International Journal of Computer Engineering in Research Trends
Multidisciplinary, Open Access, Peer-Reviewed and fully refereed
Research Paper Volume-8, Issue-12, 2021 Regular Edition E-ISSN: 2349-7084

Serverless Web Application on the Cloud framework

[1] Mereddy Samhitha,[2] Preetham Paul Bapuram, [3]Mittapally Vineeth Kumar

[1,] B.Tech Student, Department of CSE,Vidya Jyothi Institute of Technology, Hyderabad,Telangana, India.
[2,3]B.Tech Student, Department of CSE, CVR College Of Engineering, Rangareddy Dist, Telangana,India.
**Corresponding Authors Email: samhitha.mereddyx@gmail.com, bpreethampaul369@gmail.com,*
mittapallyvineethkumar444@gmail.com

Available online at: http://www.ijcert.org

Received: 16/12/2021, Revised: 22/12/2021, Accepted: 25/12/2021, Published: 31/12/2021

Abstract: - During the COVID-19 pandemic, convalescent plasma donors are dearly needed. Most of the recovered patients are not eligible to donate plasma, and from the minimal number of suitable donors, many are not coming forward to donate the plasma. The donors' count is already meager, and when someone needs plasma urgently, it has become challenging to find a donor. People are using social media to circulate their requests for plasma. Then they might find a donor, and it would be already late by that time. This process is inefficient and very much time taking. There is a chance that the plasma request could not reach the donors who are willing to donate plasma and help save a life. Instead, suppose there are platforms wherein the donors who are willing to donate plasma can register, and the needy can register their request when a request for plasma comes in. In that case, the eligible donors from the already existing pool of donors shall be presented to the requester. In this way, the donor search process can be improvised. Such a platform would end the pain in finding a donor and reduce the search time in finding a potential donor. And such a platform, built using serverless technology, will have more outstanding technical capabilities.

Keywords: Amazon Web Services, Application Programming Interface, Covid-19,Plasma

---

1. Introduction

The world today is battling COVID-19. There is neither a full-fledged vaccine, to prevent the disease, nor any medicine, to cure the disease, yet. Doctors have resorted to the traditional Convalescent plasma therapy to treat the COVID-19 patients. Convalescent plasma therapy may help people recover from COVID-19[1]. As a result, the demand of plasma is rapidly increasing. Every day, we come across many social media posts, stories, messages requesting for plasma. But most of the times, it would be too late before someone comes forward to donate. Instead of that, if there is a platform with ready donors and a request for plasma comes in, in a very short time the plasma can be donated which would save one's life. By using AWS and the cloud-based infrastructure such as BDTI, initiatives like the EU CCP platform are able to experiment with a testing environment and adopt big data technologies and data analytics skills to process data, promote open source technologies, and advance research on convalescent plasma therapy. This Server less Web Application on the Cloud Platform was developed with the least amount of operational overhead, all while including the necessary foundational elements such as security, flexibility, agility, and extensibility.

As the Convalescent Plasma Therapy is in the protocol for curing the COVID-19, the demand for plasma, of the recovered patients, has increased. The count of the voluntary plasma donors is already less and in addition to that finding a right donor under a very short time, which is very crucial, has become very difficult for the families of covid patients. So, there is the need for a platform which connects COVID patients with the plasma donors and helps them in finding a donor in the minimum time.

When used to build and deploy application to serverless platforms, functional programming delivers the following benefits:

Composability: Delivers compostable functions via a consistent, secure, web-native API that can be called from any client application and on a pay-as-you-go basis;

Accessibility: Enables applications to be easily served, composed, and consumed on-demand from every piece of computing infrastructure anywhere;

Speed: Allows developers to deploy low-latency functions and deploy them quickly and scalably across cloud-to-edge environments;

Robustness: Ensures that application performance won't degrade even as the underlying business logic is distributed far and wide;

Simplicity: Provides an abstraction for simplified development of highly scalable applications thereby sparing developers from needing to write the logic that manages containers, virtual machines, and other back-end runtime engines to which execution of microservices will be dynamically allocated;

Agility: Reduces most dependencies on underlying infrastructure and provides a polyglot development platform;

Efficiency: Boosts the density and efficiency of cloud CPU, memory, storage, and other resource usages;

Consistency: Delivers a consistently dynamic, interactive application experience from any smartphone or edge device, no matter where a user happens to be, or where the resources they're accessing are being served from.

Contribution of this paper is

- To build an application which helps COVID-19 patients to find a plasma donor within short period of time?
- To build a serverless web application making use of AWS serverless services, namely AWS Amplify, Amazon Cognito, Amazon API Gateway, AWS Lambda, Amazon DynamoDB.

2. Literature Review

We did a review of the literature on cloud computing and the development of web applications employing cloud computing technologies. As a group, we've read a number of papers on cloud computing and how cloud services may be utilised to develop web-based apps.

Inferences from Literature

Cloud Computing is a new technological development that can have an incredible effect on the world. It has numerous advantages that it gives to its clients and organizations. For example, a portion of the advantages that it gives to organizations is that it diminishes operating cost by spending less on maintenance and software upgrades and focuses more on the businesses itself. But there are many more challenges the cloud computing must overcome. Individuals are exceptionally incredulous about whether their information is secure and private. There are no guidelines or regulations around the world that provided data through cloud computing. Europe has data protection laws yet the US, being one of the most technologically advanced countries, does not have any data protection laws. Clients moreover stress over who can reveal their information and have responsibility for information. But once, there are measures and guidelines around the world, cloud computing will revolutionize the future.

The European Commission, with three of its Directorates-General (DG) (Health and Food Safety (DG SANTE), Informatics (DG DIGIT), and Communications Networks, Content and Technology (DG CONNECT)) in collaboration with the European Blood Alliance (EBA) and the European Centre for Disease Prevention and Control (ECDC), created the European Union (EU) COVID-19 Convalescent Plasma (CCP) Platform. This database is based on the concept of passive immunization (an approach promoted by the World Health Organization (WHO) Blood Regulators Network), which tests the potential of plasma collected from convalescent persons to treat or prevent viruses and diseases such as COVID-19.

The EU CCP Platform has a web interface where blood establishments can enter anonymous COVID-19 convalescent plasma donations and patient outcomes (of transfusion). The data is then analyzed and visualized using publicly available interactive dashboards. The goal is to gauge the safety and effectiveness of convalescent plasma therapy in treating the virus. Blood establishments will previously have had to register using EUSurvey. The initiative solicits participation from blood establishments and through them, clinicians, hospitals, and individual donors. The European Blood Alliance analyzes the collected data and general findings will be made available on the website for researchers to use in advancing studies on the effectiveness of convalescent plasma therapy in the treatment of COVID-19. Get more information on how to participate in this study.

The Big Data Test Infrastructure (BDTI)[7], part of the Connecting Europe Facility programme, promotes big data to support the public administration's initiatives in the EU and interconnect Europe's national digital landscapes through digital infrastructure. BDTI hosts the EU CCP Platform and provides a ready-to-use, virtual environment to gather and analyze data and the wider monitored use of the experimental therapy. The EU CCP Platform was built on the open source technology stack offered by BDTI and

complemented by Amazon Web Services (AWS) technologies.

The EU CCP Platform consists of four components: 1) information on participating blood establishments and their working protocols, 2) information on CCP donations, 3) information on clinical protocols that are being followed across the EU—whether highly structured clinical trials or broader monitored use, and 4) recipient outcomes. All participating establishments can access the EU CCP Platform.

By using AWS and the cloud-based infrastructure such as BDTI, initiatives like the EU CCP platform are able to experiment with a testing environment and adopt big data technologies and data analytics skills to process data, promote open source technologies, and advance research on convalescent plasma therapy. With the help of AWS Enterprise Support, this project received support from AWS technical account managers, solution architects and subject matter experts during the planning and implementation phase of the EU CCP platform..

3. System Study

We have developed a simple platform which connects plasma donors with the needy. Donors and Recipients has to first register with their details, then the registered recipients would be able to find donors, with the same blood group and from the same city as the recipient is in. After the recipient finds a suitable donor, the details of one will be shared with the other. The only aim of this application is to reduce the search time for donors. This web application is completely serverless. It has been developed, making use of the Amazon Web Services (AWS). The services we made use of include – AWS Amplify, Amazon Cognito, Amazon API Gateway, AWS Lambda, Amazon DynamoDB. Since the application is serverless, scalability, high availability of resource, high performance are built into the application by default.

System Architecture:

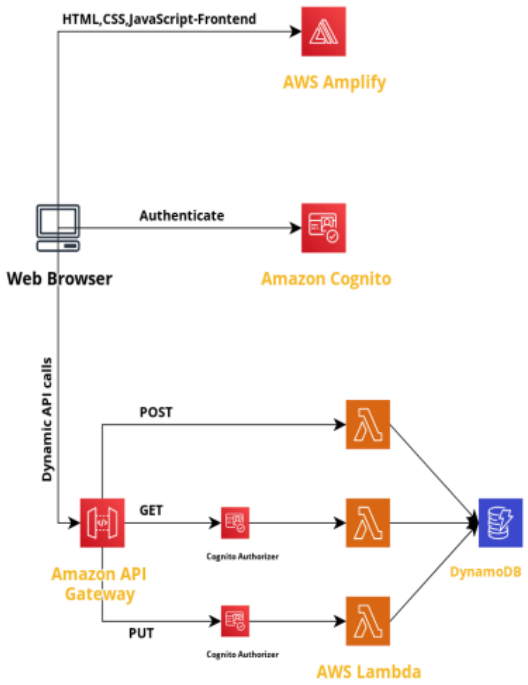

Figure 1. System Architecture

4. Methodology

4.1 Components of the Architecture:

AWS Amplify: The AWS Amplify Console is the control center for full stack web and mobile application deployments in AWS. Amplify Console provides two main services, hosting and the Admin UI. Amplify Console hosting provides a git-based workflow for hosting full stack serverless web apps with continuous deployment. The Admin UI is a visual interface for frontend web and mobile developers to create and manage app backends outside the AWS Management Console. Amplify supports popular web frameworks including JavaScript, React, Angular, Vue, Next.js, and mobile platforms including Android, iOS, React Native, Ionic, Flutter. Amplify has support for CI/CD.

Amazon Cognito : Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Users can sign in directly with a username and password, or through a third party such as Facebook, Amazon, Google or Apple. The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services.

Amazon API Gateway: Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale.

APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services. Using API Gateway, you can create RESTful APIs and WebSocket APIs that enable real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications. API Gateway handles all the tasks involved in accepting and processing up to hundreds of thousands of concurrent API calls, including traffic management, CORS support, authorization and access control, throttling, monitoring, and API version management.

AWS Lambda : Lambda is a compute service that lets you run code without provisioning or managing servers. Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. With Lambda, you can run code for virtually any type of application or backend service. All you need to do is supply your code in one of the languages that Lambda support. You organize your code into Lambda Functions[4]. Lambda runs your function only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time that you consume—there is no charge when your code is not running. You can invoke your Lambda functions using the Lambda API, or Lambda can run your functions in response to events from other AWS services.

Amazon DynamoDB : Amazon DynamoDB is a fully managed NoSQL database service that provides fast and predictable performance with seamless scalability. DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling. DynamoDB also offers encryption at rest, which eliminates the operational burden and complexity involved in protecting sensitive data. With DynamoDB[6], you can create database tables that can store and retrieve any amount of data and serve any level of request traffic. You can scale up or scale down your tables' throughput capacity without downtime or performance degradation. You can use the AWS Management Console to monitor resource utilization and performance metrics. DynamoDB provides ondemand backup capability. It allows you to create full backups of your tables for long-term retention and archival for regulatory compliance needs.

React: React is a declarative, efficient, and flexible JavaScript library for building user interfaces. It lets you compose complex UIs from small and isolated pieces of code called "components". React makes it painless to create interactive UIs. Design simple views for each state in your application and React will efficiently update and render just the right components when your data changes. Declarative views make your code more predictable and easier to debug.

4.2 Working Principle of the Components:

DynamoDB Table: serves as the persistence layer of the application. One single table is used to store the information of the users of the application. DynamoDB, being a No SQL database, there is no need of designing a schema for the table. Though, schema-less it is mandatory to provide a primary key (partition key) for the table. The DynamoDB table is set up using user id, of String data type, as partition key and with other settings, such as provisioned capacity set to 5 reads and 5 writes that AWS provides by default under free tier. The DynamoDB table is created using the AWS Management Console.

Lambda Functions: Lambda Functions perform the business logic of the application. In total, 4 lambda functions were written, each performing a unique task. Three of them interact with the database and other lambda function is used to auto confirm the user upon signup. Lambda functions were written in Java using the AWS SDK for Java. The lambda[4] functions were written in the local machine using the Eclipse IDE. Once thoroughly tested, they were deployed to the cloud using the AWS CLI. Once the lambdas are written, each lambda is packaged into an executable jar file and that jar file is deployed to the AWS through AWS CLI.

Amazon API Gateway: REST APIs – collection of resources and methods – are built at the Amazon API Gateway. The API is integrated with Lambda functions in the backend. To ensure that only authenticated users have access to the backend resources, Cognito User Pool is configured as the authorizer.

Amazon Cognito User Pool: Amazon Cognito User Pool is setup. Amazon Cognito handles complete user management and user authentication. After the user pool has been created, an App client is added to the user pool. Amazon Cognito SDK for JavaScript is used in the front end react application to write the code that performs user management tasks – sign in, sign up, sign out, etc.

AWS Amplify: Once the front end and back end are ready, and after the integration has been done, the application is deployed at the AWS Amplify Console. Initially, the git repository containing the source code is connected to the Amplify and then build settings are configured then the Amplify deploys the application directly from the Amplify console to a globally available CDN.

React Components: In react all required components are building and they are reused. Each component was having its own CSS file for styling. All requests were made using fetch API, and the data fetched is stored into state of the components. React Routing was used where ever it was required and rendered the appropriate components.

4.3 Flow model:

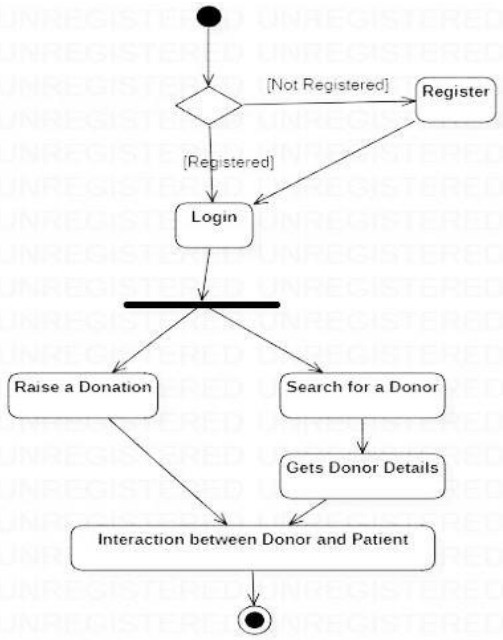

Figure 2. Flow model

4.4 Test cases:

Test Cases: Overall project comprising into 4 cases such as

Case 1: Submitting the registration form without completely filling it. i.e User will not be able to submit the form without filling it completely. User will be prompted to fill the missing details.

Case 2: Not allowing same username (email) i.e. If user tries to register with already existing username, console error will be logged and user will not be able to proceed further.

Case 3: User should not be able to login with bad credentials i.e If any unwanted user tries to login into the application, then they will not be allowed into the website until he/she exists the correct username and password.

Case 4: When a recipient logs in, only the matching donors (i.e., donors with same blood group and in the same city as the recipient) should be presented to the recipient only matching donors are presented.

Case 5: Once a recipient confirms a donation with a donor. From then on only that specific donor's information should be presented

5. Conclusion

When there is an urgent need for plasma, it is very crucial to find a potential donor in the very short time. This application will help in such a case. And the application being server less has other advantages too. Coming to the application development process, traditionally, application development involves various processes such as developing an application, hosting using a third-party mechanism, developing and maintaining whole server architecture. But cloud-ready architecture provides all solutions in one go. Cloud ready application is developed as a distributed system that uses loosely coupled components, is designed to be horizontally scalable, and run on an automated and elastic platform. Ideally, it should be possible to migrate these applications between various cloud platforms without service interruptions. As all the services used in developing the application are completely server less and fully managed by AWS, the application is scalable, there is always high availability of the resources, the application is secure and robust .Future work However, there is scope of incorporating many more features into the project. Some of them are –

- Integrating with hospital databases to get the information of people who are recovered from COVID.
- Collaborating with existing plasma banks.
- Adding notification services to notify donors, in case of urgent plasma needs, as well as recipients immediately after finding a donor.
- Extending the services of application for all plasma donation needs, not just confined to serving COVID patients.
- A sophisticated technique for one-to-one donor-recipient matching must be developed.
- Full potential of the cloud services has to be used to develop a high-performance application

References

[1] https://www.researchgate.net/publication/312040779_Building_Web_Application_Using_Cloud Computing
[2] https://en.wikipedia.org/wiki/Amazon_Web_Services.
[3] https://aws.amazon.com/lambda/web-apps
[4] https://docs.aws.amazon.com/lambda
[5] https://docs.aws.amazon.com/apigateway/latest/developerguide
[6] https://docs.aws.amazon.com/amazondynamodb/latest/APIReference.
[7] https://ec.europa.eu/cefdigital/wiki/display/CEFDIGITAL/big+data+test+infrastructure

