need to track a patient's past paper medical records and help ensure that the data is accurate and legible.

In this case we are developing REST backend of EHR system for any frontend.

2. Literature Survey:

2.1. A token-based user authentication mechanism for data exchange in RESTful API

According to research conducted by Xiang-Wen Huang, Chin-Yun Hsieh, Cheng Hao Wu and Yu-Chin Cheng Every request has a unique disposable token that is not easy to forge. The private token does not appear in transmission; attackers cannot get all information that generates a disposable token. Testing becomes more complicated because the client needs to generate disposable data every time. Both client and server compute the disposable token in every request, thus the proposed algorithm is more time-consuming.

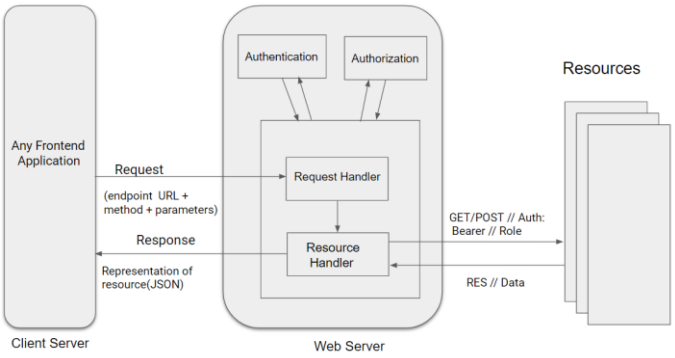
2.2. Applying representational state transfer (REST) architecture to archetype-based electronic health record systems.

According to research Erik Sundvall, Mikael Nyström , Daniel Karlsson1, Martin Eneling , Rong Chen, and Håkan Örman. The benefits of REST Electronic health record is it allows 24-hour access is provided to the key patients with history and current medication histories. Secured health records. Patient information can be accessed from rapid and remote access. Chronic diseases management is easier.

2.3. API REST Web service and backend system Of Lecturer's Assessment Information System on Politeknik Negeri Bali

According to research conducted by I B P Manuaba, E Rudiastini, entitled by "API REST Web service and backend system Of Lecturer's Assessment Information System on Politeknik Negeri Bali" has lack of standard support for security policy, reliable messaging, etc.

3. **System Architecture diagram:**



The system consists of three core components namely, client Server, web Server and Resources. The client server issues a request to the web server. The authentication and authorization of the user is checked who has made the request.For authentication bearer tokens are used and for authorization role based authorization is used. The request handler will then pass the request to the resource handler. The resource handler will fetch the data from the resources. The resources consist of a file system, Database. The fetched data will be then sent to the client server. The response will be in the JSON format.

REST API provides a great deal of flexibility. The data is not associated with resources or methods, so REST can handle multiple types of calls, return different data formats and even change hierarchically with the correct implementation of hypermedia. This flexibility allows developers to create API that meets our needs while catering to the needs of an extremely diverse clients. REST API operates on the concept that the client and the server should be separate from each other and allowed to evolve individually. The key to separating the client from the server is to have a unified interface that allows for independent development of the application without the application services, or forms and procedures, being tightly coupled to the API layer itself. Whenever a client issues a request to the web server, each call can be made independently of one another, and each call contains all of the data necessary to complete itself successfully.

The client's request is received by the request handler of the web server.

The request handler tests for client authenticity first. There are many common methods for performing user authentication, but we used token-based authentication in our system. When compared to conventional approaches such as cookies, using tokens has many advantages. Tokens are stateless, which means they are self-contained and contain all of the data needed for authentication. This improves scalability by removing the need for the server to store the session state. Tokens can be created from anywhere. Token generation and token verification are separated, allowing us to handle token signing on a separate server.

Next step is to check whether the user is authorized to execute the received request. In our system authorization is implemented using a role based authorization method. Role based authorization is a scalable method in which each user role has a predefined set of actions that it can perform. These set of actions can be altered as per system requirements. The system consists of a root user role admin which has complete access to the system, besides admin there are primary roles which are hospital, doctor and patient. If the user is authorized to execute the requested action then the request is forwarded to the resource handler module else an error message is sent to the client.

The request is forwarded to the resource handler after the client has been successfully authenticated and approved. The requested data is retrieved from the resources by the resource handler. Now, data in a health record management system may be of various forms, such as the user's general information, patient case descriptions, scan results, and so on. As a result, file systems and databases are used as resources to keep track of the various record formats. Finally, this fetched data will be sent as a response to the client. Unlike SOAP, REST is not limited to XML; instead, depending on the client's request, it can return XML, JSON, YAML, or some other format.

4. Methodology:

This study discussed the use of RESTful API. REST is not a standard but a software architecture design pattern. REST is a practical approach to web application development where systems in development need to be improved or need simple ways to interact with independent systems. REST is stateless and data-oriented, everything in the REST architecture is data.

Each request is independent, the server does not store any request status. An Application Programming Interface (API) that follows the REST Style is called a RESTful API. RESTful API uses a Uniform Resource Identifier (URI) to represent data. For operations on data, the GET method is used to obtain data, The POST method is used to create new data, The PUT method is used to update data with the resource id, and the DELETE method is used to delete data or data sets

HTTP method	Description
GET	Get a representation of the target resource's state.
POST	Let the target resource process the representation enclosed in the request.

PUT	**Set the target resource's state to the state defined by the representation enclosed in the request.**
DELETE	**Delete the target resource's state.**

5. Programming Tools:

Sr.	Description	Specification
1.	Language	JAVA
2.	Framework	Spring Boot
3.	Server	Tomcat
4.	Database	MySQL
5.	Tools	intelliJ, Postman

HTTP methods are used to store and retrieve the data from the database through REST architecture. With help of the POST method, we are creating a new user in our database as shown in the following table.

URI:	*http://localhost:8080/add/user*
Method:	POST

Body (JSON Object):	{
	'Username': 'Akshay'
	}

Response	Akshay added.

GET method is used the retrieve the details of a user. get/user_name using this mapping user details will retrieve from the database. For security In this study, bearer tokens are used for authentication as shown in the following table.

Method:	GET
Token:	wqLB_FpCiqIsJdN28I7mTuTblapnes1tg5U VrvSepl8lxr-COUPmMmkVzy5nRjqAnvPw
URI:	***http://localhost:8080/get/akshay***

Response (JSON):	{
	"username": "akshay",
	"password":"$2a$10$Ui.PxcumMcRfS",
	"enabled": true,
	"accountNonExpired": true,
	"accountNonLocked": true,
	"credentialsNonExpired": true,
	"roles": [],
	"profile":
	{
	"firstName": "Akshay",
	"lastName": "Bhoi",
	"dob": "1999-09-09",
	"address": "someBuilding",
	"city": "kalwa",
	"phone": "99999-99999",
	"age": 21,
	"image": null
	},
	"doctor": null, "patient": null,
	"patientCases": [],
	"doctorAppointments": []
	}

6. Conclusion:

REST allows a greater variety of data formats, whereas SOAP only allows XML. Coupled with JSON (which typically works better with data and offers faster parsing), REST is generally considered easier to work with. REST provides superior performance, particularly through caching for information that's not altered and not dynamic. REST is generally faster and uses less bandwidth. It's also easier to integrate with existing websites with no need to refactor site infrastructure.

7. References:

1) Xiang-Wen Huang, Chin-Yun Hsieh, Cheng Hao Wu and Yu Chin Cheng, "A Token-Based User Authentication Mechanism for Data Exchange in RESTful API," 2015 18th International Conference on Network-Based Information Systems, 2015, pp. 601-606, doi: 10.1109/NBiS.2015.89.

2) Sundvall, E., Nyström, M., Karlsson, D. et al. Applying representational state transfer (REST) architecture to archetype-based electronic health record systems. BMC Med Inform Decis Mak 13, 57 (2013). https://doi.org/10.1186/1472-6947-13-57.

3) Manuaba, I & Rudiastini, E. (2018). API REST Web service and backend system Of Lecturer's Assessment Information System on Politeknik Negeri Bali. Journal of Physics: Conference Series. 953. 012069. 10.1088/1742-6596/953/1/012069.