CSU
Channel Islands
CALIFORNIA STATE UNIVERSITY

Refactoring a Web Application using Microservices

A Thesis Presented to

The Faculty of the Computer Science Department

In (Partial) Fulfillment

of the Requirements for the Degree

Masters of Science in Computer Science

by

Student Name:
Monica Tandel

Advisor:
Dr. Michael Soltys

Month Year

APPROVED FOR MS IN COMPUTER SCIENCE

[illegible] 08/09/2021

Advisor: Michael Soltys Date

[illegible] 08/09/2021

Bahareh Abbasi Date

[illegible] 08/09/2021

Eric Kaltman Date

APPROVED FOR THE UNIVERSITY

[illegible]
Jill Leafstedt (Aug 27, 2021 12:52 PDT) 08/27/2021

Interim Dean Dr. Jill Leafstedt Date

Refactoring a Web application using Microservices
Title of Item

Microservices, AWS Lambda, Refactoring Application, AWS EC2, AWS RDS
3 to 5 keywords or phrases to describe the item

Monica Tandel
Author(s) Name (Print)

[illegible] 08/25/2021
Author(s) Signature Date

Refactoring a Web Application using Microservices

Monica Tandel

August 3, 2021

Abstract

The novel contributions in this thesis are - strategizing an efficient way to refactor microservices using business functionalities; and the challenges and migration-related issues faced during refactoring. Another element is to provide a comparison between microservices and monolithic architectures and if it is beneficial to refactor the application in a microservices architecture. Refactoring code from a monolithic architecture to microservices is a challenging process and there is no particular way for carrying refactorization. This thesis focuses on building a strategic approach that will be beneficial in terms of scaling, advancing, maintaining, and evolving the monolithic application into microservices. With the help of this strategy, migration from monolithic to microservices will be easier. Considering the above strategy, the Four Temperaments application was refactored using AWS Lambda and Amazon API Gateway, and Amazon RDS as the database.

Contents

List of Figures

List of Tables

1 Introduction

Four Temperaments is a proto-psychological theory that is incorporated into 4 fluids namely blood, yellow bile, phlegm, and black bile [5]. A Greek physician, Galen referred to these fluids in terms of senses of humor as either "sanguine", "Choleric", "melancholic" and "phlegmatic" [6]. The Four Temperament test works as a pair of temperaments usually dominates the other two and for an ideal temperament, all four senses of humor are balanced. This temperament questionnaire can be used for both fun and to understand one's relationships. Once the testing user has submitted the questionnaire with their responses, their personality is mapped to either of the 4 personalities.

The implemented Four Temperament Test is developed from scratch and not using an online template. The system is deployed on Amazon Elastic Compute Cloud (AWS EC2), a service that provides secure and resizable compute capacity in the cloud and MariaDB is the back-end of the system. This questionnaire is accessible to the users who want to register themselves to the system or can be filled as an anonymous user, i.e. in guest mode. Several Root Admin/Admin features will be discussed later. The User and Admin passwords are encrypted using the AES algorithm. This architecture is a monolithic architecture, where the entire implementation is deployed on an AWS EC2 instance. For modularity and ease of maintenance, this application is refactored into ‘Microservices’.

Refactoring code from a monolithic architecture to microservices is a challenging process. For performing the migration from monolithic to microservices, the thesis focuses on building a strategic approach. This approach involves analyzing the code functions that are developed in a monolithic architecture. In this, the larger chunks of code were broken into smaller sub-functions. The 2nd step is to search for the business capabilities in the code. The business capabilities are defined as the series of calls to the functions. The reason for using the business capabilities as a way to refactor to microservices is that the business capabilities remain stable. The requirements in each business capability may change but the overall architecture stays the same. The next step is to identify and extract the candidates from the monolithic applications. This means considering those sections of the code that are essential and creating functions for those which are repetitive. The 4th step is to analyze and assign the business requirements. In this step, the number of microservices for the application is decided based on the usage of the business functionalities. For example, the Events functionality in Four temperaments can be assigned a single microservice; Admins can be assigned a single microservice; and Users, and their Responses can be assigned a single microservice. The last step is performing all the steps practically i.e., creating the microservices. With the help of the strategic way of refactoring microservices, the system can be beneficial in terms of scaling, advancing, maintaining, and evolving.

Considering the above strategy, the Four Temperaments application was refactored using AWS Lambda and Amazon API Gateway; and Amazon RDS as the database. Some of the problems that were faced during refactorization of the application were trying to decide for which technology to use, if the framework of the current monolithic application will be a better option for microservices, structuring and planning the migration to microservices completely, and the expenses for the tools that will be used. To solve these problems, AWS Lambda is chosen as the serverless tool, Node.js language was chosen for coding the application instead of JSP and 2 Lambas were created - One for frontend and other for backend.

Another contribution for this thesis is securing the authentication of the users of the Four Temperament system using the MAC Address along with "username" and "password". This can be enabled/disabled by the Admins. The authentication process identifies the request that the user makes for accessing the system. This process considers login credentials. The login credentials most commonly include a username and a text password. Some of the other authentication technologies include biometrics and third–party applications. Apart from password-based authentication, there are multi-factor authentication, certificate-based authentication, biometric authentication, and token-based authentication. Authentication techniques are always changing and it is required to consider authentication for enhancing the user experience since cybercriminals improve their ways of attacking the systems. With the current authentication methods, the user's passwords can be vulnerable to cyber attacks like shoulder-surf, key-log, relay, eavesdrop, brute-force, and getting phished [20]. In this thesis, one of the ways to improve the security of the authentication process is carried out by securely preserving the MAC address of the user's device for the implemented Four Temperament Personality test. If the user credentials are stolen or known to unauthorized parties, they won't be able to use the credentials without the authorized user's Media Access Control (MAC) address. To login to the Four Temperament test, the user is required to use the device with which they have registered [13]. The reason for using MAC address was that MAC address varies on each device, i.e. MAC address is unique and has the potential for being an authentication parameter. The MAC address authentication is carried out on Four Temperament Personality Test.

2 Background

This section focuses on the background of monolithic architecture, microservices, and the problems and challenges that are faced while refactoring the system. It gives an insight into the related study in terms of refactorization of microservices. This section also provides information about the multi-tier architecture in the system, the issues of authentication and authorization, the issues of secure software engineering, the technologies, programming language, and the database used for the proposed system.

2.1 Related Work

In recent years, there have been a lot of companies that are wanting to or have already been successful in refactoring the monolithic application into microservices. Companies like Comcast Cable, Uber, Netflix, Amazon are an example having successfully refactored the monolith to microservices [24]. Both of the architectures have their equal share of merits and demerits leaving an unanswered question, 'Whether to convert or to keep the application as is'.

There are reasons why refactoring the monolithic architecture to microservices is challenging. Choosing which tools would be suitable for the application, structuring the monolith to work as microservices, maintaining consistency of the databases, migrating the entire monolith, making sure whether the application is working in the same manner as it was before the refactorization was performed; are some of the challenges faced while refactoring. The prior work in this research includes decomposing the monolithic applications using an architectural view. This does not provide information on how the developer is supposed to carry out the migration of their application.

According to Richardson [24], microservices can be decomposed using either of the 4 decomposition strategies; business capability, domain-driven design, verb or use case, and nouns or resources. He also mentions that a way to refactoring microservices is to implement the functionalities as services.

In [17], the authors of the paper considered a French software vendor editing named MGDIS SA. Addressing three questions for migration to microservices, how to determine suitable granularity, appropriate deployment, and efficient orchestration. They proposed that the choice of granularity must balance between the costs of Quality Assurance and that of deployment. They used the Docker tool to containerize the application for deployment. For orchestration, their first choice was Enterprise Service Buses, alternative to that was Business Process Management and they finally chose to use webhooks which helped them increase the performance.

Further in [16], these authors outline the 4 crucial aspects functional approach, norms and standards, microservices granularity and their semantics, and the technical and integration outcomes. Here, they consider an application development consists of 4 layers which are the process, the functional, software, and the hardware. Further, they claim that the splitting into granular APIs is done on the functional layer. The next step in their process was to decide on norms and standards. Their process has tens of business norms and hundreds of technical norms. For microservices granularity, they chose to base the microservices slicing on functional ones[16]. Through their semantics, technical and integration outcomes, it can be said that their process of refactorization contributes to domain-driven design.

In [15], the authors mention the problems, challenges, and benefits of refactoring the monolith to microservices. In this, the authors have taken into consideration about 50 research papers to present the challenges, problems, and solutions. They mention that refactorization of the monolith to microservices can be performed in various ways and there is not a single way to carry it out.

In [18], the authors in this are using a clustering algorithm to extract the candidates which are required for microservices architecture. The strategy that is used in this deployment is domain-driven design.

In [14], the authors mention how refactoring to microservices scalability can be improved. They also point out that microservices are expected to implement business capabilities. The migration process in their paper is business-driven.

In reference to the above work, this thesis uses the business capabilities for restructuring the monolithic application. Similar to [14], this thesis focuses on the business capabilities/functionalities and using the business capabilities as an important element. As mentioned by Chris Richardson, the author of microservices patterns and quoted here, ‘A good starting point is the Monolithic Architecture pattern and a better choice for large/complex applications is the Microservice architecture pattern’ [24]. In this thesis, the starting point for Four Temperaments application is a monolithic architecture pattern. This thesis focuses on refactorizing this monolithic architecture using its business capabilities into a microservices architecture. The implementation of the same is carried out successfully in this thesis.

2.2 Monolithic Architecture

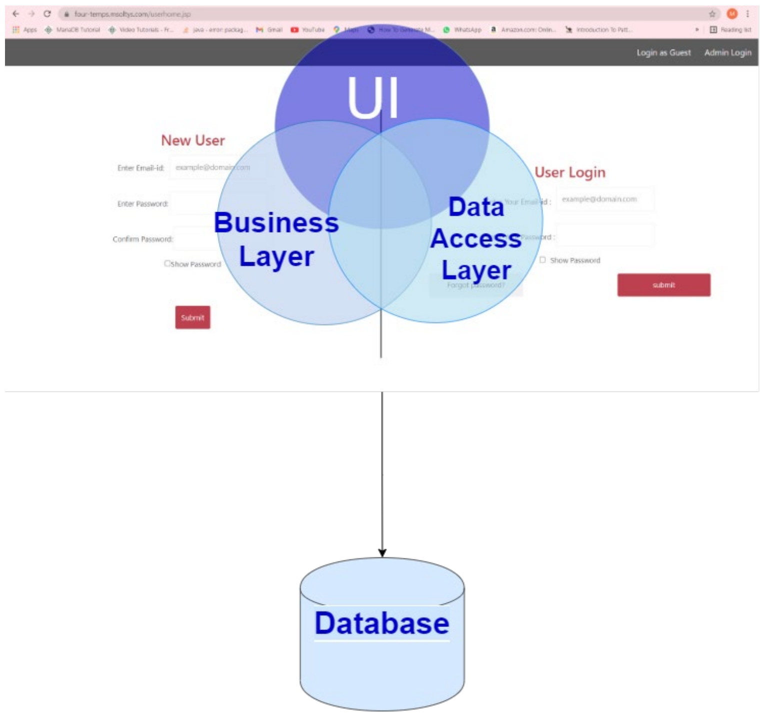


Figure 1: Monolithic Architecture

A monolithic architecture is an architecture that contains individual software components that are coupled in a single package and overall running as a single service. If one component of the application is modified, then the entire application may require rebuilding and deployment. This application can be risky and may lead to a single process failure. Initially, monolithic architecture was used for developing web applications. Although well-known companies like Netflix, Amazon.com, and eBay are now using microservices architecture, all these companies started with a monolith architecture [24].

The Figure 1 is an image of the Four Temperaments system as a monolithic architecture.

If the application is not complicated then this architecture consists of its own strengths such as being easier for the developers to develop, test, and deploy. Considering the growth of the application, the structure of the monolith architecture grows and becomes hard to handle.

The Four Temperaments application deployed on AWS EC2 instance follows the monolithic architecture. The different components of the Four Temperaments website include the user interface, the admin interface, and the events. Its deployment as a monolithic application has all the components in a single package.

2.3 Microservices

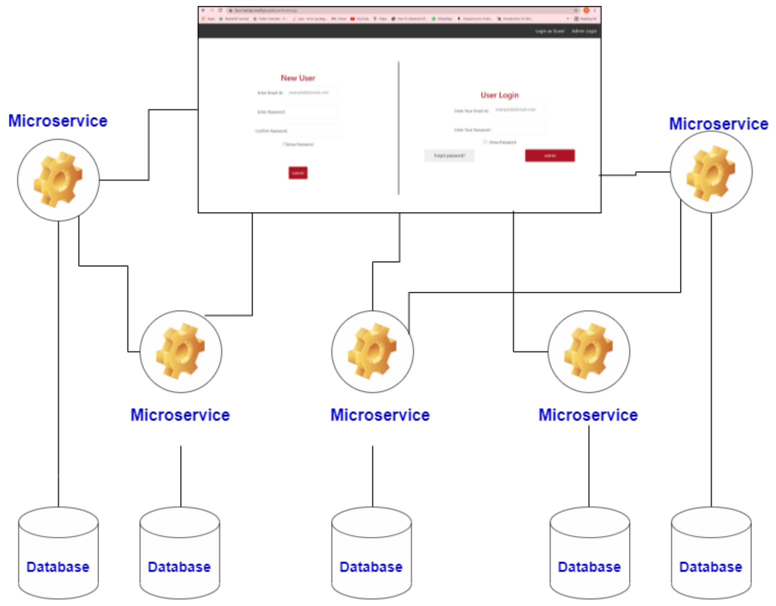


Figure 2: Microservices Architecture

Microservices consist of small, independent, self-contained services that communicate with each other over APIs. Microservices allow the developers to choose any programming language or framework for their software. The developers can select the tools required for each function that the microservice performs. Each microservice is easier and faster to develop, update, scale, and easy to manage separately. The Figure 2 is an image of the Four Temperaments system deployed as microservices architecture.

Advantages of Microservices

1. Reduces Risks and increases fault tolerance and fault isola-

tion:

Microservices are developed, tested, and deployed independently. Since all the services are separate, the risk factor reduces. For example, the remainder of the services will work independently of the service that didn't load. It helps the developers to perform changes or rollback for only the service which did not load and aren't required to deploy the complete application again. This instance explains that loosely coupled microservices are fault-tolerant, unlike the monolithic applications that are closely linked. Locating the error or issues for the function that did not load is easier in microservices.

2. Scalability and flexible Data Storage:

   This architecture provides the flexibility of storing data in multiple locations in contrast to monolithic applications. The flexible data storage approach offers developers the flexibility to decide on the storage type that is best fitted to their services. The services can be independently and horizontally scaled.

3. Reduces Clutter:

   Refactoring the monolithic application to microservices removes the functionalities that are not required. It reduces the code size since the monolithic application tends to be very large.

4. Simplifying security monitoring:

   Each service of the application is isolated. It makes it easier to trace the service that is susceptible to security threats. The isolation of services also prevents jeopardizing the other services from security threats.

Disadvantages of Microservices

1. Complexity:

   In comparison to monolithic architecture, microservices are complicated. The microservices allow developers to program in different languages that is challenging. Each service needs to be tested and monitored. The currently existing microservice tools are not likely to be compatible with the newer dependencies.

2. Expensive:

   The microservices consist of autonomous services. For communicating with the other services, remote calls are made that increase the processing cost. Each of the isolated microservice has its CPU and runtime environment that is, an increase in resources.

2.4 MVC in Four Temperaments:

There are three components in MVC, namely, the model, the view, and the controller. Model - view - controller helps separate the website information that is presented to the user from model and controller information respectively.

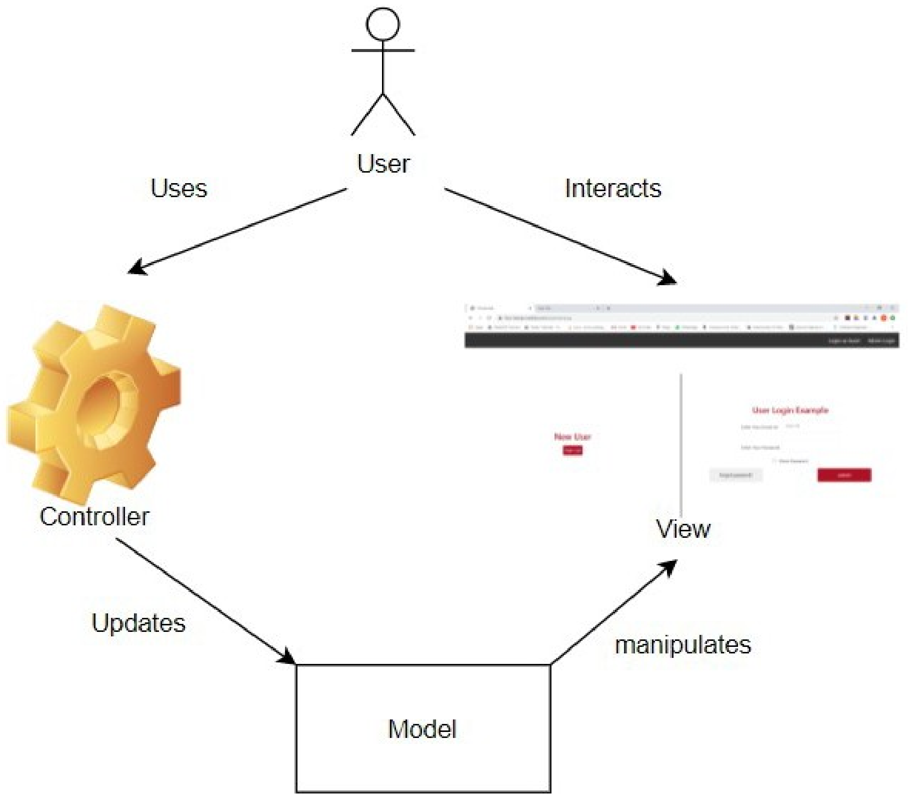


Figure 3: MVC

The model is responsible for managing data and communicating with the database. It contains the logic, data, and the business rules [26]. It renders the information that the user is going to view. It interacts with the user of the system directly and notifies the controller when the user wants to access dynamic information. The controller initiates the interaction between view and model. It gets the input and accordingly converts the information into commands which are required in either model or view. The view receives the data from the model and updates the user interface page.

Considering an example where the user is logging into the Four Temperaments system.

1. The users of the system fill in the username and the passwords and click on submit.
2. On the clicking of submit, a signal is sent to the controller through the handler.
3. From here, a notification is sent to the model that the user is trying to log into the system and the users' credentials are to be verified.
4. Once the verification is done, the model is indirectly used by the view for allowing the user access to the system taking in the username from the model and the system waits for the user to interact with the system.

2.5 The issues of authentication and authorization:

2.5.1 Authentication:

Authentication validates the users based on their user credentials. For authentication, the user needs to prove their identity for accessing the system.

Traditional Authentication Process:

Whenever a user is registering for an account, they are required to create a unique ID and key which permits them to access the system's information

Frequently, the username and password are used as a unique ID and key. For example, for accessing the Four Temperaments Test, User A only has access to see the test and will not be allowed to see the test of User B.

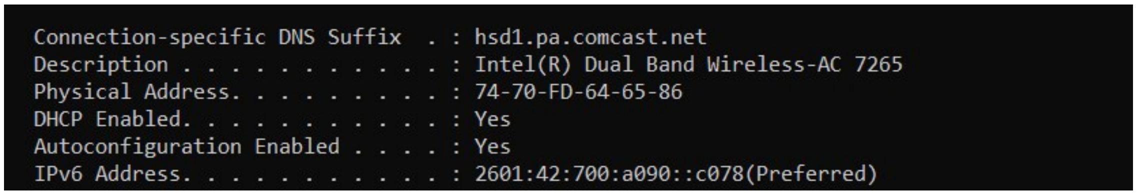
The ID and key are used to confirm the user's identity which allows the system to authorize the user. User authentication has 3 tasks:

1. Connection between the User and the Webpage's server.
2. Verification of the user's identity.
3. Allowing/Disallowing access to the system.

This process requires users to input their login credentials on the login page. The login credentials are sent to the server where information is compared to the user's credential on the Database. If a match is found, the user is authenticated by the server, and access is granted to their account. When a match is not found, then the user is prompted to re-enter the credentials.

However, the traditional user authentication process is not secure as the cybercriminals can easily gain access to the system using Brute-force or Dictionary attacks to get verified user's login credentials.

What is MAC Address?



```
Connection-specific DNS Suffix  . : hsd1.pa.comcast.net
Description . . . . . . . . . . . : Intel(R) Dual Band Wireless-AC 7265
Physical Address. . . . . . . . . : 74-70-FD-64-65-86
DHCP Enabled. . . . . . . . . . . : Yes
Autoconfiguration Enabled . . . . : Yes
IPv6 Address. . . . . . . . . . . : 2601:42:700:a090::c078(Preferred)
```

Figure 4: ipconfig

A MAC Address is primarily assigned to the device by the device manufacturers. Thus, these addresses are also known as a burned-in address, ethernet hardware address, hardware address, or physical address. This address is stored in the hardware section of the computer. A simple way to know the MAC address of one's device is to type ipconfig /all for Windows OS users on command prompt and ifconfig -a for Unix OS users on terminal. Figure 4 is example of ipconfig /all, where Physical address of that device is 74-70-FD-64-65-86.

MAC Address Based User Authentication:

MAC Address Authentication process is more secure than traditional authentication processes. For MAC Address based authentication, the user's login credentials are considered along with the MAC address of the system.

MAC address Based User authentication performs the similar 3 tasks as the traditional authentication process:

1. Connection between the User and the Webpage's server

2. Verification of the user's identity along with the MAC address as a parameter

3. Allowing/Disallowing access to the system

This simple and secure process needs the user's to input their user credentials on the login page. The login credentials are sent to the server where the user's information with MAC address of the device which user is using while logging in is compared to the user's credentials which user had used while registering. If a match is found, the user is provided access to the system else the user is prompted to re-enter the credentials. Since this is MAC address-based authentication, the user is required to use the same device which they had used while registering.

For the Four Temperaments system, the Admins can enable or disable this feature for User login. Also, it allows Users to choose if they want to remember the device which they are using while logging in for the test.

2.5.2 Authorization:

The authorization provides users the permission for accessing a resource [9]. Authorization and authentication processes are usually carried out together providing the server some information about the client that is requesting access to the system. For the Four Temperaments system, authorization of user's registration and login credentials is carried out. For registration of the users, the system verifies if the user is inputting a correct email address. If the user inputs a correct email address, then the textbox is highlighted in green. The system checks if the user is inputting the password as per the requirement of the system. If the user inputs a valid password, then the requirement points are highlighted in green. Similarly, the confirm password textbox is highlighted in green if the password and confirm password are a match.

2.6 Issues of secure software engineering

Secure software does what it is supposed to do and doesn't do anything unexpected [12]. Software engineering defines levels of maturity from 1 to 5 as individual efforts, repeatable, defined, managed, and optimized. Software engineering practices require code to be reproducible, systematic, and predictable. Most common issues threatening software security are:

1. Confidentiality: Software should be protected from unauthorized disclosure of information. It can be protected through access control, privacy, and ethics, password, encryption, and biometrics.

2. Availability: Software should be available to everyone and should be protected from unauthorized withholding of information. Authorized users should be able to access information always. This can be done through a data backup plan, business continuity, or disaster management.

3. Integrity: Software should be protected from unauthorized modification of information. It should be ensured that information is always complete and accurate without unintentional, intentional, or accidental processing methods tamper.

Software security can be enhanced by identifying the following security issues at the inception phase of software development [21].

1. Issues of access control: If the event is available and the user is authorized then the user will be able to access the questionnaire at all times.
2. Accountability issue: User activity is tracked using sessions. Once the session ends, the user will not be able to access the same questionnaire again for that Event. Questionnaire resubmission is restricted.
3. Issues related to accuracy: This software behavior is as intended irrespective of time.
4. Issues related to authorization: The authorization provides the server the information about the client who is requesting access to the system. Only authenticated personnel is allowed to operate on the questionnaire. This is implemented using access control.
5. Issues related to availability: A proper balance is maintained between security and availability. Authorized personnel can operate on the system without being denied access.

6. Issues related to confidentiality: Unauthorized persons should not be allowed access to the questionnaire.

7. Error handling: Errors are classified into User authorization issues, database access issues, data unavailable errors or missing data errors, and bad information errors.

8. Issues related to fortification: Technical details about the data are only exposed to the concerned individual and are hidden from others.

9. Issues related to authentication: Only a legitimate user is provided access to the system. This is implemented using passwords, MAC addresses.

10. Issues related to integrity: It prevents unauthorized users to alter information. This is implemented through sessions.

2.7 Technologies Used:

The technologies that were used for Four Temperaments questionnaire test are: AWS EC2 for deploying the Four Temperaments website, MariaDB was installed on AWS EC2 for the backend of the system, AWS Corretto 8 was installed on the EC2 as JAVA run time environment. AWS Code Commit for collaborating on code and also securely storing source code. The system is coded in HTML, JSP, CSS, and JavaScript. The Tomcat server is used to run the Four Temps system since the system contains Java code and requires a JAVA HTTP web server environment that is provided by Tomcat.

2.7.1 AWS EC2

Amazon Elastic Compute Cloud (AWS EC2) is a service that provides users a secure, resizable compute cloud capacity in the cloud [8]. There are two key concepts which are required for launching an instance, namely:

1. Virtual space that is dedicated to the instance
2. The software which is loaded on the instance

These are controlled by instance type and the Amazon Machine Image (AMI).

The AWS EC2 instance is backed by Amazon EBS root volume. The Availability Zone in which the Amazon EC2 instance is required to be run can be selected. When an EC2 instance is launched, it is secured by specifying a key pair and security group. While connecting to the instance, the private key of the key pair is required [8].

2.7.2 Apache Tomcat Server

Apache Tomcat server software is an open-source implementation that is developed and maintained by Apache Software Foundation. This software acts as a web server providing capabilities like data persistence and load balancing. The Apache Tomcat server software yields a basic feature that processes servlets.

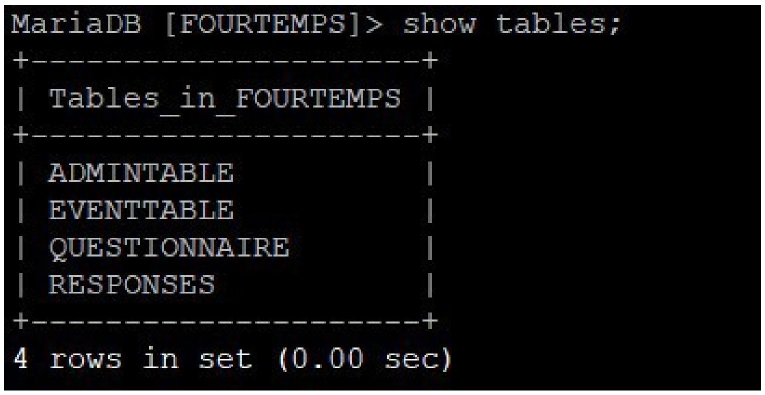
Tomcat is a JSP container and JSP is a server-side view rendering technology [25]. Tomcat handles the routing for the JSP page. Tomcat requires Java Runtime Enterprise Environment(JRE) to run.

Apache Tomcat application server is used to run the Four Temperaments system since it contains Java code and requires a Java HTTP web server environment for running which is provided by Tomcat. For establishing a connection between Tomcat and MySQL, the MySQL Connector/J jar file is required to be placed in the 'Tomcat Install Dir'/common/lib/ folder.

3 System Design

This chapter focuses on the working and authentication of users for the Four Temperaments Test. It gives detailed information about the proposed system, methodology, and design details.

3.1 System Overview



```
MariaDB [FOURTEMPS]> show tables;
+---------------------+
| Tables_in_FOURTEMPS |
+---------------------+
| ADMINTABLE          |
| EVENTTABLE          |
| QUESTIONNAIRE       |
| RESPONSES           |
+---------------------+
4 rows in set (0.00 sec)
```

Figure 5: Database of Four Temperaments System

The Four Temperaments personality site deployed on AWS EC2 has MariaDB as the back-end. Figure 5 displays the tables that are in the FourTemps Database.

Figure 6 depicts the Entity-Relationship Diagram (ERD) for the Four Temperaments system. There are four entities involved namely; Admin Table, Event Table, Questionnaire Table, and the Responses Table.

Relation between the entities:

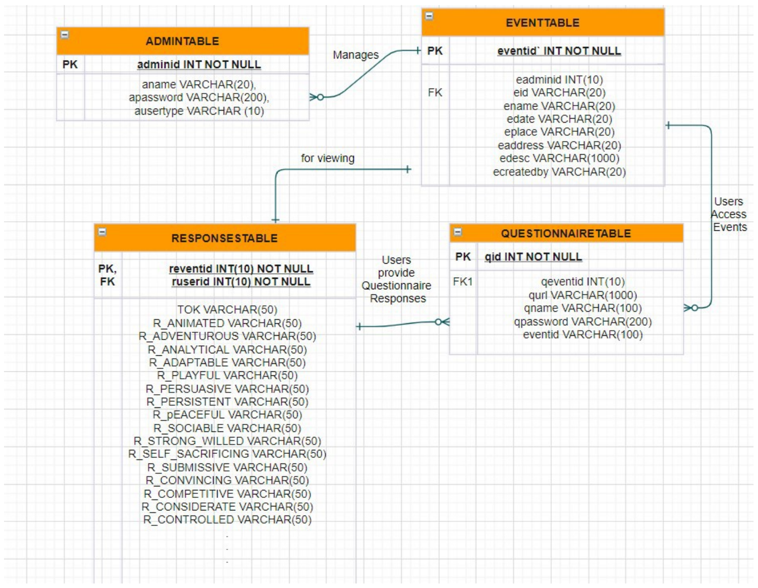


Figure 6: ERD of Four Temperaments System

The Admin Table is keeping a track of the Admins that are involved in the Four Temperaments System. The attributes for this entity are 'adminid', the 'ausername', the 'apassword', and the 'ausertype'. The Event Table is responsible for keeping track of the events that the Admin creates for testing the User personality. The attributes involved for this entity are the 'eventid' which is the primary key, 'eadminid' which is a foreign key from Admin Table, 'eid', 'ename', 'edate', 'eplace', 'edesc', and 'ecreatedby'. The relation between Admin Table and the Event Table is that the Admin manages the Events, which includes creating, deleting, and viewing the Events. The Questionnaire Table is storing the user information. The attributes in this table are ‘qid’, which is the primary key; ‘qeventid’, which is the foreign key from the Event Table; ‘qurl’; ‘qname’; ‘qpassword’, and ‘eventid’. The relation between the Event Table and Questionnaire Table is that the users are accessing the Events for submitting their responses to the personality test. If there is no event created, the users are not able to access the personality test. The Responses Table is storing the responses that the user has submitted for the questionnaire. For this entity, the attributes are, ‘reventid’, ‘ruserid’ which are the composite primary keys and foreign keys from the Event Table and Questionnaire Table; ‘TOK’; and the personality characteristics namely ‘animated’, ‘adventurous’, ‘analytical’, and so on. The relation between the Responses Table and the Questionnaire Table is that the users provide questionnaire responses which are stored in the Responses Table. The relation between the Responses Table and Event Table is for Admin viewing the Event statistics data which also includes the responses that the users had provided for the test.

3.2 Proposed System

The Four Temperaments Test is a personality test. This test is used for fun and to know the personalities of the testing users.

The Four Temperament website consists of 3 main entities: the Root Admin, Admin, and Applicant/Users. The summary of all the running processes is as follows:

1. Root Admin creates Admins to create events for testing temperaments of ‘Users’.
2. The Rood Admin/Admin creates the required event for ‘Users’ and sends the link to Users for completing their tests.
3. After receiving the Event link from the Root Admin/Admin, ‘Users’ can log into the system
4. The ‘Users’ is allowed to log in as a new user or existing user or anonymously as a guest user
5. After logging into the system, the User answers the questionnaire for the test and submits their responses.
6. Based on the User’s data, the Root Admin and Admin can view statistical results for ‘Users’ as well as the events.
7. After the Event is completed, the Root Admin, as well as the Admin, can delete the Event.
8. If the Root Admin wishes to delete the Admin, the Root Admin is capable to do the same.
9. The Root Admin, as well as the Admin, can enable and disable the MAC address authentication for the ‘Users’.

3.3 Methodology

Figure 7 depicts the systematic steps followed in developing the system. The process starts with the Root Admin logging into the Four Temperaments system. The Root Admin is responsible for creating either an Event or an Admin. If the Root Admin creates an Event then the Root Admin sends the Event information to the participants. Root Admins can also Delete an Event and an Admin. If the Root Admin creates an Admin, then the Root Admin sends the Admin information to the respective Admin. This completes the Root Admin process. Considering that the system has Admin logging into the system, then an Admin is allowed to create and delete Events and send Event information to the participants. This completes the process for an Admin. The only difference between an Admin and Root Admin is that the Root Admin creates and deletes Admins whereas Admins can't create/delete Admins.

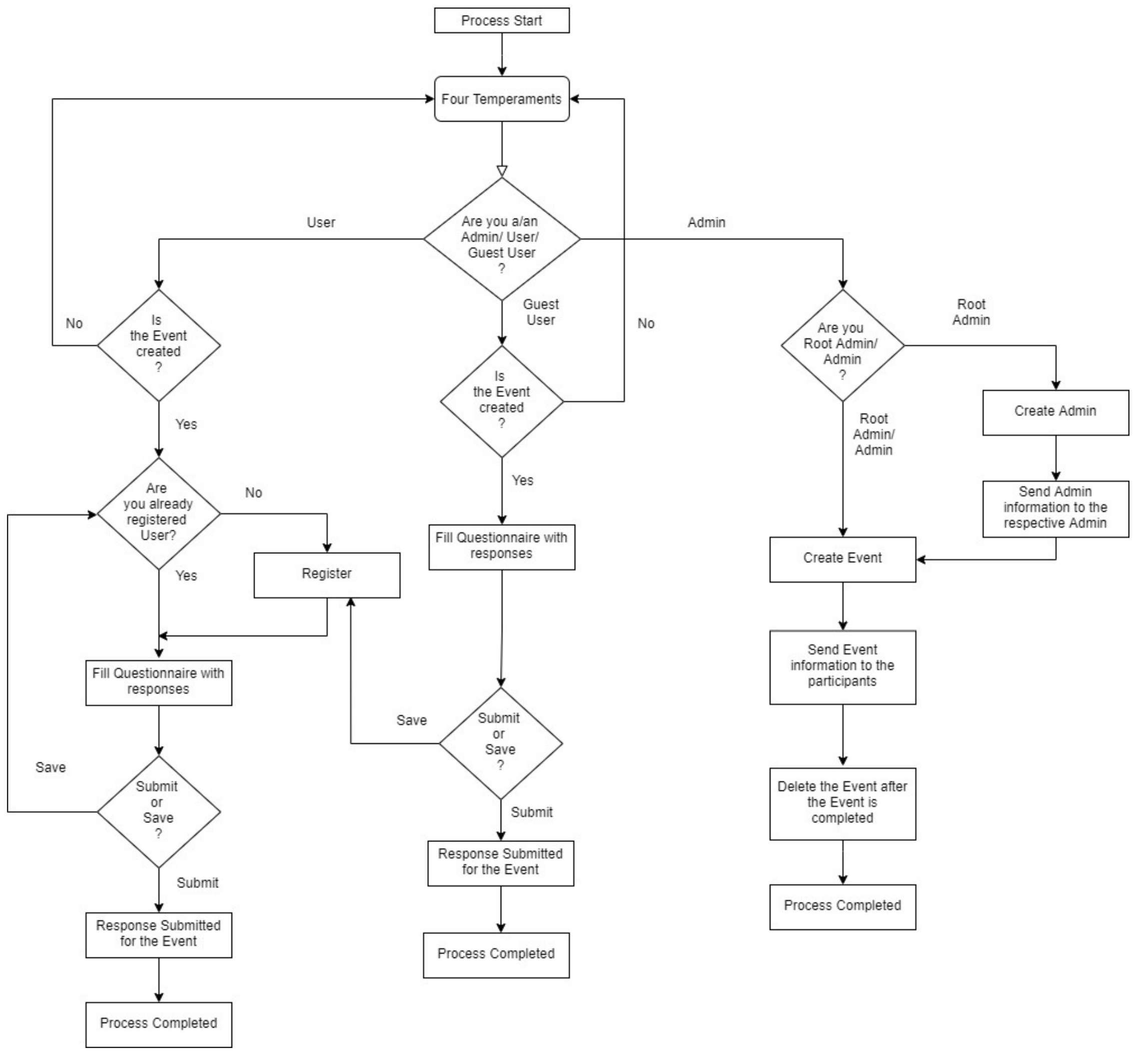

Process Start
Four Temperaments
Are you a/an Admin/ User/ Guest User ?
User
Admin
Guest User
Is the Event created ?
No
Yes
Are you already registered User?
Register
Fill Questionnaire with responses
Submit or Save ?
Save
Submit
Response Submitted for the Event
Process Completed
Are you Root Admin/ Admin ?
Root Admin
Root Admin/ Admin
Create Admin
Send Admin information to the respective Admin
Create Event
Send Event information to the participants
Delete the Event after the Event is completed

Figure 7: Flowchart of the system

Provided that the Admins have created an Event, the User can log into the system. The user can be either a registering user/existing user or an anonymous user. If the user is registering for the first time, the user is required to create their account and then log into the system and fill the questionnaire with their responses. The user can then either save or submit the questionnaire response. If the user chooses to save the questionnaire then the user is allowed to return to the testing website to complete their questionnaire and submit once they have completed responses for all the questions. If the user chooses to submit the questionnaire, the system generates a graph for the users which displays their personality type. This completes the process for registering users. If the user is an existing user then they can directly log into the system if an event is created, either submit the questionnaire or save the questionnaire for later submission and the process is completed for an existing user. Considering that the user is an anonymous user i.e. a guest user logs in to the system, then the guest user can directly fill in the questionnaire with their responses. They can either submit or save the questionnaire responses. If the guest user is saving the questionnaire, then the guest user is required to register themselves to the system. If the guest user is submitting the questionnaire, then the system generates a graph displaying the guest user's personality type and brief information about what their personality type is. This completes the process for guest users.

3.4 Design Details

3.4.1 Entities Involved:

1. Root Admin:

   There is only 1 Root Admin for the system. A Root Admin can create other Admins as well as delete them. The Root Admin is responsible for creating new Events for Users to check their temperaments. The Root Admin can also delete the Event. They are capable to view statistical data of the results received after the Users perform the test for an Event. They are capable to view User statistics as well as Event statistics. The Root Admin can enable/disable the MAC address-based authentication for the Users.

2. Admin:

   Similar to Root Admin, an Admin is also responsible to create and delete Events, view statistical data of the User performance and Event, and enable/disable MAC address-based authentication for the Users. The duties of Root Admin and Admin remain similar except for Root Admin being able to create multiple Admins.

3. User:

   Users have the role of filling out the questionnaire for an Event and submitting the same. They can be anonymous or if they wish to have a username, they are provided the option of the same. They can submit the questionnaire at that time or are allowed to return later to finish their incomplete questionnaire for that Event. If the MAC address-based authentication is enabled for the User, then the MAC of the User's device is also considered as one of the parameters of authentication.

3.4.2 Architecture:

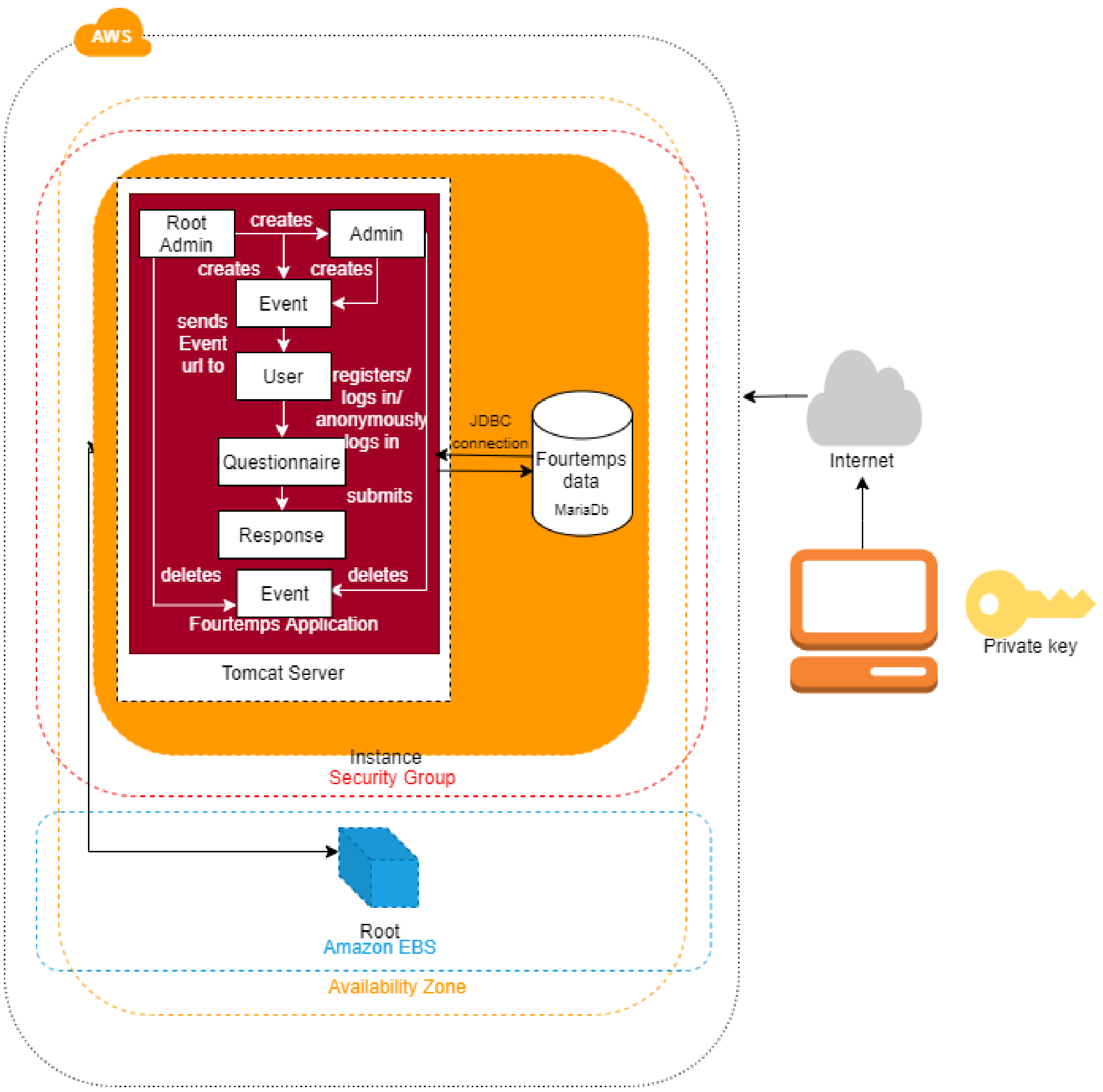


Figure 8: Architecture of the system

Figure 8 displays the architecture of the system. The architecture guides through the process of launching an AWS EC2 instance which is backed by

Amazon EBS. For running AWS EC2 instance backed with EBS, both are required to be in the same Availability Zones. In the launched AWS EC2 instance, Tomcat Server is used with MariaDB as its backend for storing the data for running the Four Temperaments system. The transfer of data from the server-side to the client-side is done using the JDBC connection. The process of the Fourtemps Application starts with the first entity i.e. the Root Admin creating either an Event or an Admin. If the Root Admin creates another Admin, this Admin has access only to create events, send event details to the Applicants, and delete events. Once an Event is created and is sent by email service for the Applicants to fill. On receiving the Email from the Root Admin/Admin, the Applicant can complete the questionnaire and submit it immediately or is given an option to save the questionnaire and return it later to complete it. Once the filled questionnaire is submitted, the Root Admin/Admin can view the Event and User statistics. Code specification for Monolithic Four Temperaments is displayed in the table 1.

Type of Information	Data
Application	Four Temperaments
Programming language	JSP
Number of files	56

Table 1: Code specification for Four Temperaments

4 Microservices:

A microservice application can solve some issues that are present in a monolithic architecture. Microservices are connected through an external Request Handling unified entry points called API Gateways.

4.1 Refactoring Monolithic to Microservices

Monolithic Four Temperaments application was developed using JSP. It is refactored to microservices. The application program is in Node.js - an interpreted language. Sequelize ORM is used for accessing the MySQL database. The application is coded in Node.js as the initial invocation time is faster when compared to using JSP which is compiled language. Also, Node.js can process multiple requests in parallel. Moreover, a Node.js application is single-threaded i.e it processes many requests simultaneously. It also consumes less memory than JSP and is more productive than JSP.

In order to refactor the application to microservices AWS technologies like AWS Lambda, AWS API Gateway, AWS RDS, and AWS Cloud Watch were used. AWS Lambda is used for running individual components of Four Temperaments. The Lambda function used connects to RDS for database connection. It uses Amazon API Gateway for making HTTP requests and Cloud watch for log tracing. AWS Code Commit is used for version control and will be used to rolling back to previous versions if required.

4.2 Migration strategy

The following strategy was implemented on Four Temperaments monolithic application for converting into microservices:

1. Analyzing the functions in Monolithic architecture:
   In this step, the functions in the monolithic architecture are checked. If there are large functions, then those functions can be split into smaller ones. Similarly, if there is an option for merging some functions then it is carried out in this step.

2. Identifying the Business functionalities:
   In this step, the business functionalities of the monolithic application are identified. For the development of the monolithic application, certain business functionalities are followed. These business functionalities are defined as the steps of calls to the functions. Identification of business functionalities is not an easy task and is not always possible. One of the approaches to learning about the business functionalities is asking the developer who had developed the monolith application and others can be using the machine learning process.

   Since Four Temperaments was implemented in monolithic architecture and is now being converted to microservices, its business functionalities are easily available and are as follows:

   (a) The Admin functionalities.

(b) The Event functionalities.

(c) The users using the system and their responses.

3. Identifying and extracting the candidates from the Mono- lithic applications:

   In this step, the unwanted and repeated code is restructured and reduced. Although the entire Four Temperaments application running in monolithic architecture contributes to importance, there are a few elements that can be considered as most important from the Monolithic application.

4. Analyzing and assigning the business functionalities:

   For analyzing the business functionalities, it is required to acquire statistical information such as the usage of the data. This statistical data provides information about which business functionality is used often compared to the others. After getting this information, the single microservice can be assigned to those business functionalities that are frequently used. The smaller business functionalities can be assigned to a single microservice. Since the four temperaments test is still in the development stage and there are no users currently using the system, there are only 2 services created for the system.

5. Converting to Microservices:

   The above steps are carried out practically in this step. The functions that are related to the business functionalities are developed in the microservices. The routes or the endpoints that are required to deploy the application are created and the service is now ready to be deployed and tested. The Table 2 provides the information on reduction of backend code files after using microservices:

Files in Monolithic	Files in Microservices
adminchangepass.jsp	index.js
admindetailsmail.jsp	adminUser.js
adminusercontroljsp.jsp	events.js
aesencryptdecrypt.jsp	eventusers.js
createadminjsp.jsp	questionnaire.js
createeventjsp.jsp	response.js
deleteadjsp.jsp	
deleteeventjsp.jsp	
forgotpasswordad.jsp	
macadd.jsp	
rootadminregjsp.jsp	
checkbox.jsp	
checkboxguest.jsp	
usererror.jsp	
userloginn.jsp	
userreg.jsp	
userrlogout.jsp	

Table 2: The backend files in Microservices v/s Monolithic

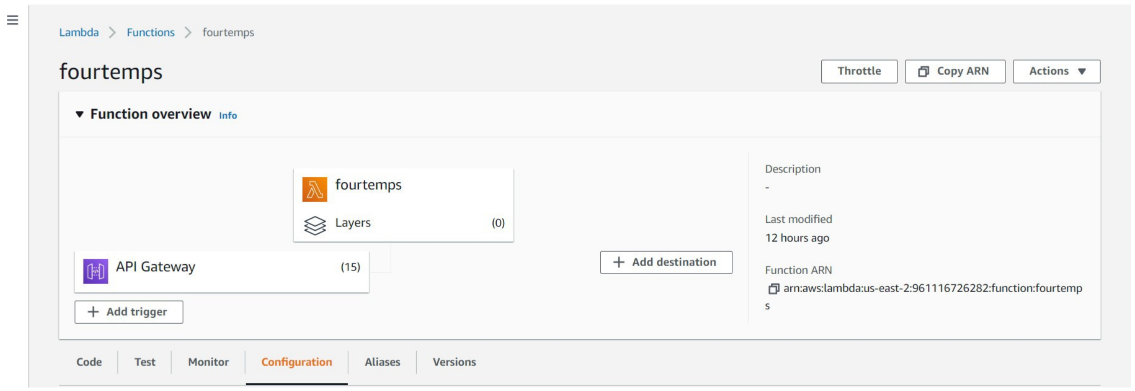
4.3 Technologies Used

AWS Lambda

AWS Lambda is being used to deploy microservices for this thesis. This AWS service offers a pay per request cost structure. This structure allows developers to write individual functions that will implement each microservice before deploying it on AWS Lambda. AWS Lambda is a serverless compute service [4]. It allows developers to run their code and a simple way to receive responses to their code. This service can make HTTP requests through Amazon API Gateway. The function which runs the code in Lambda is called a Lambda function. The steps for creating the Lambda function and uploading data for the fourtemps application are as follow:

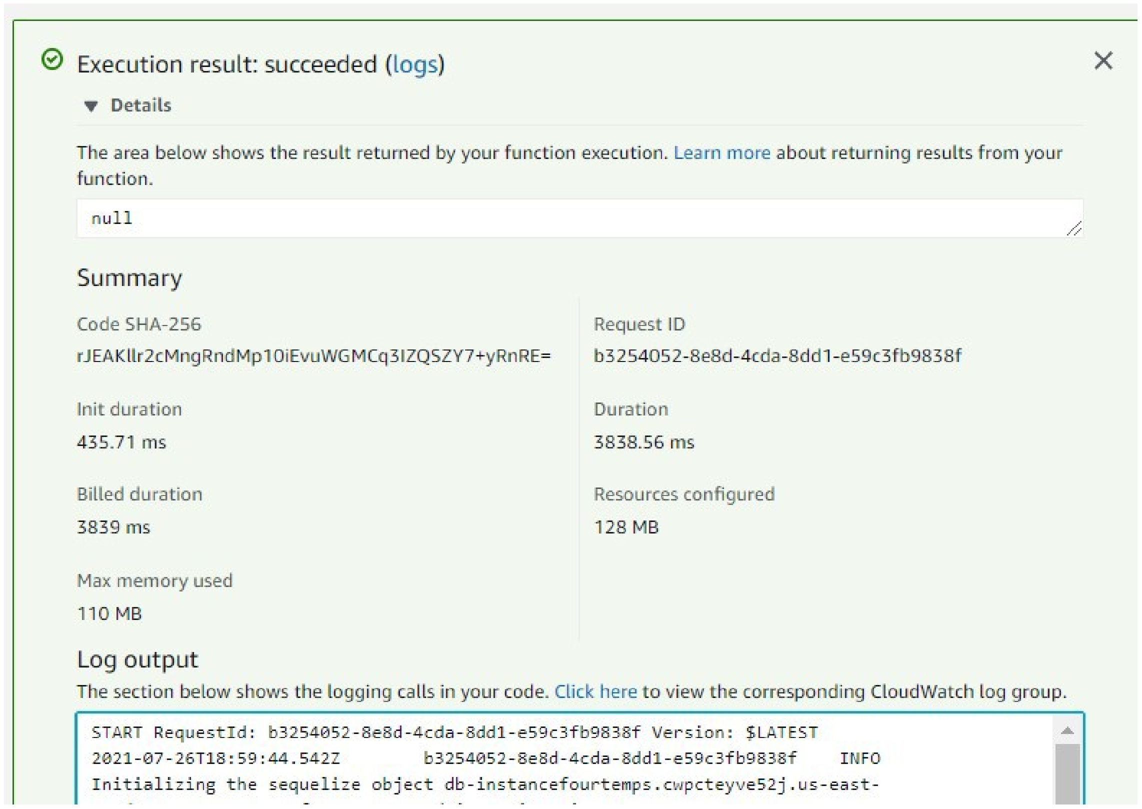
1. For creating a Lambda function, the Function Name is ‘fourtemps’, Node.js 14.x language is chosen, and a new IAM role ’fourtemps-role-2uemuuk2’ is created.

2. The zip file of the Four Temperaments application is uploaded to Lambda. The Figure 28 is an image of the Lambda function for FourTemps.



Description
-
Last modified
12 hours ago
Function ARN
arn:aws:lambda:us-east-2:961116726282:function:fourtemps

Figure 9: Lambda Function for Four Temperaments Application


Execution result: succeeded (logs)

Details

The area below shows the result returned by your function execution. Learn more about returning results from your function.

```
null
```

Summary

Code SHA-256
rJEAKllr2cMngRndMp10iEvuWGMCq3IZQSZY7+yRnRE=

Request ID
b3254052-8e8d-4cda-8dd1-e59c3fb9838f

Init duration
435.71 ms

Duration
3838.56 ms

Billed duration
3839 ms

Resources configured
128 MB

Max memory used
110 MB

Log output

The section below shows the logging calls in your code. Click here to view the corresponding CloudWatch log group.

```
START RequestId: b3254052-8e8d-4cda-8dd1-e59c3fb9838f Version: $LATEST
2021-07-26T18:59:44.542Z        b3254052-8e8d-4cda-8dd1-e59c3fb9838f    INFO
Initializing the sequelize object db-instancefourtemps.cwpcteyve52j.us-east-
```

Figure 10: Lambda function for Four Temperaments

Figure 10 is an image of successful running of the Lamdba function 'fourtemps'.

Amazon RDS

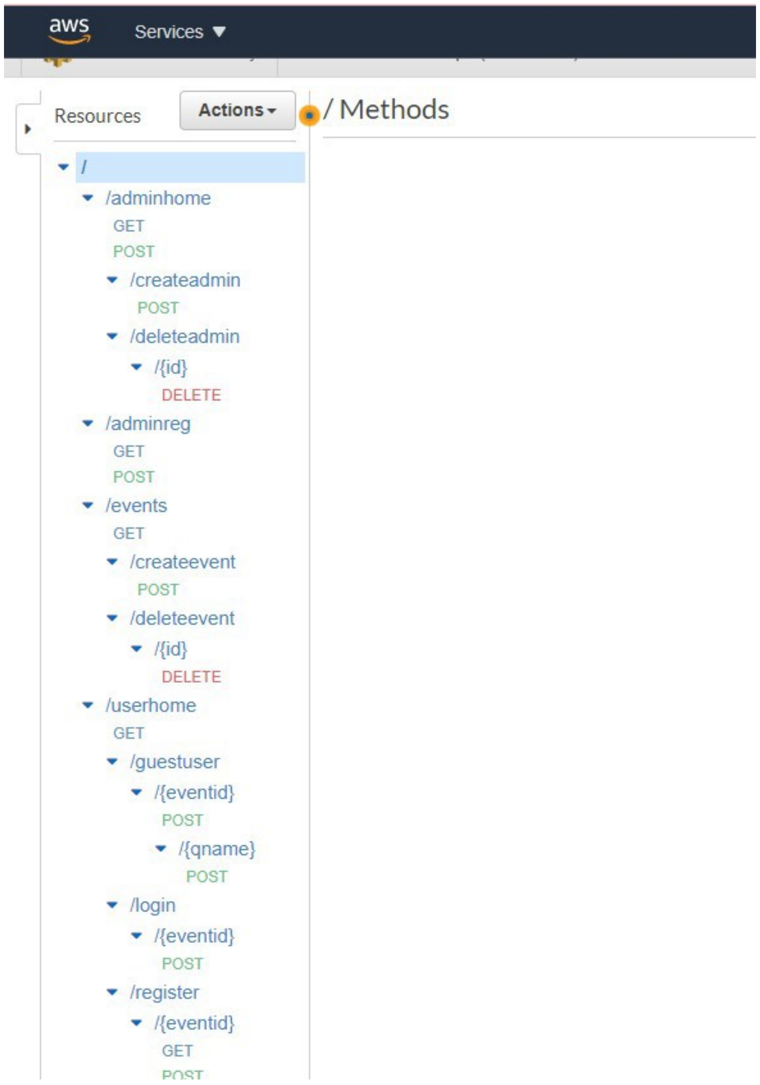
Amazon Relational Database Service (Amazon RDS) is a web service for handling a relational database in the AWS Cloud [3]. RDS is a highly available Data Store service. It provides database scalability i.e both vertically and horizontally. It provides faster Input/Output performance. It can automatically back up a snapshot of the database's data. Amazon RDS for Fourtemps is using MySQL database. The lambda function is connected to Amazon RDS using the code in Listing 1. In this code, the dbHost is the RDS DB identifier for db-instancefourtemps, the database is fourtemps and the dbUser and dbPass are the credentials for the RDS instance. Amazon RDS database db-instancefourtemps and AWS Lambda function fourtemps are in the same availability zone us-east-2 and have the same VPC.

```
// connection .js
    var dBConnection = new Sequelize(database, dbUser, dbPass
   , {
            host: dbHost,
            port: 3306 ,
            dialect: 'mysql'
        });

        dBConnection.authenticate()
            . then ( function () { console. log ("
                CONNECTED !  ");
            })
            . catch ( function ( err) { console. log
                (" NOT  CONNECTED !");
            })
            .done ();
```

Listing 1: DB connection

Amazon API Gateway



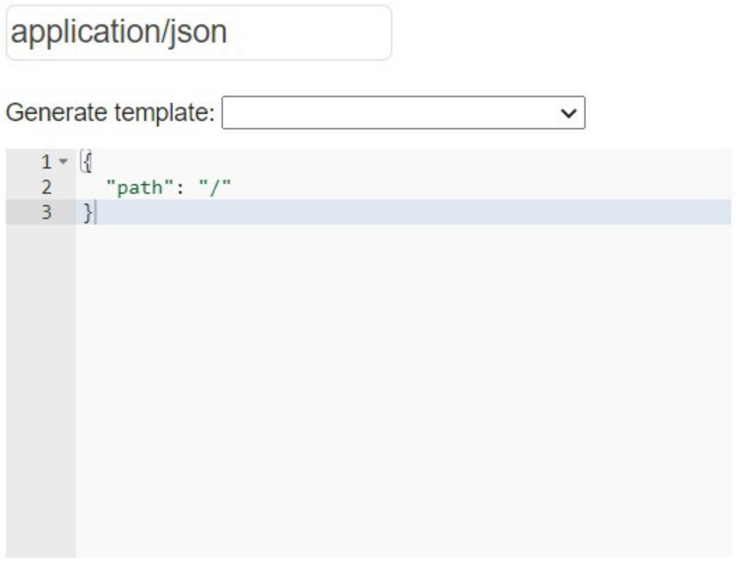
```
▾ /
  ▾ /adminhome
      GET
      POST
    ▾ /createadmin
        POST
    ▾ /deleteadmin
      ▾ /{id}
          DELETE
  ▾ /adminreg
      GET
      POST
  ▾ /events
      GET
    ▾ /createevent
        POST
    ▾ /deleteevent
      ▾ /{id}
          DELETE
  ▾ /userhome
      GET
    ▾ /guestuser
      ▾ /{eventid}
          POST
        ▾ /{qname}
            POST
    ▾ /login
      ▾ /{eventid}
          POST
    ▾ /register
      ▾ /{eventid}
          GET
          POST
```

Figure 11: Resources for Four Temperaments

Amazon API Gateway presents an entry point between the client and the backend services [1]. The resources for FourTemps are displayed in Figure 11 image with the methods for these resources.

The '/id' in the '/deleteadmin' is the parameter that is passed to the route deleteadmin path. For the integration of Lambda function and Amazon API Gateway, 'Restful API' was selected. The following steps were carried out for creating the routes:[22, 23]

1. GET Method is created by selecting the Create Method option from the Actions tab
2. The integration type chosen is Lambda function and the Lambda function selected is fourtemps
3. From Get Method Execution pane, integration request is selected. 'application/json' mapping Template is created. Figure 12 displays the path for the landing page.
4. In the Method Response, the content type is selected as text/html.
5. Similar process is carried for rest of the routes.



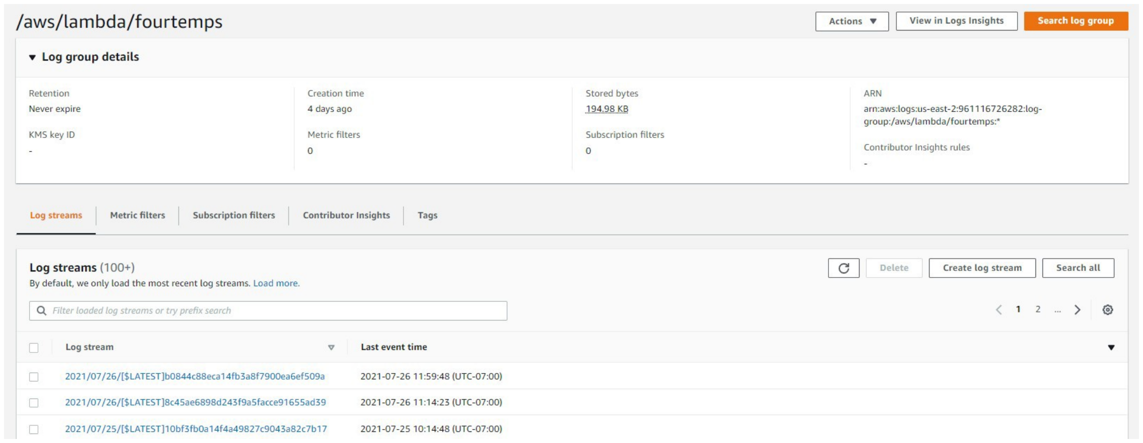
application/json

Generate template:

```
{
  "path": "/"
}
```

Figure 12: application/json

AWS CloudWatch



/aws/lambda/fourtemps

Log group details

Retention: Never expire	Creation time: 4 days ago	Stored bytes: 194.98 KB	ARN: arn:aws:logs:us-east-2:961116726282:log-group:/aws/lambda/fourtemps:*
KMS key ID: -	Metric filters: 0	Subscription filters: 0	Contributor Insights rules: -

Log streams (100+)
By default, we only load the most recent log streams. Load more.

Log stream	Last event time
2021/07/26/[$LATEST]b0844c88eca14fb3a8f7900ea6ef509a	2021-07-26 11:59:48 (UTC-07:00)
2021/07/26/[$LATEST]8c45ae6898d243f9a5facce91655ad39	2021-07-26 11:14:23 (UTC-07:00)
2021/07/25/[$LATEST]10bf3fb0a14f4a49827c9043a82c7b17	2021-07-25 10:14:48 (UTC-07:00)

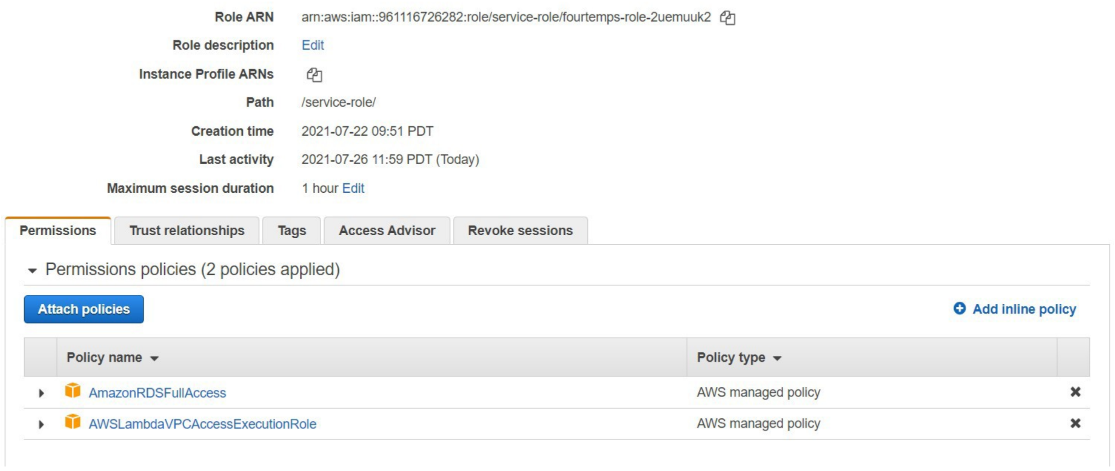
Figure 13: CloudWatch logs for fourtemps Lambda function

AWS CloudWatch, a metrics repository monitors the logs for the applications that are running on AWS [2]. The logs and changes in the fourtemps Lambda function are collected and tracked in the AWS CloudWatch.

IAM

AWS Identity and Access Management provide secure access to AWS services [7]. AWS IAM is used for managing roles and permissions for the roles.

Minimal privileges paradigm for fourtemps lambda function are as follows:



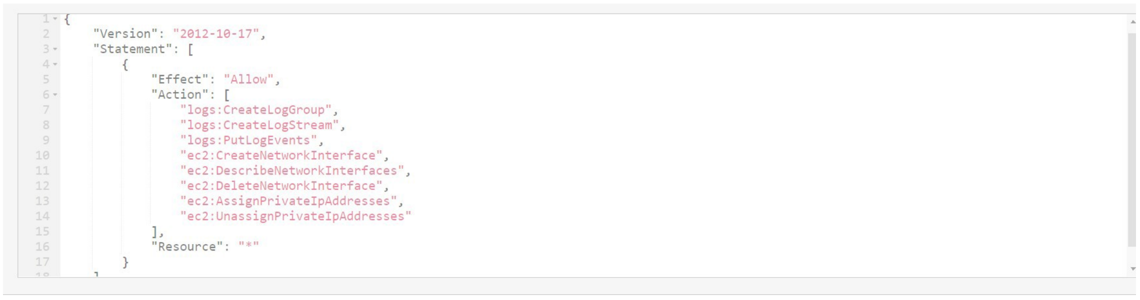
Role ARN	arn:aws:iam::961116726282:role/service-role/fourtemps-role-2uemuuk2
Role description	Edit
Instance Profile ARNs	
Path	/service-role/
Creation time	2021-07-22 09:51 PDT
Last activity	2021-07-26 11:59 PDT (Today)
Maximum session duration	1 hour Edit

Policy name	Policy type
AmazonRDSFullAccess	AWS managed policy
AWSLambdaVPCAccessExecutionRole	AWS managed policy

Figure 14: Minimal privilages for fourtemps Lambda function

The permissions in AWS Lambda functions have 2 important policies, the resource-based policy, and the execution role policy. For invoking the

Lambda function the event source uses the resource-based policy. The Lambda function then uses the execution role policy to check the roles that are allowed to the function.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "ec2:CreateNetworkInterface",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DeleteNetworkInterface",
                "ec2:AssignPrivateIpAddresses",
                "ec2:UnassignPrivateIpAddresses"
            ],
            "Resource": "*"
        }
```

Figure 15: Lambda Policy for fourtemps Lambda function

After creating the Fourtemps Lambda function and a new IAM role fourtemps-role-2uemuuk2 is created only for the Fourtemps Lambda function. The fourtemps-role-2uemuuk2 role is having full access to RDS and is having access to AWSLambdaVPCAccessExecutionRole. The AWSLambdaVPCAccessExecutionRole role has permissions for connecting to AWS EC2 and for generating AWS CloudWatch log group.

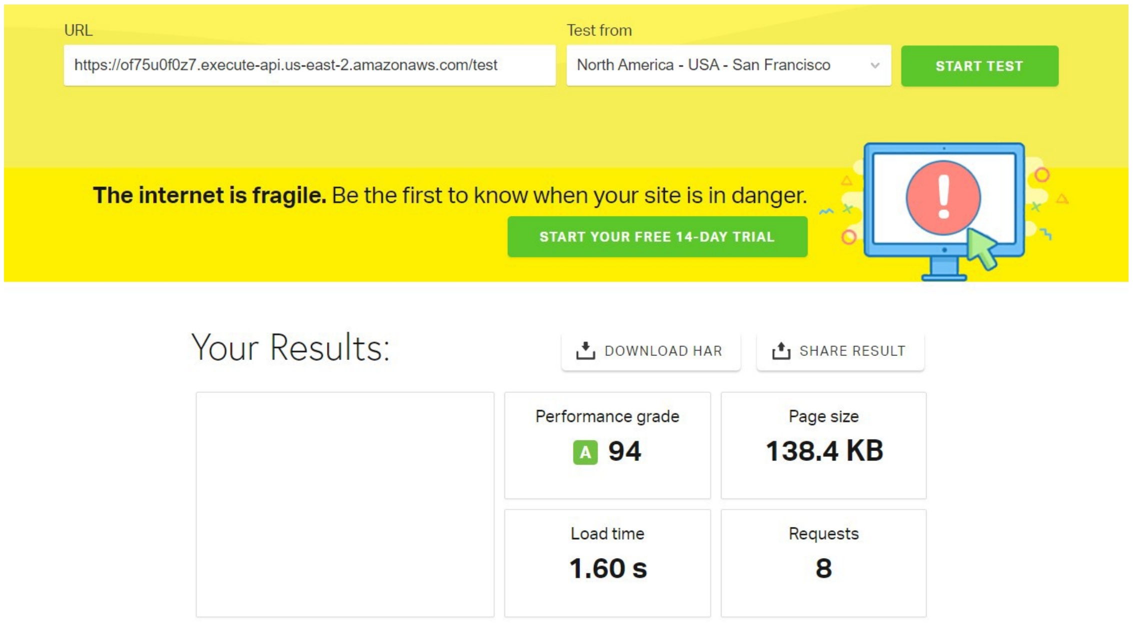
Type of Information	Data
Application	Four Temperaments
Programming language	Node.js
Number of files	5

Table 3: Code specification for Four Temperaments Lambda 1

Type of Information	Data
Application	Four Temperaments
Programming language	Node.js
Number of files	22

Table 4: Code specification for Four Temperaments Lambda 2

After creating the Lambdas the code specification of Lambda 1 is provided in Table 3 and code specification of Lambda 2 is provided in Table 4. This provides an information that converting Four Temperaments to microservices reduced the amount of code significantly and was a better option than monolithic. For testing the speed of the application, Pingdom was used. The response speed of the Four Temperaments application, when deployed to microservices, has increased compared to that of the monolithic application deployed on the AWS EC2 instance. The Figure 16 and Figure 17 are the images displaying the speed of Four Temperament deployed on monolithic and microservices architectures respectively.



URL: https://of75u0f0z7.execute-api.us-east-2.amazonaws.com/test

Test from: North America - USA - San Francisco

START TEST

The internet is fragile. Be the first to know when your site is in danger.

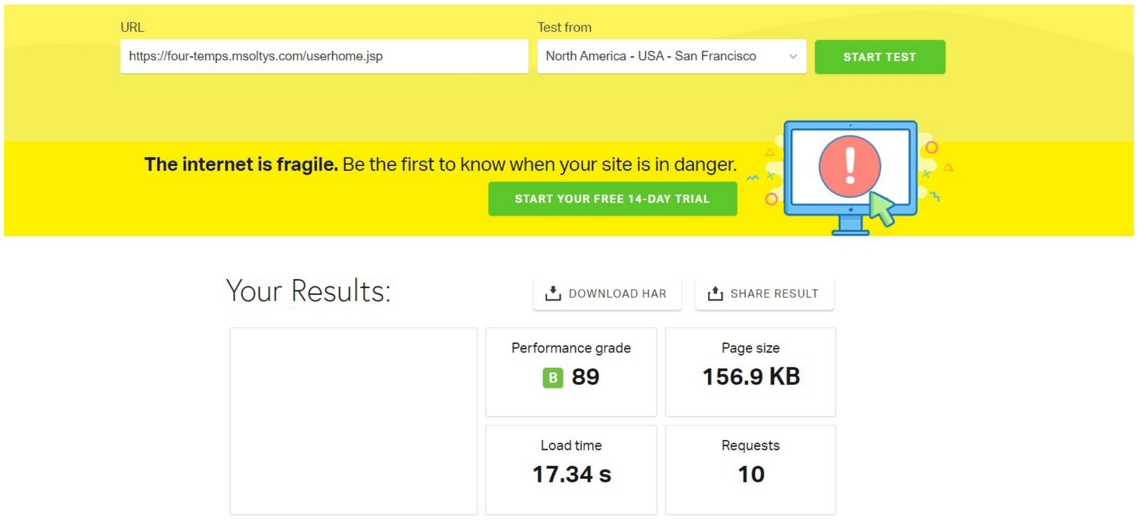
START YOUR FREE 14-DAY TRIAL

Your Results:

DOWNLOAD HAR | SHARE RESULT

Performance grade	Page size
A 94	138.4 KB
Load time	**Requests**
1.60 s	8

Figure 16: Speed of Four Temperaments as Microservice Architecture



URL: https://four-temps.msoltys.com/userhome.jsp

Test from: North America - USA - San Francisco

START TEST

The internet is fragile. Be the first to know when your site is in danger.

START YOUR FREE 14-DAY TRIAL

Your Results:

DOWNLOAD HAR | SHARE RESULT

Performance grade	Page size
B 89	156.9 KB
Load time	**Requests**
17.34 s	10

Figure 17: Speed of Four Temperaments as Monolithic Architecture

5 Implementation

This section provides information about the methods that were used in the system. The following are the methods that were implemented in the system:

1. JavaBeans Activation Framework and Mail
2. Advanced Encryption Standard
3. CanvasJS
4. Token generation for Events
5. MAC Address based Authentication

It also provides the details about the system flow, that is the pages consisting of the Four Temperaments system.

5.1 Methods:

5.1.1 JavaBeans Activation Framework:

Jakarta Activation (JAF) is also known as JavaBeans Activation Framework enables developers to:

1. determines the type of arbitrary piece of data,
2. encapsulate access to it,

3. discover the operations available on it and

4. to instantiate the appropriate bean to perform the operations [11].

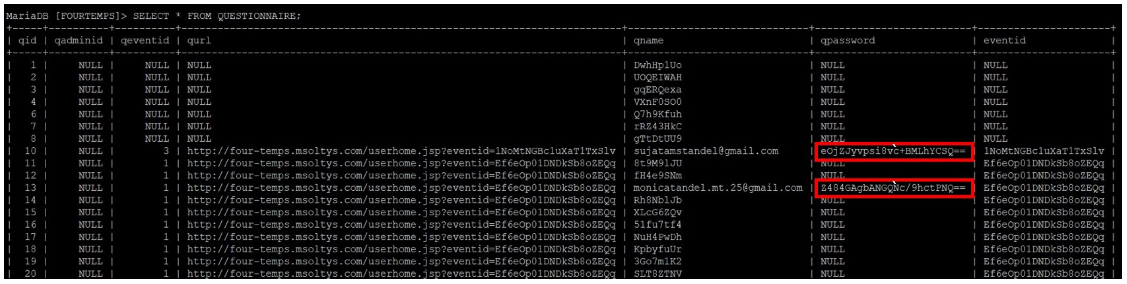
JAF can be operated with external data types. The external data types can be media retrieved from files and streams. It contains classes that wrap arbitrary data sources that provide access to the data as streams or objects, identifying the Multipurpose Internet Mail Extensions (MIME) type of data, and enumerating a registered set of "commands" for operating on the data. In the Four Temperaments system, JAF(activation.1.1.1.jar) with JavaMail API is used for sending mails to the Users as well as the Admins. JavaMail API consists of classes that are essential for designing the mail system. The javax.mail.internet package contains the classes which are related to Internet mail systems. This package specifies the mail systems based on internet standard MIME and simple mail transfer protocol (SMTP).

For sending emails to the users, the sender's email id and password credentials are required. The host used for sending mails is needed to be defined. For example, in the Four Temperaments system, the Gmail host was defined. The host properties like the hostname, the transport protocol, authorization, StartTLS, username, password, and the port number are defined before creating a session for the sender using javax.mail.Authenticator() from javax.mail.internet.

The javax.mail.Authenticator checks if the sender's credentials are legitimate. A default MIMEMessage object is created setting the from, to, subject, and message field, and the email is sent using Transport.send(). For the Four Temperament System, the to field email id is taken from the database.

5.1.2 Advanced Encryption Standard:

Advanced Encryption Standard (AES) is based on a substitution - permutation network which, unlike DES, AES doesn't make use of the Fiestal network. It is a cryptographic algorithm that is used for protecting electronic sensitive data. It uses a block cipher algorithm for ensuring that the data can be stored securely. AES consists of three sizes of ciphers 128, 192, and 256-bit sizes that represent AES-128, AES-192, and AES-256, respectively. Encryption of data is carried out on each block basis in AES. Since it is the symmetric block (secret key) cipher, AES encrypts and decrypts data using the same secret key. The Encryption of data converts data to an unreadable text called ciphertext and decryption converts the data back to its original form called the plaintext. For converting the plaintext to ciphertext, the key size is used in this encryption method which specifies the number of rounds or repetitions the plaintext is put in the cipher block. The number of rounds or repetition can be 10, 12, and 14 rounds corresponding to 128, 192, and 256 - bit keys respectively [19]. For hashing passwords of Admin user and the testing user of the Four Temperament System, AES with 128 bit size i.e. AES-128 is being used. The Figure 18 is the image displaying the hashed passwords for the Testing User.

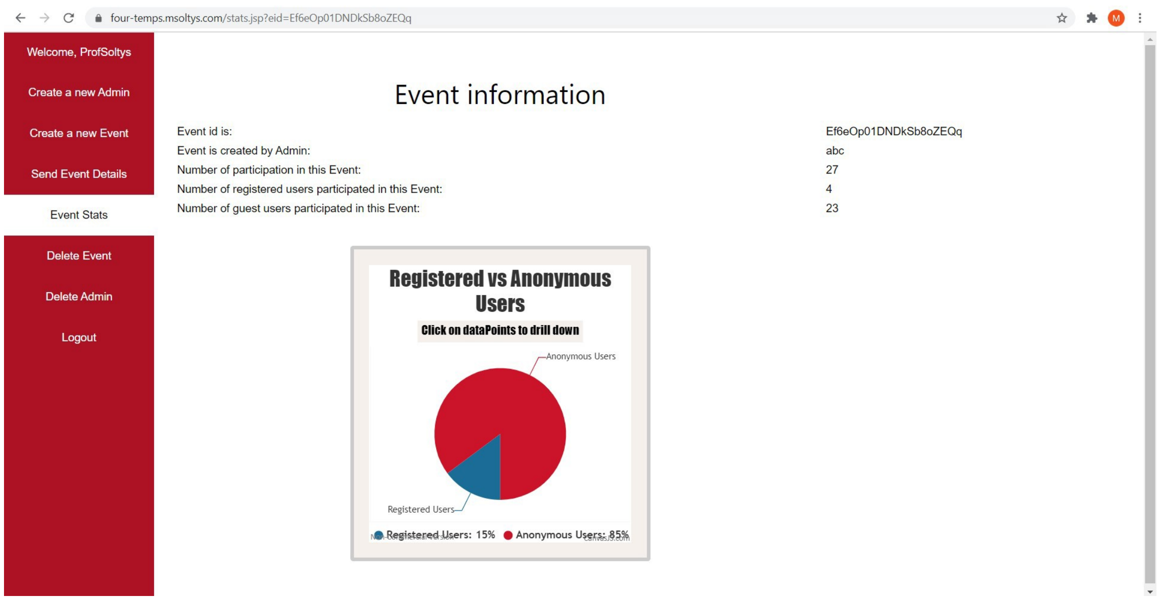


```
MariaDB [FOURTEMPS]> SELECT * FROM QUESTIONNAIRE;
+-----+----------+----------+-------------------------------------------------------------------------+------------------------------+--------------------------+----------------------+
| qid | qadminid | qeventid | qurl                                                                    | qname                        | qpassword                | eventid              |
+-----+----------+----------+-------------------------------------------------------------------------+------------------------------+--------------------------+----------------------+
|   1 |     NULL |     NULL | NULL                                                                    | DwhHp1Uo                     | NULL                     | NULL                 |
|   2 |     NULL |     NULL | NULL                                                                    | UOQEIWAH                     | NULL                     | NULL                 |
|   3 |     NULL |     NULL | NULL                                                                    | gqERQexa                     | NULL                     | NULL                 |
|   4 |     NULL |     NULL | NULL                                                                    | VXnF0SO0                     | NULL                     | NULL                 |
|   6 |     NULL |     NULL | NULL                                                                    | Q7h9Kfuh                     | NULL                     | NULL                 |
|   7 |     NULL |     NULL | NULL                                                                    | rRZ43HkC                     | NULL                     | NULL                 |
|   8 |     NULL |     NULL | NULL                                                                    | gTtDtUU9                     | NULL                     | NULL                 |
|  10 |     NULL |        3 | http://four-temps.msoltys.com/userhome.jsp?eventid=1NoMtNGBc1uXaT1TxSlv | sujatamstandel@gmail.com     | eOjZJyvpsi8vc+BMLhYCSQ== | 1NoMtNGBc1uXaT1TxSlv |
|  11 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | 8t9M91JU                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
|  12 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | fH4e9SNm                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
|  13 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | monicatandel.mt.25@gmail.com | Z484GAgbANGQNc/9hctPNQ== | Ef6eOp01DNDkSb8oZEQq |
|  14 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | Rh8NblJb                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
|  15 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | XLcG6ZQv                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
|  16 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | 51fu7tf4                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
|  17 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | NuH4PwDh                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
|  18 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | KpbyfuUr                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
|  19 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | 3Go7m1K2                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
|  20 |     NULL |        1 | http://four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq | SLT8ZTNV                     | NULL                     | Ef6eOp01DNDkSb8oZEQq |
```

Figure 18: Hashed passwords Using AES for Four Temperaments System

5.1.3 CanvasJS:

CanvasJS is a JavaScript Charting Library. It allows the creation of rich Dashboards and does not compromise on maintainability or functionality [10]. It is built with the help of the Canvas element. In a minuscule moment, this charting library renders thousands of Data Points. CanvasJS is used for displaying the User Statistics graph on the Admin side. Figure 19 is an example of User Statistic graph on the Admin side.

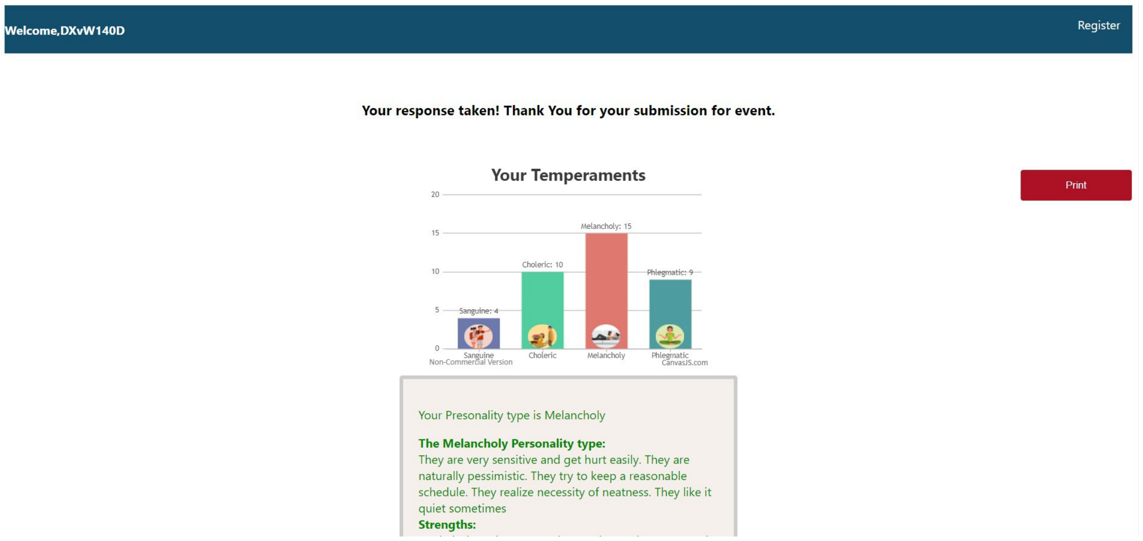

four-temps.msoltys.com/stats.jsp?eid=Ef6eOp01DNDkSb8oZEQq
Welcome, ProfSoltys
Create a new Admin
Create a new Event
Send Event Details
Event Stats
Delete Event
Delete Admin
Logout
Registered vs Anonymous Users
Click on dataPoints to drill down
Anonymous Users
Registered Users
● Registered Users: 15% ● Anonymous Users: 85%

Event information

Event id is:	Ef6eOp01DNDkSb8oZEQq
Event is created by Admin:	abc
Number of participation in this Event:	27
Number of registered users participated in this Event:	4
Number of guest users participated in this Event:	23

Figure 19: Event Statistics graph for Four Temperaments using CanvasJS

It is also used for displaying the personality Type of an user. Figure 20 is an example of the personality type of an user.



Your response taken! Thank You for your submission for event.

Your Presonality type is Melancholy

The Melancholy Personality type:
They are very sensitive and get hurt easily. They are naturally pessimistic. They try to keep a reasonable schedule. They realize necessity of neatness. They like it quiet sometimes
Strengths:

Figure 20: Personality Type of an user graph using CanvasJS

5.1.4 Token Generation for Events:

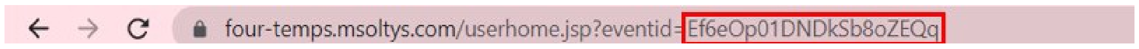

four-temps.msoltys.com/userhome.jsp?eventid=Ef6eOp01DNDkSb8oZEQq

Figure 21: Event Token

The ‘eventid’ for an Event is randomly generated when an Event is created. The ‘eventid’ is of string length 20 and can contain numbers or uppercase/lowercase characters. This ‘eventid’ is used as a token in the URL. Thus, the URL generated for each Event is different. Figure 21 is an example of the event token generated for that Event.

This token is also sent to the ‘send event information to the users’ page as a subject line.

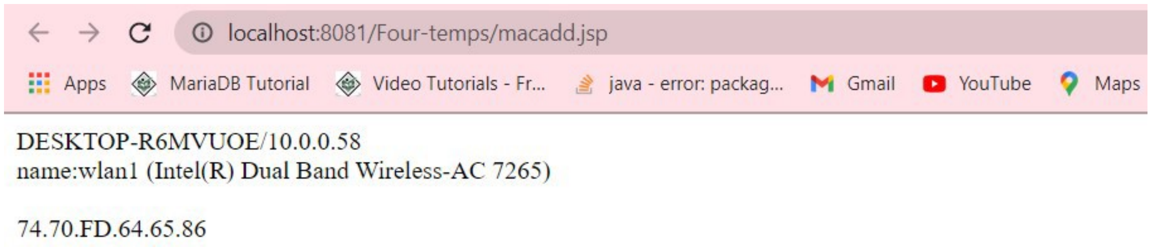
A User is given access to the system only when the Event token exists. If there is no event token in the URL that means the Event is not created and the User is supposed to wait till the Event is created by either of the Admins.

5.1.5 MAC Address Authenticaiton

The MAC Address Authentication uses the MAC address of the device the User is using while registering themselves to the system. While registering themselves to the Four Temperaments system, a call is made from the client-side to the server-side where the Java code for extracting MAC address is run. For knowing the MAC Address of the user's device, it is necessary to know the localhost address from where the Internet Protocol (IP) Address is read. With the help of an IP address, the MAC address is stored in a byte array.

```
    inetAddress = InetAddress.getLocalHost();
    NetworkInterface network = NetworkInterface.
getByInetAddress(inetAddress);
    byte[] hw = network.getHardwareAddress();
```

The MAC address-based authentication makes the system secure by not allowing unauthorized users, who have authorized user's login credentials, the permission to access the system. This authentication is resistant to Dictionary attacks. Figure 22 is the example of getting the MAC from the server-side using the above code.



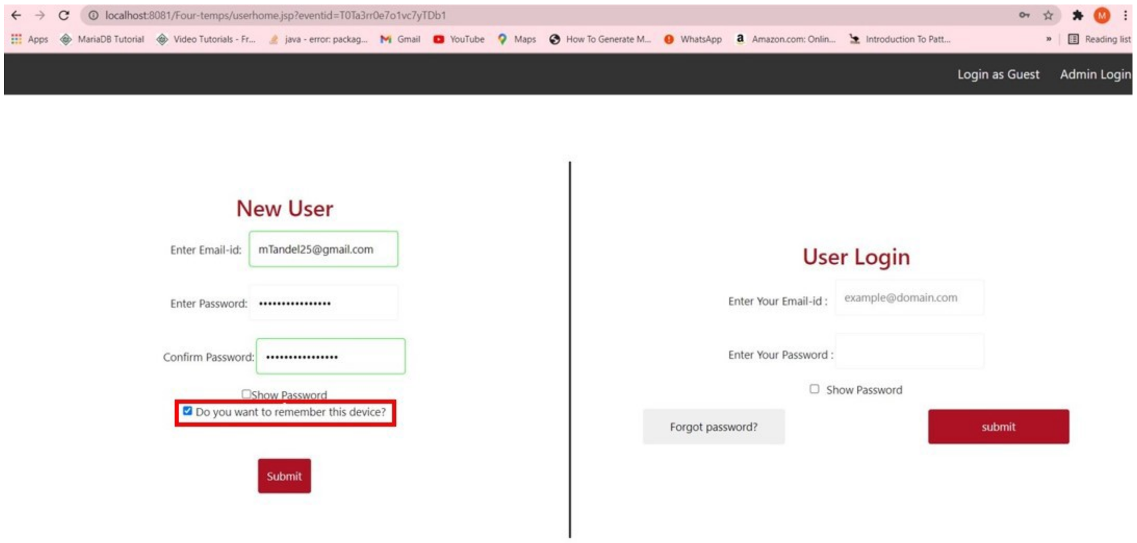
```
DESKTOP-R6MVUOE/10.0.0.58
name:wlan1 (Intel(R) Dual Band Wireless-AC 7265)

74.70.FD.64.65.86
```

Figure 22: Mac Address

The following process takes place for allowing a user to login successfully:

1. The user when logging into the system has clicked on the checkbox for the system to remember their device.


localhost:8081/Four-temps/userhome.jsp?eventid=T0Ta3rr0e7o1vc7yTDb1
Login as Guest
Admin Login
New User

Enter Email-id: mTandel25@gmail.com

Enter Password: ••••••••••••••••

Confirm Password: ••••••••••••••••

Show Password
Do you want to remember this device?
Submit
User Login

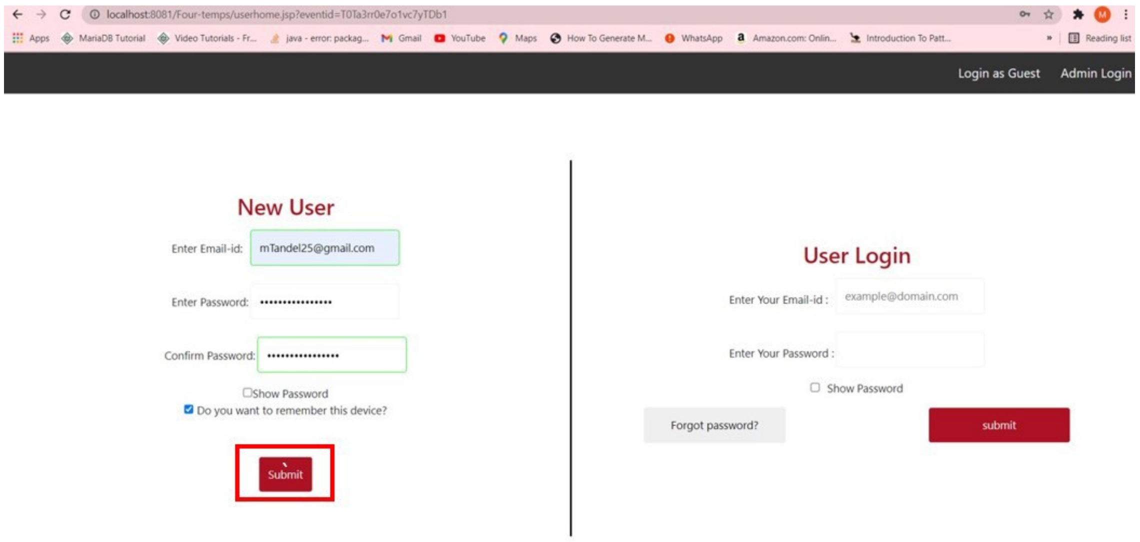
Enter Your Email-id : example@domain.com

Enter Your Password :

Show Password
Forgot password?
submit

Figure 23: User login with Mac Address as the third parameter

2. When the user clicks Submit, the code on the server side which calls the method for extracting the MAC address of the user.



Enter Email-id: mTandel25@gmail.com

Enter Password: ••••••••••••••••

Confirm Password: ••••••••••••••••

☐ Show Password

☑ Do you want to remember this device?

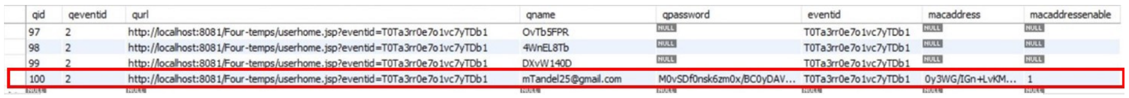
Enter Your Email-id : example@domain.com

Enter Your Password :

☐ Show Password

Figure 24: On click of submit button submission of data to server side

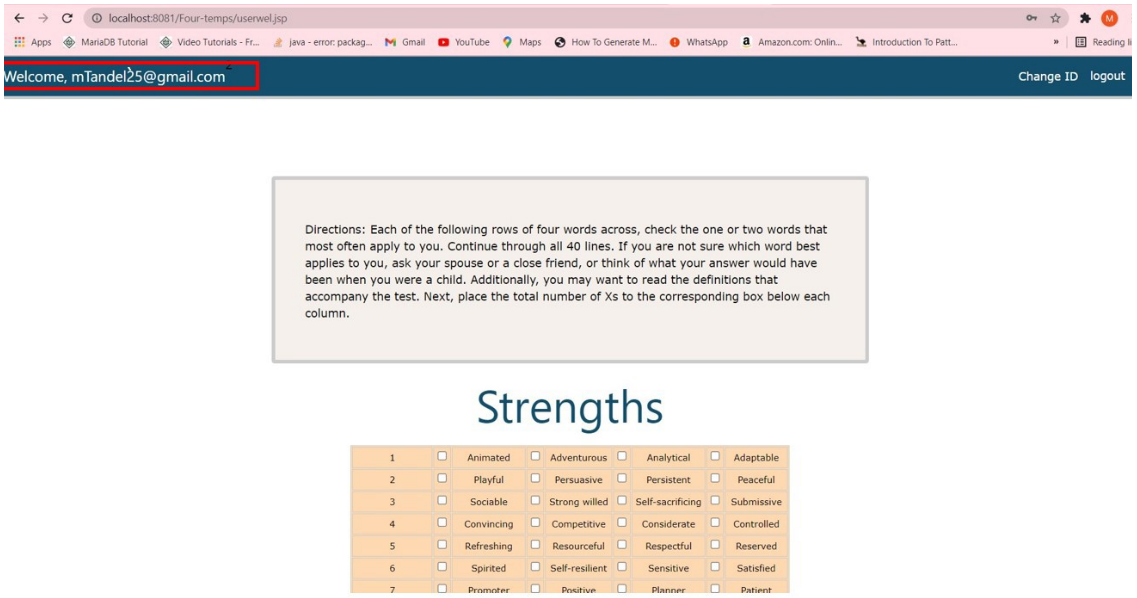
3. Figure 25 is the entry of user's credentials in the database. This MAC address entry is hashed using AES encryption.



qid	qeventid	qurl	qname	qpassword	eventid	macaddress	macaddressenable
97	2	http://localhost:8081/Four-temps/userhome.jsp?eventid=T0Ta3rr0e7o1vc7yTDb1	OvTb5FPR	NULL	T0Ta3rr0e7o1vc7yTDb1	NULL	NULL
98	2	http://localhost:8081/Four-temps/userhome.jsp?eventid=T0Ta3rr0e7o1vc7yTDb1	4WnEL8Tb	NULL	T0Ta3rr0e7o1vc7yTDb1	NULL	NULL
99	2	http://localhost:8081/Four-temps/userhome.jsp?eventid=T0Ta3rr0e7o1vc7yTDb1	DXvW140D	NULL	T0Ta3rr0e7o1vc7yTDb1	NULL	NULL
100	2	http://localhost:8081/Four-temps/userhome.jsp?eventid=T0Ta3rr0e7o1vc7yTDb1	mTandel25@gmail.com	M0vSDf0nsk6zm0x/BC0yDAV...	T0Ta3rr0e7o1vc7yTDb1	0y3WG/IGn+LvKM...	1
NULL		NULL	NULL	NULL	NULL	NULL	NULL

Figure 25: Database entry for the user

4. When user logs into the system, the method above is called for getting the MAC address of the user and comparing that with the entry in the database.



Welcome, mTandel25@gmail.com

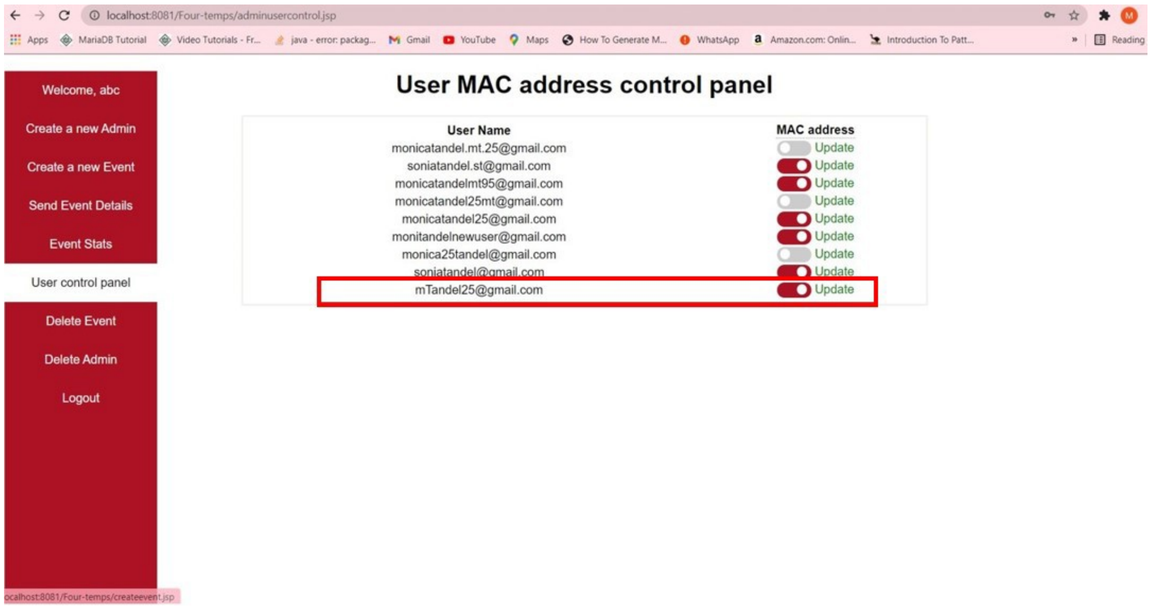
Directions: Each of the following rows of four words across, check the one or two words that most often apply to you. Continue through all 40 lines. If you are not sure which word best applies to you, ask your spouse or a close friend, or think of what your answer would have been when you were a child. Additionally, you may want to read the definitions that accompany the test. Next, place the total number of Xs to the corresponding box below each column.

Strengths

1	☐	Animated	☐	Adventurous	☐	Analytical	☐	Adaptable
2	☐	Playful	☐	Persuasive	☐	Persistent	☐	Peaceful
3	☐	Sociable	☐	Strong willed	☐	Self-sacrificing	☐	Submissive
4	☐	Convincing	☐	Competitive	☐	Considerate	☐	Controlled
5	☐	Refreshing	☐	Resourceful	☐	Respectful	☐	Reserved
6	☐	Spirited	☐	Self-resilient	☐	Sensitive	☐	Satisfied
7	☐	Promoter	☐	Positive	☐	Planner	☐	Patient

Figure 26: Successful login of the user

5. The user control panel from Admin's side which shows that MAC address authentication is enabled for the user.


localhost:8081/Four-temps/adminusercontrol.jsp

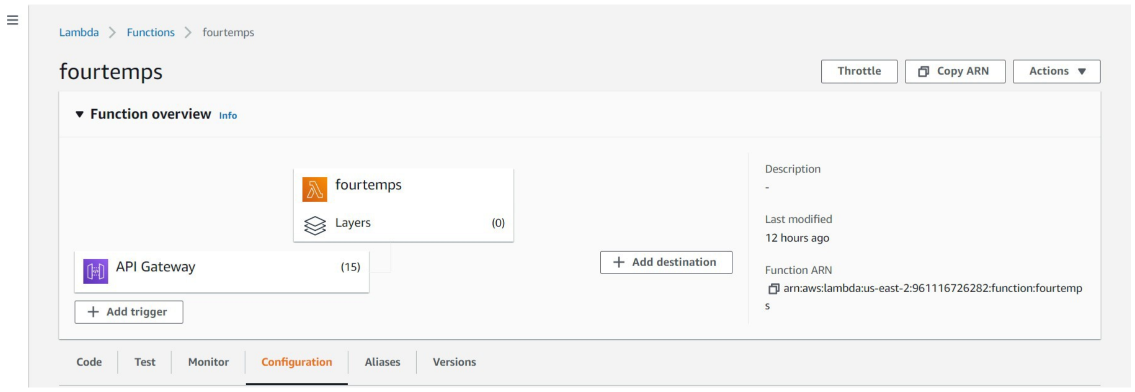
User MAC address control panel

User Name	MAC address
monicatandel.mt.25@gmail.com	○ Update
soniatandel.st@gmail.com	● Update
monicatandelmt95@gmail.com	● Update
monicatandel25mt@gmail.com	○ Update
monicatandel25@gmail.com	● Update
monitandelnewuser@gmail.com	● Update
monica25tandel@gmail.com	○ Update
soniatandel@gmail.com	● Update
mTandel25@gmail.com	● Update

localhost:8081/Four-temps/createevent.jsp

Figure 27: User control panel on Admin's side

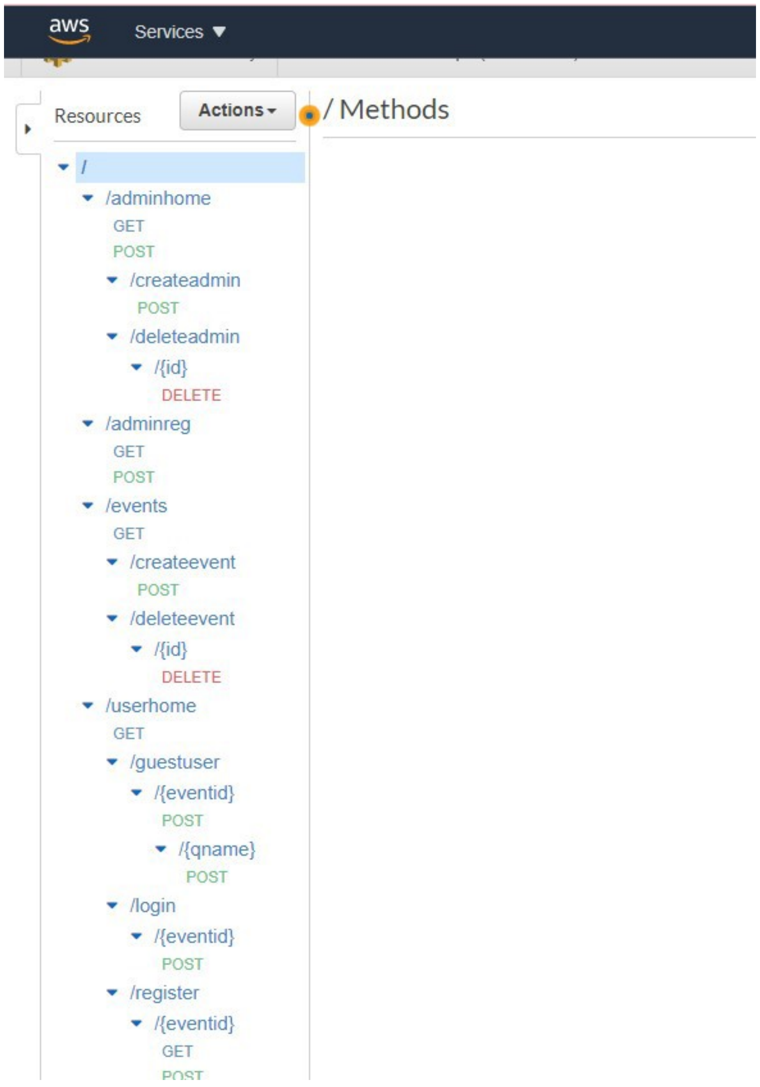
5.2 Implementation of Microservices



Description
-
Last modified
12 hours ago
Function ARN
arn:aws:lambda:us-east-2:961116726282:function:fourtemps

Figure 28: Lambda Function for Four Temperaments Application

The Figure 28 displays the Lambda function for the Four Temperaments Application.



```
/
  /adminhome
    GET
    POST
    /createadmin
      POST
    /deleteadmin
      /{id}
        DELETE
  /adminreg
    GET
    POST
  /events
    GET
    /createevent
      POST
    /deleteevent
      /{id}
        DELETE
  /userhome
    GET
    /guestuser
      /{eventid}
        POST
        /{qname}
          POST
    /login
      /{eventid}
        POST
    /register
      /{eventid}
        GET
        POST
```

Figure 29: Routes for Four Temperaments

The Figure 29 displays the routes that are created for the Four Temperaments Application.

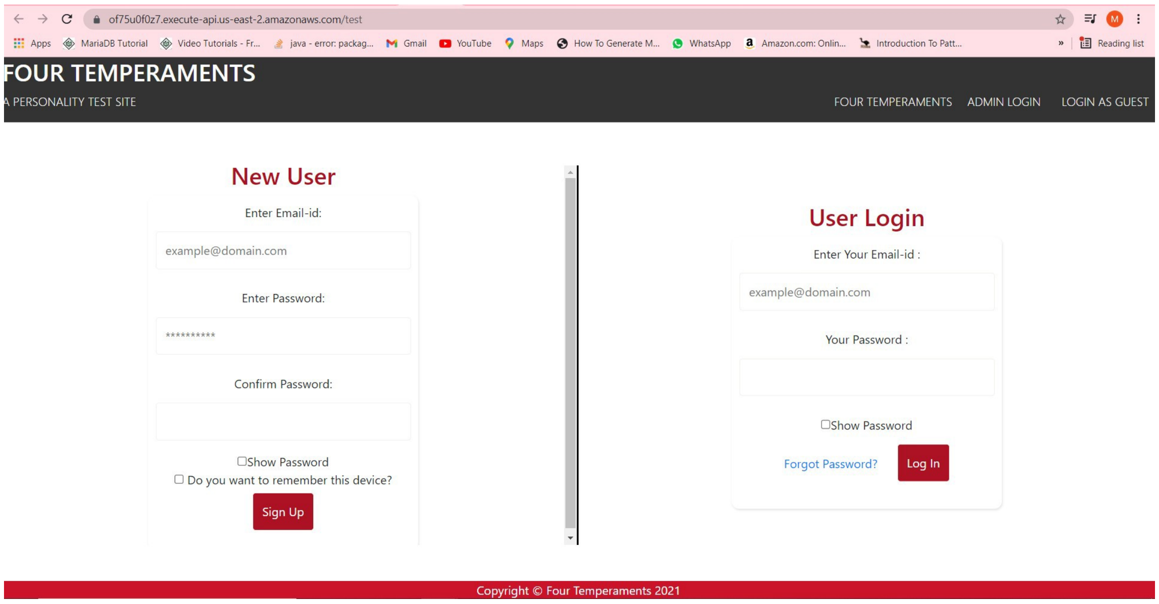
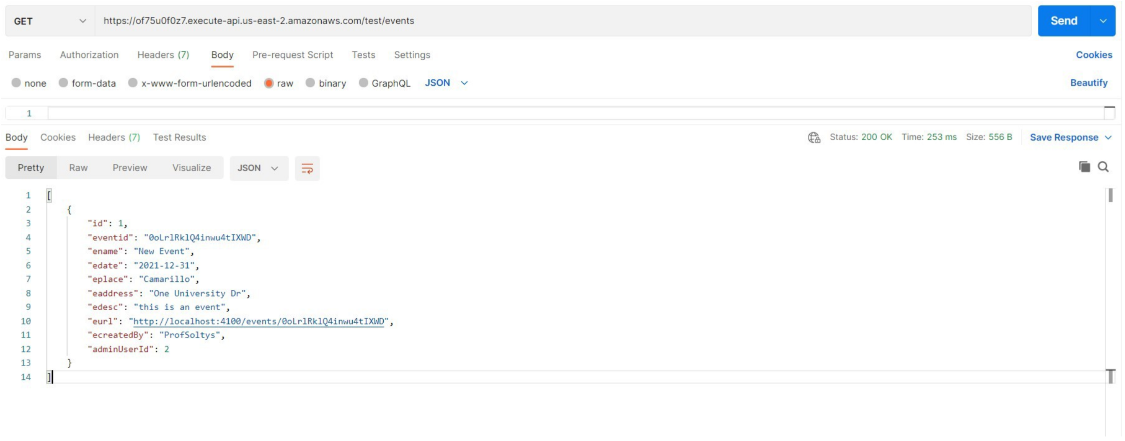


Figure 30: Four Temperaments Website on AWS Lambda

The Figure 30 displays the Four Temperaments application deployed using AWS Lambda. The Figure 31 displays the details of events in Four Temperaments application connected to Amazon RDS.



```
[
    {
        "id": 1,
        "eventid": "0oLrlRklQ4inwu4tIXWD",
        "ename": "New Event",
        "edate": "2021-12-31",
        "eplace": "Camarillo",
        "eaddress": "One University Dr",
        "edesc": "this is an event",
        "eurl": "http://localhost:4100/events/0oLrlRklQ4inwu4tIXWD",
        "ecreatedBy": "ProfSoltys",
        "adminUserId": 2
    }
]
```

Figure 31: Four Temperaments events

5.3 System Flow:

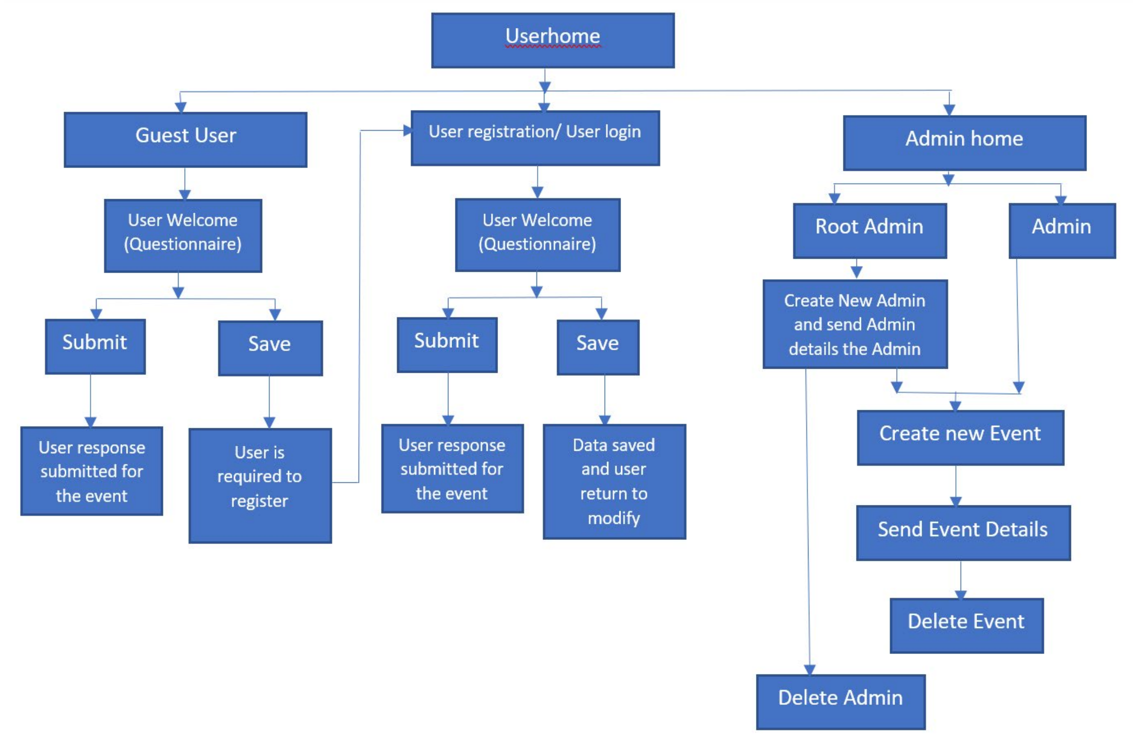

Userhome
Guest User
User registration/ User login
Admin home
User Welcome (Questionnaire)
User Welcome (Questionnaire)
Root Admin
Admin
Submit
Save
Submit
Save
Create New Admin and send Admin details the Admin
User response submitted for the event
User is required to register
User response submitted for the event
Data saved and user return to modify
Create new Event
Send Event Details
Delete Event
Delete Admin

Figure 32: PageFlow of the system

The webpages for this system are designed using HTML and CSS designing languages. The system's design is original and is not using any template of some other system or those available online. The PageFlow of the system gives an overview of how the pages are linked. The process for the system starts with the Userhome page which gives 3 user options to choose Guest User, User registration/User login, or Admin. The next step is to have an event created which is done by Admin or Root Admin. On the selection of

Admin from the Userhome page, the system is redirected to the Admin login page.

5.3.1 Userhome page:

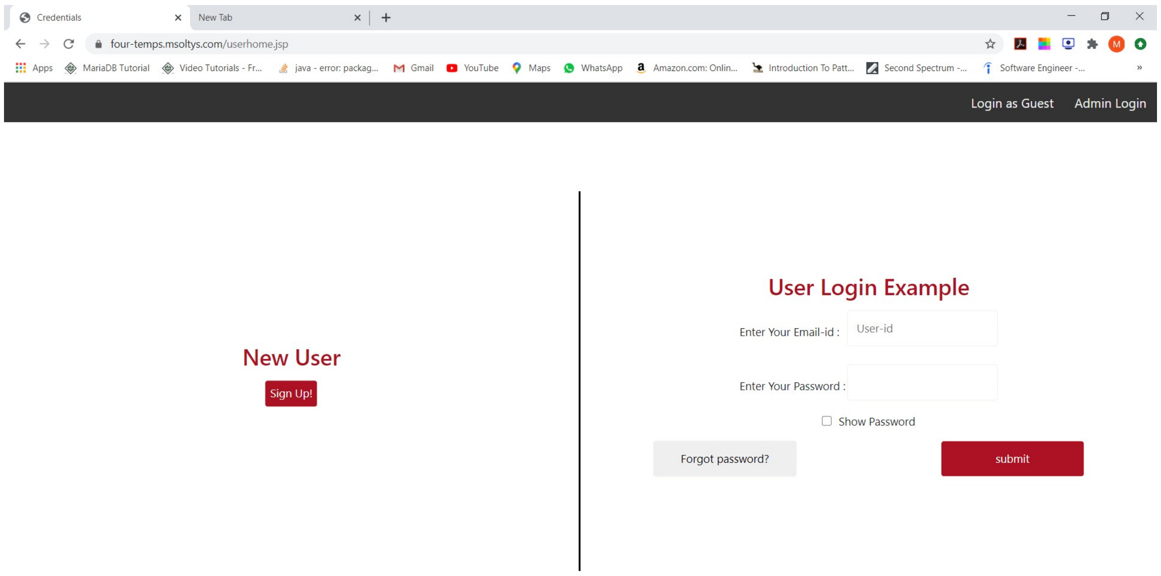


Figure 33: User page of the system

The Userhome page consists of user registration, which allows users to create a new account and user login. For user registration, the user is required to enter their email address, password and confirm their password. The system validates if the user has entered a proper email address. The system highlights the textbox border to green if the password and confirmed password match. 'User' password is stored in encrypted format in the database using the AES algorithm. While logging into the system, if the user has forgotten the password, the system asks for the user to enter their mail address and a randomly generated password is sent to the user via mail for the user to log in. If the user has not correctly entered their credentials, then the system throws an error regarding the same and redirects the 'User' to the home page with the error message asking the user to try logging into the system or use Forgot Password instead. Users can also login anonymously as guest users which can be accessed from the Userhome page. The Userhome page also provides a link that directs to the Admin login page. Figure 33 shows the Userhome page.

5.3.2 Adminhome page:

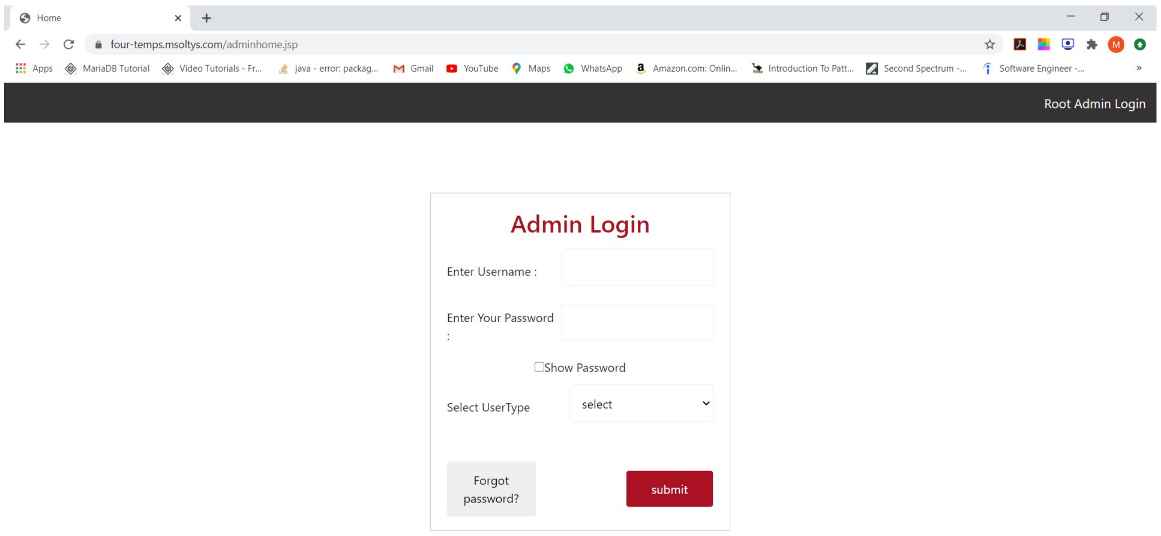
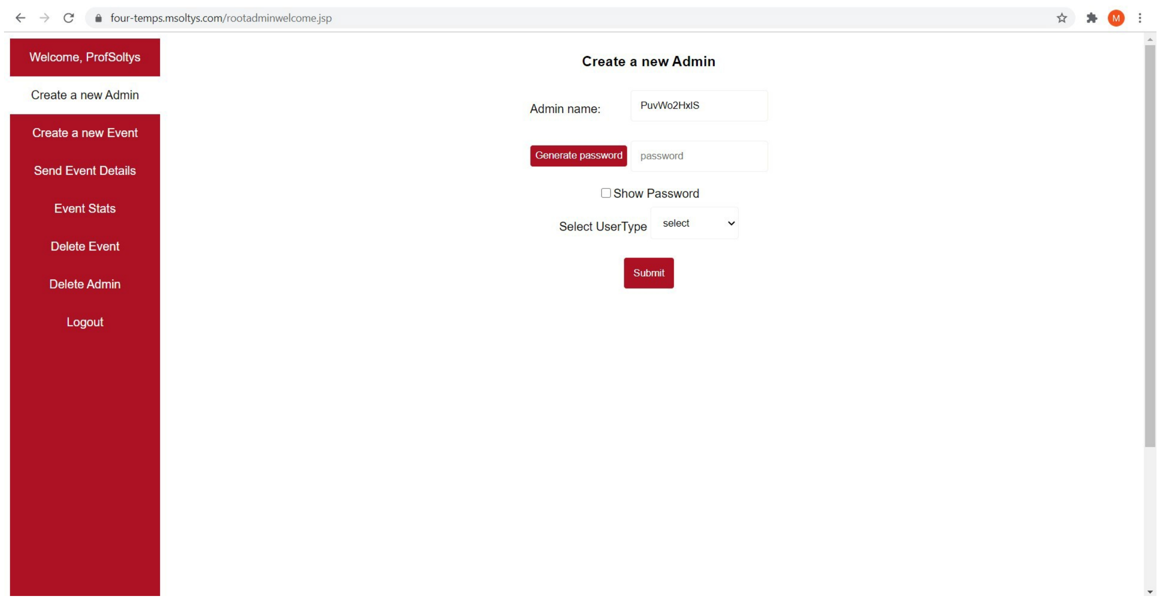


Figure 34: Admin page of the system

Adminhome page consists of admin login and the Root Admin registration link. Admins can log in from the Adminhome page where the admin inputs the ‘username’, ‘password’, and the ‘admintype’ i.e. either RootAdmin or Admin. Also, the system allows only one RootAdmin to be registered. New Admins are created by the RootAdmin with CreateAdmin page and the Admin information is sent to the Admin via their emails. Figure 35 is the page for Creating a new Admin. The RootAdmin can also delete an Admin if they want by using the DeleteAdmin page. Figure 36 is the DeleteAdmin page. For the Admin, the system provides an Admin profile where Admin can change the username or password if they want. Similar to the user password, even the Admin passwords are stored in encrypted format in the database. For a registered Admin, while logging into the system, if they have forgotten their password, the system asks for the username and their email address. A randomly generated password is sent to the Admin for logging into the system via Gmail, details of sending the mail from the system are explained in section Send Mail. Similar to Userhome, if the admin has not entered their credentials properly, then they receive an error message and are asked to reenter their credentials or use Forgot Password to log in. Figure 34 displays Adminhome page.


four-temps.msoltys.com/rootadminwelcome.jsp

Welcome, ProfSoltys
Create a new Admin
Create a new Event
Send Event Details
Event Stats
Delete Event
Delete Admin
Logout

Create a new Admin

Admin name: PuvWo2HxlS

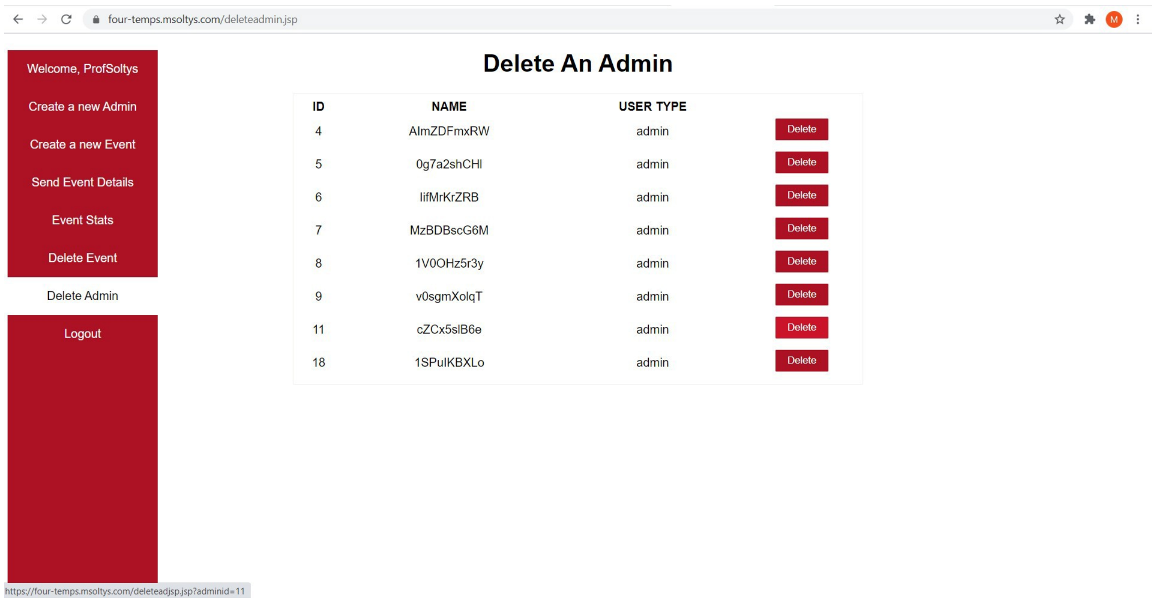
Generate password | password

Show Password

Select UserType: select

Submit

Figure 35: Create Admin page for the system


four-temps.msoltys.com/deleteadmin.jsp

Welcome, ProfSoltys
Create a new Admin
Create a new Event
Send Event Details
Event Stats
Delete Event
Delete Admin
Logout

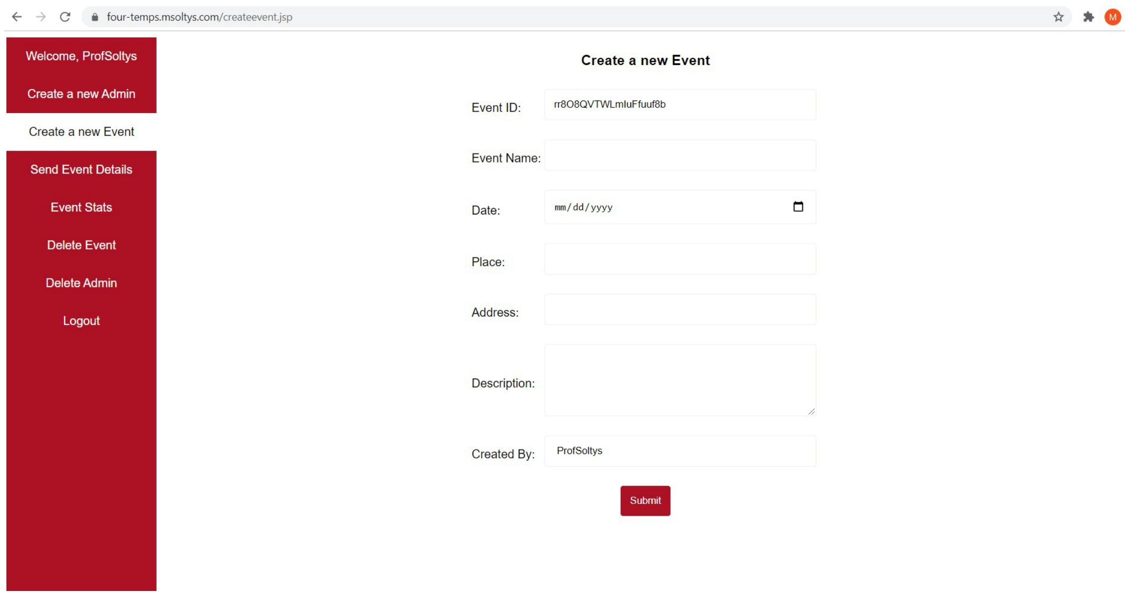
Delete An Admin

ID	NAME	USER TYPE	
4	AlmZDFmxRW	admin	Delete
5	0g7a2shCHl	admin	Delete
6	IifMrKrZRB	admin	Delete
7	MzBDBscG6M	admin	Delete
8	1V0OHz5r3y	admin	Delete
9	v0sgmXolqT	admin	Delete
11	cZCx5slB6e	admin	Delete
18	1SPulKBXLo	admin	Delete

https://four-temps.msoltys.com/deleteadjsp.jsp?adminid=11

Figure 36: Delete Admin page for the system

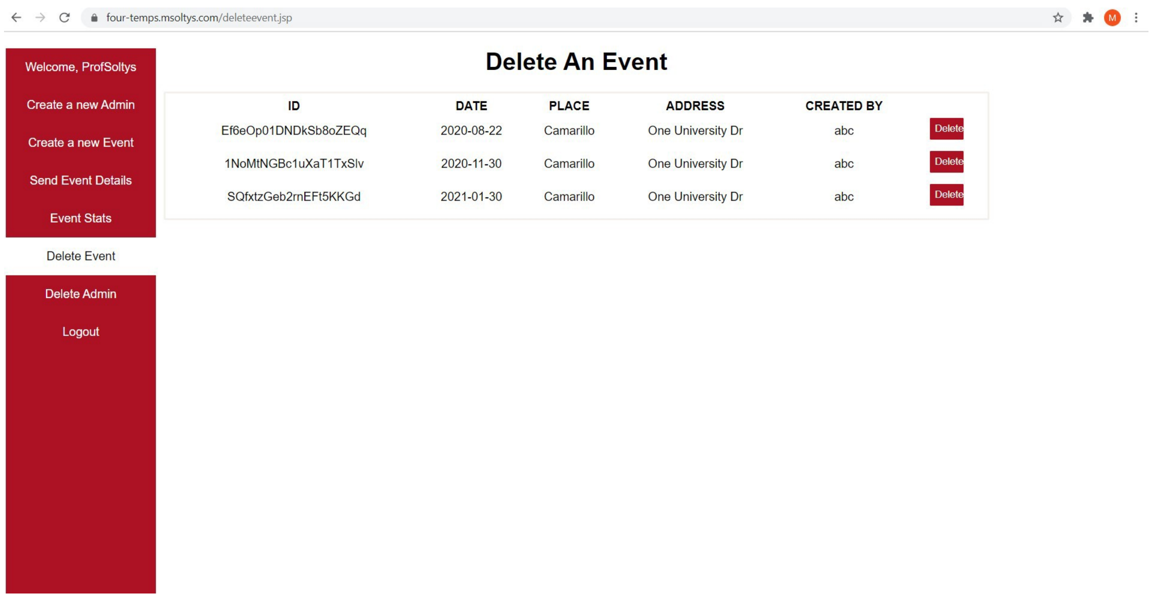
5.3.3 Create and Delete Event pages:


four-temps.msoltys.com/createevent.jsp

Create a new Event

Event ID:	rr8O8QVTWLmIuFfuuf8b
Event Name:	
Date:	mm/dd/yyyy
Place:	
Address:	
Description:	
Created By:	ProfSoltys

Figure 37: Create event for the system



Delete An Event

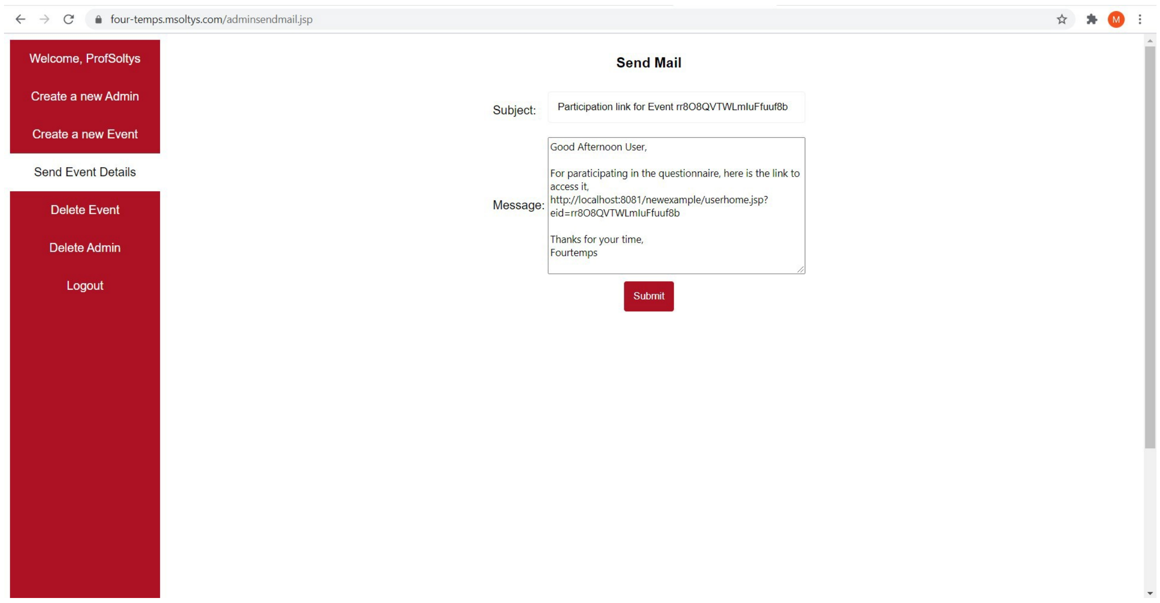
ID	DATE	PLACE	ADDRESS	CREATED BY	
Ef6eOp01DNDkSb8oZEQq	2020-08-22	Camarillo	One University Dr	abc	Delete
1NoMtNGBc1uXaT1TxSlv	2020-11-30	Camarillo	One University Dr	abc	Delete
SQfxtzGeb2rnEFt5KKGd	2021-01-30	Camarillo	One University Dr	abc	Delete

Figure 38: Delete event for the system

RootAdmin and Admin after logging into the system can create new events. The 'eventid' is randomly generated. For a new event to be created, the Admin needs to enter 'Event Name', 'Date', 'Place', 'Address', and 'Description'. The event 'Created By' text is filled by the system itself. After creating the event, the event id is passed to the URL link. This event questionnaire link is then sent to the users to fill the questionnaire and submit their responses. Both the Admins have permission to delete the Event. Figure 37 and Figure 38 displays images for Create Event and Delete Event respectively.

5.3.4 Send Mail:

Currently, the Admins can send the event information to the users who have already registered in the system. Send Mail takes three arguments, the recipient, subject, and the message. The recipient information is currently being taken from the database, event id information is added to both subject and the message text area of the system. Figure 39 is an image for Send Mail page.



Subject: Participation link for Event rr8O8QVTWLmIuFfuuf8b

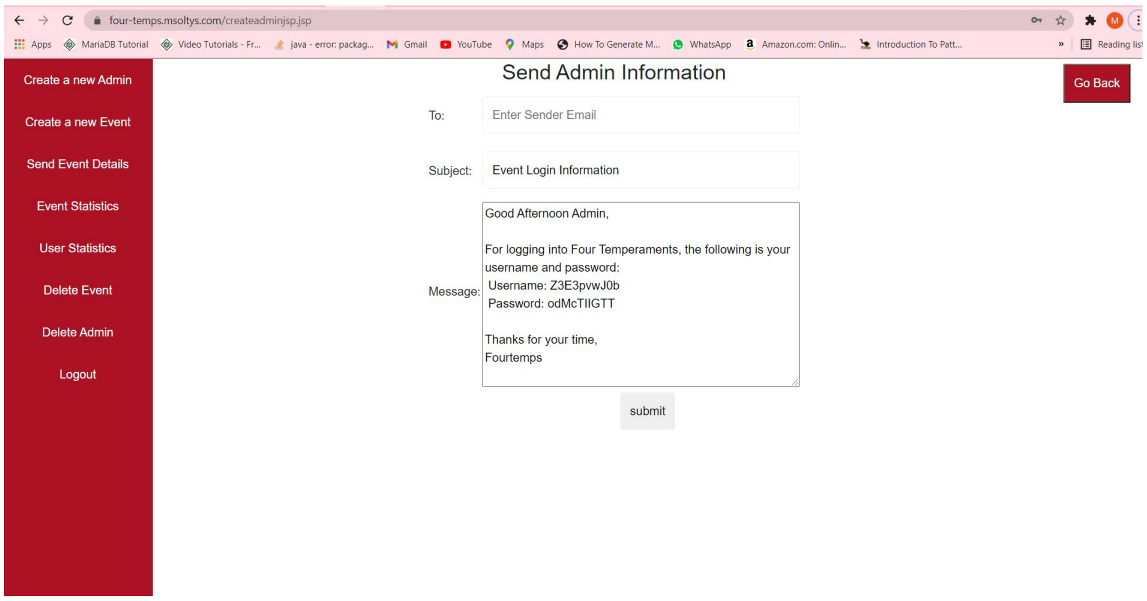
Message:
Good Afternoon User,

For paraticipating in the questionnaire, here is the link to access it,
http://localhost:8081/newexample/userhome.jsp?eid=rr8O8QVTWLmIuFfuuf8b

Thanks for your time,
Fourtemps

Figure 39: Send mail for the system

The email sending function is also used while sending email by the Root Admin to the Admin when Root Admin creates a new Admin. Figure 40 is an example of Root Admin sending information to Admin.



To: Enter Sender Email

Subject: Event Login Information

Message:
Good Afternoon Admin,

For logging into Four Temperaments, the following is your username and password:
Username: Z3E3pvwJ0b
Password: odMcTIIGTT

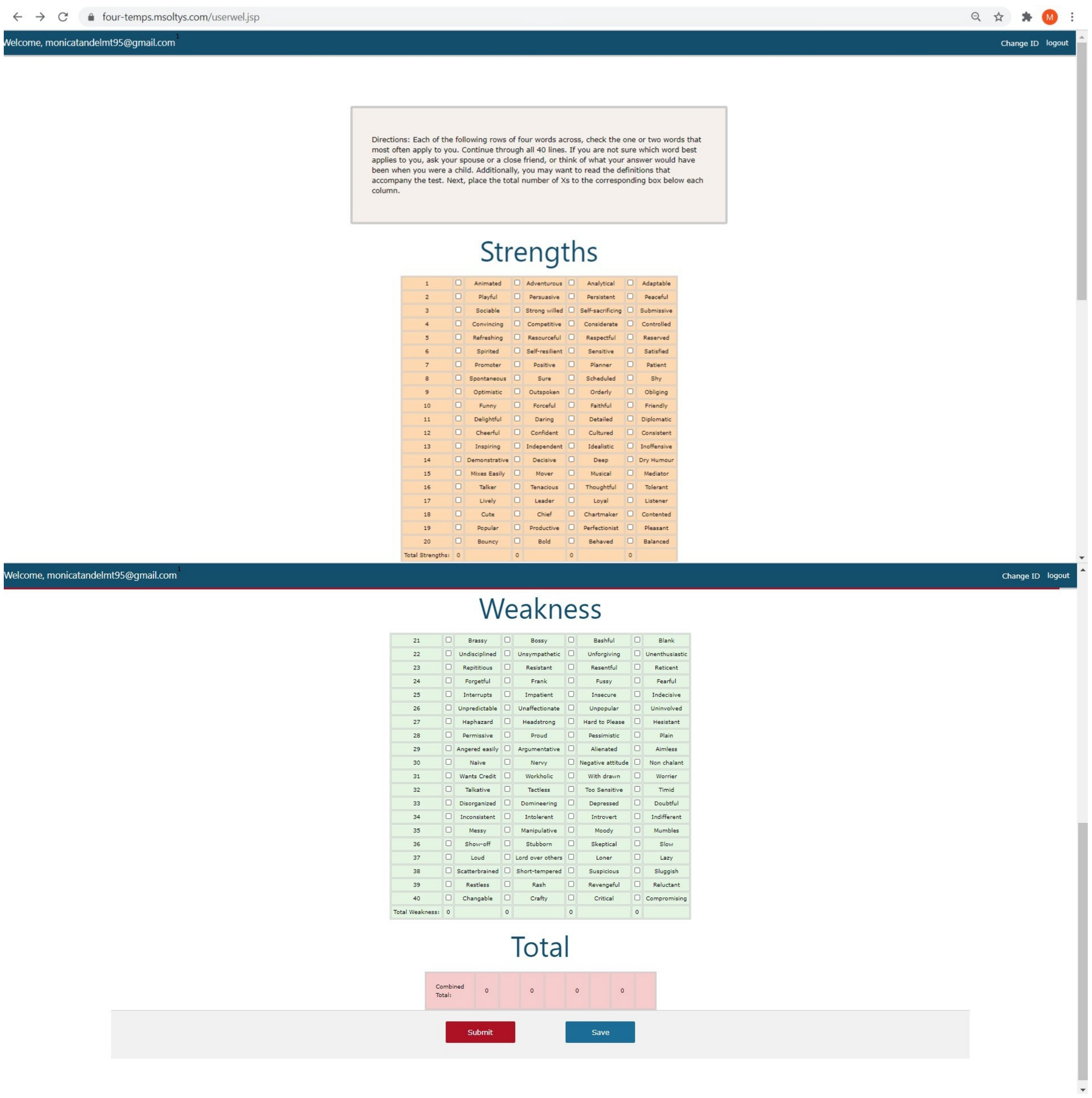
Thanks for your time,
Fourtemps

Figure 40: Root Admin sends Admin information

5.3.5 Userwelcome page:

When the user clicks on the event questionnaire link sent to them by the admin, the user can either register themself as a new user or login to the system or be anonymous i.e. login as a guest. For a registering or already registered user, the user can fill the questionnaire -and either 'Submit' or 'Save' their questionnaire data to the system. Submit button saves the user's response to the questionnaire in the Four Temperaments system and the user won't be able to change that information. For any particular event, the user is allowed to submit only one questionnaire response. The save button allows user to save their information into the system but does not submit a response. Since this user has clicked on the save button, so the user is allowed to return to the event for completing their incomplete submission and submit the questionnaire response once completed.

If the user wishes to change their password, the Userwelcome page also provides the user with an option to change their password which requires the ‘current password’, ‘new password’, and ‘confirm password’ as the input.



four-temps.msoltys.com/userwel.jsp

Welcome, monicatandelmt95@gmail.com[1]

Directions: Each of the following rows of four words across, check the one or two words that most often apply to you. Continue through all 40 lines. If you are not sure which word best applies to you, ask your spouse or a close friend, or think of what your answer would have been when you were a child. Additionally, you may want to read the definitions that accompany the test. Next, place the total number of Xs to the corresponding box below each column.

Strengths

1	☐	Animated	☐	Adventurous	☐	Analytical	☐	Adaptable
2	☐	Playful	☐	Persuasive	☐	Persistent	☐	Peaceful
3	☐	Sociable	☐	Strong willed	☐	Self-sacrificing	☐	Submissive
4	☐	Convincing	☐	Competitive	☐	Considerate	☐	Controlled
5	☐	Refreshing	☐	Resourceful	☐	Respectful	☐	Reserved
6	☐	Spirited	☐	Self-resilient	☐	Sensitive	☐	Satisfied
7	☐	Promoter	☐	Positive	☐	Planner	☐	Patient
8	☐	Spontaneous	☐	Sure	☐	Scheduled	☐	Shy
9	☐	Optimistic	☐	Outspoken	☐	Orderly	☐	Obliging
10	☐	Funny	☐	Forceful	☐	Faithful	☐	Friendly
11	☐	Delightful	☐	Daring	☐	Detailed	☐	Diplomatic
12	☐	Cheerful	☐	Confident	☐	Cultured	☐	Consistent
13	☐	Inspiring	☐	Independent	☐	Idealistic	☐	Inoffensive
14	☐	Demonstrative	☐	Decisive	☐	Deep	☐	Dry Humour
15	☐	Mixes Easily	☐	Mover	☐	Musical	☐	Mediator
16	☐	Talker	☐	Tenacious	☐	Thoughtful	☐	Tolerant
17	☐	Lively	☐	Leader	☐	Loyal	☐	Listener
18	☐	Cute	☐	Chief	☐	Chartmaker	☐	Contented
19	☐	Popular	☐	Productive	☐	Perfectionist	☐	Pleasant
20	☐	Bouncy	☐	Bold	☐	Behaved	☐	Balanced
Total Strengths:	0		0		0		0	

Welcome, monicatandelmt95@gmail.com[1]

Weakness

21	☐	Brassy	☐	Bossy	☐	Bashful	☐	Blank
22	☐	Undisciplined	☐	Unsympathetic	☐	Unforgiving	☐	Unenthusiastic
23	☐	Repititious	☐	Resistant	☐	Resentful	☐	Reticent
24	☐	Forgetful	☐	Frank	☐	Fussy	☐	Fearful
25	☐	Interrupts	☐	Impatient	☐	Insecure	☐	Indecisive
26	☐	Unpredictable	☐	Unaffectionate	☐	Unpopular	☐	Uninvolved
27	☐	Haphazard	☐	Headstrong	☐	Hard to Please	☐	Hesistant
28	☐	Permissive	☐	Proud	☐	Pessimistic	☐	Plain
29	☐	Angered easily	☐	Argumentative	☐	Alienated	☐	Aimless
30	☐	Naive	☐	Nervy	☐	Negative attitude	☐	Non chalant
31	☐	Wants Credit	☐	Workholic	☐	With drawn	☐	Worrier
32	☐	Talkative	☐	Tactless	☐	Too Sensitive	☐	Timid
33	☐	Disorganized	☐	Domineering	☐	Depressed	☐	Doubtful
34	☐	Inconsistent	☐	Intolerent	☐	Introvert	☐	Indifferent
35	☐	Messy	☐	Manipulative	☐	Moody	☐	Mumbles
36	☐	Show-off	☐	Stubborn	☐	Skeptical	☐	Slow
37	☐	Loud	☐	Lord over others	☐	Loner	☐	Lazy
38	☐	Scatterbrained	☐	Short-tempered	☐	Suspicious	☐	Sluggish
39	☐	Restless	☐	Rash	☐	Revengeful	☐	Reluctant
40	☐	Changable	☐	Crafty	☐	Critical	☐	Compromising
Total Weakness:	0		0		0		0	

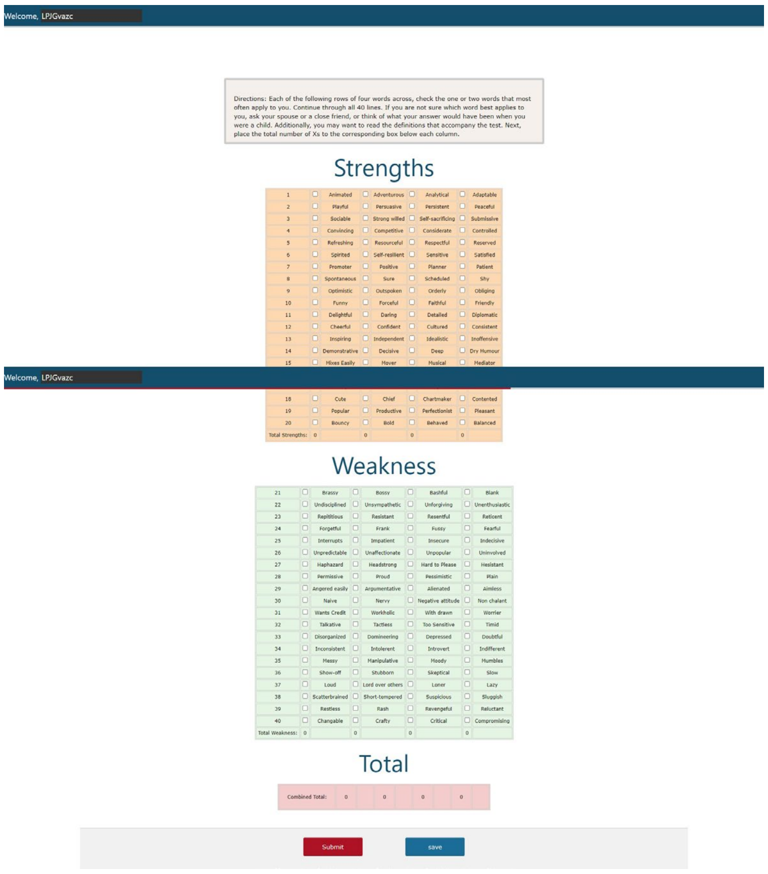
Total

Combined Total:	0		0		0		0	

Figure 41: Userwelcome page of the system

5.3.6 GuestUser page:

If the 'User' chooses to be anonymous, then the 'User' can log in as a guest user. The guest user is allowed to submit the questionnaire for the event. If the user clicks on 'save', then the user is provided an option to either register or, go back and click on submit instead.



Directions: Each of the following rows of four words across, check the one or two words that most often apply to you. Continue through all 40 lines. If you are not sure which word best applies to you, ask your spouse or a close friend, or think of what your answer would have been when you were a child. Additionally, you may want to read the definitions that accompany the test. Next, place the total number of Xs to the corresponding box below each column.

Strengths

1	☐	Animated	☐	Adventurous	☐	Analytical	☐	Adaptable
2	☐	Playful	☐	Persuasive	☐	Persistent	☐	Peaceful
3	☐	Sociable	☐	Strong willed	☐	Self-sacrificing	☐	Submissive
4	☐	Convincing	☐	Competitive	☐	Considerate	☐	Controlled
5	☐	Refreshing	☐	Resourceful	☐	Respectful	☐	Reserved
6	☐	Spirited	☐	Self-resilient	☐	Sensitive	☐	Satisfied
7	☐	Promoter	☐	Positive	☐	Planner	☐	Patient
8	☐	Spontaneous	☐	Sure	☐	Scheduled	☐	Shy
9	☐	Optimistic	☐	Outspoken	☐	Orderly	☐	Obliging
10	☐	Funny	☐	Forceful	☐	Faithful	☐	Friendly
11	☐	Delightful	☐	Daring	☐	Detailed	☐	Diplomatic
12	☐	Cheerful	☐	Confident	☐	Cultured	☐	Consistent
13	☐	Inspiring	☐	Independent	☐	Idealistic	☐	Inoffensive
14	☐	Demonstrative	☐	Decisive	☐	Deep	☐	Dry Humour
15	☐	Mixes Easily	☐	Mover	☐	Musical	☐	Mediator
18	☐	Cute	☐	Chief	☐	Chartmaker	☐	Contented
19	☐	Popular	☐	Productive	☐	Perfectionist	☐	Pleasant
20	☐	Bouncy	☐	Bold	☐	Behaved	☐	Balanced
Total Strengths:	0		0		0		0	

Weakness

21	☐	Brassy	☐	Bossy	☐	Bashful	☐	Blank
22	☐	Undisciplined	☐	Unsympathetic	☐	Unforgiving	☐	Unenthusiastic
23	☐	Repititious	☐	Resistant	☐	Resentful	☐	Reticent
24	☐	Forgetful	☐	Frank	☐	Fussy	☐	Fearful
25	☐	Interrupts	☐	Impatient	☐	Insecure	☐	Indecisive
26	☐	Unpredictable	☐	Unaffectionate	☐	Unpopular	☐	Uninvolved
27	☐	Haphazard	☐	Headstrong	☐	Hard to Please	☐	Hesistant
28	☐	Permissive	☐	Proud	☐	Pessimistic	☐	Plain
29	☐	Angered easily	☐	Argumentative	☐	Alienated	☐	Aimless
30	☐	Naive	☐	Nervy	☐	Negative attitude	☐	Non chalant
31	☐	Wants Credit	☐	Workholic	☐	With drawn	☐	Worrier
32	☐	Talkative	☐	Tactless	☐	Too Sensitive	☐	Timid
33	☐	Disorganized	☐	Domineering	☐	Depressed	☐	Doubtful
34	☐	Inconsistent	☐	Intolerent	☐	Introvert	☐	Indifferent
35	☐	Messy	☐	Manipulative	☐	Moody	☐	Mumbles
36	☐	Show-off	☐	Stubborn	☐	Skeptical	☐	Slow
37	☐	Loud	☐	Lord over others	☐	Loner	☐	Lazy
38	☐	Scatterbrained	☐	Short-tempered	☐	Suspicious	☐	Sluggish
39	☐	Restless	☐	Rash	☐	Revengeful	☐	Reluctant
40	☐	Changable	☐	Crafty	☐	Critical	☐	Compromising
Total Weakness:	0		0		0		0	

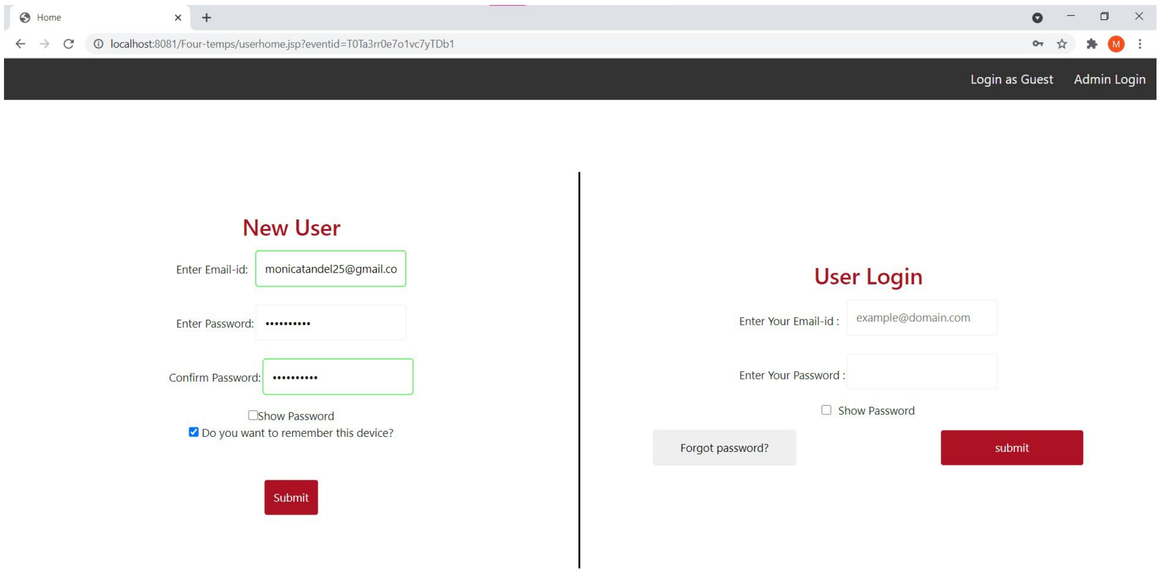
Total

Combined Total:	0		0		0		0	

Figure 42: GuestUser of the system

The guest user can register themselves into the system at any given time by clicking on the register button provided to them on the page

5.3.7 User MAC Address based Authentication page:


Home

localhost:8081/Four-temps/userhome.jsp?eventid=T0Ta3rr0e7o1vc7yTDb1

Login as Guest Admin Login

New User

Enter Email-id: monicatandel25@gmail.co

Enter Password: ••••••••••

Confirm Password: ••••••••••

Show Password

[✓] Do you want to remember this device?

Submit

User Login

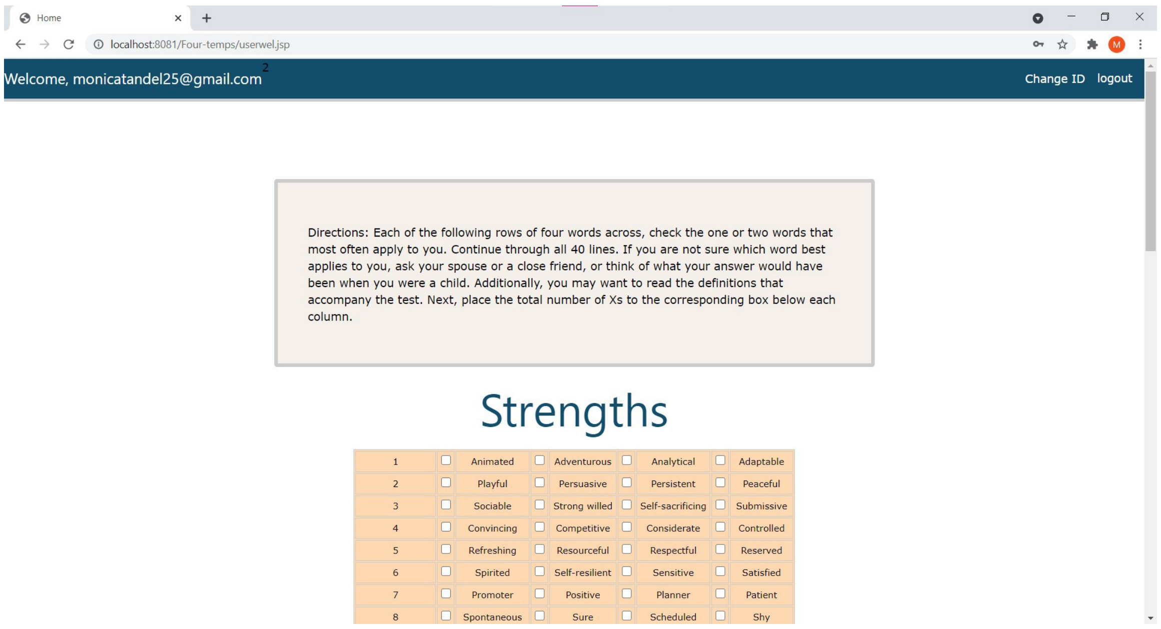
Enter Your Email-id : example@domain.com

Enter Your Password :

Show Password

Forgot password? submit

Figure 43: MacAddressUser of the system



Welcome, monicatandel25@gmail.com

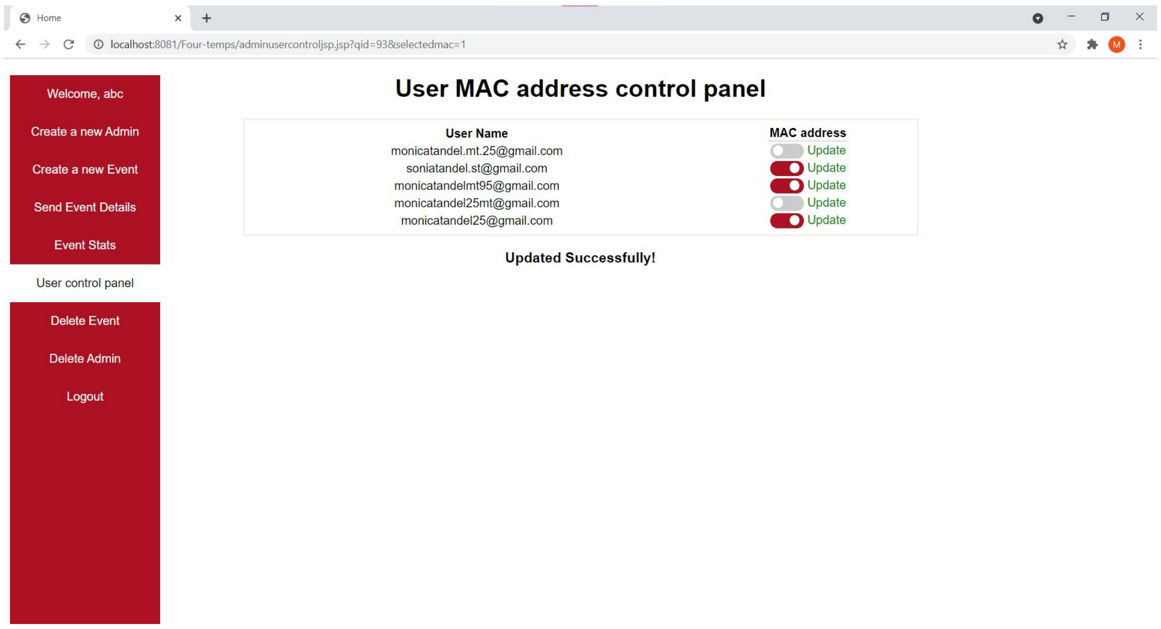
Directions: Each of the following rows of four words across, check the one or two words that most often apply to you. Continue through all 40 lines. If you are not sure which word best applies to you, ask your spouse or a close friend, or think of what your answer would have been when you were a child. Additionally, you may want to read the definitions that accompany the test. Next, place the total number of Xs to the corresponding box below each column.

Strengths

1	☐ Animated	☐ Adventurous	☐ Analytical	☐ Adaptable
2	☐ Playful	☐ Persuasive	☐ Persistent	☐ Peaceful
3	☐ Sociable	☐ Strong willed	☐ Self-sacrificing	☐ Submissive
4	☐ Convincing	☐ Competitive	☐ Considerate	☐ Controlled
5	☐ Refreshing	☐ Resourceful	☐ Respectful	☐ Reserved
6	☐ Spirited	☐ Self-resilient	☐ Sensitive	☐ Satisfied
7	☐ Promoter	☐ Positive	☐ Planner	☐ Patient
8	☐ Spontaneous	☐ Sure	☐ Scheduled	☐ Shy

Figure 44: Sucessful Login of Mac Address enabled user

If the 'User' wants the system to remember the device, they can click on the checkbox after inputting the login credentials for the system. If the checkbox is selected, the MAC address of their device is considered as the third parameter for authentication. Figure 43 is the User login page where the user can select enabling MAC Address Authentication. Figure 44 is the image with successful login for user: 'monicatandel25mt@gmail.com' and password: 'Monica25MT' whose device MAC address is used for authentication.

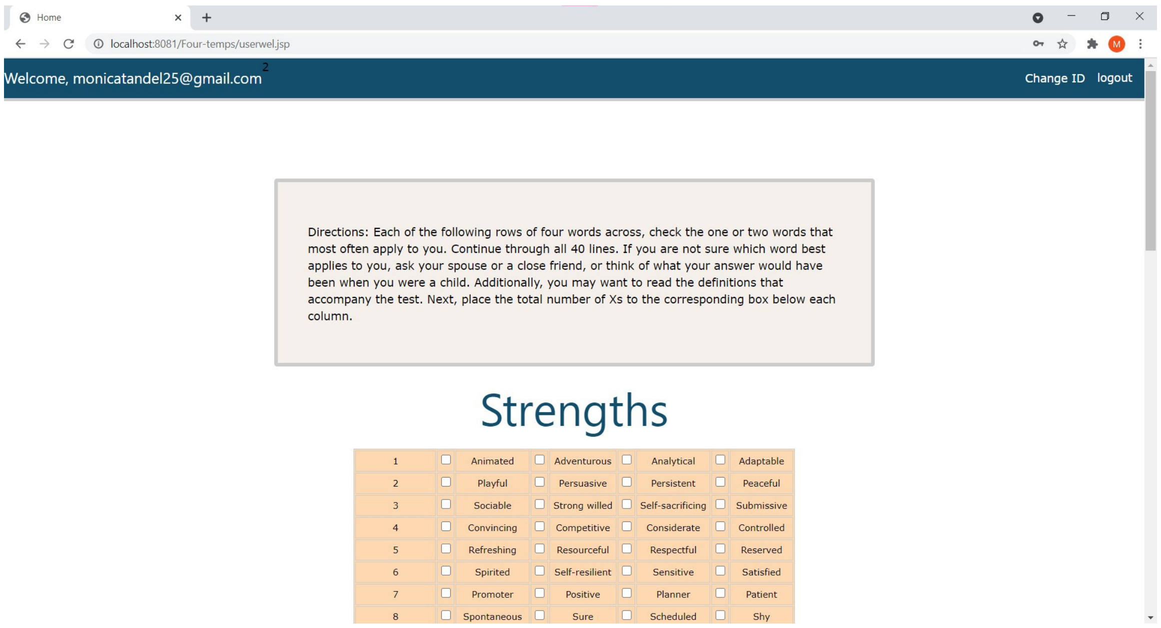


User MAC address control panel

User Name	MAC address
monicatandel.mt.25@gmail.com	Update
soniatandel.st@gmail.com	Update
monicatandelmt95@gmail.com	Update
monicatandel25mt@gmail.com	Update
monicatandel25@gmail.com	Update

Updated Successfully!

Figure 45: Admin User Control Panel for the system


Home
localhost:8081/Four-temps/userwel.jsp
Welcome, monicatandel25@gmail.com
Change ID
logout

Directions: Each of the following rows of four words across, check the one or two words that most often apply to you. Continue through all 40 lines. If you are not sure which word best applies to you, ask your spouse or a close friend, or think of what your answer would have been when you were a child. Additionally, you may want to read the definitions that accompany the test. Next, place the total number of Xs to the corresponding box below each column.

Strengths

1	☐ Animated	☐ Adventurous	☐ Analytical	☐ Adaptable
2	☐ Playful	☐ Persuasive	☐ Persistent	☐ Peaceful
3	☐ Sociable	☐ Strong willed	☐ Self-sacrificing	☐ Submissive
4	☐ Convincing	☐ Competitive	☐ Considerate	☐ Controlled
5	☐ Refreshing	☐ Resourceful	☐ Respectful	☐ Reserved
6	☐ Spirited	☐ Self-resilient	☐ Sensitive	☐ Satisfied
7	☐ Promoter	☐ Positive	☐ Planner	☐ Patient
8	☐ Spontaneous	☐ Sure	☐ Scheduled	☐ Shy

Figure 46: Sucessful Login of Mac Address enabled user

The MAC address authentication for users can also be enabled/disabled by the ‘Admins’. Figure 45 is the page for Admin enabling MAC authentication for the user. Figure 46 is the page of the ‘User’ logging in successfully.

Considering that the cyber attacker is already knowing the user’s login credentials whose MAC address authentication is turned on. The cyber attacker would not be able to log in since the cyber attacker’s MAC address won’t match the verified user’s. Thus the login attempt will not be successful.

6 Conclusion and future work

The thesis focuses on providing a strategic method for refactoring the Four Temperament Test system to microservices. This strategic method involves the business functionalities concept while performing the refactorization. From the analysis, it can be seen that refactoring Four Temperaments into microservices architecture from monolithic was helpful as it lowered the code files and response rate. The reason for choosing business functionalities as an important key for refactorization is that the business functionalities do not change. In future work, the 4th step from the strategy i.e. the analysis of the business functionality can be carried out using machine learning or human interaction.

For securing the system, the 'User' and 'Admin' passwords are encrypted using the AES algorithm. Another way for securing the system was to add MAC address as the third parameter for authentication. This application is designed using Microservices to make the application more modular and easier to maintain.

The Four Temperaments system follows MVC that is the Model - View - Controller architecture. The system handles the software-related issues, namely, the issues related to access control, accuracy, authorization, availability, confidentiality, fortification, authentication and integrity; accountability issues; and error handling.

The system consists of 3 main entities, namely, the testing 'User', the 'Root Admin', and the 'Admin'. The 'Root Admin' and 'Admin' can create an Event for 'Users' to attempt the questionnaire. The Event link information is sent to the 'Users'.

The system provides 'Users' a simple interface for answering the questionnaire. The 'Users' are provided with a facility to save the questionnaire if they wish or submit it directly. On submission of the questionnaire responses, a graphical representation of the User's personality temperament, a brief description of their personality, their strengths, weaknesses, and career options respective to their temperament along with some solutions to overcome their weaknesses are provided to the 'User'.

The 'Admins' of the system are capable to create and delete events. They are also able to check the Event statistics. The Event statistics consists of a graph that displays the number of registered users and anonymous users registered for the system. The 'Admins' are currently sending the emails to the registered users of the system. For future work, the implementation of sending emails for unregistered users can be performed.

The test helps during screening applicants for employment and job training by measuring the characteristics of a person like their attitude, emotional adjustment, interest, and motivation determining if the person is a good fit. These tests also help understand one's relationship with their coworkers, screening applicants for employment and job training, criminal cases, custody battles, and accessing psychological disorders

References

[1] Amazon api gateway — amazon web services. https://aws.amazon.com/api-gateway/.

[2] Amazon cloudwatch - application and infrastructure monitoring. https://aws.amazon.com/cloudwatch/.

[3] Amazon rds — cloud relational database — amazon web services. https://aws.amazon.com/rds/.

[4] Aws lambda – serverless compute - amazon web services. https://aws.amazon.com/lambda/.

[5] The four temperaments - menu. http://temperaments.fighunter.com/.

[6] Guide: Four temperaments and humors — personality psychology. https://personality-psychology.com/four-temperaments-and-humors/.

[7] Iam role - aws identity and access management. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html.

[8] Tutorial: Getting started with amazon ec2 windows instances - amazon elastic compute cloud. https://docs.aws.amazon.com/AWSEC2/latest/WindowsGuide/EC2_GetStarted.html.

[9] Authentication vs authorization. https://www.okta.com/identity-101/authentication-vs-authorization/, 2021.

[10] Html5 charts and graphs. https://canvasjs.com/, 2021.

[11] the java ee 6 tutorial. https://docs.oracle.com/cd/E19798-01/ 821-1841/bnactindex.html, 2021.

[12] What is secure software engineering? https://www.cerias.purdue.edu/site/blog/post/what-is-secure-software-engineering/, 2021.

[13] Bintang Maulana Prasetya Pagar Alam, Rycka Septiasari, and Amiruddin Amiruddin. Applying mac address-based access control for securing admin's login page. *Proceeding of the Electrical Engineering Computer Science and Informatics*, 6(0), 2019.

[14] Antonio Bucchiarone, Nicola Dragoni, Schahram Dustdar, Stephan T. Larsen, and Manuel Mazzara. From monolithic to microservices: An experience report from the banking domain. *IEEE Software*, 35(3):50–55, 2018.

[15] Pamela Flores and Victor Velepucha. Monoliths to microservices - migration problems and challenges: A sms. *Second International Conference on Information Systems and Software Technologies*, 2021.

[16] Jean-Philippe Gouigoux and Dalila Tamzalit. ”functional-first” recommendations for beneficial microservices migration and integration. *IEEE International Conference on Software Architecture (ICSA-C)*, 2019.

[17] Jean-Phillipe Gouigoux and Dailia Tamzalit. From monolith to microservices lessons learned on an industrial migration to web oriented architecture. *IEEE International Conference on Software Architecture Workshops*, 2017.

[18] Manabu Kamimura, Keisuke Yano, Tomomi Hatano, and Akihiko Matsuo. Extracting candidates of microservices from monolithic application code. *Asia-Pacific Software Engineering Conference (APSEC)*, 2018.

[19] Rob Mardisalu. What is advanced encryption standard (aes): Beginner’s guide. https://thebestvpn.com/ advanced-encryption-standard-aes/what, May 2019.

[20] Kostantinos Papadamou, Steven Gevers, Christos Xenakis, Michael Sirivianos, Savvas Zannettou, Bogdan Chifor, Sorin Teican, George Gugulea, Alberto Caponi, and Annamaria et al. Recupero. Killing the password and preserving privacy with device-centric and attribute-based authentication. *IEEE Transactions on Information Forensics and Security*, 15:2183–2193, 2020.

[21] Nikhat Parveen, Md. Rizwan Beg, and M. H. Khan. Software security issues: Requirement perspectives. 2014.

[22] Danilo Poccia. Aws lambda: Calling functions from a web browser. https://medium.com/@danilop/ aws-lambda-calling-functions-from-a-web-browser-338fbcb6a44d, Aug 2016.

[23] Danilo Poccia. *AWS Lambda in action: event-driven serverless applications*. Manning Publications, 2017.

[24] Chris Richardson. *Microservices patterns: with examples in Java.* Simon and Schuster, 2018.

[25] What is: Java By Matthew Tyson and Matthew Tyson. What is tomcat? the original java servlet container. https://www.infoworld.com/article/3510460/ what-is-apache-tomcat-the-original-java-servlet-container. html, Dec 2019.

[26] Afzaal Ahmad Zeeshan. Programming in java using the mvc architecture. https://www.codeproject.com/Articles/879896/ Programming-in-Java-using-the-MVC-Architecture, Feb 2015.