HEART HEALTH STATUS TRACKING APPLICATION

Project report submitted in complete fulfillment of the requirement for the degree of Bachelor of Technology
in

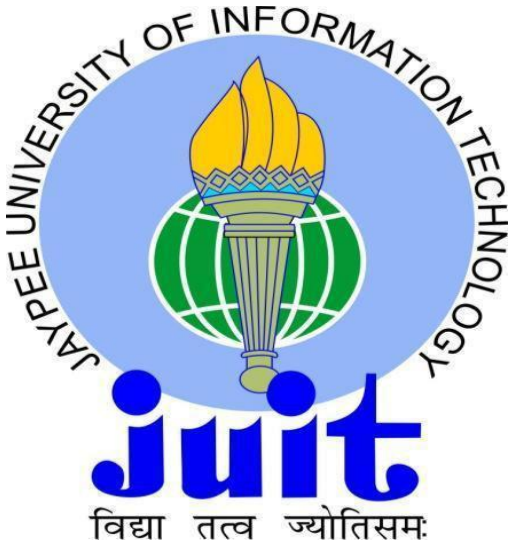
Computer Science and Engineering/Information Technology

By
Shivam Jindal (141271)
Vaibhav Nagpal (141415)

Under the supervision of

Dr. Yugal Kumar

To



Department of Computer Science & Engineering and Information Technology
Jaypee University of Information Technology Waknaghat, Solan-173234, Himachal Pradesh

Certificate

Candidate's Declaration

We hereby declare that the work presented in this report entitled **"Health Heart Status Tracking Application**" in complete fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering Technology** and **Information Technology** submitted in the department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from January 2018 to May 2018 under the supervision of **Dr. Yugal Kumar** (**Assistant Professor**).

Shivam Jindal(141271) Vaibhav Nagpal(141415)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Dr. Yugal Kumar

Assistant Professor

Computer Science & Engineering Department

Dated:

ACKNOWLEDGEMENT

We are thankful and obligated to Dr. Yugal Kumar, Assistant Professor, Department of Computer Science and Engineering for his assistance and guidance in fruition of this task report. We likewise express our profound feeling of appreciation and thankfulness to our guide for his steady supervision, motivation and support ideal from the earliest starting point of this venture. We additionally need to thank our folks and companions for their monstrous help and certainty upon us. We regard it a wonderful obligation to put on record our earnest and genuine appreciation to our task manage for his long sightedness, shrewdness and co-activity which helped us in handling essential parts of the venture in an exceptionally consistent and down to earth way.

Vaibhav Nagpal (141415)

Shivam Jindal (141271)

TABLE OF CONTENTS

LIST OF FIGURES

.

LIST OF TABLES

LIST OF ABBREVATIONS

API(s)	Application Programming Interface(s)
AWS	Amazon Web Services
CBIHCS	Cloud Based Intelligent Health Care Service
CPU	Central Processing Unit
HTTP	Hypertext Transfer Protocol
S3	Amazon Simple Storage Service
SOA	Service-Oriented Architecture
CVD	Cardiovascular hazard

ABSTRACT

Coronary illness influences roughly 70 million individuals worldwide where the vast majority doesn’t know the manifestations. This exploration analyzes the model of early cautioning framework for coronary illness by android application. It plans to encourage clients to early distinguish coronary illness which can be utilized autonomously.

The promising capability of distributed computing and its meeting with advancements, for example, versatile figuring, remote systems, sensor advances takes into consideration creation and conveyance of more up to date kind of cloud administered mobile applications.

In this paper a real-time heart disease monitoring system which alerts user about their current health status is introduced. The system uses various factors such as smoking cholesterol level etc. into consideration for calculating the risk score using Framingham Risk Score.

CHAPTER 1

INTRODUCTION

Heart failure and stroke cause a major weight on society because of their high expenses of care. It causes millions of deaths around the world. The checking of patient's physiological data is vital for the further treatment. Numerous patients can be profited by persistent observing as a piece of an analytic strategy, ideal support of a constant condition or amid managed recuperation from an intense occasion.

Due to the recent technological advancements in the past two decades, more and more population has started using mobile phones, this has created a market for various applications which can ease their lives. A heart health status tracking application improves the quality of life along with offering heart health-related user centered health care contents actively.

The proposed project is an implementation of an android based application which is compatible with all the android devices running on version greater than 5.0(Lollipop) and relies on patient's heart health record data and various other factors such as smoking habits, cholesterol level, heart rate etc. to calculate coronary disease risk and give each patient a risk score using Framingham Risk Score Algorithm.

1.1) Problem Statement

Most of today's heart healthcare applications are not patient and doctor oriented. This creates a problem that there is a lack of proper communication between patients and doctors, and doctors are unable to see some of the symptoms of patients reacted coronary condition.

In particular, this android application explicitly creates a link between patients and doctors and also eases the work of doctor by collecting data by patients regarding their day to day activities and general habits, the application also calculates the risk for coronary diseases using Framingham Risk Score Algorithm, patients at high risk are also alerted by the application.

1.2) Objective

Short-term –

The short term objective of this paper is to ponder the writing on portable frameworks and applications right now accessible, and additionally the current applications identified with cardiology and then to characterize the outcomes to perceive what is accessible and what is missing in these applications, concentrating especially on business applications..

Long-term –

Long-term objective includes that students are exposed to the industrial environment which should help us in future when we will work in real-time projects. One should gain experience of working with a team and learn to cooperate with our team members. The major objective of the research is to utilize the technical knowledge to create a mobile application for the benefit of the society by reducing risk factor by calculating heart disease risk core using Framingham Risk Score calculation algorithm as a base.

1.3) Methodology

UI Design- We make an application which goes about as an interface between the client and the cloud. The application collects the data from user about general habits such as smoking, diabetes, cholesterol level etc. and calculates heart disease risk score.

Cloud Implementation- We have implemented the cloud server using Amazon Web Services (AWS). The technology allows subscribers to have at their disposal a full-fledged virtual cluster of computers available all the time, through the Internet.

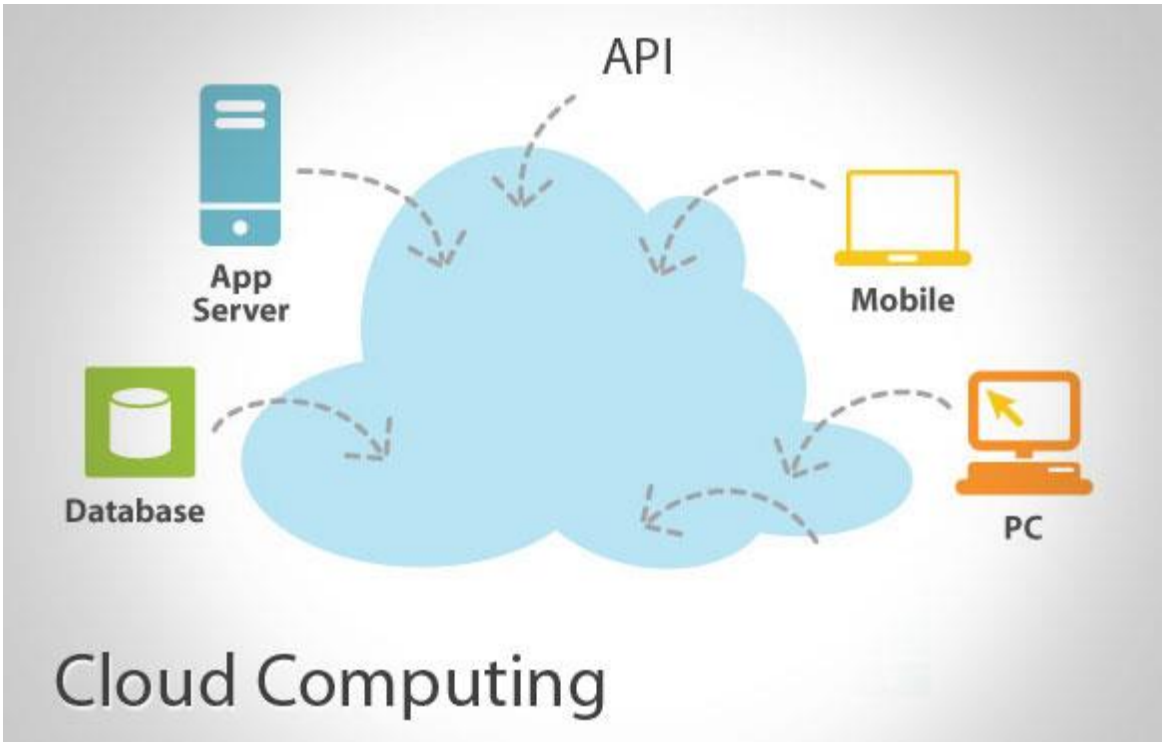


Fig-1: Cloud Computing Model.

Implementation of application using AWS Lambda Function: We have created an API using Amazon Web Services Lambda Function which can be implemented on various platforms such as mobile and web. The code you keep running on AWS Lambda is known as a "Lambda function." After you make your Lambda function it is constantly prepared to keep running when it is activated, like an equation in a spreadsheet. Each capacity incorporates your code and also some related arrangement data, including the function name and resource requirements. Lambda capacities are "stateless," with no liking to the hidden framework, so Lambda can quickly dispatch the same number of duplicates of the capacity as expected to scale to the rate of approaching occasions.

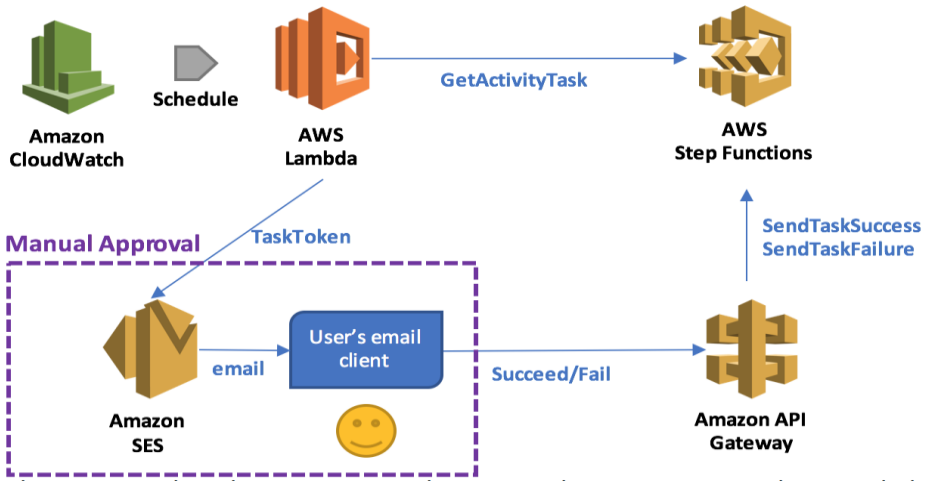


Fig-2: AWS Lambda Function Model.

Framingham Risk Score: The Framingham Risk Score is a sex particular calculation used to appraise the 10-year cardiovascular danger of a person. The Framingham Risk Score was first created in light of information got from the Framingham Heart Study, to appraise the 10-year danger of creating coronary illness.

CHAPTER 2

LITERATURE SURVEY

2.1) Title: “Cloud based intelligent system for delivering health care as a service.”

The promising capability of distributed computing and its union with advances, for example, portable processing, remote systems, sensor innovations considers creation and conveyance of more current sort of cloud administrations. In this paper, we advocate the utilization of distributed computing for the creation and administration of cloud based medicinal services administrations. As an agent contextual analysis, we plan a Cloud Based Intelligent Health Care Service that performs continuous observing of client wellbeing information for conclusion of perpetual ailment, for example, diabetes. In addition, infrastructure level systems are proposed to give dynamic asset versatility. Test comes about exhibit that arrangement exactness of 92.59% is accomplished with our model framework and the anticipated examples of CPU utilization offer better open doors for versatile asset flexibility.

2.1.1) Introduction

As of late, mechanical advancements have prompted the improvement of new processing foundations, for example, distributed computing that gives framework and programming on rent to end clients. The "pay for utilize" estimating model, on-request registering and pervasive system get to permit cloud administrations to be open to anybody, whenever, anyplace. The inalienable advantages like quick organization bring down costs, versatility, fast provisioning, moment flexibility and more prominent strength, fast reconstitution of administrations, ease debacle recuperation and information stockpiling arrangements guarantees the possibilities of distributed computing.

2.1.2) Architectural Overview

Research's goal is to plan a Cloud Based Intelligent HealthCare Service (CBIHCS) that performs constant checking of user wellbeing information for recognizable proof of perpetual ailment such as diabetes. Our framework is non sufficiently specific to oblige diagnosis and identification of numerous ailments by dissecting the user wellbeing information put away in cloud archives. Be that as it may, this research centers around just a single particular utilize case, to be specific, identifying clients as "Diabetic" or "Non-Diabetic".

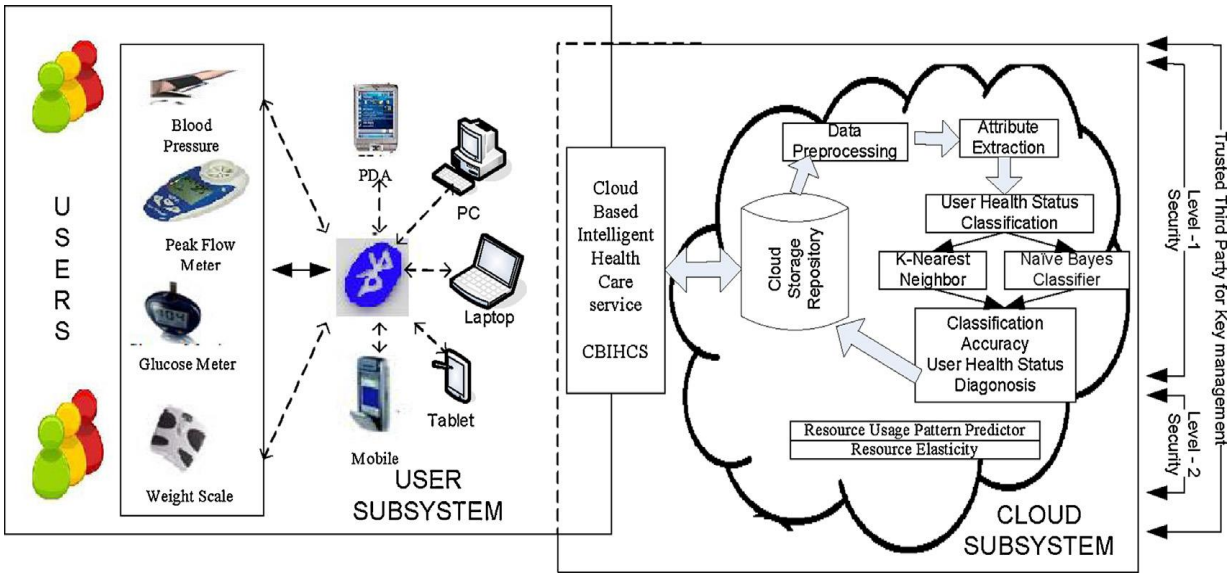


Fig-3: Detailed architecture of CBIHCS.

2.1.3) Cloud Subsystem

Research's goal is to plan a Cloud Based Intelligent HealthCare Service (CBIHCS) that performs constant checking of user wellbeing information for recognizable proof of perpetual

Ailment such as diabetes. Our framework is non sufficiently specific to oblige diagnosis and Identification of numerous ailments by dissecting the user wellbeing information put away in cloud archives. Be that as it may, this research centers around just a single particular utilize case, to be specific, identifying clients as "Diabetic" or "Non-Diabetic" .The analysis procedure includes a few calculations on stored information that principally comprise of information pre processing, attribute determination and arrangement. Once the investigation is complete the data would then be able to be handed-off remotely to doctors, paramedics and patient's cell phones for definite validation and clinical determination. The undertaking named transmit information is accountable for transmitting information with the goal that approval can be performed. After the information is affirmed, the specialists may recommend medicine and spare in client open cloud storage. The errand validates and stores the data.

2.1.4) Security aspects of CBIHCS

To address the security challenges in a cloud facilitated android application, we execute security instruments at multiple levels and give part based access control to guarantee the protection of basic restorative information of patients. The two sorts of user parts characterized by CBIHCS are (1) Owner; (2) Trusted Partner .The tolerant whose restorative information dwells in our cloud hosted web application is assigned as the Owner 'O' of information. Additionally, now and again, the proprietor may wish to share his personal medical information with a gathering of individuals for counsel or other purposes. We call such individuals as Trusted Partners (TP) of the owner. To execute security components, we propose to inter grind the utilization of symmetric cryptosystems for authentication and part based access control (RBAC) instruments for authorization. Clients of CBIHCS are recognized by an extraordinary client name. As opposed to putting away the client secret key in plaintext on cloud based capacity, we scramble the watchword with a Private Key (PK) issued by a Trusted Third Party (TTP). A TTP is an element that guarantees the security

Bolster for information and communications traded between the depending client parties. In expansion, it guarantees that exclusive true blue clients who are registered with CBIHCS can get to its functionalities while preventing counterfeit clients (who might be framework proprietors or users with director benefits) to get to the information of other users.

2.1.5) Data Preprocessing

Current medicinal services gadgets come equipped with assortment of instruments that enable patients to automatically transfer information from various meters. Be that as it may, such data might contain some clamor segments or missing sample necessitating the execution of preprocessing steps. Information pre-handling involves exhaustive examination of crude data followed by resulting information reconciliations, transformations and decrease for formal investigation. Misshaped esteems or missing readings regularly end up deluding in the formal analysis. Henceforth, adequate number of value tests must be collected and dissected to take into account basic assessment.

S.No.	Attribute	Description
1	Age	Age of user in years.
2	Gender	Whether the user is male or female.
3	Weight	Weight of user in Kgs.
4	BMI	Body Mass Index of user.
5	Total Cholesterol	Total Cholesterol level of user(mm/Hg).
6	Smoking Habits	Whether a person smokes (Scale 1-5).

Table-1: Health attributes of user.

2.1.6) Data Acquisition

In our diabetic contextual investigation, we grouped client wellbeing status based on the

Preparation information gathered from 65 subjects including 35 guys and 30 females. The subjects go in the age of18– 85 years and the mean age is 52 years with a standard deviation of 7.5. Client's own information is recorded one time by our web benefit interface and resulting wellbeing information esteems are recorded persistently consistently for a time of 7 days using electronic well being gadgets. After the seventh day, the recorded values are spared to a document and used in our examinations. Out of the aggregate 65 subjects, 40 subjects are "Diabetic" and 25 are "Non-Diabetic". Information of 23 diabetic subjects is chosen for.

S.No.	Attribute	Mean	Standard Deviation
1	HDL Cholesterol	60.41	10.91
2	BP Systolic	138.54	15.22
3	Weight	75.68	8.48
4	BMI	29.21	6.69
5	Total Cholesterol	169.52	15.32
6	LDL Cholesterol	80.10	10.42

Table-2: Statistical Analysis of gathered data.

2.1.7) Conclusion

In this report, researchers have displayed a Cloud based Intelligent Heart Health Care Service (CBIHCS) that performs continuous screening of client well being information gathered from

Different remote sensory health mind types of gear. It applies Principal Component analysis for trait determination and K-NN and Naïve Bayes for user health status characterization. In our work, we characterized the user as diabetic and Non-diabetic. K-NN accomplishes better classification precision and more prominent affectability and specificity measures indicating that it has better potential for infection identification in genuine situations. Aside from that, we used standard statistical forecast systems that infer the resource usage designs for CBIHCS and propose basic heuristics to perform dynamic foundation flexibility. Experimental results led on Amazon EC2 obviously show the effectiveness of our approach in guaranteeing a steady level of application execution. A practical, universally accessible and a profoundly focalized human services arrangement along these lines become achievable with CBIHCS.

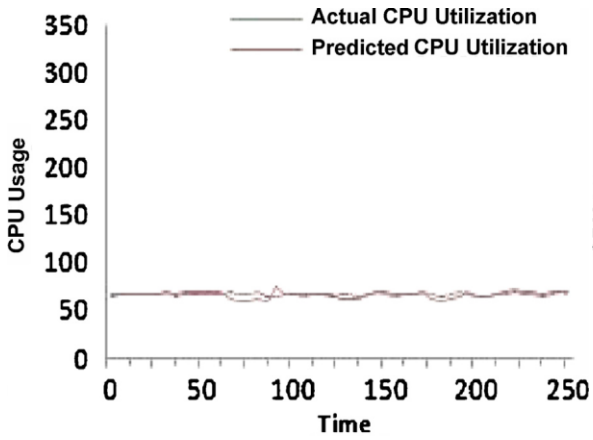


Figure-4: CPU usage for varying data.

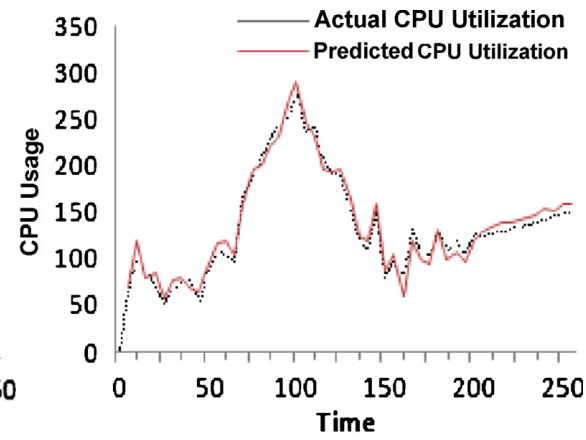


Figure-5: CPU usage for steady data.

2.2) Title: "Patient Monitoring System Using Android."

Telemedicine is a quickly creating use of facility pharmaceutical where restorative data is

Exchanged through the telephone or web or different systems to consult and performing remote therapeutic systems or examinations. Telemedicine can be connected to a more prominent stretch out in the field of cardiology where ECG fills in as the significant device. This task expounds the experience; a philosophy received and features different outline angles to be considered for making telemedicine in persistent observing framework compelling. In this technique, the patient's crucial signs like ECG, heart rate, breathing rate, temperature, SpO2 are caught and the qualities are gone into the database. It is then transferred into the electronic server also, sent to the specialist's telephone utilizing ANDROID innovation. It likewise empowers the specialists to right away send back their input to the attendant station.

2.2.1) Introduction

The advanced visionary of medicinal services industry is to give better social insurance to individuals whenever and anyplace on the planet in a more financial and patient well disposed way. Subsequently to increase the patient care effectiveness, there emerges a need to enhance the patient observing gadgets and make them more versatile. The medicinal world today faces two essential issues with regards to quiet checking. Right off the bat, the requirements of wellbeing care's supplier's essence close to the bedside of the patient and furthermore, the patient is confined to informal lodging to extensive machines. With a specific end goal to accomplish better quality patient care, the above referred to issues must be understood. As the bio instrumentation, PCs and media communications advances are propelling, it has progressed toward becoming possible to plan more gateway essential sign tele checking frameworks to procure, record, show and to transmit the physiological flag from the human body to any area. Latest works in correspondence advances have enlivened the advancement of telemedicine to a huge degree. Telemedicine benefits not just the clients who can get human services all the more effectively; it additionally benefits the specialists who can streamline their endeavors to help more patients.

2.2.2) Software Used

Java -

Java is utilized as a part of a wide assortment of registering stages from inserted gadgets and cell phones on the low end, to big business servers and supercomputers on the top of the line. While less normal, Java applets are some of the time used to give enhanced and secure capacities while perusing the World Wide Web on PCs.

Android -

Android is a Linux-based working framework for cell phones, for example, advanced mobile phones and tablet PCs. It is created by the Open Handset Alliance drove by Google. Google discharges the Android code as open-source, under the Apache License. The Android Open Source Project (AOSP) is entrusted with the upkeep and advance improvement of Android. The rendition utilized here is Android 8.0 Oreo was released, in light of Linux portion 2.6.32 is utilized to complete out our task work**.**

Eclipse -

Eclipse is an open source group, whose undertakings are centered around building an open improvement stage involved extensible structures, devices and runtimes for building, sending and overseeing programming over the lifecycle. The Eclipse SDK comprises of the Eclipse Platform, Java advancement apparatuses and the Plug-in advancement Environment.

XAMPP –

This is planned for utilize just as an improvement apparatus, to permit web specialists and software engineers to test their deal with their own PCs with no entrance to the Internet. XAMPP 1.8.1 for Windows.

2.2.3) Patient Parameters

Electrocardiography is a trans thoracic (over the thorax or chest) elucidation of the electrical movement of the heart over some undefined time frame, as identified by anodes appended to the surface of the skin and recorded by a gadget outer to the body. The chronicle delivered by this non-intrusive technique is named an electrocardiogram (additionally ECG or EKG).An ECG is utilized to quantify the rate and consistency of heartbeats, and in addition the size and position of the chambers, the nearness of any harm to the heart, and the impacts of medications or gadgets used to direct the heart, for example, a pacemaker. Most ECGs are performed for symptomatic or explore purposes on human hearts, however may likewise be performed on creatures, for the most part for conclusion of heart variations from the norm or research.

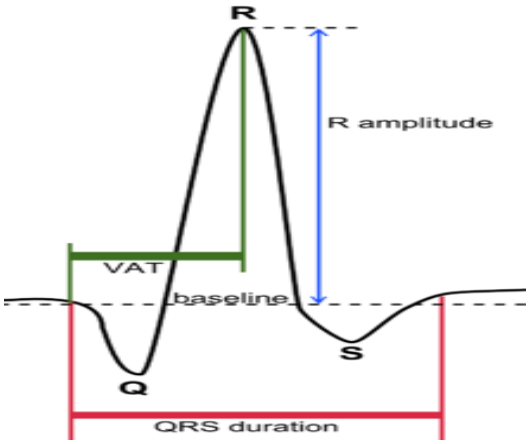


Figure-6: Wave Description

An ordinary resting heart rate for grown-up's reaches from 60 to 100 thumps per minute. For the most part, a lower heart rate at rest suggests more productive heart capacity and better cardiovascular wellness. There are numerous manners by which the Heart Rate accelerates or backs off. Ordinary resting heart rates go from 60-100 bpm. Bradycardia is characterized as a

resting heart rate underneath 60 bpm. Be that as it may, heart rate from 50 to 60 bpm is regular among sound individuals and don't really require unique consideration. Tachycardia is characterized as a resting heart rate over 100 bpm, however tireless rest rates between 80-100 bpm, primarily on the off chance that they are available amid rest, may be indications of hyperthyroidism or frailty.

Hyper acute T waves	Perhaps the primary indication of intense myocardial localized necrosis, where T waves turn out to be more conspicuous, symmetrical, and pointed
Shortened QT interval	Hyperkalaemia, some drugs, certain genetic abnormalities, hyperkalaemia.
Prolonged QT interval	Hypocalcaemia, some drugs, certain genetic abnormalities.
Flattened or inverted T waves	Coronary ischemia, hypokalaemia, left ventricular hypertrophy, digoxin effect, some drugs.
Peaked T wave, QRS wide, prolonged PR, QT short	Hyperkalemia, treat with calcium chloride, glucose and insulin or dialysis.
Prominent U waves	Hypokalemia.

Table-3: Pathological Variations in ECG.

2.2.4) Database Creation

The Control board is utilized to make the database .The primary programming utilized for this is XAMPP. Most of the part comprises of Apache, MySQL, FileZilla, and Mercury

Tomcat. In this undertaking, Apache and MySQL are utilized to make the database. The alternatives Apache and MySQL are begun. Later the administrator of MySQL opens to another window. On opening the database ICU Biomedical and the alternative patient points of interest will lead us to the patient points of interest window.

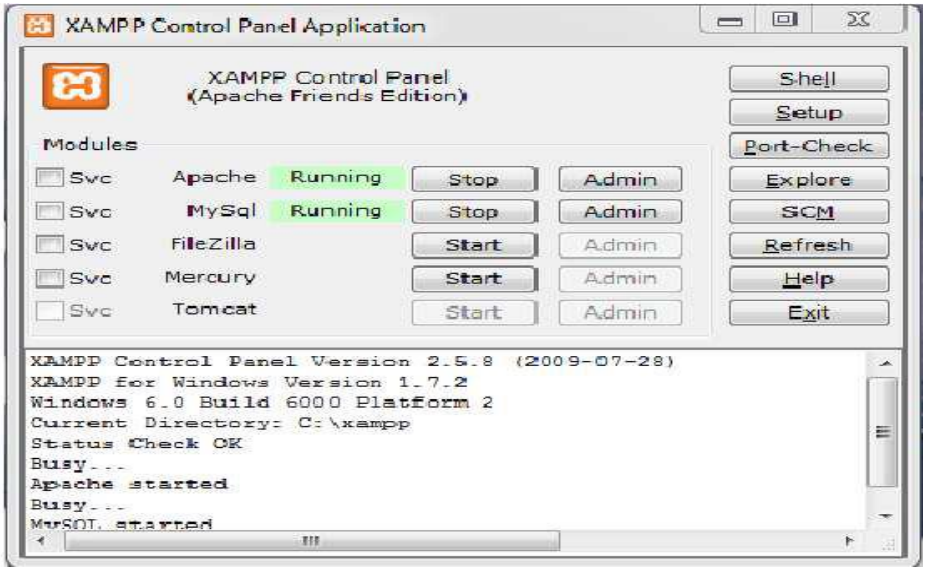


Figure-7: Control Panel

The 'patient detail window' is utilized to make a database on quiet subtle elements containing the essential data of the patients which incorporates persistent ID, name, portable number, address, age, sexual orientation and input. There are separate alternatives to alter and make the database with the goal that the information can be refreshed on time.

The patient parameters window helps in including a database containing the patient parameters such as patient ID, ECG, pulse rate, heart rate, SpO2, temperature, R wave amplitude and QRS complex width. There are separate options to edit and create the database so that the data can be updated on time

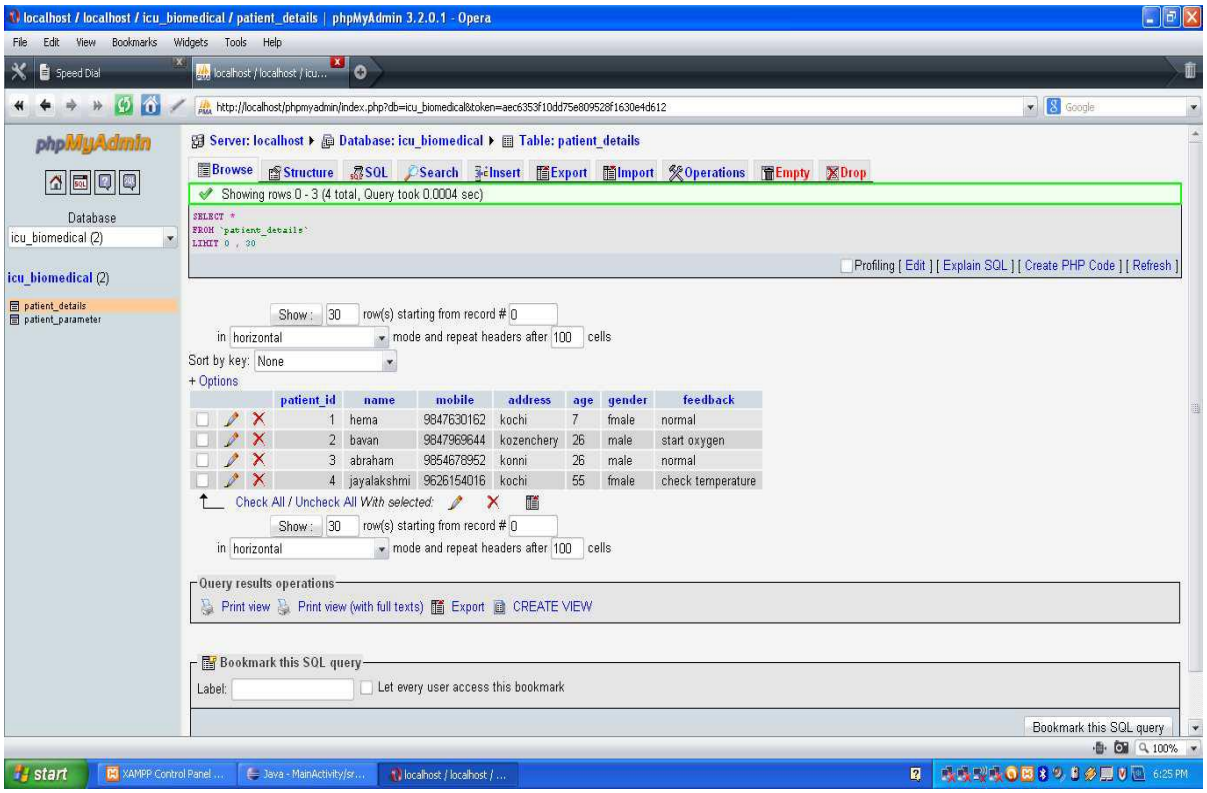


Figure-8: Patient Details

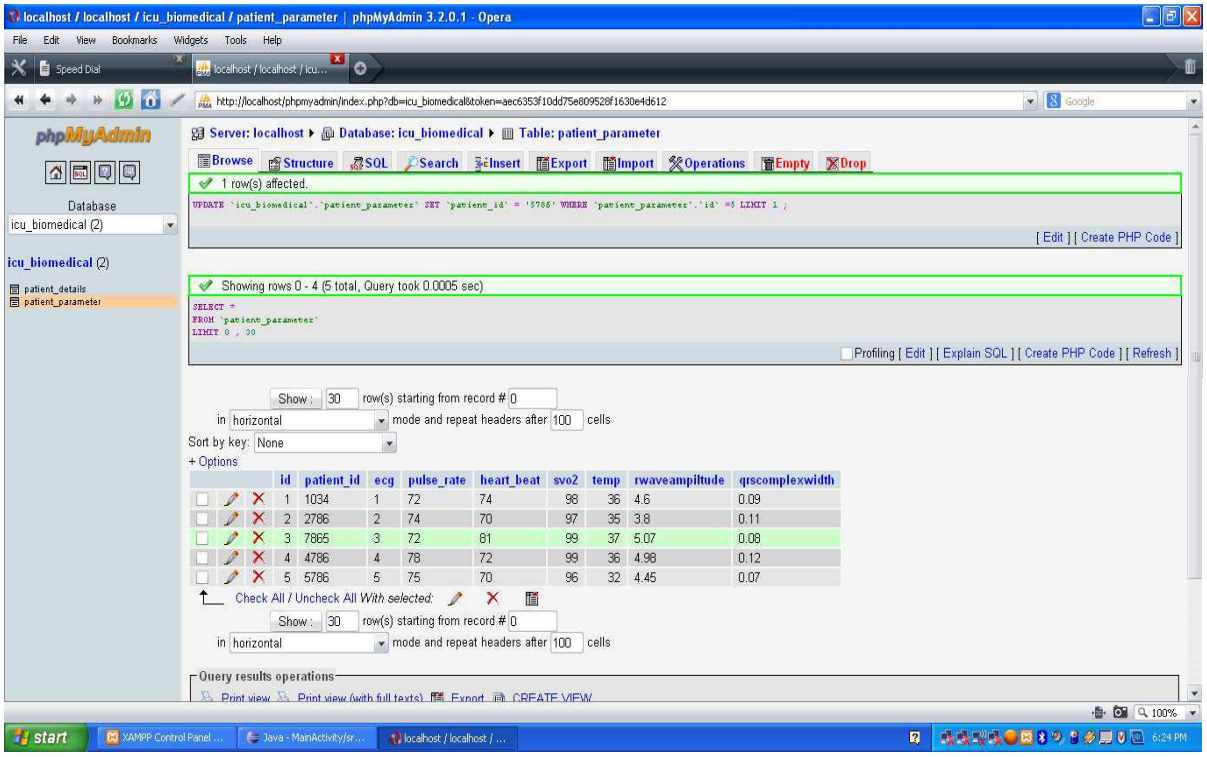


Figure-9: Patient Parameters

2.2.5) Conclusion

The above discussed project demonstrates the patient's essential parameters, for example, ECG, heart rate, SpO2, beat rate and temperature are estimated utilizing a patient checking framework. These qualities are sent into a database and are transferred into an online server physically; there is a function for entering the data in database automatically later on. Additionally, the whole points of interest of the patient experiencing different constant maladies like malignancy, Alzheimer's and so on can be sent to a specialist sitting abroad to dissect and prescribe the sort of treatment and drugs for the analysis of the sickness.

2.3) Title: “Heart Disease Monitoring System Using Web and Smartphone Android.”

The heart Diseases cause a great many demises overall due to the expansion in the maturing populace and the ascending of social insurance costs. There is likewise a request of value medicinal services from remote areas. Innovative progressions in the field of restorative hardware and correspondence can help diminishing the cost of social insurance. In this paper an ongoing coronary illness checking framework is presented. The system extricates the ECG motion from the patient, sends it through the Internet and stores it into a doctor's facility server. The framework additionally forms the ECG utilizing MATLAB to alarms the specialist and doctor's facility staff by sending email and SMS message if any irregularity is distinguished. The framework likewise actualizes an application in view of Android stage to give online data about the patient status, for example, the patient's heart beat rate, ECG, tolerant history and gives new perusing each 30 minutes. The framework empowers specialists to remotely catch up the status of their patient utilizing their PC and PDAs. The framework was tried and checked by therapeutic group for approval.

2.3.1) Introduction

Heart disappointment and stroke cause a major weight on society because of their high expenses of care bring down personal satisfaction and unexpected passing. It causes million

of death overall. The observing of patients physiological data are vital for the further treatment. Numerous patients can profit by ceaseless observing as a piece of a symptomatic method, ideal upkeep of an unending condition or amid regulated recuperation from an intense occasion or surgical technique. The remote observing additionally empowers the patient to experience his ordinary live and help diminishing the cost of human services. With the current progress in IC plan, the figuring power and the memory size of cell phone have expanded extensively. This improvement makes numerous cell phones, fit for doing complex figuring undertakings and consequently can be utilized as a part of checking coronary illness persistent remotely medicinal services from remote areas. Innovative progressions in the field of restorative hardware and correspondence can help diminishing the cost of social insurance.

In this paper a constant ECG framework that encourages the observing and follows up of the patient's condition is executed. The electrocardiogram (ECG) is a test that records the electrical movement of the heart. ECG deciphers the heart's electrical action into line tracings on paper. These records hold critical data that can be utilized to quantify how quick the heart is thumping and to identify the anomaly of the heartbeats. The calculation energy of cell phones is used in this framework. The proposed framework has great extensibility and can without much of a stretch consolidate other physiological signs to suit different tele-wellbeing situations. The framework has two sections one conveyed by clients, e.g. interminable patient, and the second conveyed by specialist organizations e.g., the restorative specialists.

2.3.2) System Architecture

The figure below shows the system architecture design.

Patient unit subsystem: This incorporates terminals that sense the electrical action experiencing the heart, flag intensification circuit, molding circuit, information securing

Circuit and home entryway: The circuit takes a perusing at regular intervals and sends it to home passage PC.

Web Server and Database subsystem: To store the patient ECG flag information, recognize any variation from the norm in the ECG flag and distribute the outcomes that can be accessed just by approved individuals.

Android unit subsystem: Android based application that empowers specialists to get to the patient points of interest utilizing advanced cell.

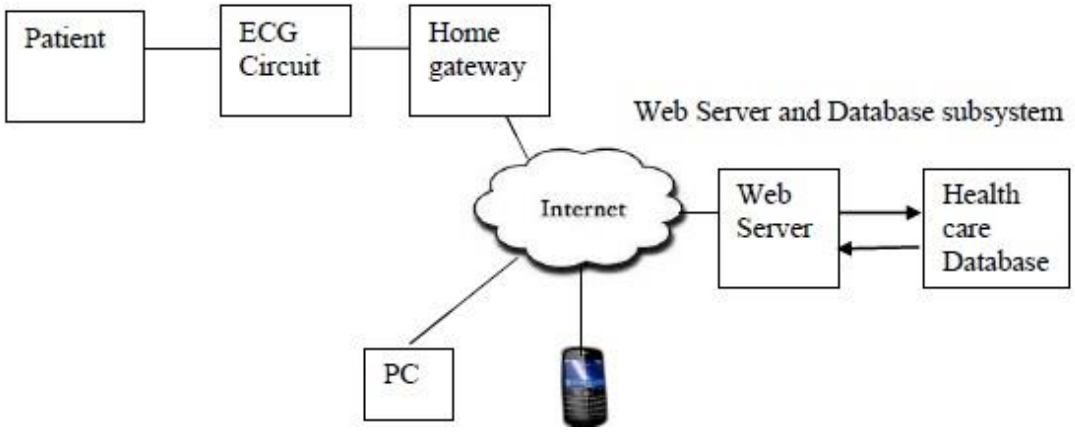


Figure-10: Remote ECG System Architecture

2.3.3) Patient Subsystem Architecture

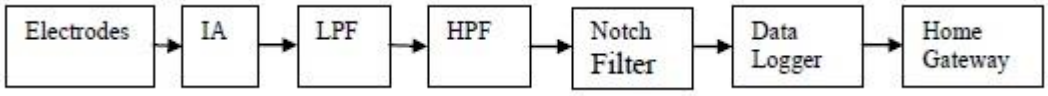


Figure-11: Patient Subsystem Architecture

2.3.4) Web Server Implementation

The ASP.NET website gives the doctors the access to patient's current heart condition ubiquitously and remotely.

The website has three types of users-

Web Admin-

In charge of making new records for the two patients and specialists, deal with their profiles, and the link between specialists to patients.

Doctor-

Can get to patients' ECG readings, include and see remarks, include and see medicinal reports of his patient composed by him/her or another specialist responsible.

Patients-

Can access his/her medical reports and view the doctor / doctors assigned to him/her.
The doctor can "Filter" or search the patients either by name, File number, ID, or ID type. In order to get the full list he / she just can click the "Select" button.

2.3.5) Android Unit Subsystem

Doctor Application (ECG Note)-

Specialist home page contains a rundown of the patient's names. The status of every patient can be seen inside every patient name, to show whether he/she are on the web or not. The Green light or the Red will be appeared to demonstrate whether the patients are in threat or

not. This is valuable for the crisis cases, so the specialist can discuss specifically with the patient. The specialist can get to any patient information such as ECG condition and other vital data like heart beat rate by pushing on his/her photograph.

Patient Application (My Note)-

The patient home page contains a rundown of symbols. These symbols are "Your condition", "schedule", "offer", and "options". The status of the patient can be seen at the highest point of the page, which is shown in a Green or Red square.

2.3.6) Results and Discussions

The framework have been executed, tried and accomplished the plan goals said in segment above which are the outline of moderate and modest ECG framework as contrasted and the accessible ECG frameworks and additionally the gushing of the patient ECG to the social insurance supplier site so specialists can screens their patient status progressively. Included highlights like distinguishing the variations from the norm in quiet ECG and send ready message promptly to specialists through email and SMS is likewise executed and tried. Finally Android based application that encourages specialists to catch up their patients remotely by means of PDA is executed and tried.

Figure-12: Patient Page Designed

2.4) Title: “Developing web services using server less architecture.”

Service- oriented architecture (SOA) and distributed systems have turned out to be well known arrangement to settle holes between business needs and Information Technology administrations. Having generally embraced as urgent routine with regards to SOA web administrations empower correspondence between programming segments or applications over the web. This exhibits an imaginative way to deal with manufacture web administrations without worries about server arrangement. The approach is plan relic yielded from the outline science process which comprises of server less design and a training to execute into web administrations venture. So as to demonstrate the idea, the server less engineering and proposed rehearse are utilized to build up a model web benefit application.

2.4.1) Introduction

SOA and Distributed systems are produced by ventures to serve different business spaces, from letter mail and managing account administrations. The central of SOA is to discharge autonomous and self-depicting programming segments which are all around accessible and open over a system by means of standard correspondence conventions. As a noteworthy execution web administrations innovation has embraced this idea in building present day web application.

The design characterizes the partition of worries among frontend (the show of information and connection with clients), information sources (the tireless stockpiling of utilization's information), backend (the intercedes amongst frontend and information sources that handles business rationale). The decoupling idea enables engineers to execute noteworthy changes in a layer without affecting alternate levels and subsequently of effectively viable programming. From the point of view of SOA outline business rationale and information layers can be actualized as web administrations, empowering correspondence by means of standard web based conventions. As of late, web administrations distribute APIs to teach their customers how to set up association and trade information.

2.4.2) Advantages and Disadvantages

We worked with AWS cloud condition that aides boosting up the exploration advance. Second, Amazon is the pioneer of giving essential administrations to building server less applications , and thus of sufficient encounters in investigating and support.

Amazon offers a complementary plan program to get to AWS cloud items at sensible cost, fitting motivation behind test and spending plan. The execution is fundamentally done utilizing Node.js and JavaScript programming dialect, which is appropriate to the model of occasion driven and non-blocking I/O. The examination is constrained to this programming dialect and its relating advances. The investigate does not put database plan into thought with the goal that database advances are not specified. Likewise, the data security is barred from the examination. As the execution is done on the AWS, the creator appoints this obligation to the supplier. Amazon's most noteworthy need is to construct security-touchy server farm and system engineering for clients. Hence the execution can be led with trust in a protected cloud condition.

2.4.3) Terminology

Web benefit is broadly conceded as a significant routine with regards to benefit situated designs empowering interchanges between programming parts or business applications over the web. As various programming or applications are executed with various programming dialects, equipment particulars of a web benefit are completely autonomous of them to give a standard and all inclusive technique for information trade. In different words, the basic is that no information of buyers is required to build up a web administration, and the other way around. This additionally encourages simple and brisk combination of existing programming arrangements and advancement of disseminated applications. Fensel et al demonstrated the six fundamental rules that depict the basic definitions and attributes of web benefit, which is quickly represented in the accompanying table.

Loose coupling	Service is stateless, not tightly coupled to consumers and to its logic and Implementation.
Contracted	Contract speaks to an administration, characterizing its sources of info, yields, and get to approaches, quality necessities, and Error-handling Procedures.
Discoverable	Service is discoverable at execution time.
Addressable	Service is uniquely identifiable in the network.
Distributed	As being isolated by geographic and machine limits benefit is competent for tending to loss of communication.
Point-to-Point	A consumer uses only one producer at a time.

Table-4: Basic Principles of Web Service

Web administrations empower remote access to capacities by means of the web by using an arrangement of guidelines. The key strategy is to give designers a mean of information correspondence and trade which is free from the worries towards programming dialect and working framework.

2.4.4) Architecture Components

Segments of server less engineering are server less registering stage, API (Application Programming Interface) holder and framework task checking apparatus, database and other supplement administrations. These segments are distinctive administrations gave and kept up by cloud benefit seller. Among them, figuring stage executes application code to deal with business rationale, including database access and control.

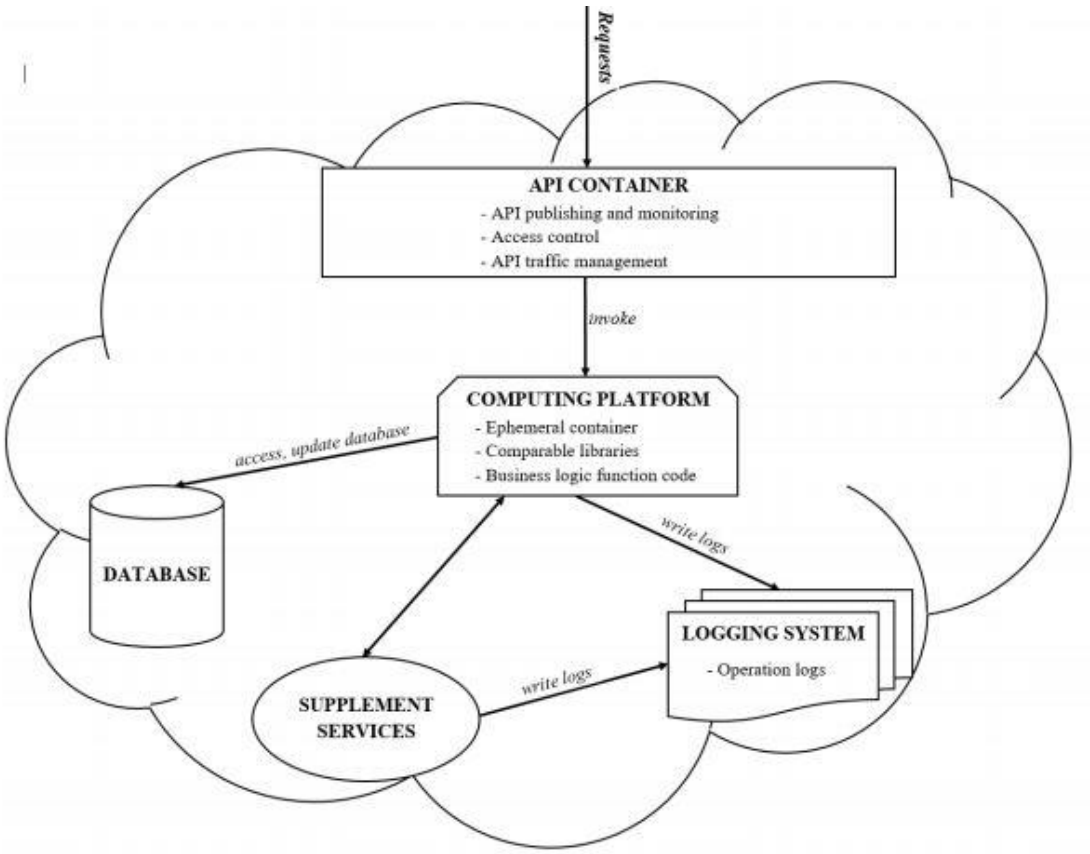


Figure-13: Major Components of Server Less Architecture

The above figure shows such real parts of the server less design. A Request to application is done through API call, which initially goes to API holder. The holder checks whether this call is approved, and extricates information that being sent in the body or inquiry parameters. Relating capacities in figuring stage are summoned to keep preparing the demand. The capacities execute business rationale with input information, set up association with database to recover or control information records if important, at that point set up a reaction to send back to the requester. Meanwhile, point by point data identifying with the procedure are followed and signed into the task observing device. They are accessible to be recovered for examination if there should arise an occurrence of episode. Observing device ought to have the capacity to track and store task information caused by any supplement administrations being incorporated into server less application.

2.5) Research Motivation

In 2014, Amazon presented Lambda work on AWS - an occasion driven and registering stage. The stage permits code to be executed in light of occasions without arrangement of servers or register assets. Together with the dispatch of API (Application Programming Interface) Gateway, AWS Lambda has prompted the idea of server less design. Since most recent two years, server less has turn into a rising subject in the realm of programming designs. The group has seen its rise with a few open-source structures and devoted meetings.

Amazon API Gateway gives a completely oversaw cloud-based condition for discharging and looking after APIs. It goes about as passageway for applications to associate with business rationales, functionalities or recover information from different AWS administrations. Code running on AWS Lambda can be unquestionably summoned by sending solicitations to Programming interface Gateway, empowering probability to build up an entire web benefit. In this manner, AWS has given sufficient foundation to make web administrations with no worries about servers (physical or virtual) design and screens.

The server less idea has slowly been in spotlight in which there are various choices for improvement stage offered by lofty suppliers. In addition, in the product organizations, there is a requirement for web benefits in which can be grown quickly, profoundly adaptable, and require irrelevant support exertion. Since the declaration of AWS Lambda and API Gateway utilizing them had been of most extreme intrigue. Be that as it may, a bringing up issue has been the means by which to execute proficiently and sort out forming of source code with regards to ceaseless reconciliation and sending.

CHAPTER 3

SYSTEM DEVELOPMENT

3.1) Software Requirements:

- Postman
- AWS Lambda function
- JDK(Java development kit)
- Maven
- Android Studio

3.2) Hardware Requirements:

- CPU: 2.2 GHz Processor and above
- RAM: 4 GB or above
- OS: Windows or Linux

These hardware requirements are for the computer on which the code has to be written.
The actual code will be computed using AWS resources. A Lambda function is created to return a simple JSON message in each invocation. In a web service, to call this function, a dedicated API endpoint is defined together with a corresponding HTTP method. API Gateway is adopted to publish and manage such endpoint.

3.3) System Design:

3.3.1) AWS Lambda Function:

Amazon gives an online UI for snappy access and arrangement of their administrations, known as Amazon Web Services (AWS) administration support. However, it is easy to make new Lambda Function and offer settings to it from the AWS support. The most essential

new Lambda Function and offer settings to it from the AWS support. The most essential setting is runtime - execution condition of capacity code. The runtime is picked relying upon decision of improvement group for programming dialect and System.

In this research, we decided to code server less application in Android studio, therefore the runtime should be JDK. AWS Lambda also supports Python, Java and C# languages. Handler, is a method dedicated for each Lambda function, which is the beginning point whenever the function is called. On the one hand, memory is desired computation resources allocated for a Lambda function, meaning that higher memory is more powerful. Timeout defines maximum execution time of a function to prevent infinite run. In other words, whenever reaching timeout AWS Lambda stops a running function. While configuring Lambda function, developers can place code directly on the AWS console.

3.3.2) API Gateway:

To construct an API with Lambda integrations, you can utilize either the Lambda proxy integration or the Lambda custom integration. All in all, you should utilize the Lambda proxy integration for a deft and streamlined API set up while giving adaptable and capable highlights. The custom joining might be a superior offer in the event that it is essential for API Gateway to pre-process approaching solicitation information before it ranges to the backend Lambda work. In any case, it is an inheritance innovation. Setting up a Lambda custom coordination is more required than setting up the Lambda proxy joining and the current setup is probably going to be inoperable when the backend Lambda work requires changes in its info or yield.

With the Lambda proxy integration, the contribution to the incorporated Lambda capacity can be communicated as any blend of demand headers, way factors, question string parameters, and body. Furthermore, the Lambda capacity can utilize the API design settings to impact its execution rationale. For an API engineer, setting up a Lambda intermediary incorporation is straightforward. Programming interface Gateway arranges the joining

Solicitation and reconciliation reaction for you. When set up, the coordinated API technique can develop with the backend without changing the current settings. This is conceivable on the grounds that the backend Lambda work engineer parses the approaching solicitation information and reacts with wanted outcomes to the customer when nothing turns out badly or reacts with blunder messages when anything turns out badly.

With the Lambda custom coordination, you should guarantee that the contribution to the Lambda work is provided as the mix asks for payload. As an API engineer, must guide any information the customer provided as demand parameters into the best possible reconciliation ask for body.

3.3.3) Spring Boot:

The general gist here is that this library provides a few abstract classes which build on top of the AWS Lambda Java API's **RequestHandler** interfaces. Instead of implementing AWS's interfaces directly, we extend one of these new abstract classes and we can now use spring's dependency injection with your Lambda functions.

The meat of the Spring AWS Lambda library is the **SpringRequestHandler** class. Basically, instead of implementing **RequestHandler** directly, we extend this class and provide it with a Spring **Application Context**. Then we can implement as many **RequestHandler** as we want and provide injectable dependencies to them.

MainHandler is the entry point to application. On the AWS Lambda configuration console, we actually set this as your "Handler" (see screenshot below). Note that **MainHandler** will rarely have any logic itself. Instead, we simply tell it the request and response types via the generic type parameters and implement the **getApplicationContext** abstract method, which should provide a reference to our application's main **ApplicationContext**.

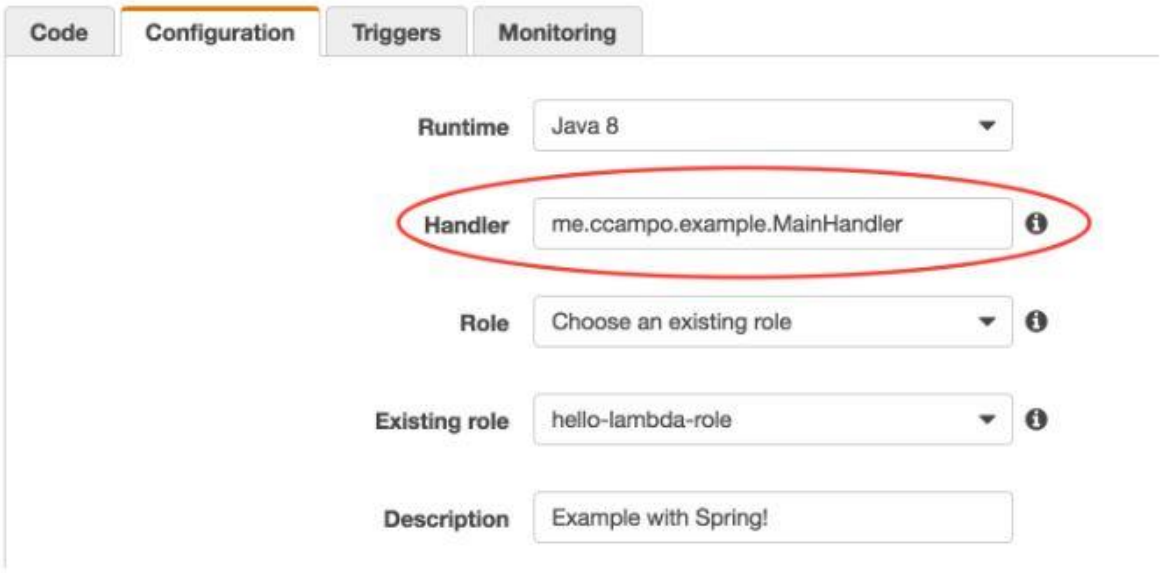


Figure-14: AWS Lambda Configuration Console

If we look at **SpringRequestHandler**, we'll notice that it actually implements the **RequestHandler** class itself and provides a default **handleRequest** method implementation, that actually retrieves a spring bean of type **Request Handler** and calls that bean's **handlerequest** method.

Since **SpringRequestHandler** gets a bean of type **RequestHandler** and calls that bean's **handleRequest** method that means that all of our main application logic should go in our own implementation of **RequestHandler** and this class should be declared a bean.

This might seem not much different than the default AWS programming model. The key difference here though is that now any **RequestHandler** we implement our self can take advantage of using spring's annotations and dependency injection. All we have to do is register it as a bean.

3.4) Use Cases:

Fully server less application use cases include:

- **Web or Mobile Backends** – Create fully server less, mobile applications or websites by creating user-facing content in a native mobile application or static web content in an S3 bucket. Then have your front-end content integrate with Amazon API Gateway as a backend service API. Lambda functions will then execute the business logic you've written for each of the API Gateway methods in your backend API.
- **Chatbots and Virtual Assistants** – Build new server less ways to interact with your customers, like customer support assistants and bots ready to engage customers on your company-run social media pages. The Amazon Alexa Skills Kit (ASK) and Amazon Lex have the ability to apply natural-language understanding to user-voice and freeform-text input so that a Lambda function you write can intelligently respond and engage with them.
- **Internet of Things (IoT) Backends** – AWS IoT has direct-integration for device messages to be routed to and processed by Lambda functions. That means you can implement server less backends for highly secure, scalable IoT applications for uses like connected consumer appliances and intelligent manufacturing facilities.

Using AWS Lambda as the logic layer of a server less application can enable faster development speed and greater experimentation – and innovation — than in a traditional, server-based environment.

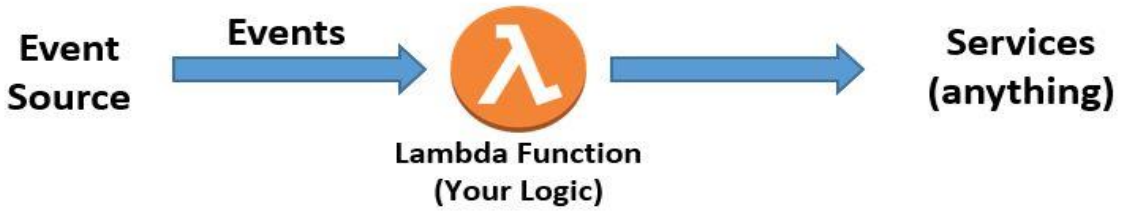


Figure-15: AWS Lambda Use case

3.5) Architectural components:

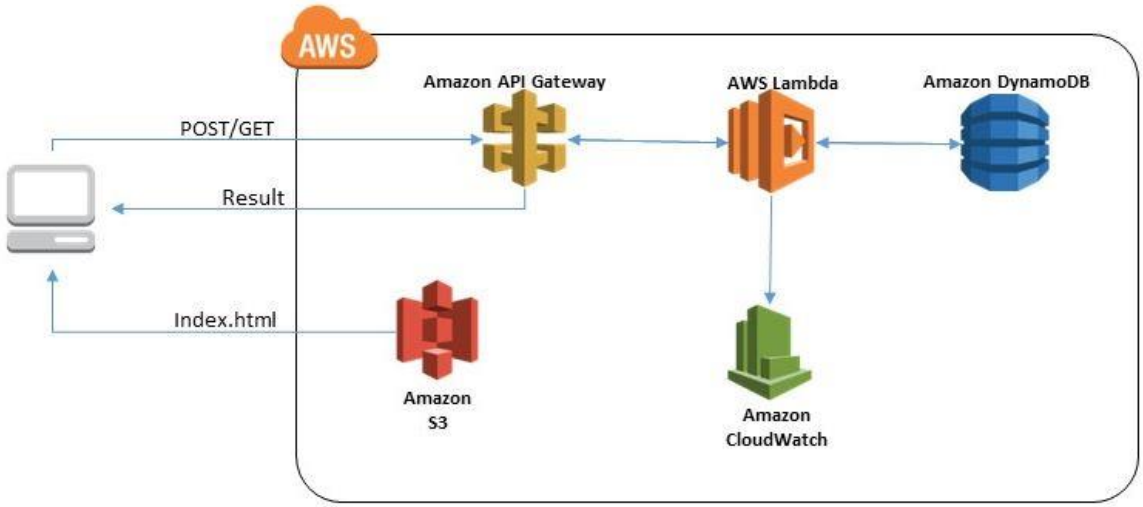


Figure-16: AWS Lambda Architecture

- **AWS S3:** We stored our front end static webpage content and images in an S3 bucket. The pages use java script to talk to a server less backend. When clients request static content it is served by S3.
- **AWS API Gateway:** Using the API Gateway service we were able to create routes to Lambda functions. The Lambda functions are then invoked in response to normal GET and POST requests. In effect the API Gateway routes act like a trigger to run the Lambda functions.
- **AWS Lambda:** This is the core AWS service for deploying applications using a server less architecture. We wrote the core components of the application using python and deployed these python components to AWS as Lambda functions.
- **AWS DynamoDB**: This managed No SQL database service is not required for a server less deployment however we made use of it as it integrates very smoothly with other AWS services. DynamoDB has great flexibility and near infinite scalability.
- **AWS Cloud Watch:** To monitor the deployed AWS resources which made up the application.

3.6) Algorithm:

Framingham Risk Score

The Framingham Risk Score is a sex particular calculation used to evaluate the 10-year cardiovascular danger of a person. The Framingham Risk Score was first created in light of information acquired from the Framingham Heart Study to appraise the 10-year danger of creating coronary heart disease. Keeping in mind the end goal to evaluate the 10-year cardiovascular infection chance, cerebrovascular occasions, fringe corridor sickness and heart disappointment were along these lines included as malady results for the 2008 Framingham Risk Score, over coronary illness.

The Framingham Risk Score is one of various scoring frameworks used to decide a person's odds of creating cardiovascular malady. Some of these scoring frameworks are accessible on the web. Cardiovascular hazard scoring frameworks give a gauge of the likelihood that a man will create cardiovascular malady inside a predetermined measure of time, more often than not 10 to 30 years. Because they give a sign of the danger of creating cardiovascular illness, they additionally demonstrate who is well on the way to profit by counteractive action. Thus, cardiovascular hazard scores are utilized to figure out who ought to be offered preventive medications, for example, medications to bring down circulatory strain and medications to bring down cholesterol levels. For instance, about 30% of coronary illness (CHD) occasions in the two people were independently owing to circulatory strain levels that surpassed high typical (≥130/85), demonstrating that pulse administration and observing is principal both to cardiovascular well being and forecast of results.

Since hazard scores, for example, the Framingham Risk Score give a sign of the presumable advantages of aversion, they are valuable for both the individual patient and for the clinician in choosing whether way of life alteration and preventive restorative treatment, and for understanding training, by recognizing people at expanded hazard for future cardiovascular events.

Coronary Heart Disease (CHD) chance at 10 years in percent can be ascertained with the assistance of the Framingham Risk Score. People with generally safe have 10% or less CHD hazard at 10 years, with middle of the road chance 10-20%, and with high hazard at least 20%. In any case it ought to be recollected that these categorizations are self-assertive.

A more helpful metric is to think about the impacts of treatment. In the event that a gathering of 100 people all have a 20% ten-year danger of cardiovascular infection it implies that we ought to expect that 20 of these 100 people will create cardiovascular malady (coronary illness or stroke) in the following 10 years and eighty of them won't create cardiovascular ailment in the following 10 years. If they somehow happened to take a mix of medicines (for instance medications to bring down cholesterol levels in addition to medications to bring down circulatory strain) that diminished their danger of cardiovascular sickness considerably it implies that 10 of these 100 people ought to be relied upon to create cardiovascular infection in the following 10 years and 90 of them ought not be required to create cardiovascular ailment. In the event that that was the situation then 10 of these people would have maintained a strategic distance from cardiovascular ailment by taking treatment for a long time; 10 would get cardiovascular malady regardless of whether they took treatment; and 80 would not have cardiovascular ailment regardless of whether they took treatment.

Regardless of their broad prominence, randomized trials surveying the effect of utilizing cardiovascular sickness chance scores demonstrate constrained effect on understanding results. Despite the fact that there is great confirmation that focusing on people with high aggregate CVD chance is the most productive approach to lessen CVD-related bleakness and mortality, to date trials evaluating the handiness of hazard scores at helping clinicians target high hazard patients indicate restricted advantage.

Recognize that the most grounded indicator of cardiovascular hazard in any hazard condition is age. All people matured 70 and over are at >20% ten year cardiovascular hazard and nearly no one matured under 40 is at >20% ten year cardiovascular hazard.

Since the individuals who advantage most from treatment are those at most astounding danger So the treatment of patients with raised circulatory strain and brought cholesterol step up in their thirties. While treatment of patients with "typical" circulatory strain and "ordinary" cholesterol levels in their seventies advantages many. This provides reason to feel ambiguous about the astuteness of arranging people as having hypertension or raised cholesterol and treating these individual hazard factors without a thought of both their general danger of cardiovascular illness and of the likelihood that they will profit.

CHAPTER 4

PERFOMANCE ANALYSIS

4.1) System Testing:

Framework testing of programming is trying led on a total, incorporated framework to assess the framework's consistence with its predefined prerequisites. Framework testing falls inside the extent of discovery testing, and all things considered, ought to require no learning of the internal outline of the code or rationale.

It is comparable experiment composing. In experimental thinking we ought to compose the test situations and utilize cases.

4.1.1) Black Box Testing:

Black Box testing is a strategy for programming testing that looks at the usefulness of an application without peering into its inner structures or workings.

Particular information of the application's code/inner structure and programming knowledge isn't required. The analyzer knows about what the product should do yet doesn't know about how it does it. For example, the analyzer knows that a specific info restores a specific, perpetual outcome however doesn't know about how the product generates the output in any case.

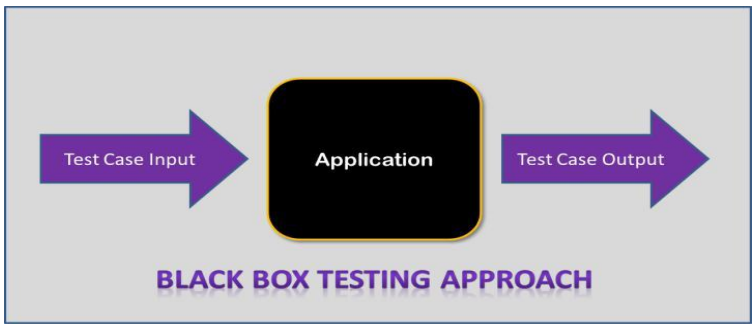


Figure-17: Black Box Testing

4.1.2) Unit Testing:

In computer programming, unit testing is a product testing technique by which singular units of source code, sets of at least one computer program modules together with related control information, use systems, and working strategies, are used to decide if they are fit to utilize. Naturally, one can see a unit as the smallest testable piece of an application. In procedural programming, a unit could be a whole module; however it is all the more ordinarily an individual capacity or technique.

The objective of unit testing is to separate each piece of the program and demonstrate that the individual parts are right.

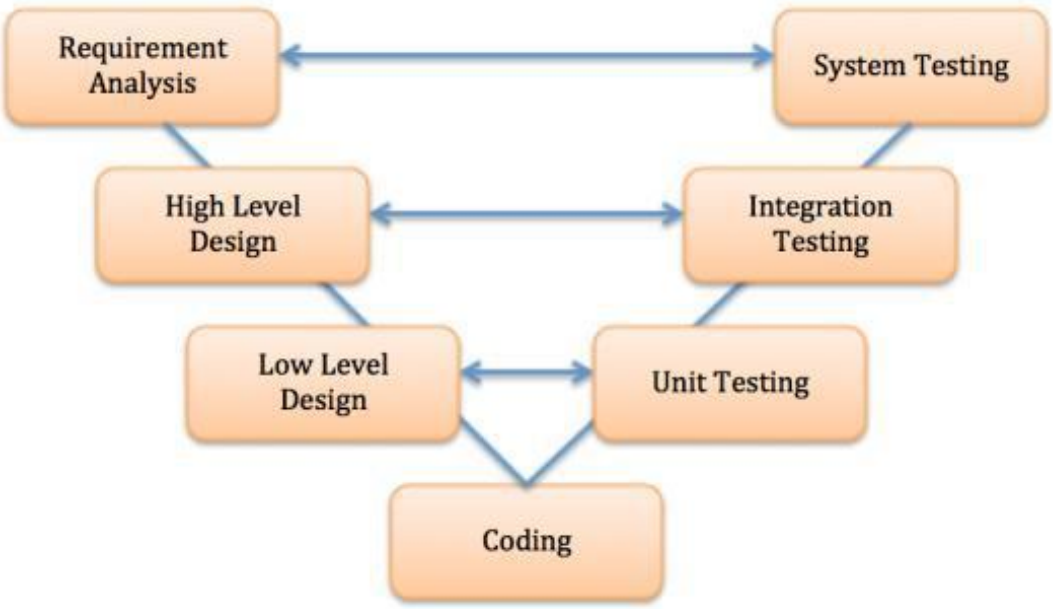


Figure-18: Unit Testing

4.1.3) Performance Testing:

Performance testing is finished by social event data with respect to activities of utilizations, and examining those gathered information to comprehend its practices and capacity to process substantial workloads.

The Performance testing is to reproduce end client's solicitations to a web application and study reactions under various client stacking conditions. By reproduction test generator characterizes virtual clients, perform over and again client's solicitations to make high movement stream to the application. There are three sorts of Performance testing, specifically (i) stress testing (ii) load testing (iii) Strength testing. Stress or weight test goes for investigating current design and similarity amongst equipment and programming at most extreme load to figure out conceivable framework bottlenecks. Load test focuses at deciding application's capacity by looking at most extreme load condition that the application can stand. Load test and stress test can be executed in conjunction. Strength test is performed in impressively longer length to explore baffling bugs that are not effectively imitated, for example, memory holes, or database exchange.

Metrics	**Measurement Units**	**Description**
Response time	Time unit(milliseconds)	Waiting time for client to receive server's response after sending a request.
Throughput	Requests/second	Number of requests being handled by application in a time unit
Resource Utilization	Percentage	Usage of resources, such as CPU (central processing unit), memory, network bandwidth

Table-5: Metrics for Performance testing

CHAPTER 5

CONCLUSION

This report briefly discussed about Server less architecture in which we have chosen to use AWS lambda Function and Spring boot. We have made an API for heart health status checking application which can be integrated on web or mobile. Now we have gained some experience with AWS Lambda too which is not much researched, with this we can see where the future of cloud computing is headed. Instead of buying computing resources, we can pay a little rent to AWS for use of their computational resources.

We were successfully able to design an API on AWS Lambda which can be deployed on any platform as what we have done is Platform independent. We can deploy API on Android and create a Heart health status tracking android application, which then can be easily availed to people to give them a check on their heart health status.

As more and more people will get the facility to check their health sitting on their mobile or desktop, the overall risk of heart failure can be reduced to very much large extent.

CHAPTER 6

REFERENCES

[1] Pankaj Deep Kaur, Inderveer Chana "Cloud based intelligent system for delivering health care as a service" (2013) - *Computer methods and program in biomedicine.*

http://dx.doi.org/10.1016/j.cmpb.2013.09.013

[2] Prema Sundaram, "Patient Monitoring System Using Android **"**(2013) - *International Journal of Computer Science and Mobile Computing*

https://www.ijcsmc.com/docs/papers/May2013/V2I5201382.pdf

[3] Alauddin Al-Omary, Wael El-Medany, Riyad Al-Hakim" Heart Disease Monitoring System Using Web and Smartphone **"** (2014) - *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*

http://www.ijareeie.com/upload/2014/april/1A_Heart_HardCopy_Forg.pdf

[4] Tran, Trung Hieu "Developing Web Services With Server less Architecture" (2017) - *Lappeenranta University of Technology School of Business and Management*

http://www.doria.fi/bitstream/handle/10024/147672/Developing%20Web%20Services%20With%20Serverless%20Architecture.pdf?sequence=1

Codes:

Request Handler:

```
package work.aws;

import com.amazonaws.services.lambda.runtime.Context;

public interface RequestHandler<I,O> {
   public O handleRequest(I input, Context context);
}
```

Risk Request Handler:

```
package work.aws;

import com.amazonaws.services.lambda.runtime.Context;
import work.pojo.PatientData;
import work.pojo.ResultData;

import java.io.InputStream;

public class RiskRequestHandler implements RequestHandler<PatientData, ResultData> {
   @Override
   public ResultData handleRequest( PatientData input, Context context) {
      PatientData patientData = new PatientData();
      return calculateRisk(input);
   }

   private ResultData calculateRisk(PatientData input) {
      {
         int points = 0;
         int age = input.getAge();
         float totchol = input.getTotalChoestrol();
         float HDLChol = input.getHdlChol();
         boolean treated = input.isTreated();
         int SBP = input.getSpb();
         if (age == 1) points = points - 9;
         else if (age == 2) points = points - 4;
         else if (age == 4) points = points + 3;
         else if (age == 5) points = points + 6;
         else if (age == 6) points = points + 8;
         else if (age == 7) points = points + 10;
         else if (age == 8) points = points + 11;
         else if (age == 9) points = points + 12;
         else if (age == 10) points = points + 13;

         // Total cholesterol points
```

```
if (age == 1 || age == 2) {
   // # Age 20â€“39 years
   if (totchol == 2) points = points + 4;
   else if (totchol == 3) points = points + 7;
   else if (totchol == 4) points = points + 9;
   else if (totchol == 5) points = points + 11;
} else if (age == 3 || age == 4) {
   //# Age 40â€“49 years
   if (totchol == 2) points = points + 3;
   else if (totchol == 3) points = points + 5;
   else if (totchol == 4) points = points + 6;
   else if (totchol == 5) points = points + 8;
} else if (age == 5 || age == 6) {
   //#Age 50â€“59 years
   if (totchol == 2) points = points + 2;
   else if (totchol == 3) points = points + 3;
   else if (totchol == 4) points = points + 4;
   else if (totchol == 5) points = points + 5;
} else if (age == 7 || age == 8) {
   //#Age 60â€“69 years
   if (totchol == 2) points = points + 1;
   else if (totchol == 3) points = points + 1;
   else if (totchol == 4) points = points + 2;
   else if (totchol == 5) points = points + 3;
} else if (age == 9 || age == 10) {
   //#Age 70â€“79 years
   if (totchol == 4) points = points + 1;
   else if (totchol == 5) points = points + 1;
}
int smoker = 0;
if (input.isSmoker()) {
   smoker = 1;
}
//#Cigarette smoker points
if (smoker == 1) {
   if (age == 1 | age == 2) points = points + 8;
   else if (age == 3 || age == 4) points = points + 5;
   else if (age == 5 || age == 6) points = points + 3;
   else if (age == 7 || age == 8) points = points + 1;
   else if (age == 9 || age == 10) points = points + 1;
}

// #HDL cholesterol points
if (HDLChol == 1) points = points - 1;
else if (HDLChol == 3) points = points + 1;
else if (HDLChol == 4) points = points + 2;
```

```
        // #Systolic blood pressure points
        if (treated) {
           if (SBP == 2) points = points + 1;
           else if (SBP == 3) points = points + 2;
           else if (SBP == 4) points = points + 2;
           else if (SBP == 5) points = points + 3;
        } else {
           if (SBP == 3) points = points + 1;
           else if (SBP == 4) points = points + 1;
           else if (SBP == 5) points = points + 2;
        }
        ResultData data = new ResultData();
        int risk = points;
        if (risk == 0) data.setRiskPercentage("Risk is lesser than <1% with 10-year risk " +
points + " %");
        else if (risk >= 1 & risk <= 4)
           data.setRiskPercentage("Risk is lesser than 1% with 10-year risk " + points + "
%");
        else if (risk >= 5 & risk <= 6) data.setRiskPercentage("2% with 10-year risk " +
points + " %");
        else if (risk == 7) data.setRiskPercentage("3% with 10-year risk " + points + " %");
        else if (risk == 8) data.setRiskPercentage("4% with 10-year risk " + points + " %");
        else if (risk == 9) data.setRiskPercentage("5% with 10-year risk " + points + " %");
        else if (risk == 10) data.setRiskPercentage("6% with 10-year risk " + points + " %");
        else if (risk == 11) data.setRiskPercentage("8% with 10-year risk " + points + " %");
        else if (risk == 12) data.setRiskPercentage("10% with 10-year risk " + points + " %");
        else if (risk == 13) data.setRiskPercentage("12% with 10-year risk " + points + " %");
        else if (risk == 14) data.setRiskPercentage("16% with 10-year risk " + points + " %");
        else if (risk == 15) data.setRiskPercentage("20% with 10-year risk " + points + " %");
        else if (risk == 16) data.setRiskPercentage("25% with 10-year risk " + points + " %");
        else if (risk >= 17) data.setRiskPercentage("30% with 10-year risk " + points + " %");
        return data;
     }
  }
}
```

Result Data:

```
public String getRiskPercentage() {
     return riskPercentage;
  }

  public void setRiskPercentage(String riskPercentage) {
     this.riskPercentage = riskPercentage;
  }
```

```
    public String getInfo() {
        return info;
    }

    public void setOtherData(String otherData) {
        this.info = otherData;
    }
}
```

Patient Data:

```
package work.pojo;


//points.men <- function(age, totchol, smoker, HDLChol, treated, SBP)
public class PatientData {
    private String sex;
    private int age;
    private float totalChoestrol;
    private boolean smoker;
    private float hdlChol;
    private boolean treated;
    private int spb;

    public String getSex() {
        return sex;
    }

    public void setSex(String sex) {
        this.sex = sex;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public float getTotalChoestrol() {
        return totalChoestrol;
    }

    public void setTotalChoestrol(float totalChoestrol) {
        this.totalChoestrol = totalChoestrol;
    }
```

```
    public boolean isSmoker() {
        return smoker;
    }

    public void setSmoker(boolean smoker) {
        this.smoker = smoker;
    }

    public float getHdlChol() {
        return hdlChol;
    }

    public void setHdlChol(float hdlChol) {
        this.hdlChol = hdlChol;
    }

    public boolean isTreated() {
        return treated;
    }

    public void setTreated(boolean treated) {
        this.treated = treated;
    }

    public int getSpb() {
        return spb;
    }

    public void setSpb(int spb) {
        this.spb = spb;
    }
}
```

Main Application:

```
package work.main;


public class MainApplication {
    public static void main(String[] args) {

    }
}
```