**A NODE LEVEL SECURITY ALGORITHM FOR CLOUD-ASSISTED IoT**

A PROJECT REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE

REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING



by

**Batch** - **A14**

CH. Lakshmi Prasanna(18JG1A0517) K. Hanna Shiny (18JG1A0532)

G. Sireesha Devi (18JG1A0529) B. Vasantha (18JG1A0508)

Under the esteemed guidance of

**Mr. S. Sumahasan**

Asst. Professor

CSE Department

**Department of Computer Science and Engineering**

**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN**

[Approved by AICTE NEW DELHI, Affiliated to JNTUK Kakinada]

[Accredited by National Board of Accreditation(NBA) for B.Tech. CSE, ECE & IT – Valid from 2019-20 to 2021-22] Kommadi, Madhurawada, Visakhapatnam – 530048

**2018 – 2022**

**GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING FOR WOMEN**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**CERTIFICATE**

This is to certify that the project report titled **“NODE LEVEL SECURITY ALGORITHM FOR CLOUD-ASSISTED IoT”** is a bonafide work of following IV/IV B. Tech students in the Department of Computer Science and Engineering, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNT University, Kakinada during the academic year 2021-2022, in fulfilment of the requirement for the award of the degree of Bachelor of Technology of this university.

This is to certify that the project report titled **“Privacy Preserving Cloud storage in MIOT using Cryptography Algorithm”**is a bona-fide work of following IV/II B. Tech students in the Department of Computer Science Engineering, Gayatri Vidya Parishad College of Engineering for Women affiliated to JNTU University, Kakinada during the academic year 2017-2021, in fulfillment of the requirement for the award of the degree of Bachelor of

Technology of this university.

**Ch. Lakshmi Prasanna**(18JG1A0517) **G. Sireesha Devi**(18JG1A0529)

**K. Hanna Shiny**(18JG1A0532)  **B. Vasantha**(18JG1A0508)

#### **Mr. S. Sumahasan** **Dr. P. V. S. L. Jagadamba**

Asst. Professor Professor

**(Internal Guide)** (**Head of department)**

## External Examiner

**ACKNOWLEDGEMENT**

We take the opportunity to one and all who have helped in making the project possible. We are thankful to **Gayatri Vidya Parishad College of Engineering for Women,** for giving us the opportunity to work on a project as part of the curriculum.

Our sincere thanks to our guide **Mr. S. Sumahasan**, Asst. Professor in Department of Computer Science Engineering for his simulating guidance and assistance from the beginning of the project.

Our sincere thanks to **Mr. Satyanarayana**, Professor in Department of Basic Sciences and Humanities for his simulating guidance and assistance regarding the project.

We are very much thankful to our Head of Computer Science Engineering **Prof. P. V.**

**S. L. Jagadamba**, Head of the Department for her help and encouragement and also for providing the lab facility to complete the project work.

Our sincere thanks to our beloved Vice-Principal **Prof. G. Sudheer** for providing the best faculty and lab facility throughout these academic years.

Our sincere thanks to our beloved principal **Prof. R K Goswami** for providing the best faculty and lab facility throughout these academic years.

Finally, we are thankful to our entire faculty and our lab technicians for their good wishes and constructive criticism, which led to the successful completion of the project.

|  |  |  |
| --- | --- | --- |
|  | **CONTENTS** |  |
| Abstract |  | i |
| List of figures |  | ii |
| List of Tables |  | iii |
| List of Screens |  | iv |

1. **INTRODUCTION**
   1. Problem definition 1
   2. Motivation 1
   3. Objective of Project 2
   4. [Limitations of Project](#_TOC_250031) 2
   5. Organization of Documentation 2
2. [LITERATURE SURVEY](#_TOC_250030)
   1. [Introduction](#_TOC_250029) 3
   2. [Existing system](#_TOC_250028) 16
   3. [Disadvantages of Existing system](#_TOC_250027) 16
   4. [Proposed system](#_TOC_250026) 16
   5. Conclusion 19
3. REQUIREMENT ANALYSIS
   1. [Introduction](#_TOC_250024) 20
   2. [Software Requirements](#_TOC_250021) 20
   3. Hardware Requirements 21
   4. System architecture 24
   5. [Algorithms and Flowcharts](#_TOC_250017) 25
4. [DESIGN](#_TOC_250016)
   1. Introduction 26
      1. [Class Diagram](#_TOC_250015) 27
      2. [Use Case Diagram](#_TOC_250014)  29
      3. [Sequence Diagram](#_TOC_250013) 29
   2. DFD/ER/UML diagram(any other project diagrams) 31
5. IMPLEMENTIONS & RESULTS
   1. [Introduction](#_TOC_250012) 36
      1. PostgreSQL Installation 36
      2. Firebase Installation 37
      3. PyCharm Installation 38
      4. Django Installation 38
   2. [Software Design](#_TOC_250010) 
      1. [Front end Design 3](#_TOC_250009)8
      2. Back end Design 39
   3. Sample codes 39
   4. [Output Screens](#_TOC_250008) 53
6. TESTING & VALIDATION
   1. [Introduction](#_TOC_250007) 59
   2. [Types of Testing](#_TOC_250006) 61
   3. Test Cases 62
   4. [Validation](#_TOC_250004) 63
7. [CONCLUSION](#_TOC_250003)
   1. [Project Conclusion](#_TOC_250002) 64
8. [REFERENCES](#_TOC_250000) 65

# **ABSTRACT**

Cloud-assisted Internet of Things (IoT) provides a favorable solution to data Booming problems for the ability constraints of individual objects. However, with the influence of cloud, IoT faces new security challenges for data mutuality between two parties, which is novel in this work and have been not currently addressed at the present time by the conventional approaches. This work carries out a systematic inquiry using a secure cloud-assisted IoT data managing method to keep data confidentiality when collecting, storing and accessing IoT data with the help of a cloud with the consideration of regular increase in users. The proposed system applies a node level security algorithm. Hence, A secure IoT under our proposed method could resist most attacks from both insiders and outsiders of IoT to break data confidentiality.

**Key words:** IoT, Cloud-assisted IoT, Confidentiality, Security Challenges, Node level security algorithm, Attack.

# **LIST OF FIGURES**

ii

**Figure No. Figure Name Page No.**

|  |  |  |
| --- | --- | --- |
| Fig. 1. | Structure Of Heart | 4 |
| Fig. 2. | Radial Method | 5 |
| Fig. 3. | Carotid Method | 6 |
| Fig. 4. | IOT Application | 8 |
| Fig. 5. | Elliptic Curve | 9 |
| Fig. 6. | Key Generation Mechanism | 10 |
| Fig. 7. | Diffie-Hellman Key Exchange Mechanism | 11 |
| Fig. 8. | Elliptic Curve Diffie-Hellman | 13 |
| Fig. 9. | ElGamal Algorithm | 14 |
| Fig. 10. | Heart Monitoring System | 17 |
| Fig. 11. | Providing Security at Node Level | 18 |
| Fig. 12. | Uploading Data to the Cloud | 18 |
| Fig. 13. | Pulse Sensor | 22 |
| Fig. 14. | Raspberry Pi | 23 |
| Fig. 15. | System Architecture | 24 |
| Fig. 16. | Flow of The Process | 25 |
| Fig. 17. | Class Diagram | 31 |
| Fig. 18. | Use-Case Diagram | 33 |
| Fig. 19. | Sequence Diagram | 35 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# **LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Table Name** | **Page No** |
| Table 1. | Heart Beat rate and Respiratory for different ages | 5 |
| Table 2. | Design of test cases and scenario | 62 |

**LIST OF SCREENS**

**Screen No. Name of Screen Page No**

|  |  |  |
| --- | --- | --- |
| Screen 1. | User Registers on website | 60 |
| Screen 2. | Login Page for User and User | 60 |
| Screen 3. | User clicks on View or Download Report | 61 |
| Screen 4. | File Displayed after Clicking | 61 |
| Screen 5. | User Gets Secret Key | 62 |
| Screen 6. | User enters Secret Key to download | 62 |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**1.INTRODUCTION**

* 1. **PROBLEM DEFINITION**

According to the WHO, CVD (Cardio vascular disease) kills 17 million people worldwide, accounting for 31% of all deaths. As a result, a way to prevent or assist in the reduction of human life losses. Heart stroke is a condition that occurs when your body overheats, usually as a result of extended exposure to or physical effort in hot weather. If your body temperature rises to 104 degrees Fahrenheit (40 degrees Celsius) or higher, you may get heatstroke, the most dangerous type of heat damage. The condition is especially frequent during the summer. Heatstroke demands medical intervention right away. Heat stroke can swiftly harm your brain, heart, kidneys, and muscles if left untreated. The longer you wait for treatment, the worse the damage becomes, raising your chance of serious complications or death. Cryptography is only used at cloud level or in the cloud once the data is stored. But data breaches can happen on the way the raw data is being travelled from the node to the cloud. So, there is no protection for the data at the node level.

**1.2 MOTIVATION TO THE PROJECT**

In many nations, heart disease is one of the leading causes of mortality, accounting for approximately 15 million fatalities yearly. In addition, cardiovascular illness disables a large number of people. The time between the onset of the first sign of any heart problem and the need for medical help varies greatly across people and can be deadly. One critical inference drawn from epidemiological data is that deployment of resources for early detection and treatment of heart disease has a higher potential of reducing fatality associated with cardiac disease than improved care after hospitalization. Hence new strategies are needed in order to reduce time before treatment. Monitoring of patients is one possible solution. This project can be used in hospitals (Calling Ambulance.) and also for patients who can be under continuous monitoring while traveling from place to place (using heart rate band).

**1.3 OBJECTIVE OF THE PROJECT**

We achieve this goal by developing the security to the data at the node itself where the data is captured and stored. We implement certain encryption algorithms in the node level itself to encrypt the data and then send it to the cloud for storage. When the user wants to receive the data the data is decrypted at the users side and then the requested data is shown to the user.

* 1. **LIMITATIONS OF THE PROJECT**
* Even a small mistake in configuration or coding may result in a diminishment of some or all of the cryptographic security.
* There needs to be very careful attention paid to any hybrid cryptographic implementation in order to ensure that it does not make us less secure.
  1. **ORGANIZATION OF THE PROJECT**

The documentation of our project has been divided into the following sections:

**Chapter 1:** A concise overview of the rest of the documentation work is explained below. Definitions of a few terms which are involved in building this software were reviewed.

**Chapter 2:** Literature survey describes the primary terms involved in the development of this software. It gives an overview of the existing system and features of the proposed system.

**Chapter 3:** Analysis deals with detailed analysis of the project. Software requirement specification which further contain User requirement analysis, Software requirement analysis, and Hardware requirement analysis.

**Chapter 4:** Design includes UML diagrams along with explanation of module design and organization.

**Chapter 5:** Contains screenshots of Output Screens and its implementation.

**Chapter 6**: Gives the testing and validation details with design of test cases and scenarios along with validation screenshots.

**Chapter 7:** Contains project conclusion and future enhancements.

**2. LITERATURE SURVEY**

**2.1 INTRODUCTION**

The significance of vital signs stems from the fact that they can be used to assess a person's health. Any change in these signs' measurements indicates a problem with the patient's physical condition. Variations in one or more vital signs can be used to diagnose a variety of medical disorders. The specialist gadgets used to measure vital signs are not portable and are hard to come by. As a result, the concept of using an armband (portable heart rate monitor) and a mobile phone as a diagnosing tool is presented in this thesis. There are four vital signs which are standard in most medical settings:

1. Pulse rate.

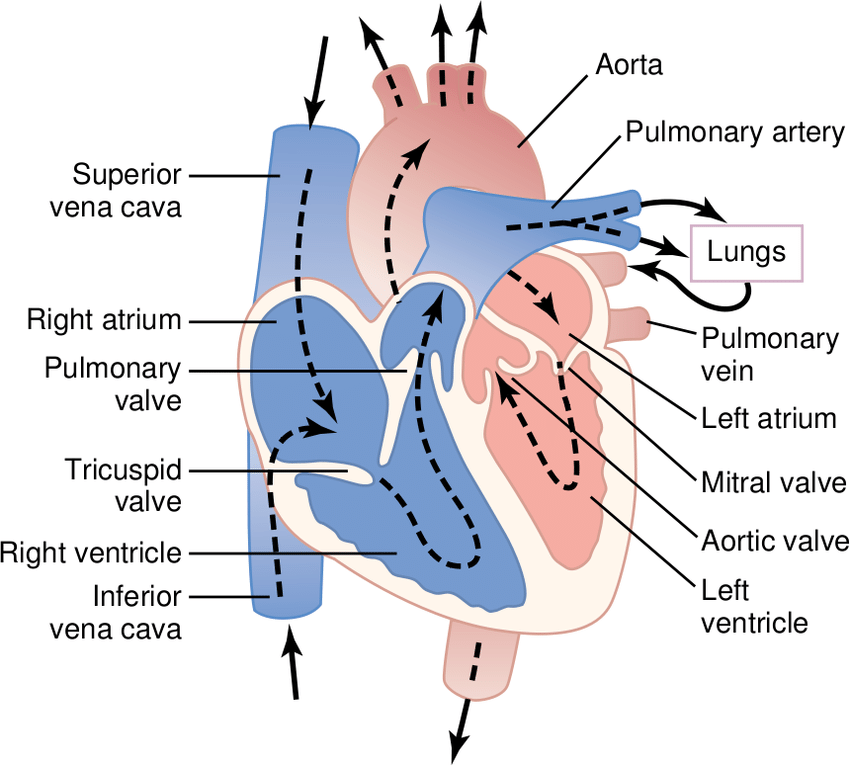
2. Respiratory rate.

3. Blood pressure.

4. Body temperature

**2.1.1 Heart:**

The coronary heart in Fig. 1. is the primary organ withinside the human body, that is liablefor pumping blood during the body. It is placed withinside the centre of the thorax, barely offset to the left and surrounded with the aid of using the lungs. The coronary heart is made of separate pumps: a proper coronary heart that pumps blood via the lungs, and a left coronary heart that pumps blood via the peripheral organs. At the equal time, every of those hearts is a pulsatile chamber pump composed of an atrium and a ventricle.

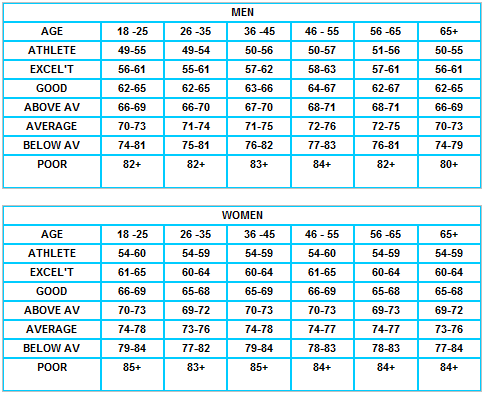


**Fig. 1.** Structure of heart and blood flow

**2.1.2 Pulse and Heart Rate**

**Heart Beat**: The rhythmic movement or sound made by the heart as it pumps blood around your body. The heart's pulse, which includes one complete systole and diastole.

**Heart Rate:** The number of pulses per unit of time, usually per minute, is referred to as heart rate. The number of contractions of the ventricles determines the heart rate (the lower chambers of the heart). It's possible that your heart rate is too fast (tachycardia) or too sluggish (bradycardia) (bradycardia). Sedentary males have an average heart rate of 72 beats per minute and sedentary females have an average heart rate of 80 beats per minute in Table 1. however these rates are typically drastically different for experienced athlete.



**Table 1.** Heart rate of different ages

**2.1.3 Heart Rate Measurements Methods**

There are several methods to measure the heart rate, the most used methods:

**Radial:**

The Wrist Pulse is being taken in Fig. 2. Count the number of beats for six seconds while holding the palm of the right hand facing upwards and placing the tips of the middle three fingers from the left hand on the wrist joint. The BPM is calculated by multiplying this number by ten.



**Fig. 2.** Radial Method

**Carotid:**

On the Throat, it is taking the pulse. By lightly placing two fingers (first and second) on the side of the throat slightly below the jaw's angle. The jugular vein in Fig. 3. has a pulse that can be felt. For six seconds, count the number of beats. The BPM is calculated by multiplying this number by ten.



**Fig. 3.** Carotid Method

**Heart Rate Monitors:**

Many heart monitoring devices ensure a more accurate measure of heart rate than manual methods. In most physical training situations, they are the preferred method.

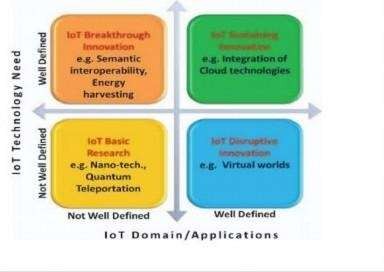
**2.1.4 Pulse Monitoring Using IoT**

A pulse sensor is an electronic device that measures the heart rate, or the rate at which the heart beats. The main things we do to stay healthy are monitor our body temperature, heart rate, and blood pressure. To monitor the Arterial Pressure or Blood Pressure, we utilise thermometers and a sphygmomanometer to measure the body temperature.

Heart rate can be measured in two ways: one by manually checking the pulse at the wrists or neck, and the other by using a Pulse Sensor. We created a Heart Rate Monitor System with Arduino and a Pulse Sensor in this project. You can find the Principle of Pulse Sensor, working of the Pulse Sensor and Arduino based Heart Rate Monitoring System using a practical heart beat Sensor.

**IoT**

The Internet of Things (IOT) is a network of interconnected computing devices, mechanical and digital machinery, items, animals, and people with unique identifiers and the ability to transfer data without requiring human-to-human or human-to-computer interaction. The Internet of Things makes life easier for people. Its applications have been found in a variety of industries, including health, automobiles, agriculture, and traffic systems. We use micro-controllers and micro-processors in IoT. A microcontroller (MCU for microcontroller unit) is a **small computer** on a single metal-oxide-semiconductor (MOS) integrated circuit (IC) chip. A microcontroller contains one or more CPUs (processor cores) along with memory and programmable input/output peripherals. A microprocessor is a computer processor that incorporates data processing logic and control on a single integrated circuit or a small number of integrated circuits. The microprocessor comprises the arithmetic, logic, and control circuitry required to accomplish the duties of the central processing unit of a computer. The integrated circuit can interpret and execute computer instructions as well as conduct arithmetic calculations. The microprocessor is a digital integrated circuit that accepts binary data as input, processes it according to instructions stored in its memory, and outputs the results. Microprocessors comprise both combinational logic and sequential digital logic, and they operate on binary numbers and symbols. The Internet of Things (IoT) enables objects to be sensed or controlled remotely over existing network infrastructure, allowing for more direct integration of the physical world into computer-based systems and, as a result, improved efficiency, accuracy, and economic benefit, as well as reduced human intervention.



**Fig. 4.** IoT Applications

**2.1.5 Cryptosystem Scheme**

With asymmetric cryptography: Each user has two keys: a public key and a private key. Both keys are mathematically related (both keys together are called the key pair). The public key is made available to anyone. The private key is kept secret. Both keys are required to perform an operation. Asymmetric algorithms are based on the computational security that offers encryption made in polynomial time complexity and that require cryptanalysis with exponential time complexity. • Asymmetric algorithms are efficient at encrypting small volumes of data, which is why they are generally used for key exchange and digital signature. There are a number of key Mathematical Algorithms that serve as the crux for Asymmetric Cryptography, and of course, use widely differing Mathematical Algorithms than the ones used with Symmetric Cryptography. The Mathematical Algorithms used in Asymmetric Cryptography include the following:

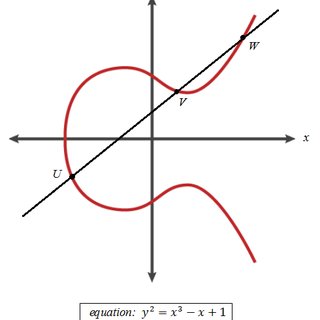
* The RSA Algorithm
* The Diffie-Hellman Algorithm
* The Elliptical Wave Theory Algorithm

Cryptosystem Schemes we use are used for securing data at the node itself. We use the ECC, ECDH and ElGamal encryption and decryption algorithms.

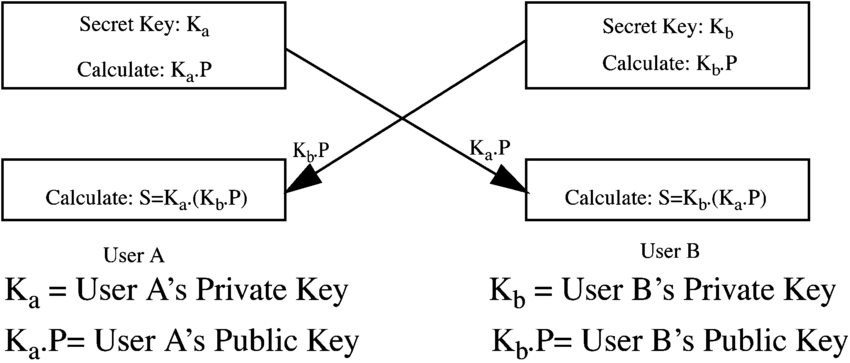
**Algorithms Used**

**ECC (Elliptic-curve cryptography) Algorithm:**

Elliptic-curve cryptography (ECC) in Fig. 6. is an approach to public-key cryptography based on the algebraic structure of elliptic curves in Fig. 5. over finite fields. ECC allows smaller keys compared to non-EC cryptography (based on plain Galois fields) to provide equivalent security. A trapdoor function is a function that is easy to compute in one direction, yet difficult to compute in the opposite direction (finding its inverse) without special information, called the "trapdoor". Trapdoor functions are a special case of one-way functions and are widely used in public-key-cryptography. In the multiplicative group Zp\*, the discrete logarithm problem is: given elements r and q of the group, and a prime p, find a number k such that r = qk mod p. If the elliptic curve groups is described using multiplicative notation, then the elliptic curve discrete logarithm problem is: given points P and Q in the group, find a number that Pk = Q; k is called the discrete logarithm of Q to the base P. When the elliptic curve group is described using additive notation, the elliptic curve discrete logarithm problem is: given points P and Q in the group, find a number k such that Pk = Q.



**Fig. 5.** Elliptic Curve



**Fig. 6.** Key Generation Mechanism

**Key Generation**

Key generation is an important part where we have to generate both public key and private key. The sender will be encrypting the message with receiver’s public key and the receiver will decrypt its private key.

Now, we have to select a number ‘d’ within the range of ‘n’.

Using the following equation, we can generate the public key

Q=d\*P

d = The random number that we have selected within the range of ( 1 to n-1 ). P is the point on the curve.

‘Q’ is the public key and ‘d’ is the private key.

**Encryption**

Let ‘m’ be the message that we are sending. We have to represent this message on the curve. This has an in-depth implementation of details.

Consider ‘m’ has the point ‘M’ on the curve ‘E’. Randomly select ‘k’ from [1 – (n-1)].

Two cipher texts will be generated let it be C1 and C2.

C1 = k\*P

C2 = M + k\*Q

C1 and C2 will be send.

**Decryption**

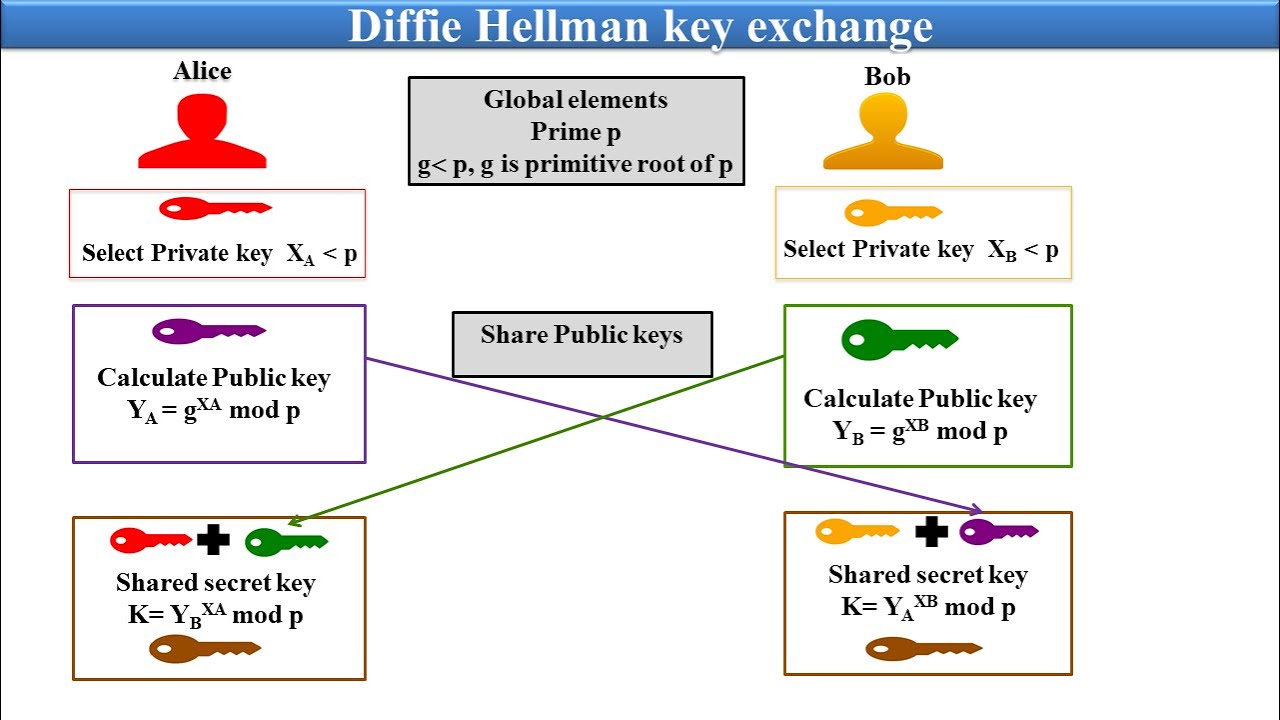
We have to get back the message ‘m’ that was send to us,

M=C2-d\*C1

M is the original message that we have send.

**Diffie–Hellman key exchange**

Diffie–Hellman key exchange in Fig. 7. is a method of securely exchanging cryptographic keys over a public channel and was one of the first public-key protocols as conceived by Ralph Merkle and named after Whitfield Diffie and Martin Hellman. DH is one of the earliest practical examples of public key exchange implemented within the field of cryptography. Published in 1976 by Diffie and Hellman, this is the earliest publicly known work that proposed the idea of a private key and a corresponding public key.



**Fig. 7.** Diffie-Hellman Key Exchange Mechanism

The simplest and the original implementation of the protocol uses the multiplicative group of integers modulo p, where p is prime, and g is a primitive root modulo p. These two values are chosen in this way to ensure that the resulting shared secret can take on any value from 1 to p–1. Here is an example of the protocol, with non-secret values in blue, and secret values in red.

Alice and Bob publicly agree to use a modulus p = 23 and base g = 5 (which is a primitive root modulo 23).

Alice chooses a secret integer a = 4, then sends Bob A = ga mod p

A = 54 mod 23 = 4

Bob chooses a secret integer b = 3, then sends Alice B = gb mod p

B = 53 mod 23 = 10

Alice computes s = Ba mod p

s = 104 mod 23 = 18

Bob computes s = Ab mod p

s = 43 mod 23 = 18

Alice and Bob now share a secret (the number 18).

**ECDH (Elliptic curve Diffie-Hellmann Algorithm)**

Elliptic-curve Diffie–Hellman in Fig. 8. is a key agreement protocol that allows two parties, each having an elliptic-curve public–private key pair, to establish a shared secret over an insecure channel. This shared secret may be directly used as a key, or to derive another key. The key, or the derived key, can then be used to encrypt subsequent communications using a symmetric-key cipher. It is a variant of the Diffie–Hellman protocol using elliptic-curve cryptography. ECDH is based on the following property of EC points:

(**a** \* **G**) \* **b** = (**b** \* **G**) \* **a**

1.Alice generates a random ECC key pair: {alicePrivKey, alicePubKey = alicePrivKey \* G}

2.Bob generates a random ECC key pair: {bobPrivKey, bobPubKey = bobPrivKey \* G}

3.Alice and Bob exchange their public keys through the insecure channel (e.g. over Internet)

4.Alice calculates sharedKey = bobPubKey \* alicePrivKey

5.Bob calculates sharedKey = alicePubKey \* bobPrivKey

6.Now both Alice and Bob have the same sharedKey == bobPubKey \* alicePrivKey == alicePubKey \* bobPrivKey

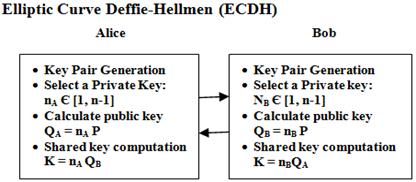
If we have two secret numbers a and b (two private keys, belonging to Alice and Bob) and an

ECC elliptic curve with generator point G, we can exchange over an insecure channel the values

(a \* G) and (b \* G) (the public keys of Alice and Bob) and then we can derive a shared secret:

secret = (a \* G) \* b = (b \* G) \* a. Pretty simple. The above equation takes the following form:

alicePubKey \* bobPrivKey = bobPubKey \* alicePrivKey = **secret**

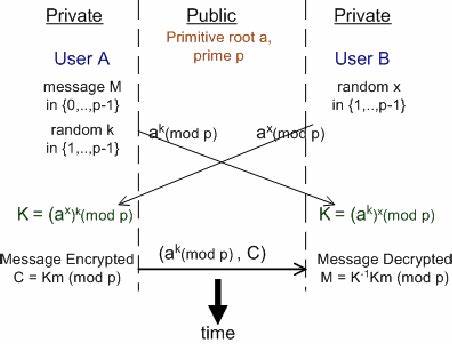


**Fig. 8.** Elliptic Curve Deffie-Hellmen

The ECDH algorithm (Elliptic Curve Diffie–Hellman Key Exchange) is trivial:

**ElGamal Algorithm**

ElGamal encryption system is an asymmetric key encryption algorithm in Fig.9. for public-key cryptography which is based on the Diffie–Hellman key exchange. It was described by Taher ElGamal in 1985. ElGamal encryption is used in the free GNU Privacy Guard software, recent versions of PGP, and other cryptosystems. The Digital Signature Algorithm (DSA) is a variant of the ElGamal signature scheme, which should not be confused with ElGamal encryption.



**Fig. 9.** ElGamal Algorithm

We use ElGamal algorithm for decryption purpose only:

* The computation of the values chosen in the first and second phases will then be utilised by user1 to determine the right number, which will be used to decrypt the encrypted message. The User1 will process bap, and the result will be used to divide by Z to obtain the decrypted value. The value that has been decrypted is anything that has been encrypted in the second step. In the preceding scenario, user1 started the process by calculating the private and public keys, which are the soul of the algorithm. The key is then used by user2 in the second stage to encrypt the method. The message is encrypted so that they value computed in that initial phase could be leveraged to decrypt the message. In the third step, it could be witnessed that after diving the entire value with the number that is computed in the third step itself totally decrypts the message making it readable for the end-user. The same approach is followed every when the urge to pass the message securely occurs.

**Elliptic Curve ElGamal Encryption**

We often use elliptic curves for public key cryptography tasks such as key exchange and digital signature tasks. Because these curves serve faster implementations than other trusted algorithms such as Diffie Hellman or RSA. Rarely, we can adapt elliptic curves for symmetric key encryption tasks. This idea is mainly based on ElGamal encryption schema and elliptic curves. We will create a python implementation of this concept. All points on the elliptic curve can be mapped directly to an ASCII value. Koblitz represents message as a point or a set of points on an elliptic curve.

y2p=x3+ax+bmodP

Message Encoding Algorithm

• Divide the message into blocks of fixed size.

• Express each block as a number x.

• This converts the message into a series of numbers between 0 and 35.

• α=x3+ax+b mod p

• Γ=α(p-1)/2

• If Γ ≠11 then set x=x+1

• b=sqrt(α mod P)

• y=b

**Public key generation**

We can produce our public key. Public key would be independent from plaintext. We will pick a secret key and calculate secret key times base point. That would be our public key.

**Encryption**

C1= Generator point \* Private key of node

C2 = (public key of node)\*(private key of node)+Pm

**Decryption**

ElGamal decryption scheme is based on the following equation.

decryption = C2 – secretKey \* C1

We will adapt this equation into elliptic curves. We have already known how to add points over elliptic curves but the term in the above includes subtraction. Reflecting the sign to multiplier point handles subtraction.

decryption = C2 + secretKey \* (-C1)

**2.2 EXISTING SYSTEM**

* The security is provided to the data in the cloud when data is transferred from on point to the other.
* Data breaches can happen by any intruder on the way of the data being travelled by any sort of intruder attacks.
* Cryptography security algorithms like AES, Triple DES and Blowfish to provide security at the cloud.
* Security techniques are not applied in the protection of offloaded data from attacks.

**2.3 DISADVANTAGES OF THE EXISTING SYSTEM**

* The existing system has data vulnerability as it is being exposed to data attacks and data breaches by any intruders.
* There is no concept of providing security to the data at the node itself i.e the device in which the data is being collected.
* The currently used encryption algorithms used much storage data and a larger key size to store their data which leads to the wastage of storage.

**2.4 PROPOSED SYSTEM**

This work is mainly divided into four parts:

* Designing a pulse monitoring system.
* Providing security at node level.
* Uploading encrypted data into cloud.
* Accessing the data through webpage.

**Part1: Pulse monitoring system**

A pulse sensor is an electronic device that measures the heart rate, or the pace at which the heart beats. The main things we do to stay healthy are monitor our body temperature, heart rate, and blood pressure. To monitor the Arterial Pressure or Blood Pressure, we utilise thermometers and a sphygmomanometer to measure the body temperature. Heart rate can be measured in two ways: one by manually checking the pulse at the wrists or neck, and the other by using a Pulse Sensor. We created a Heart Rate Monitor System in Fig. 10. with Raspberry Pi and a Pulse Sensor in this project.



**Fig. 10.** Heart Monitoring System

**Part 2: Providing security at the node level**

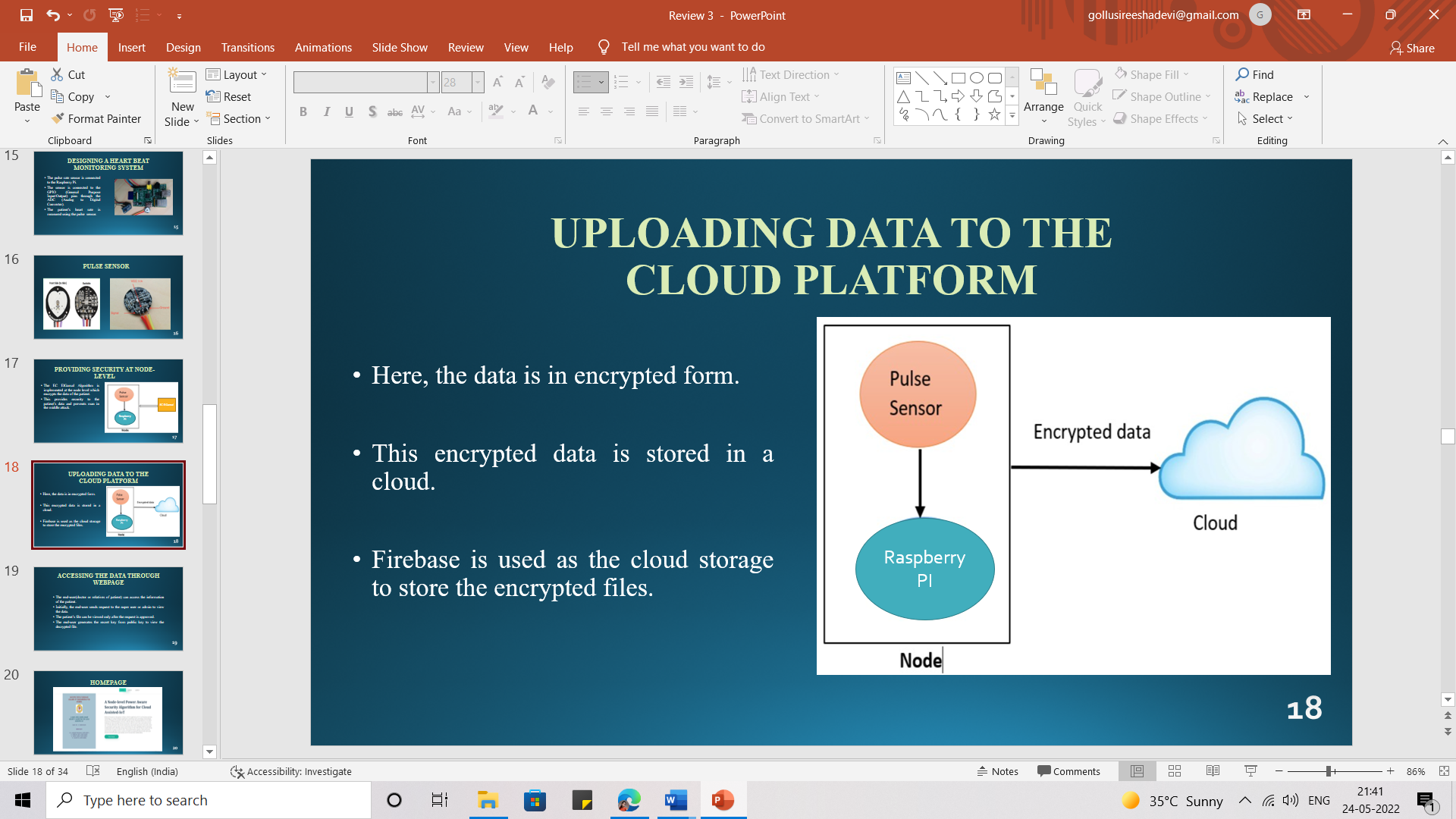
## In Fig. 11. the node comprises of the Raspberry PI, pulse sensor, a bread board and some connectors. The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.  A pulse wave is the change in the volume of a blood vessel that occurs when the heart pumps blood, and a detector that monitors this volume change is called a pulse sensor.The heart-beat sensor collects data and stores it in the Raspberry PI. Here we implement data encryption algorithms like Elliptic curve Cryptographic Algorithm (ECC) , Elliptic-curve Diffie–Hellman(ECDH) and encrypt the data at the Node and then send the data to the cloud.

## 

**Fig. 11.** Providing Security at the Node Level

**Part 3: Uploading Encrypted data into the cloud**

Data encryption is a way of translating data from plaintext (unencrypted) to ciphertext (encrypted). Users can access encrypted data with an encryption key and decrypted data with a decryption key in Fig. 12. Protecting your data. We use Firebase cloud for storing data. Firebase is a platform developed by Google for creating mobile and web applications. It was originally an independent company founded in 2011. In 2014, Google acquired the platform and it is now their flagship offering for app development. Once the data is encrypted at the node, the data is uploaded to the cloud which is stored in the form of tables that can be retrieved by the user whenever needed or necessary.



**Fig. 12.** Uploading Data to The Cloud

**Part 4: Accessing the data through the web page**

Decryption is taking encoded or encrypted text or other data and converting it back into text you or the computer can read and understand. The recipient of decryption receives a prompt or window in which a password can be entered to access the encrypted data. For decryption, the system extracts and converts the garbled data and transforms it into words and images that are easily understandable not only by a reader but also by a system. Decryption can be done manually or automatically. The user requests specific data through the web page

## CONCLUSION

Every system has its own advantages and disadvantages. As the proposed system we have chosen is objective specific, it is dependent on the factor we need to justify and criticality of the system. Here the proposed system is more secure in providing data storage than the existing system.

**3. REQUIREMENT ANALYSIS**

**3.1 INTRODUCTION**

The system requirements can be hardware and software. These system requirements are also known as prerequisites. Each computer system requires certain hardware components or software resources to be present on a computer. The need for system requirements increases with the increase in demand for higher processor power and resources of newer versions.

**3.2 SOFTWARE REQUIREMENTS**

* PostgreSQL
* Firebase
* Python
* PyCharm
* Django

1. **PostgreSQL**

PostgreSQL is a powerful object-relational database system that is free and open source. It has over 20 years of active development and a proven architecture, earning it a strong reputation for dependability, data integrity, and correctness. PostgreSQL is compatible with all major operating systems, including Linux, UNIX, and Windows. It can handle text, images, sounds, and video, and has programming interfaces for C/C++, Java, Perl, Python, Ruby, and Open Database Connectivity (ODBC). Many web, mobile, and analytics applications use PostgreSQL as their primary data store or data warehouse. PostgreSQL has a long history of supporting advanced data types, and it offers the same level of performance optimization as its commercial database counterparts, such as Oracle and SQL Server.

1. **Firebase**

Follow the below steps to install Firebase

**Step 1:** Create a Firebase project and register your app

**Step 2:** Install the SDK and initialize Firebase

1. **Python**

Python is a high-level, interpreted scripting language created in the late 1980s by Guido van Rossum at the Netherlands National Research Institute for Mathematics and Computer Science. The first version was posted to the alt. sources newsgroup in 1991, and version 1.0 was released in 1994. Python 2.0 was released in 2000, and the 2.x versions were the most common until December 2008. At the time, the development team decided to release version 3.0, which included a few relatively minor but significant changes that were not backward compatible with the 2.x versions. Python 2 and 3 are very similar, and some Python 3 features have been backported to Python 2. However, they are still incompatible in general. Python 2 and 3 have both been maintained and developed, with both receiving periodic release updates.

1. **Django**

Django is a Python-based web framework, free and open-source, that follows the model–template–views architectural pattern. It is maintained by the Django Software Foundation, an independent organization established in the US as a 501 non-profit.  Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Built by experienced developers, Django takes care of much of the hassle of web development, so you can focus on writing your app without needing to reinvent the wheel.

**3.3 HARDWARE REQUIREMENTS**

* Pulse Sensor
* Raspberry Pi
* Connecting Wires
* MCP 3208

1. **Pulse Sensor**

Pulse sensor in Fig. 13. is an open-source heart rate monitor used to monitor non-invasive heart rate. It monitors real-time pulses and calculates BPM using Arduino algorithms. This sensor has two sides, the front of which is shaped like a heart and is the side that will be connected to the skin. As illustrated in the diagram below, the pulse sensors have three pins.



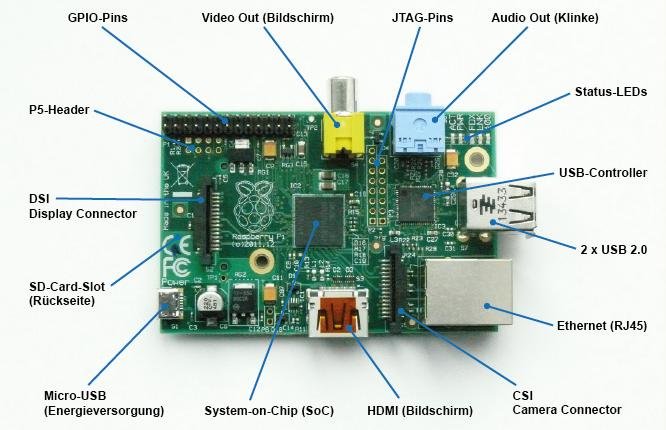
**Fig. 13:** Pulse Sensor

If the front side is facing you, the most left pin is GND, while the center one is the input voltage, which will be connected to the Arduino's +5v. The final one will be used to output electricity and will be connected to the Arduino's analogue pins.

The pulse sensor turns physical PPG signals into electrical signals. The sensor generates a raw signal of analogue voltage variations, which it amplifies and normalizes at v/2. With each pulse, a pulse wave passes along the arteries to the tissues where the pulse sensor is connected.

1. **Raspberry Pi**

The Raspberry Pi Foundation, a UK charity that aims to educate people in computing and make computing education more accessible, created the Raspberry Pi series of single-board computers. The Raspberry Pi in Fig. 14. was introduced in 2012, and several iterations and variations have been released since then. The original Pi had a single-core 700MHz CPU and only 256MB RAM, whereas the latest model has a quad-core CPU that clocks in at more than 1.5GHz and 4GB RAM. People all over the world use the Raspberry Pi to learn programming, build hardware projects, automate their homes, and even use it in industrial applications. The Raspberry Pi is a low-cost computer that runs Linux, but it also has GPIO (general purpose input/output) pins that allow you to control electronic components for physical computing and explore the Internet of Things (IoT).

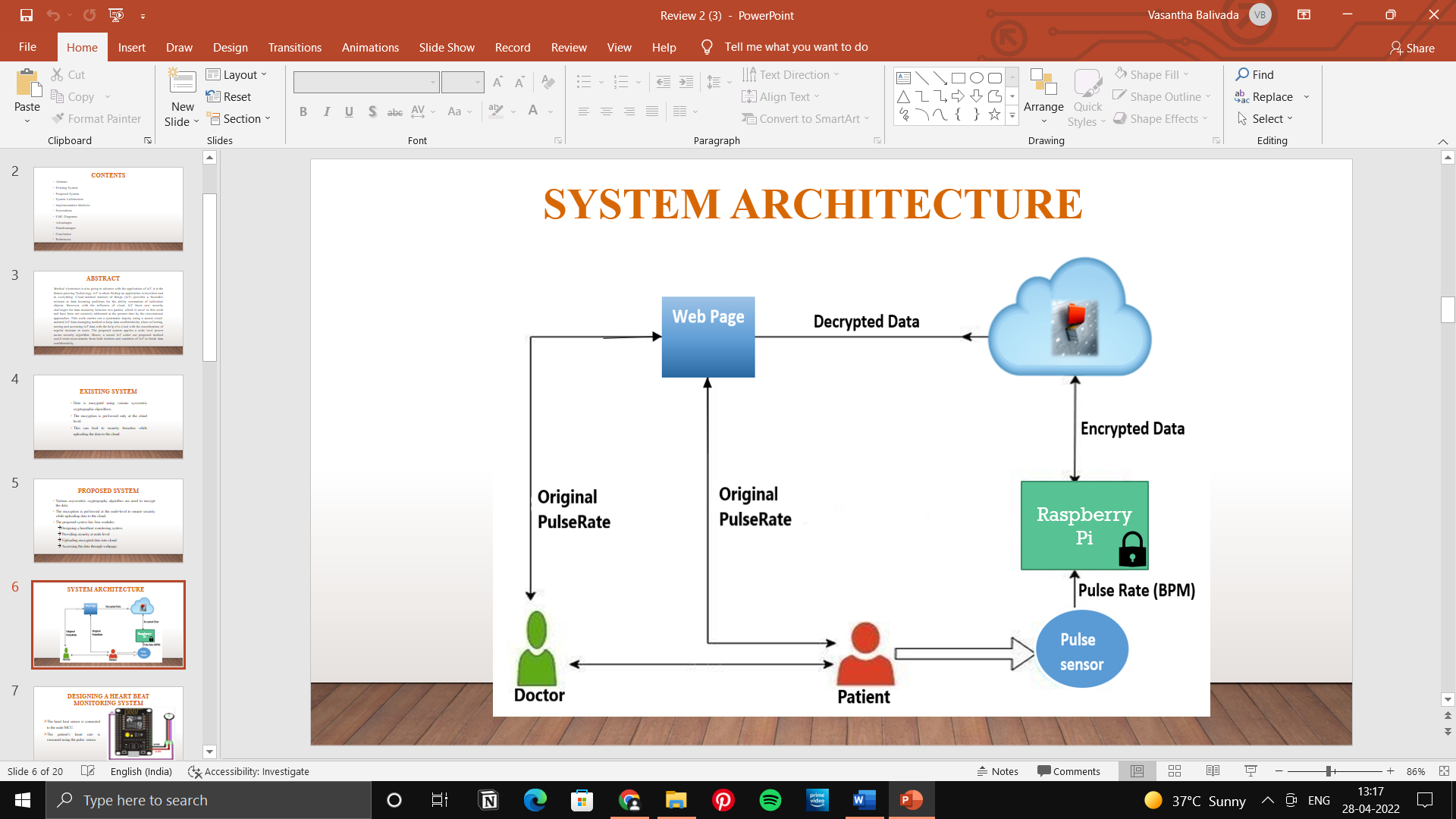


**Fig. 14.** RaspberryPi

1. **MCP3208**

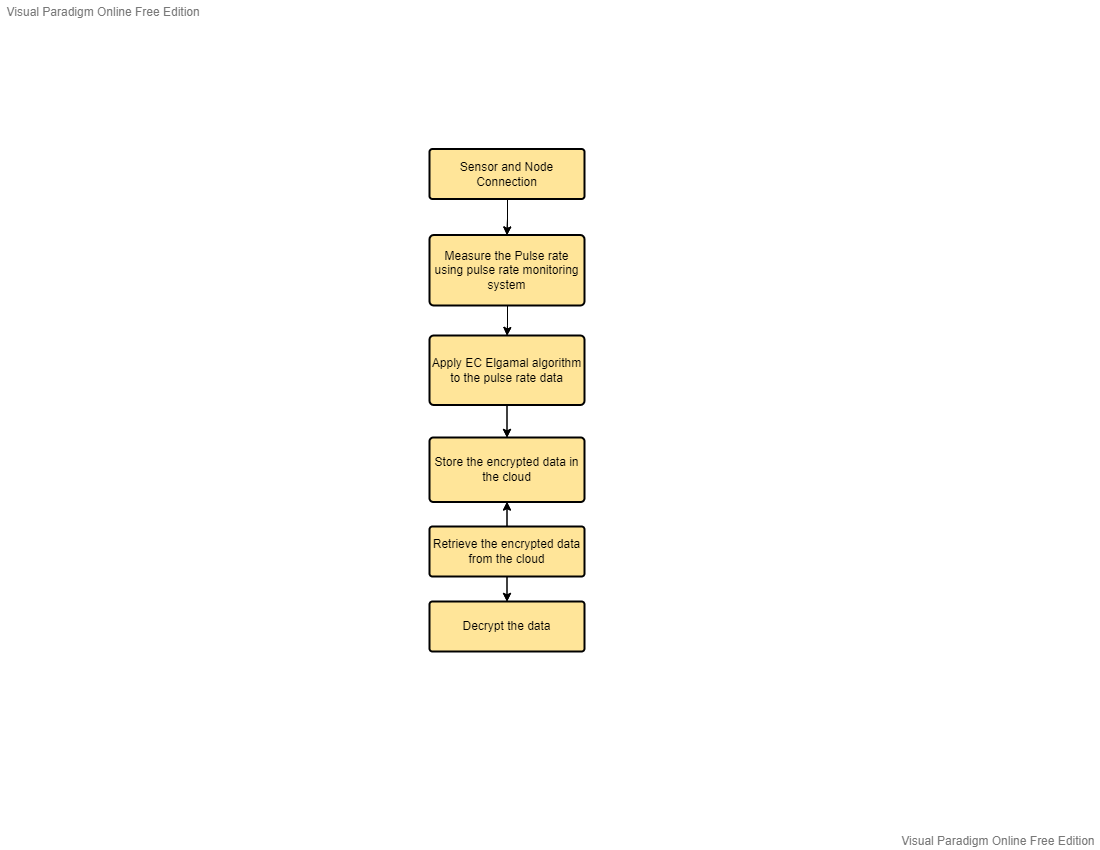
The MCP3208-CI/P is an 8 channel, 12bit Analogue to Digital Converter (ADC) with SPI interface in 16 pin DIP package. This ADC combines high performance and low power consumption in a small package by making it as an ideal for embedded control applications. The MCP3208 12-bit Analog-to-Digital Converter (ADC) combines high performance and low power consumption in a small package, making it ideal for embedded control applications. The MCP3208 features a successive approximation register (SAR) architecture and an industry-standard SPI™ serial interface, allowing 12-bit ADC capability to be added to any PICmicro® microcontroller.

**3.4 SYSTEM ARCHITECTURE**



**Fig. 15.** System Architecture

**3.5 ALGORITHMS AND FLOWCHARTS**

**Fig. 16.** Flow of the process

**4.DESIGN**

**4.1 INTRODUCTION**

Grady Booch, James Raumbaug h, and Ivar Jacobson collaborated to merge the finest elements of their unique object-oriented analysis and design approaches into a single method called the unified modelling language, which was released in January 1997 as version 1.0. The Booch, OMT, and OOSE methods are the foundations of UML. UML's objectives are as follows:

1.To use object-oriented concepts to represent systems.

2.To provide an explicit link between the conceptual and the executable.

3.To deal with the scale difficulties that come with sophisticated, mission-critical systems.

4.Develop a modelling language that can be used by both people and machines.

**Basic Building Blocks of UML**

Things and relationships are the basic building blocks in UML; they are joined in various ways according to different rules to produce various sorts of diagrams. There are nine different types of diagrams in UML; a list and brief description of each is provided below. The document's more detailed discussions will concentrate on the first five diagrams on the list, which are the most general and are sometimes referred to as the UML core diagrams.

1.Use case Diagram: depicts a set of use cases and how they can be used by actors.

2. Class Diagram: depicts the system's structure, which is separated into classes with various linkages and interconnections.

3. Sequence Diagram: The interaction between a group of items is depicted in a diagram by the messages that can be sent between them.

**4.1.1 CLASS DIAGRAM**

In a Class diagram, the classes reflect both the major objects and interactions in the application, as well as the classes that need to be programmed. Classes are represented in the diagram by boxes that include three parts:

• The name of the class appears at the top. The initial letter is capitalised and displayed in bold and in the centre.

• The class's attributes are found in the middle section. They're aligned to the left, and the first letter is in lowercase.

• The class's methods are listed at the bottom of the page. They're also positioned to the left, and the initial letter is in lowercase.

In the design of a system, a number of classes are identified and grouped together in a class diagram which helps to determine the static relations between those objects. With detailed modelling, the classes of the conceptual design are often split into a number of subclasses.

**Visibility:** To specify the visibility of a class member (i.e., any attribute or method), these notations must be placed before the member's name: + Public • Private # Protected / Derived ~ Packages **Relationships:** A relationship is a general term covering the specific types of logical connections found on class and object diagrams. UML shows the following relationships:

**• Links:** A Link is the basic relationship among objects.

**• Association:** An association represents a family of links. An association can 27 Be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties.

**• Aggregation:** Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship. As a type of association, an aggregation can be named and have the same adornments that an association can. In UML, it is graphically represented as a hollow diamond shape on the containing class with a single line that connects it to the contained class. The aggregate is semantically an extended object that is treated as a unit in many operations, although physically it is made of several lesser objects.

**• Composition:** Composition is a stronger variant of the "has a" association relationship; composition is more specific than aggregation. Composition usually has a strong lifecycle dependency between instances of the container class and instances of the contained class (es). The UML graphical representation of a composition relationship is a filled diamond shape on the containing class end of the tree of lines that connect contained class (es) to the containing class.

**• Generalization:** The Generalization relationship ("is a") indicates that one of the two related classes (the subclass) is considered to be a specialized form of the other (the super type) and the superclass is considered a 'Generalization' of the subclass. The UML graphical representation of a Generalization is a hollow triangle shape on the superclass end of the line (or tree of lines) that connects it to one or more subtypes. The generalization relationship is also known as the inheritance or "is a" relationship. Generalization can only be shown on class diagrams and on Use case diagrams.

**• Realization:** In UML modelling, a realization relationship is a relationship between two model elements, in which one model element realizes (implements or executes) the behaviour that the other model element specifies. The UML graphical representation of a Realization is a hollow triangle shape on the interface end of the dashed line (or tree of lines) that connects it to one or more implementers. A plain arrowhead is used on the interface end of the dashed line that connects it to its users. Realizations can only be shown on class or component diagrams.

**• Dependency:** Dependency is a weaker form of bond which indicates that one class depends on another because it uses it at some point in time. One class depends on another if the independent class is a parameter variable or local variable of a method of the dependent class.

**• Multiplicity:** This association relationship indicates that (at least) one of the two related classes makes reference to the other. The UML representation of an association is a line with an optional arrowhead indicating the role of the object(s) in the relationship, and an optional notation at each end indicating the multiplicity of instances of that entity (the number of objects that participate in the association).

**4.1.2 USE CASE DIAGRAM**

Use case diagrams are used to analyse a system's high-level requirements. As a result, when a system's requirements are examined, the functionality are captured in use cases. The system functionalities are written in an ordered manner as use cases. The actors are the elements that are relevant to the use cases. The term "actor" refers to anything that interacts with the system. Human users, internal apps, or external applications can all be considered actors. So, in a nutshell, when preparing to construct a use case diagram, the following things should be identified:

• Functionalities to be represented as use cases.

• Actors

• Relationships among the use cases and actors. Use case diagrams are drawn to capture the functional requirements of a system. So after Identifying the above items we have to follow the following guidelines to draw an efficient use case diagram:

• The name of a use case is very important. So the name should be chosen in such a way so that it can identify the functionalities performed.

• Give a suitable name for actors.

• Show relationships and dependencies clearly in the diagram.

• Do not try to include all types of relationships. Because the main purpose of the diagram is to identify requirements.

• Use notes whenever required to clarify some important points.

**4.1.3 SEQUENCE DIAGRAM**

A sequence diagram is a diagram that depicts how processes interact with one another and in what order they occur. A sequence diagram depicts item interactions in chronological order. It illustrates the scenario's objects and classes, as well as the sequence of messages sent between them in order to carry out the scenario's functionality. A sequence diagram depicts multiple processes or things that exist simultaneously as parallel vertical lines (lifelines), and the messages passed between them as horizontal arrows, in the order in which they occur. A sequence diagram is used to depict how functionality flows through a use case.

The key points are:

• The main purpose is to represent the logical flow of data with respect to a process

• A sequence diagram displays the objects and not the classes.

• Messages are written with horizontal arrows with the message name written above them, displaying interaction.

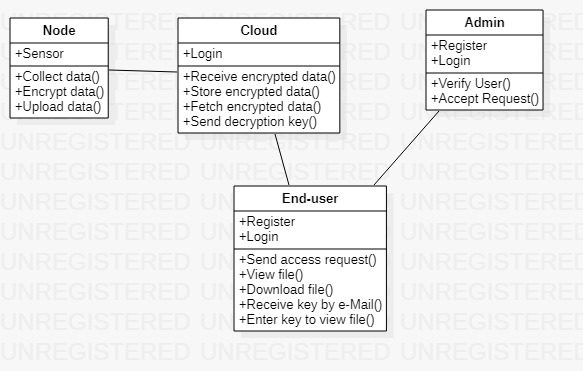
• Solid arrowheads represent synchronous calls, open arrowheads represent asynchronous messages, and dashed lines represent reply messages.

• When an object is destroyed (removed from memory), an X is drawn on top of the lifeline, and the dashed line ceases to be drawn below it (this is not the case in the first example though).

• It should be the result of a message, either from the object itself, or another.

**4.2 UML DIAGRAMS**

**4.2.1 CLASS DIAGRAM**



**Fig. 17.** Class Diagram

**Explanation:**

There are four classes in the class diagram in the fig. 17.

1.Node Class

2. Cloud Class

3.User Class

4.End-User Class

**Node Class**

The attributes of the Node class are sensor and it’s functions are:

1. Collect data ()- to collect the data from the sensor
2. Encrypt data ()- to use encryption algorithms to encrypt data
3. Upload data ()- to upload data to the cloud

**Cloud Class**

The attributes of the cloud class are login and its functions are:

1. Receive encrypted data ()- to receive encrypted data from the sensorby
2. Store encrypted data ()- to store encrypted data in the form of databases
3. Fetch encrypted data ()-to fetch data when a request is passed
4. Send decryption key ()-to send key to verify user

**User Class**

The attributes of User class are Register and Login and its methods are:

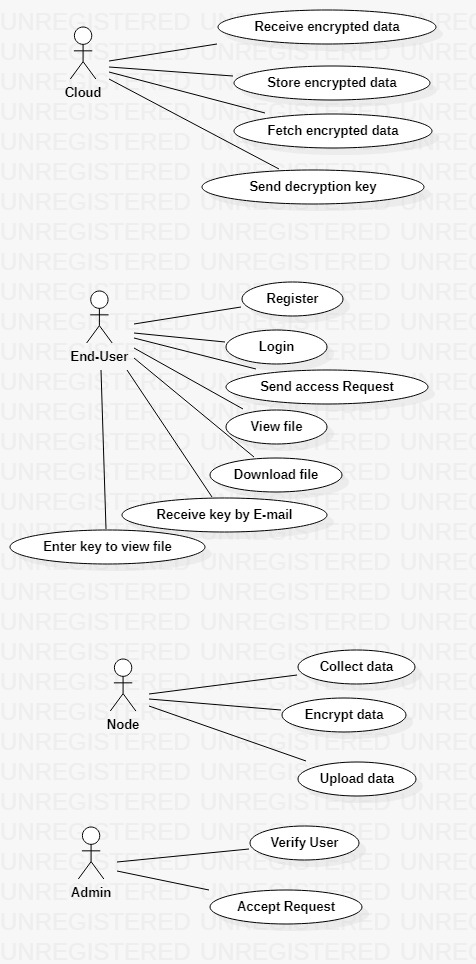
1. Verify User () – to verify the user registered and send the status
2. Accept Request ()- to accept any request sent by the user

**End-User Class**

The attributes of End-User class are Register and Login and its methods are:

1. Send Access Request ()- to send data access request to the User
2. View File ()- to view the patient’s report’s
3. Download File ()- to download the patient’s reports
4. Receive key by E-mail ()- to receive the decryption key through e-mail
5. Enter key to view file ()-to enter key to view the file

**4.2.2 USE CASE DIAGRAM**



**Fig. 18.** Use-Case Diagram

**Explanation:**

In the Use Case diagram Fig. 18., we have four actors Node, User, Cloud, End-User

**Node**

This use case diagram has the following use cases

1. Collect data - to collect the data from the sensor
2. Encrypt data - to use encryption algorithms to encrypt data
3. Upload data - to upload data to the cloud

**Cloud**

This use case diagram has the following use cases

1. Receive encrypted data - to receive encrypted data from the sensorby
2. Store encrypted data - to store encrypted data in the form of databases
3. Fetch encrypted data -to fetch data when a request is passed
4. Send decryption key -to send key to verify user

**User**

This use case diagram has the following use cases

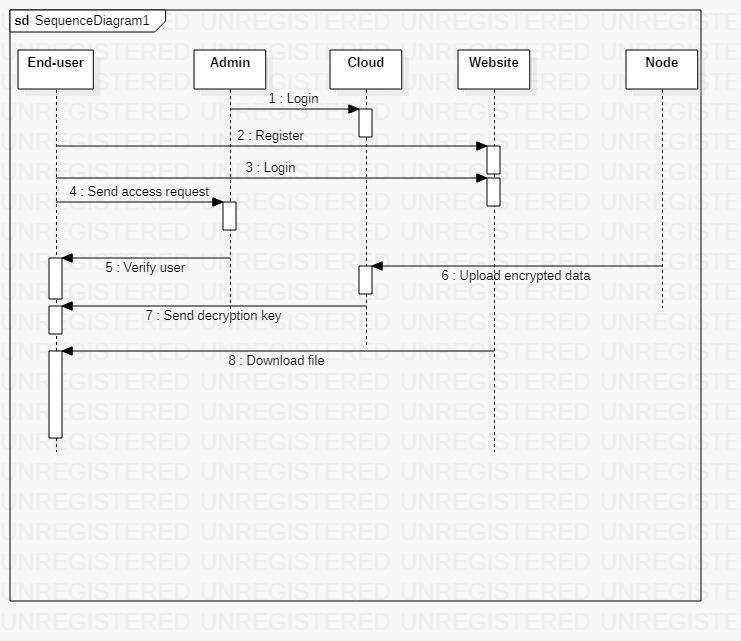
1. Verify User – to verify the user registered and send the status
2. Accept Request - to accept any request sent by the user

**End-User**

This use case diagram has the following use cases

1. Send Access Request - to send data access request to the User
2. View File - to view the patient’s report’s
3. Download File - to download the patient’s reports
4. Receive key by E-mail - to receive the decryption key through e-mail
5. Enter key to view file -to enter key to view the file.

**4.2.3 SEQUENCE DIAGRAM**



**Fig. 19.** Sequence Diagram

**Explanation**

In Sequence Diagram Fig. 19. the actions are performed between the End-User, User, node, cloud

1. The logins first registers with the cloud
2. The end-user registers through the website to access data
3. The end-user logs in through the website
4. End-User sends the access request to the User
5. The User then verifies the User
6. The data is collected at the node then encrypted and sent to the cloud
7. The User sends the decryption key to the user to access files
8. The user now download’s any file

**5. IMPLEMENTATION AND RESULTS**

**5.1 INTRODUCTION**

**5.1.1 PostgreSQL Installation**

**Step-1:** To install PostgreSQL go to  <https://www.postgresql.org/download> and select Windows.

**Step-2:** Download the installer under the Interactive Installer by Enterprise DB.

**Step-3:** Select PostgreSQL version.

1. You will be prompted to desired PostgreSQL version and operating system. Select the latest PostgreSQL version and OS as per your environment.
2. Click the Download Button.

**Step-4:** Open exe file. Once you download PostgreSQL, open the downloaded exe and click next on the install welcome screen.

**Step-5:** Update location.

1. Change the Installation directory if required, else leave it to default.
2. Click Next.

**Step-6:** Select components.

1. You may choose the components you want to install in your system. You may uncheck Stack Builder.
2. Click Next.

**Step-7:** Check data location.

1. You may change the data location.
2. Click Next.

**Step-8:** Enter password.

1. Enter super user password. Make a note of it.
2. Click Next.

**Step-9:** Check port option.

1. Leave the port number default.
2. Click Next.

**Step-10:** Check summary.

1. Check the pre-installation summary.
2. Click Next.

**Step-11:** Ready to install. Click the next button.

**Step-12:** Check stack builder prompt. Once install is complete you will see the Stack Builder prompt

1. Uncheck that option. We will use Stack Builder in more advance tutorials.
2. Click Finish.

**Step-13:** Launch PostgreSQL. To launch PostgreSQL, go to Start Menu and search pgUser4

**Step-14:** Check pgUser. You will see pgUser homepage.

**Step-15:** Find PostgreSQL 10. Click on Servers > PostgreSQL 10 in the left tree

**Step-16:** Enter password.

1. Enter super user password set during installation
2. Click OK

**Step-17:** Check Dashboard. You will see the Dashboard of PostgreSQL.

**5.1.2 Installing Firebase**

**Step-1:** Download firebase library from following link: <https://github.com/FirebaseExtended/firebase-arduino>

**Step-2:** Open Sketch > include library > add.ZIP library and add ZIP library to Arduino IDE.

**Step-3:** Open Tools > Manage Library and install Arduino Json **version 5.13.5**library.

**Step-4:** Make sure you select ESP8266 board present in the tools in the board menu.

**Step-5:** Open firebase and create a project. Then open that project and initialize it for the webpage.

**Step-6:** Now a unique code containing the configuration will be displayed.

**Step-7:** Go to storage and rules and allow the permissions for the data operations such as read and write.

## 5.1.3 Installing PyCharm?

Here is a step-by-step process on how to download and install Pycharm IDE on Windows:

**Step 1:**To download PyCharm visit the website and Click the "DOWNLOAD" link under the Community Section. [https://www.jetbrains.com/pycharm/download/.](https://www.jetbrains.com/pycharm/download/)

**Step 2:** Once the download is complete, run the exe to install PyCharm. The setup wizard should have started. Click “Next”.

**Step 3:** On the next screen, Change the installation path if required. Click “Next”.

**Step 4:** On the next screen, you can create a desktop shortcut if you want and click on “next”.

**Step 5:** Choose the start menu folder. Keep selecting JetBrains and click on “Install”.

**Step 6:** Wait for the installation to finish.

**Step 7:** Once installation finished, you should receive a message screen that PyCharm is installed. If you want to go ahead and run it, click the “Run PyCharm Community Edition” box first and click “Finish”.

**Step 8:** After you click on "Finish," the Following screen will appear.

**5.1.4 Installing Django**

**Step 1-** Install pip- Open command prompt and enter following command

**Step 2 -** Install virtual environment- Enter following command in cmd

**Step 4 -** Set Virtual environment **-**Create a virtual environment by giving this command in cmd

**Step 5-** Install Django- Install Django by giving following command

**Step 6-**Return to the env\_site directory

**Step 7-**Change directory

**Step 8-**Start the server- Start the server by typing following command in cmd

**5.2 SOFTWARE DESIGN**

**5.2.1 Front End Design**

**HTML:** HTML is a computer-readable format that instructs a computer on how to display a web page. The documents are simple text files with special "tags" or codes that a web browser uses to decode and show information on your computer screen. HTML is an abbreviation for Hyper Text Markup Language; an HTML file is a text file with small markup elements. The markup tags instruct the web browser on how to display the page.

**CSS**: CSS is an abbreviation for Cascading Style Sheets. It's a simple design language that's meant to make the process of presenting webpages easier. CSS is in charge of a webpage's appearance. CSS allows you to manage the color of the text, font style, paragraph spacing, how columns are scaled and laid out, what background images or colors are used, layout designs and variations in display for different devices and screen sizes, and a range of other effects. CSS is simple to learn and understand, but it provides significant control over the presentation of an HTML document. CSS is most frequently used in conjunction with the markup languages HTML or XHTML.

**5.2.2 Back End Design PostgreSQL**

1. PostgreSQL is a powerful object-relational database system that is free and open source.
2. It has over 20 years of active development and a proven architecture, earning it a strong reputation for dependability, data integrity, and correctness.
3. PostgreSQL is compatible with all major operating systems, including Linux, UNIX, and Windows.
4. It can handle text, images, sounds, and video, and has programming interfaces for C/C++, Java, Perl, Python, Ruby, and Open Database Connectivity (ODBC).
5. Many web, mobile, and analytics applications use PostgreSQL as their primary data store or data warehouse.
6. PostgreSQL has a long history of supporting advanced data types, and it offers the same level of performance optimization as its commercial database counterparts, such as Oracle and SQL Server.
   1. **SAMPLE CODE**

from ecelgamal import decryption,secp256k1,Point

import pyrebase

config = {

"apiKey" : "9Va7NU7FhlX6krcgVVZj41z3wT1k5qJnY7TYXZFI",

"authDomain" : "majorproject-node1234.firebaseapp.com",

"databaseURL" : "https://majorproject-node1234-default-rtdb.firebaseio.com/",

"storageBucket" : "majorproject-node1234.appspot.com"

}

def retrieveDataFromCloud():

firebase = pyrebase.initialize\_app(config)

db = firebase.database()

encrypted\_ciphers=db.child("pulseRate").get().val()

priv\_key=int(encrypted\_ciphers["private key"])

C1\_x=int(encrypted\_ciphers["cipher1"]["X"])

C1\_y=int(encrypted\_ciphers["cipher1"]["Y"])

C2\_x=int(encrypted\_ciphers["cipher2"]["X"])

C2\_y=int(encrypted\_ciphers["cipher2"]["Y"])

C1=…

[4:23 PM, 6/17/2022] Siri Gvp: import random, math

from os import urandom

from typing import Callable, Tuple, Optional

from dataclasses import dataclass

from abc import ABC, abstractmethod

from binascii import hexlify

def egcd(a, b) :

if a == 0 :

return b, 0, 1

else :

g, y, x = egcd(b % a, a)

return g, x - (b // a) \* y, y

def modinv(a, m) :

a = a % m

g, x, y = egcd(a, m)

if g != 1 :

raise Exception("modular inverse does not exist")

else :

return x % m

def modsqrt(a, p) :

""" Find a quadratic residue (mod p) of 'a'. p

must be an odd prime.

Solve the congruence of the form:

x^2 = a (mod p)

And returns x. Note that p - x is also a root.

0 is returned is no square root exists for

these a and p.

The Tonelli-Shanks algorithm is used (except

for some simple cases in which the solution

is known from an identity). This algorithm

runs in polynomial time (unless the

generalized Riemann hypothesis is false).

"""

# Simple cases

#

if legendre\_symbol(a, p) != 1 :

return 0

elif a == 0 :

return 0

elif p == 2 :

return p

elif p % 4 == 3 :

return pow(a, (p + 1) // 4, p)

# Partition p-1 to s \* 2^e for an odd s (i.e.

# reduce all the powers of 2 from p-1)

#

s = p - 1

e = 0

while s % 2 == 0 :

# Interesting bug. s /= 2 and s = int(s) not equals to s //= 2

# s /= 2

# s = int(s)

s //= 2

e += 1

# Find some 'n' with a legendre symbol n|p = -1.

# Shouldn't take long.

#

n = 2

while legendre\_symbol(n, p) != -1 :

n += 1

# Here be dragons!

# Read the paper "Square roots from 1; 24, 51,

# 10 to Dan Shanks" by Ezra Brown for more

# information

#

# x is a guess of the square root that gets better

# with each iteration.

# b is the "fudge factor" - by how much we're off

# with the guess. The invariant x^2 = ab (mod p)

# is maintained throughout the loop.

# g is used for successive powers of n to update

# both a and b

# r is the exponent - decreases with each update

#

x = pow(a, (s + 1) // 2, p)

b = pow(a, s, p)

g = pow(n, s, p)

r = e

while True :

t = b

m = 0

for m in range(r) :

if t == 1 :

break

t = pow(t, 2, p)

if m == 0 :

return x

gs = pow(g, 2 \*\* (r - m - 1), p)

g = (gs \* gs) % p

x = (x \* gs) % p

b = (b \* g) % p

r = m

def legendre\_symbol(a, p) :

""" Compute the Legendre symbol a|p using

Euler's criterion. p is a prime, a is

relatively prime to p (if p divides

a, then a|p = 0)

Returns 1 if a has a square root modulo

p, -1 otherwise.

"""

ls = pow(a, (p - 1) // 2, p)

return -1 if ls == p - 1 else ls

def int\_length\_in\_byte(n: int) :

assert n >= 0

length = 0

while n :

n >>= 8

length += 1

return length

@dataclass

class Point :

x: Optional[int]

y: Optional[int]

curve: "Curve"

def is\_at\_infinity(self) -> bool :

return self.x is None and self.y is None

def \_post\_init\_(self) :

if not self.is\_at\_infinity() and not self.curve.is\_on\_curve(self) :

raise ValueError("The point is not on the curve.")

def \_str\_(self) :

if self.is\_at\_infinity() :

return f"Point(At infinity, Curve={str(self.curve)})"

else :

return f"Point(X={self.x}, Y={self.y}, Curve={str(self.curve)})"

def \_repr\_(self) :

return self.\_str\_()

def \_eq\_(self, other) :

return self.curve == other.curve and self.x == other.x and self.y == other.y

def \_neg\_(self) :

return self.curve.neg\_point(self)

def \_add\_(self, other) :

return self.curve.add\_point(self, other)

def \_radd\_(self, other) :

return self.\_add\_(other)

def \_sub\_(self, other) :

negative = - other

return self.\_add\_(negative)

def \_mul\_(self, scalar: int) :

return self.curve.mul\_point(scalar, self)

def \_rmul\_(self, scalar: int) :

return self.\_mul\_(scalar)

@dataclass

class Curve(ABC) :

name: str

a: int

b: int

p: int

n: int

G\_x: int

G\_y: int

def \_str\_(self) :

return self.name

def \_repr\_(self) :

return self.\_str\_()

def \_eq\_(self, other) :

return (

self.a == other.a and self.b == other.b and self.p == other.p and

self.n == other.n and self.G\_x == other.G\_x and self.G\_y == other.G\_y

)

@property

def G(self) -> Point :

return Point(self.G\_x, self.G\_y, self)

@property

def INF(self) -> Point :

return Point(None, None, self)

def is\_on\_curve(self, P: Point) -> bool :

if P.curve != self :

return False

return P.is\_at\_infinity() or self.\_is\_on\_curve(P)

@abstractmethod

def \_is\_on\_curve(self, P: Point) -> bool :

pass

def add\_point(self, P: Point, Q: Point) -> Point :

if (not self.is\_on\_curve(P)) or (not self.is\_on\_curve(Q)) :

raise ValueError("The points are not on the curve.")

if P.is\_at\_infinity() :

return Q

elif Q.is\_at\_infinity() :

return P

if P == -Q :

return self.INF

if P == Q :

return self.\_double\_point(P)

return self.\_add\_point(P, Q)

@abstractmethod

def \_add\_point(self, P: Point, Q: Point) -> Point :

pass

@abstractmethod

def \_double\_point(self, P: Point) -> Point :

pass

def mul\_point(self, d: int, P: Point) -> Point :

"""

https://en.wikipedia.org/wiki/Elliptic\_curve\_point\_multiplication

"""

if not self.is\_on\_curve(P) :

raise ValueError("The point is not on the curve.")

if P.is\_at\_infinity() :

return self.INF

if d == 0 :

return self.INF

res = self.INF

is\_negative\_scalar = d < 0

d = -d if is\_negative\_scalar else d

tmp = P

while d :

if d & 0x1 == 1 :

res = self.add\_point(res, tmp)

tmp = self.add\_point(tmp, tmp)

d >>= 1

if is\_negative\_scalar :

return -res

else :

return res

def neg\_point(self, P: Point) -> Point :

if not self.is\_on\_curve(P) :

raise ValueError("The point is not on the curve.")

if P.is\_at\_infinity() :

return self.INF

return self.\_neg\_point(P)

@abstractmethod

def \_neg\_point(self, P: Point) -> Point :

pass

@abstractmethod

def compute\_y(self, x: int) -> int :

pass

def encode\_point(self, plaintext: bytes) -> Point :

plaintext = len(plaintext).to\_bytes(1, byteorder="big") + plaintext

while True :

x = int.from\_bytes(plaintext, "big")

y = self.compute\_y(x)

if y :

return Point(x, y, self)

plaintext += urandom(1)

def decode\_point(self, M: Point) -> bytes :

byte\_len = int\_length\_in\_byte(M.x)

plaintext\_len = (M.x >> ((byte\_len - 1) \* 8)) & 0xff

plaintext = ((M.x >> ((byte\_len - plaintext\_len - 1) \* 8))

& (int.from\_bytes(b"\xff" \* plaintext\_len, "big")))

return plaintext.to\_bytes(plaintext\_len, byteorder="big")

class ShortWeierstrassCurve(Curve) :

"""

y^2 = x^3 + a\*x + b

https://en.wikipedia.org/wiki/Elliptic\_curve

"""

def \_is\_on\_curve(self, P: Point) -> bool :

left = P.y \* P.y

right = (P.x \* P.x \* P.x) + (self.a \* P.x) + self.b

return (left - right) % self.p == 0

def \_add\_point(self, P: Point, Q: Point) -> Point :

# s = (yP - yQ) / (xP - xQ)

# xR = s^2 - xP - xQ

# yR = yP + s \* (xR - xP)

delta\_x = P.x - Q.x

delta\_y = P.y - Q.y

s = delta\_y \* modinv(delta\_x, self.p)

res\_x = (s \* s - P.x - Q.x) % self.p

res\_y = (P.y + s \* (res\_x - P.x)) % self.p

return - Point(res\_x, res\_y, self)

def \_double\_point(self, P: Point) -> Point :

# s = (3 \* xP^2 + a) / (2 \* yP)

# xR = s^2 - 2 \* xP

# yR = yP + s \* (xR - xP)

s = (3 \* P.x \* P.x + self.a) \* modinv(2 \* P.y, self.p)

res\_x = (s \* s - 2 \* P.x) % self.p

res\_y = (P.y + s \* (res\_x - P.x)) % self.p

return - Point(res\_x, res\_y, self)

def \_neg\_point(self, P: Point) -> Point :

return Point(P.x, -P.y % self.p, self)

def compute\_y(self, x) -> int :

right = (x \* x \* x + self.a \* x + self.b) % self.p

y = modsqrt(right, self.p)

return y

class MontgomeryCurve(Curve) :

"""

by^2 = x^3 + ax^2 + x

https://en.wikipedia.org/wiki/Montgomery\_curve

"""

def \_is\_on\_curve(self, P: Point) -> bool :

left = self.b \* P.y \* P.y

right = (P.x \* P.x \* P.x) + (self.a \* P.x \* P.x) + P.x

return (left - right) % self.p == 0

def \_add\_point(self, P: Point, Q: Point) -> Point :

# s = (yP - yQ) / (xP - xQ)

# xR = b \* s^2 - a - xP - xQ

# yR = yP + s \* (xR - xP)

delta\_x = P.x - Q.x

delta\_y = P.y - Q.y

s = delta\_y \* modinv(delta\_x, self.p)

res\_x = (self.b \* s \* s - self.a - P.x - Q.x) % self.p

res\_y = (P.y + s \* (res\_x - P.x)) % self.p

return - Point(res\_x, res\_y, self)

def \_double\_point(self, P: Point) -> Point :

# s = (3 \* xP^2 + 2 \* a \* xP + 1) / (2 \* b \* yP)

# xR = b \* s^2 - a - 2 \* xP

# yR = yP + s \* (xR - xP)

up = 3 \* P.x \* P.x + 2 \* self.a \* P.x + 1

down = 2 \* self.b \* P.y

s = up \* modinv(down, self.p)

res\_x = (self.b \* s \* s - self.a - 2 \* P.x) % self.p

res\_y = (P.y + s \* (res\_x - P.x)) % self.p

return - Point(res\_x, res\_y, self)

def \_neg\_point(self, P: Point) -> Point :

return Point(P.x, -P.y % self.p, self)

def compute\_y(self, x: int) -> int :

right = (x \* x \* x + self.a \* x \* x + x) % self.p

inv\_b = modinv(self.b, self.p)

right = (right \* inv\_b) % self.p

y = modsqrt(right, self.p)

return y

class TwistedEdwardsCurve(Curve) :

"""

ax^2 + y^2 = 1 + bx^2y^2

https://en.wikipedia.org/wiki/Twisted\_Edwards\_curve

"""

def \_is\_on\_curve(self, P: Point) -> bool :

left = self.a \* P.x \* P.x + P.y \* P.y

right = 1 + self.b \* P.x \* P.x \* P.y \* P.y

return (left - right) % self.p == 0

def \_add\_point(self, P: Point, Q: Point) -> Point :

# xR = (xP \* yQ + yP \* xQ) / (1 + b \* xP \* xQ \* yP \* yQ)

up\_x = P.x \* Q.y + P.y \* Q.x

down\_x = 1 + self.b \* P.x \* Q.x \* P.y \* Q.y

res\_x = (up\_x \* modinv(down\_x, self.p)) % self.p

# yR = (yP \* yQ - a \* xP \* xQ) / (1 - b \* xP \* xQ \* yP \* yQ)

up\_y = P.y \* Q.y - self.a \* P.x \* Q.x

down\_y = 1 - self.b \* P.x \* Q.x \* P.y \* Q.y

res\_y = (up\_y \* modinv(down\_y, self.p)) % self.p

return Point(res\_x, res\_y, self)

def \_double\_point(self, P: Point) -> Point :

# xR = (2 \* xP \* yP) / (a \* xP^2 + yP^2)

up\_x = 2 \* P.x \* P.y

down\_x = self.a \* P.x \* P.x + P.y \* P.y

res\_x = (up\_x \* modinv(down\_x, self.p)) % self.p

# yR = (yP^2 - a \* xP \* xP) / (2 - a \* xP^2 - yP^2)

up\_y = P.y \* P.y - self.a \* P.x \* P.x

down\_y = 2 - self.a \* P.x \* P.x - P.y \* P.y

res\_y = (up\_y \* modinv(down\_y, self.p)) % self.p

return Point(res\_x, res\_y, self)

def \_neg\_point(self, P: Point) -> Point :

return Point(-P.x % self.p, P.y, self)

def compute\_y(self, x: int) -> int :

# (bx^2 - 1) \* y^2 = ax^2 - 1

right = self.a \* x \* x - 1

left\_scale = (self.b \* x \* x - 1) % self.p

inv\_scale = modinv(left\_scale, self.p)

right = (right \* inv\_scale) % self.p

y = modsqrt(right, self.p)

return y

P256 = ShortWeierstrassCurve(

name="P256",

a=-3,

b=41058363725152142129326129780047268409114441015993725554835256314039467401291,

p=0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff,

n=0xffffffff00000000ffffffffffffffffbce6faada7179e84f3b9cac2fc632551,

G\_x=0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296,

G\_y=0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5

)

secp256k1 = ShortWeierstrassCurve(

name="secp256k1",

a=0,

b=7,

p=0xfffffffffffffffffffffffffffffffffffffffffffffffffffffffefffffc2f,

n=0xfffffffffffffffffffffffffffffffebaaedce6af48a03bbfd25e8cd0364141,

G\_x=0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798,

G\_y=0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

)

Curve25519 = MontgomeryCurve(

name="Curve25519",

a=486662,

b=1,

p=0x7fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffed,

n=0x1000000000000000000000000000000014def9dea2f79cd65812631a5cf5d3ed,

G\_x=0x9,

G\_y=0x20ae19a1b8a086b4e01edd2c7748d14c923d4d7e6d7c61b229e9c5a27eced3d9

)

M383 = MontgomeryCurve(

name="M383",

a=2065150,

b=1,

p=0x7fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff45,

n=0x10000000000000000000000000000000000000000000000006c79673ac36ba6e7a32576f7b1b249e46bbc225be9071d7,

G\_x=0xc,

G\_y=0x1ec7ed04aaf834af310e304b2da0f328e7c165f0e8988abd3992861290f617aa1f1b2e7d0b6e332e969991b62555e77e

)

E222 = TwistedEdwardsCurve(

name="E222",

a=1,

b=160102,

p=0x3fffffffffffffffffffffffffffffffffffffffffffffffffffff8b,

n=0xffffffffffffffffffffffffffff70cbc95e932f802f31423598cbf,

G\_x=0x19b12bb156a389e55c9768c303316d07c23adab3736eb2bc3eb54e51,

G\_y=0x1c

)

E382 = TwistedEdwardsCurve(

name="E382",

a=1,

b=-67254,

p=0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff97,

n=0xfffffffffffffffffffffffffffffffffffffffffffffffd5fb21f21e95eee17c5e69281b102d2773e27e13fd3c9719,

G\_x=0x196f8dd0eab20391e5f05be96e8d20ae68f840032b0b64352923bab85364841193517dbce8105398ebc0cc9470f79603,

G\_y=0x11

)

@dataclass

class ElGamal :

curve: Curve

def encrypt(self, plaintext: bytes, public\_key: Point,

randfunc: Callable = None) -> Tuple[Point, Point] :

return self.encrypt\_bytes(plaintext, public\_key, randfunc)

def decrypt(self, private\_key: int, C1: Point, C2: Point) -> bytes :

return self.decrypt\_bytes(private\_key, C1, C2)

def encrypt\_bytes(self, plaintext: bytes, public\_key: Point,

randfunc: Callable = None) -> Tuple[Point, Point] :

# Encode plaintext into a curve point

M = self.curve.encode\_point(plaintext)

return self.encrypt\_point(M, public\_key, randfunc)

def decrypt\_bytes(self, private\_key: int, C1: Point, C2: Point) -> bytes :

M = self.decrypt\_point(private\_key, C1, C2)

return self.curve.decode\_point(M)

def encrypt\_point(self, plaintext: Point, public\_key: Point,

randfunc: Callable = None) -> Tuple[Point, Point] :

randfunc = randfunc or urandom

# Base point G

G = self.curve.G

M = plaintext

random.seed(randfunc(1024))

k = random.randint(1, self.curve.n)

C1 = k \* G

C2 = M + k \* public\_key

return C1, C2

def decrypt\_point(self, private\_key: int, C1: Point, C2: Point) -> Point :

M = C2 + (self.curve.n - private\_key) \* C1

return M

def gen\_keypair(curve: Curve,

randfunc: Callable = None) -> Tuple[int, Point] :

randfunc = randfunc or urandom

private\_key = gen\_private\_key(curve, randfunc)

public\_key = get\_public\_key(private\_key, curve)

return private\_key, public\_key

def gen\_private\_key(curve: Curve,

randfunc: Callable = None) -> int :

order\_bits = 0

order = curve.n

while order > 0 :

order >>= 1

order\_bits += 1

order\_bytes = (order\_bits + 7) // 8

extra\_bits = order\_bytes \* 8 - order\_bits

rand = int(hexlify(randfunc(order\_bytes)), 16)

rand >>= extra\_bits

while rand >= curve.n :

rand = int(hexlify(randfunc(order\_bytes)), 16)

rand >>= extra\_bits

return rand

def get\_public\_key(d: int, curve: Curve) -> Point :

return d \* curve.G

def heart\_beat\_measure() :

digits = "789"

OTP = ""

for i in range(2) :

OTP += digits[math.floor(random.random() \* 3)]

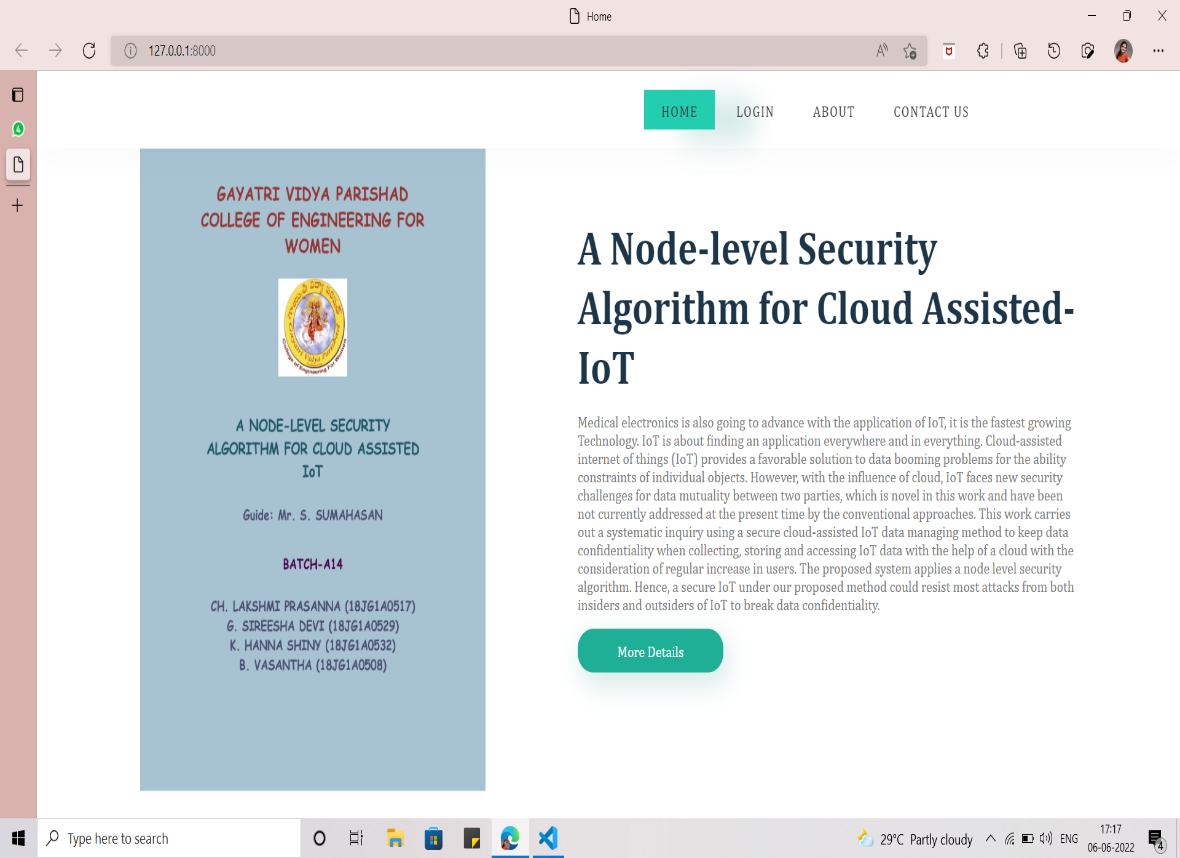
return OTP

def decryption(pri\_key, C1, C2) :

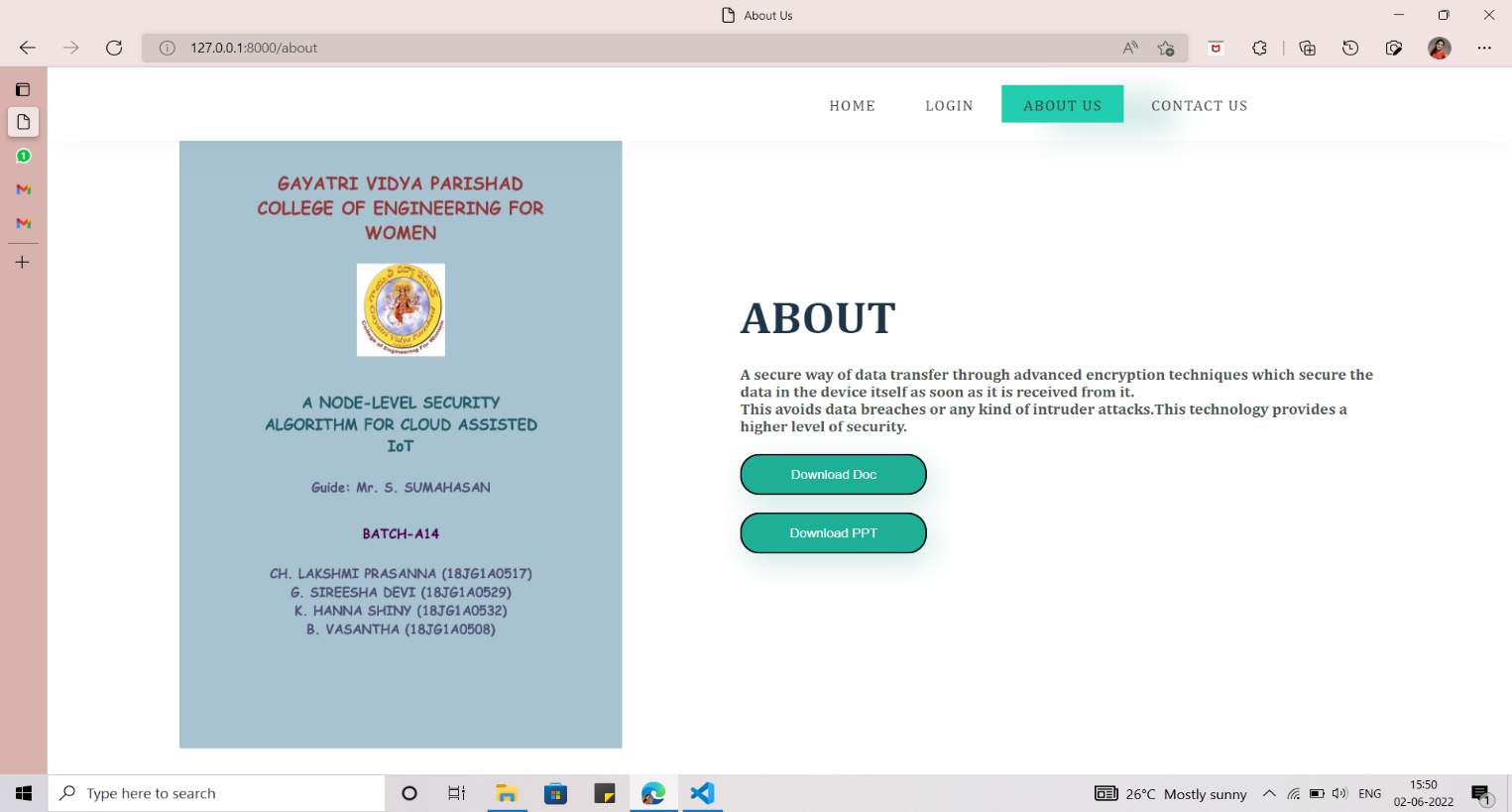
cipher\_elg = ElGamal(secp256k1)

return cipher\_elg.decrypt(pri\_key, C1, C2).decode('utf-8')

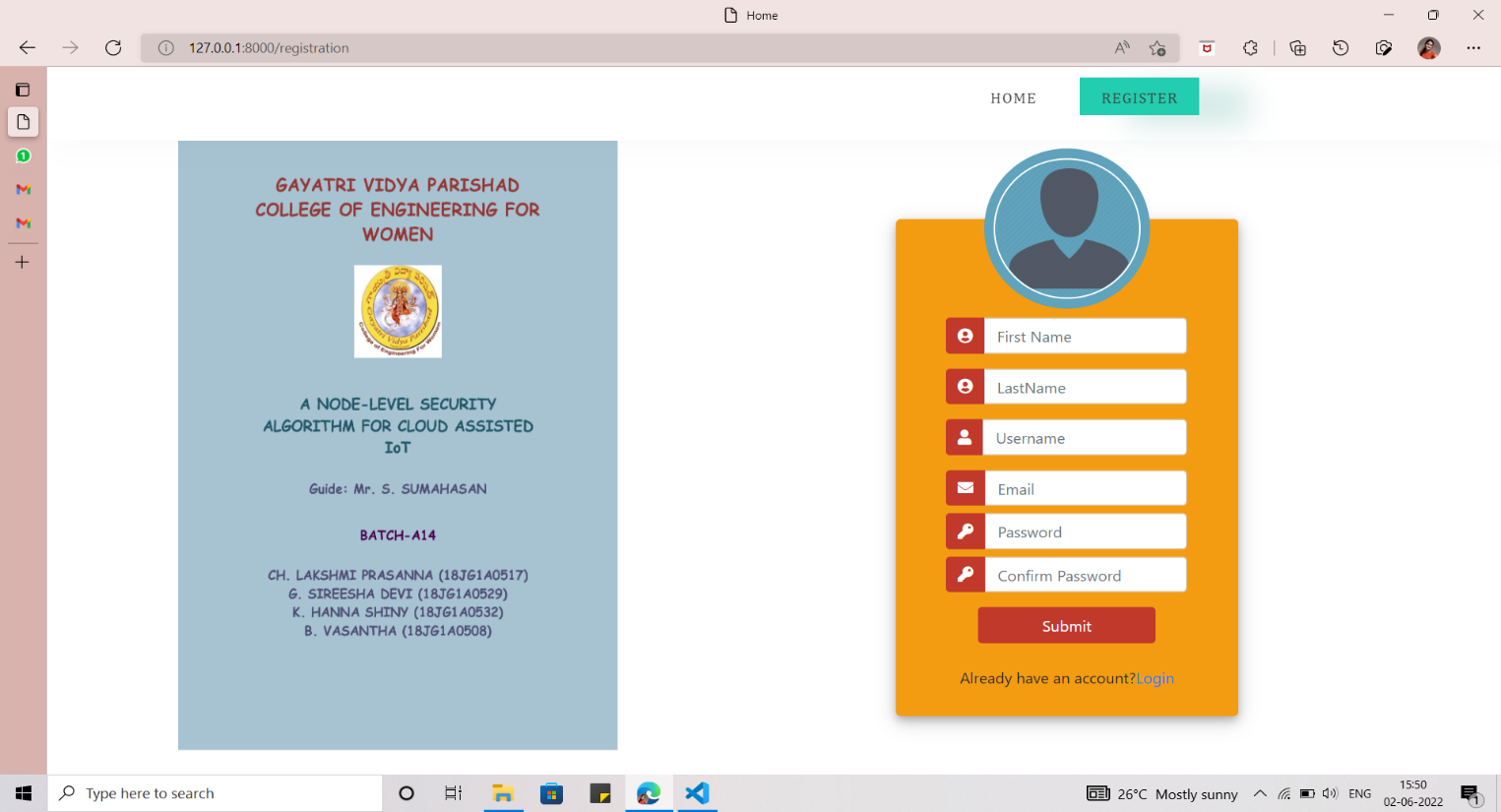
* 1. **OUTPUT SCREENS**



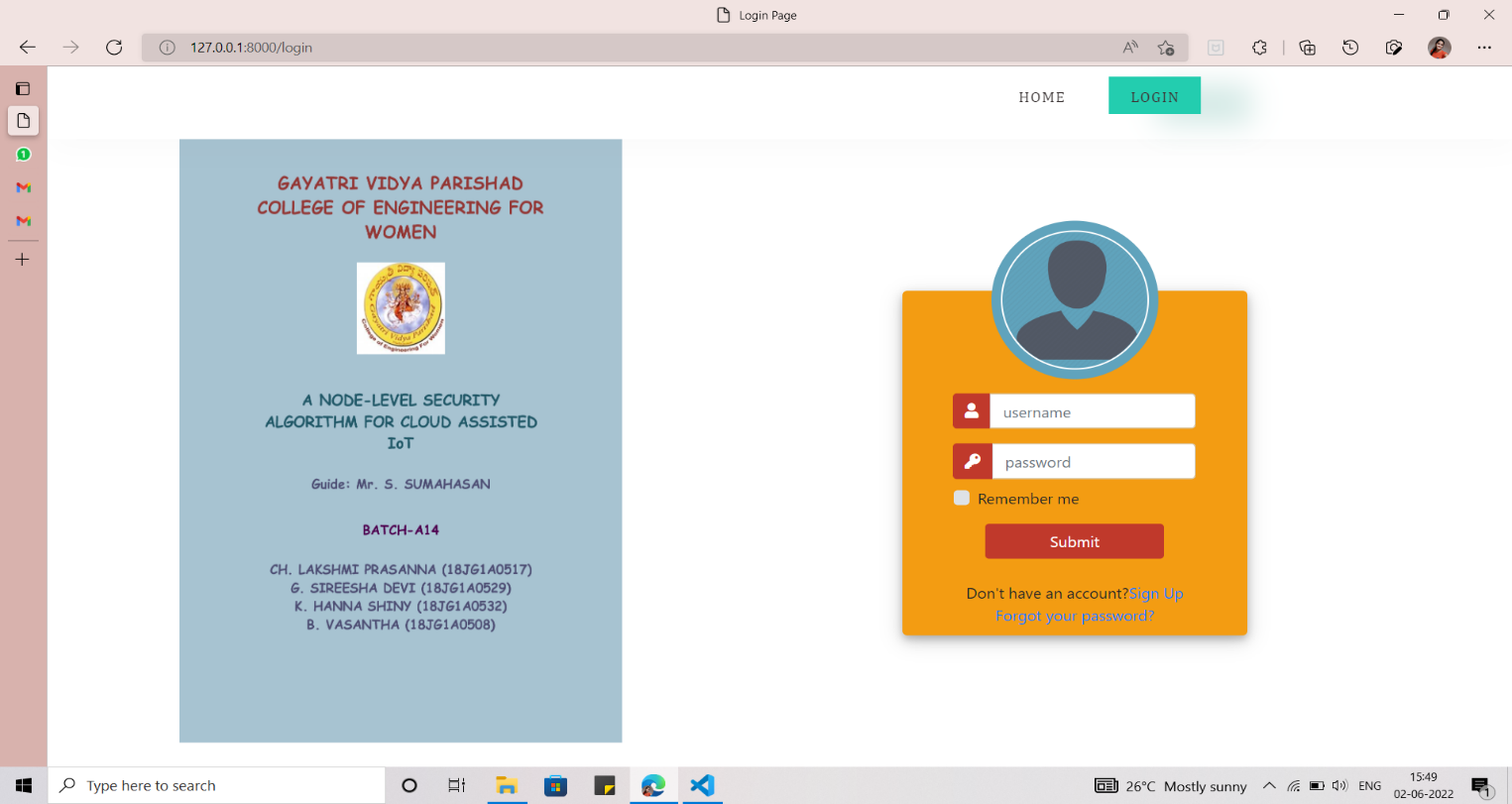
**Screen 1.** Home Page- It consists of a brief description of the project. The navbar consists of ‘Home’, ‘Login’, ‘About’ and ‘Contact Us’ buttons. The More Details consists of the details of the team members.



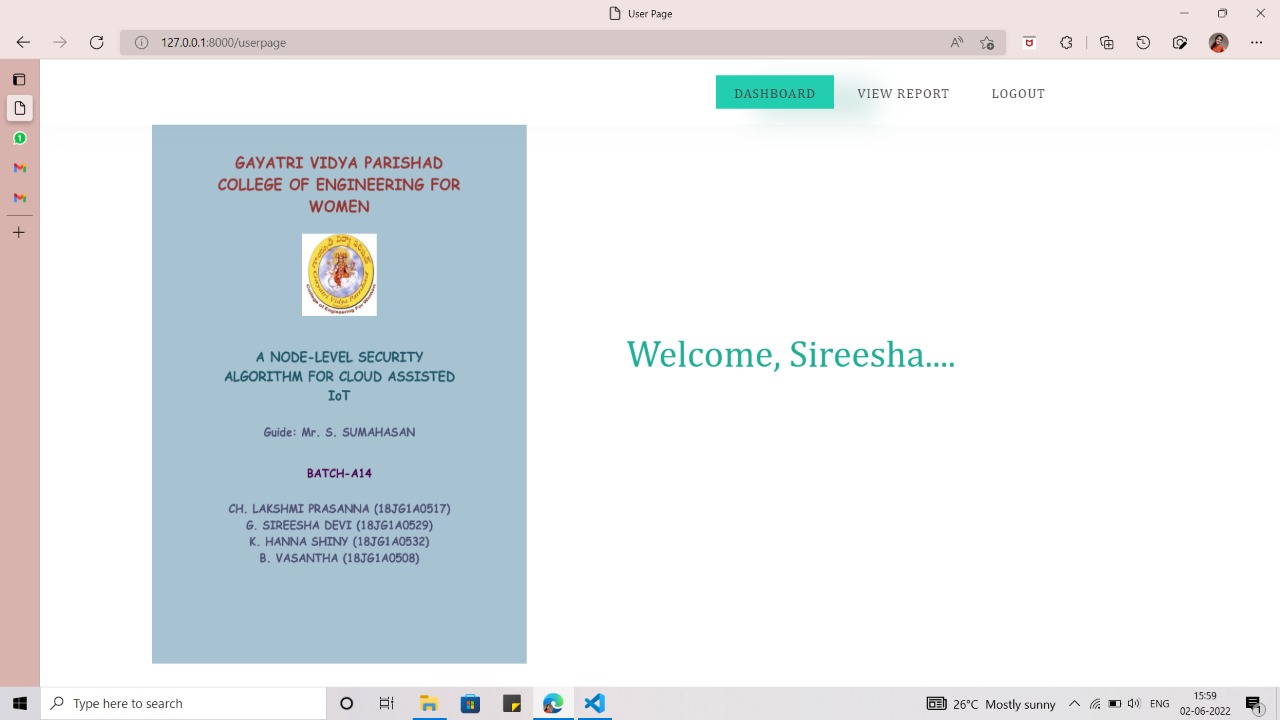
**Screen 2.** About Page- It consists of Download Doc and Download PPT buttons used to download the ppt’s and documents



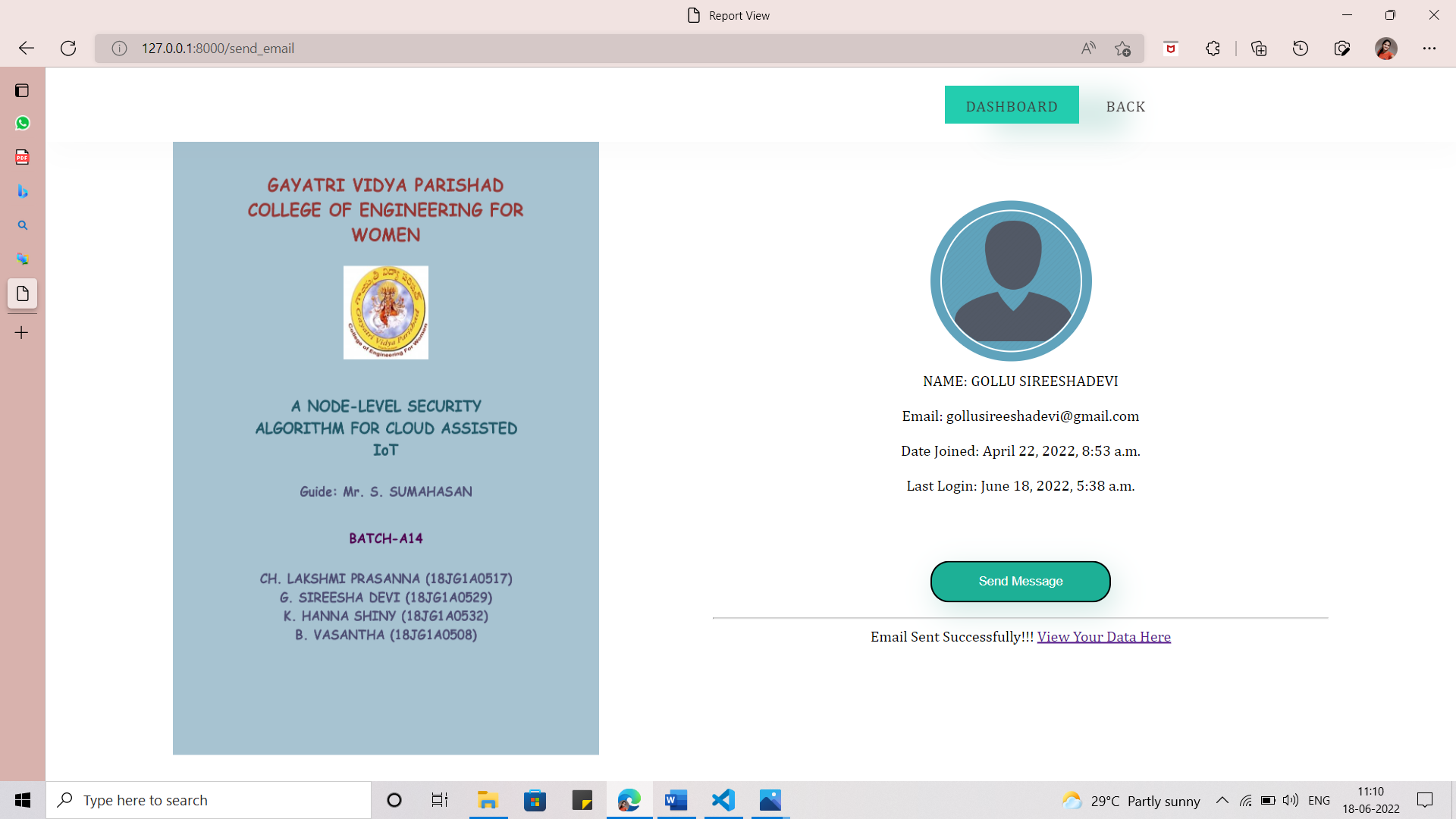
**Screen 3.** Registration Page- It takes the details of the user that registers for first time



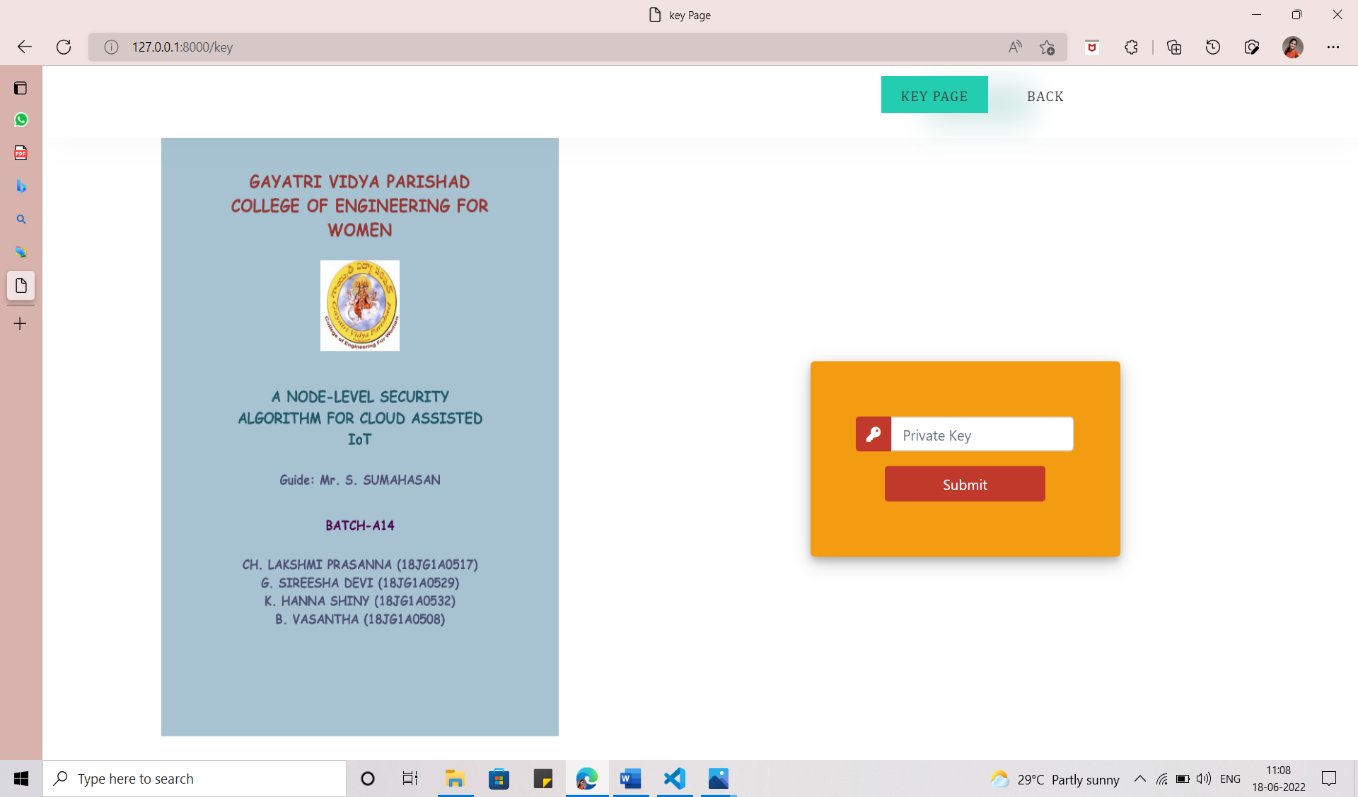
**Screen 4.** Login page- It validates the user details that tries to log in



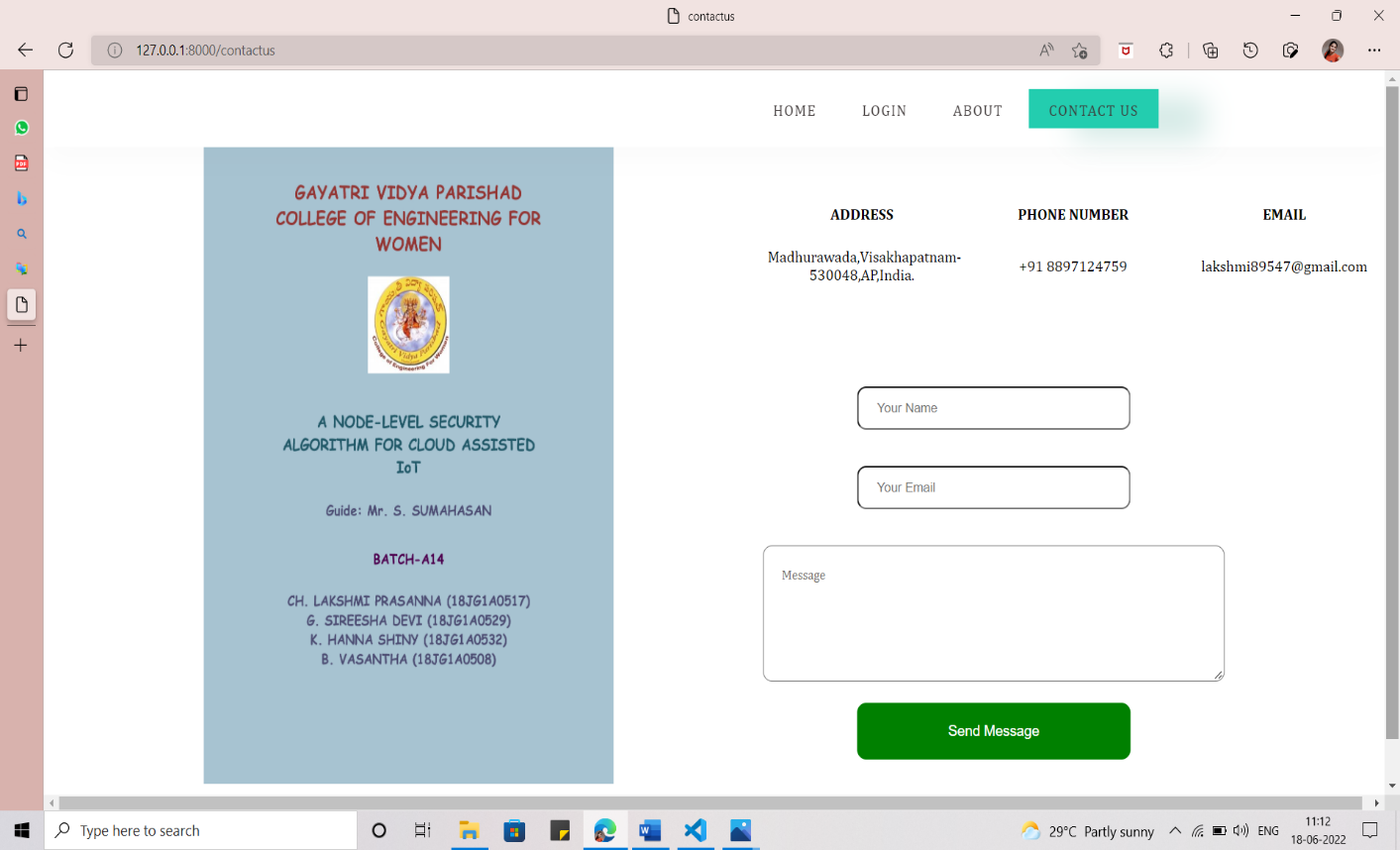
**Screen 5.** Welcome Page- It consists of a welcome message

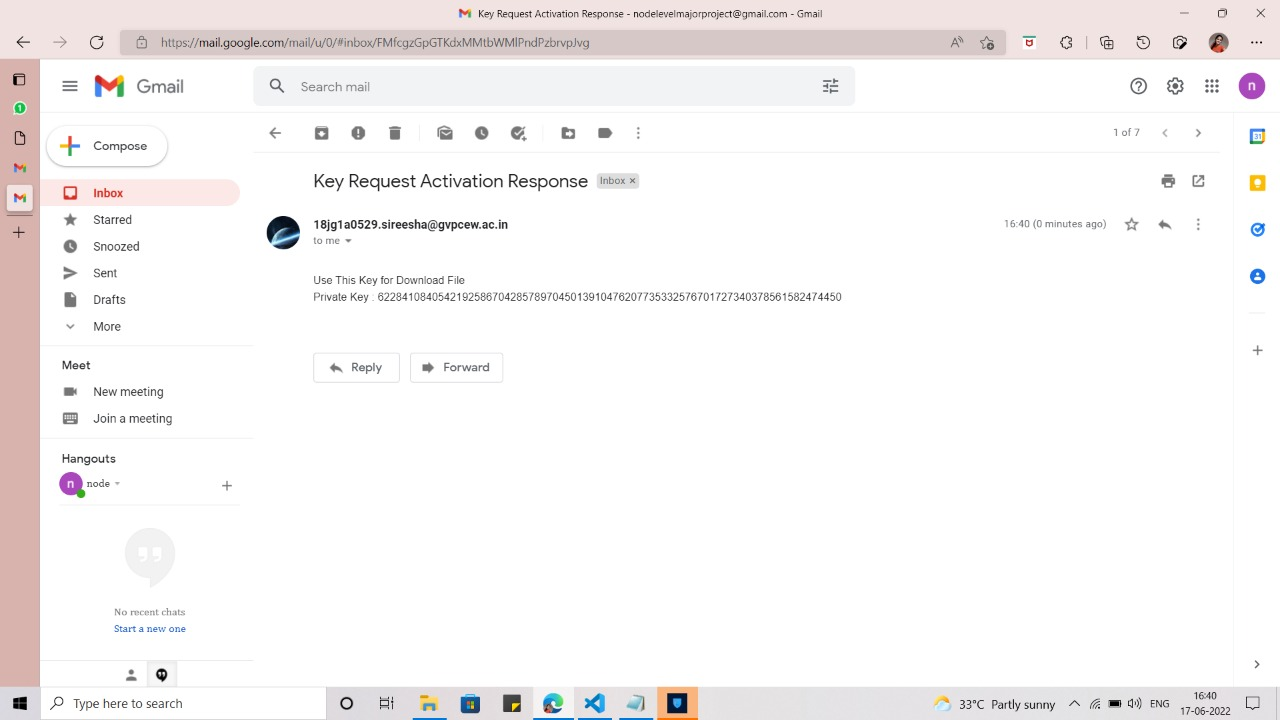


**Screen 6.** User Verification Page- Consists of User login details

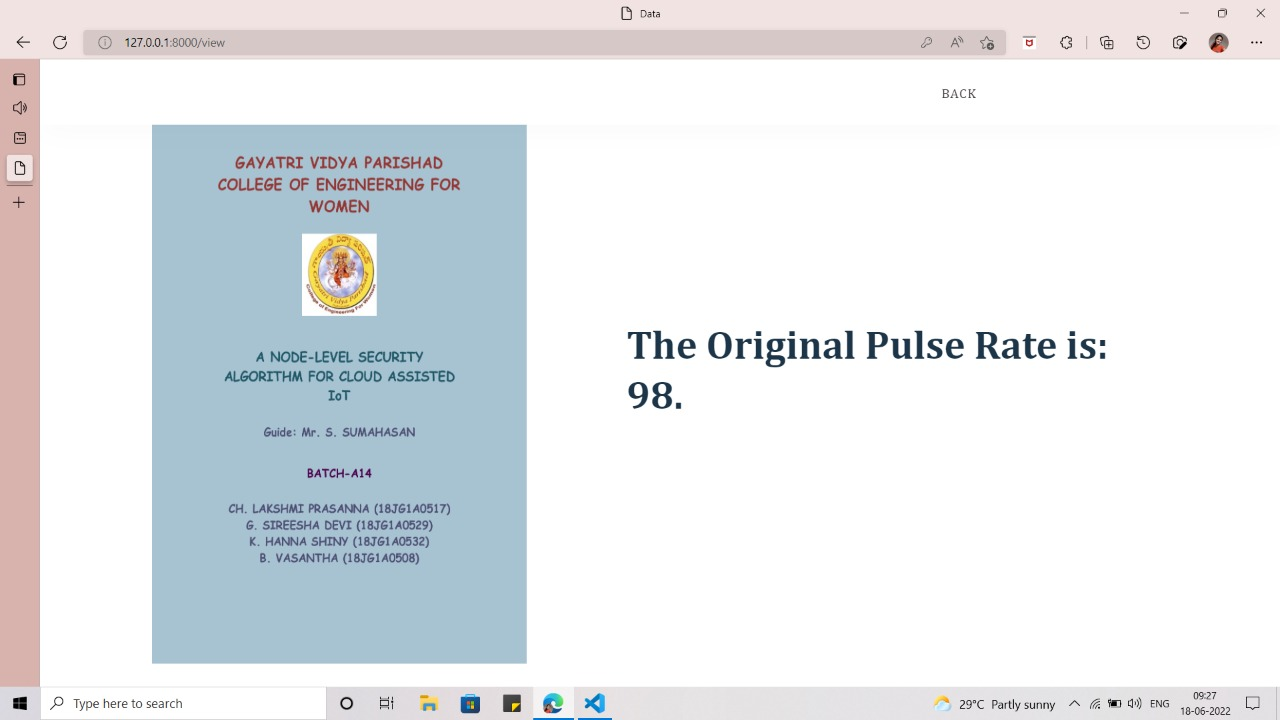


**Screen 7:** Key Verification Page- The user enters the public and private keys to verify the download request.



 **Screen 8:** Contact Us Page- It consists of all the details of all the group members

**Screen 9:** Receiving E-mail- The user receives the public and private keys to download the file.



**Screen 10:** After entering the correct private key sent through the mail in Screen 7, the decryption is done and the original pulse rate is printed.

**6.TESTING AND VALIDATION**

**6.1 INTRODUCTION**

**Scope**

The basic goal of testing is to detect software failures so that flaws can be identified and remedied. This is not a simple task. Testing cannot prove that a product works properly under all conditions; it can only prove that it does not work properly under some conditions. The scope of software testing frequently involves the inspection of code as well as the execution of such code in various contexts and conditions, as well as the evaluation of code elements such as: does it do what it is expected to do and do what it needs to do in today's software development culture, a testing organisation may be separate from the development team. There are various roles for testing team members. Information derived from software testing may be used to correct the process by which software is developed.

**Defects and Failures**

Not all software flaws are the result of coding errors. One prominent source of costly defects is requirement gaps, such as unacknowledged requirements, which result in omission errors by the programme designer. Non-functional needs such as testability, scalability, maintainability, usability, performance, and security are major sources of requirements gaps. Software flaws occur as a result of the following procedures. An error (mistake) is made by a programmer, resulting in a defect (fault, bug) in the software source code. If this flaw is exploited, the system will produce incorrect results in certain situations, resulting in a failure. Not all flaws will definitely lead to failures. Defects in dead code, for example, will never result in failures. When the environment changes, a flaw can become a failure. Examples of these changes in environment include the software being run on a new hardware platform, alterations in source data or interacting with different software. A single defect may result in a wide range of failure symptoms.

**Compatibility**

Compatibility with another application, a new operating system, or, increasingly, a web browser version is a common cause of software failure. In the case of backward compatibility, this can occur because the programmer only considered writing their programme for, or testing the software on, "the most recent version of "this-or-that operating system. The unintended repercussions of this reality include that their most recent work may not be totally compatible with prior software/hardware combinations, or it may not be fully compatible with another significant operating system. In any event, these disparities, whatever they were, might have resulted in software failures, as witnessed by a sizable community of computer users. This could be considered a “prevention oriented strategy” that fits well with the latest testing phase suggested by Dave Gelperin and William C.Hetzel, as cited below.

**Input combinations and preconditions**

A fundamental issue with software testing is that testing under all possible combinations of inputs and preconditions is impossible, even with a simple product. This means that the number of errors in a software product might be rather significant, and that defects that occur infrequently are difficult to detect during testing. More importantly, non-functional dimensions of quality (how it is supposed to be versus what it is supposed to do) can be highly subjective; what is valuable to one person may be intolerable to another.

**Static vs Dynamic testing**

Software testing can be approached in a variety of ways. Static testing consists of reviews, walkthroughs, or inspections, whereas dynamic testing consists of actually executing programmed code with a provided set of test cases. The former can be skipped, whereas the latter occurs when programmes are used for the first time - which is typically regarded as the start of the testing stage. This may really begin before the programme is completely finished in order to test a specific area of code. Spreadsheet programmes, for example, are by definition tested "on the fly" during the development process, as the result of some computation or text manipulation is displayed interactively immediately after each formula is entered.

**Software verification and validation**

Software testing is used in association with verification and validation:

➔ **Verification:** Have we built the software right (i.e., does it match the specifications?)? It is process based.

➔ **Validation:** Have we built the right software (i.e., this what the customer wants?)? It is product based.

**Software Quality Assurance (SQA)**

Though contentious, software testing may be seen as an important component of the software quality assurance (SQA) process. Software process specialists and auditors in SQA have a broad view of software and its evolution. They investigate and modify the software engineering process itself in order to reduce the number of flaws that end up in the produced programme: the defect rate. The nature of the programme determines what makes a "acceptable fault rate." For example, an arcade video game designed to imitate flying an aeroplane would probably have a considerably higher tolerance for flaws than mission critical software used to handle the functions of a real-life airline! Although there are close links with SQA, testing departments often exist independently, and there may be no SQA function in some companies. Software testing is a task intended to detect defects in software by contrasting a computer program’s expected results with its actual results for a given set of 71 inputs. By contrast, QA (Quality assurance) is the implementation of policies and procedures intended to prevent defects from occurring in the first place.

**6.2 TYPES OF TESTING**

Testing methods Software testing methods are traditionally divided into black box testing and white box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

➢ **Black Box Testing:** Black Box Testing is testing without knowledge of the internal workings of the item being tested. For example, when black box testing is applied to software engineering, the tester would only know the “legal” inputs and what the expected outputs should be, but not how the program actually arrives at those outputs. Synonyms for black-box include: behavioural, functional, opaque-box and closed box.

➢ **Unit Testing:** Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

➢ **White Box Testing:** In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

a) Guarantee that all independent paths have been executed.

b) Execute all logical decisions on their true and false sides.

c) Execute all loops at their boundaries and within their operational bounds.

d) Execute internal data structures to ensure their validity.

**6.3 TEST CASES**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **TEST**  **ID** | **TEST CASE** | **EXPECTED RESULT** | **OBSERVED RESULTS** | **POSITIVE/**  **NEGATIVE** | **PASS/FAIL** |
| 1. | Open website | Website Opened | Website Opened | P | P |
| 2. | Registered credentials are entered to login | Validation done successfully | Validation done successfully | P | P |
| 3. | Registered credentials are entered to login | Login Fails | Validation is not successful | N | F |
| 4. | No Wi-Fi connection for Downloading the Patients records | No Results are displayed | Connection was unsuccessful | N | F |
| 5. | User clicks in view or download file | The user asked to verify himself | The user asked to verify himself | P | P |
| 6. | User enters keys received through mail | The file is downloaded | The file is downloaded | P | P |
| 7. | User enters the wrong keys | Error Page | The file isn’t downloaded | N | F |
| 8. | User clicks on Contact Us page | Details of the group members are displayed | Details of the group members are displayed | P | P |

**6.4 VALIDATION**

This chapter is introduced with explanation of testing and validation in software development life cycle in general. Design of test cases and different scenarios is given in detail along with screenshots, followed by validations of generated test cases and scenarios.

**7.CONCLUSION**

**7.1 PROJECT CONCLUSION**

The numerous benefits provided by encryption have driven many large multi-level organizations to transfer their data based on it. This paper begins by pointing out major security concerns data owners have when sharing their data on the cloud. Next, the most widely implemented and researched data sharing schemes are briefly discussed revealing points of weakness in each. To address the concerns, this paper proposes a way in which data can be transferred in a secure way without any data breaches. For securing the data we use encryption algorithm EC Elgamal which is based on Elliptic-Curve Diffie Hellman Algorithm (ECDH) which produce a public and a private key to produce an encrypted format of the original text to be transferred. The messages are transferred between two parties who share their public keys with each other. The message is sent using the public key of the sender and it is decrypted by using the private key of the receiver. EC Elgamal is a key agreement protocol that allows two parties, each having an elliptic-curve public–private key pair, to establish a shared secret over an insecure channel. This shared secret may be directly used as a key, or to derive another key. The key, or the derived key, can then be used to encrypt subsequent communications using a symmetric-key cipher. It is based on the Elliptic Curve Diffie–Hellman(ECDH) protocol using elliptic-curve cryptography. Our comprehensive encryption algorithms provide data security and avoid data breaches from any kind of intruder attacks.

**7.2 FUTURE ENHANCEMENT**

The different users getting registered will be verified by using different security procedures before giving access to data. The whole process can be done by more power-aware devices like Node-MCU. The Encryption and Decryption processes can be done by better and efficient algorithms. Data can be more efficiently shown to different persons in different roles.

**8.REFERENCES**

1. V.S. Mahalle , A. K. Shahade, “Enhancing the Data Security in Cloud by Implementing Hybrid (Rsa & Aes) Encryption Algorithm”, IEEE , INPAC,pp 146-149,Oct.2014. https:/[/www.ij](http://www.ijcseonline.org/full_paper_view.php?paper_id=3548)c[seonline.org/full\_paper\_view.php?paper\_id=3548](http://www.ijcseonline.org/full_paper_view.php?paper_id=3548)
2. Abu Marjan, Palash Uddin, “Developing Efficient Solution to Information Hiding through text steganography along with cryptography”,IEEE,IFOST,pages,14-17,October,2014. <https://www.researchgate.net/publication/345209022_Security_of_Fog_> Storage\_by\_using\_Hybrid\_Cryptography
3. P.Bhendwade and R.T.Patil, “Steganographic Secure Data Communication”,IEEE,International Conference on Communication and Signal Processing,pages,953-956,April,2014

<https://www.ijeat.org/wp>[content/uploads/Souvenir\_Volume-8%20Issue-4\_Apri l\_2019.pdf](https://www.ijeat.org/wp-content/uploads/Souvenir_Volume-8%20Issue-4_April_2019.pdf)

1. S. Hesham and Klaus Hofmann , “High Throughput Architecture for the Advanced Encryption Standard Algorithm” IEEE,International Symposium on Design and Diagnostics of Electronic Circuits & Systems, pages 167- 170,April 2014.

<https://www.ijcaonline.org/archives/volume175/number18/31552>[-2020920693](https://www.ijcaonline.org/archives/volume175/number18/31552-2020920693)

1. Sandholm, T., Magnusson, B., Johnsson, B.A.: Anon-demand WebRTC and IoT device tunneling service for hospitals. In: International Conference on Future InternetofThings and Cloud, Barcelona, pp. 53–60(2014)

https://doi.org/10.1109/FiCloud.2014.19

1. Agrawal, Sarita, and Manik Lal Das. "Internet ofThings—A paradigm shift of future

Internet Applications." In Engineering (NUiCONE), 2011 Nirma University

International Conference on, pp.1-7.IEEE, 2011 <https://doi.org/10.1109/NUiConE.2011.6153246>

1. Wang, X., Wang, J.T., Zhang, X., Song, J.: A multiple communication standards compatible IoT system for medical usage. In: IEEE Faible Tension Faible Consommation(FTFC), Paris, pp. 1–4 (2013) <https://doi.org/10.1109/FTFC.2013.6577775>
2. S. M. Babu, A. J. Lakshmi and B. T. Rao, "A study on cloud based Internet of Things: CloudIoT," 2015 Global Conference on Communication Technologies (GCCT), 2015, pp. 60-65, doi: 10.1109/GCCT.2015.7342624.
3. H. Deng, Z. Qin, L. Sha and H. Yin, "A Flexible Privacy-Preserving Data Sharing Scheme in Cloud-Assisted IoT," in IEEE Internet of Things Journal, vol. 7, no. 12, pp. 11601-11611, Dec. 2020, doi: 10.1109/JIOT.2020.2999350.
4. A. Alsaidi and F. Kausar, "Security Attacks and Countermeasures on Cloud Assisted IoT Applications," 2018 IEEE International Conference on Smart Cloud (SmartCloud), 2018, pp. 213-217, doi: 10.1109/SmartCloud.2018.00043.
5. A. Sajid, H. Abbas and K. Saleem, "Cloud-Assisted IoT-Based SCADA Systems Security: A Review of the State of the Art and Future Challenges," in IEEE Access, vol. 4, pp. 1375-1384, 2016, doi: 10.1109/ACCESS.2016.2549047.
6. L. Yang and F. Li, "Cloud-Assisted Privacy-Preserving Classification for IoT Applications," 2018 IEEE Conference on Communications and Network Security (CNS), 2018, pp. 1-9, doi: 10.1109/CNS.2018.8433157.
7. S. Rajkumar, M. Srikanth and N. Ramasubramanian, "Health monitoring system using Raspberry PI," 2017 International Conference on Big Data, IoT and Data Science (BID), 2017, pp. 116-119, doi: 10.1109/BID.2017.8336583.
8. N. N. Nagamma, M. V. Lakshmaiah and T. Narmada, "Raspberry Pi based biometric authentication vehicle door locking system," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), 2017, pp. 2348-2351, doi: 10.1109/ICPCSI.2017.8392138.
9. R. S. Kumar, P. K. Stanley and A. S. Gandhi, "Raspberry Pi based vehicle collision avoidance system," 2017 International Conference on Innovations in Electrical, Electronics, Instrumentation and Media Technology (ICEEIMT), 2017, pp. 211-215, doi: 10.1109/ICIEEIMT.2017.8116838.
10. S. N. Ishak, N. N. N. A. Malik, N. M. A. Latiff, N. E. Ghazali and M. A. Baharudin, "Smart home garden irrigation system using Raspberry Pi," 2017 IEEE 13th Malaysia International Conference on Communications (MICC), 2017, pp. 101-106, doi: 10.1109/MICC.2017.8311741.
11. S. Dhar, A. Gandhi, G. Ahuja and G. Anusooya, "Raspberry Pi based real time tracking system," 2017 International Conference on Inventive Computing and Informatics (ICICI), 2017, pp. 442-451, doi: 10.1109/ICICI.2017.8365390.
12. A. Mishra, D. Patil, N. Karkhanis, V. Gaikar and K. Wani, "Real time emotion detection from speech using Raspberry Pi 3," 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2017, pp. 2300-2303, doi: 10.1109/WiSPNET.2017.8300170.
13. J. Marot and S. Bourennane, "Raspberry Pi for image processing education," 2017 25th European Signal Processing Conference (EUSIPCO), 2017, pp. 2364-2366, doi: 10.23919/EUSIPCO.2017.8081633.
14. A. Pawar and V. M. Umale, "Internet of Things Based Home Security Using Raspberry Pi," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), 2018, pp. 1-6, doi: 10.1109/ICCUBEA.2018.8697345.
15. S. Sruthy and S. N. George, "WiFi enabled home security surveillance system using Raspberry Pi and IoT module," 2017 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES), 2017, pp. 1-6, doi: 10.1109/SPICES.2017.8091320.
16. T. Zitta, M. Neruda and L. Vojtech, "The security of RFID readers with IDS/IPS solution using Raspberry Pi," 2017 18th International Carpathian Control Conference (ICCC), 2017, pp. 316-320, doi: 10.1109/CarpathianCC.2017.7970418.
17. S. Rajkumar, M. Srikanth and N. Ramasubramanian, "Health monitoring system using Raspberry PI," 2017 International Conference on Big Data, IoT and Data Science (BID), 2017, pp. 116-119, doi: 10.1109/BID.2017.8336583.
18. A. Gharib and M. Ibnkahla, "Security Aware Cluster Head Selection with Coverage and Energy Optimization in WSNs for IoT," ICC 2021 - IEEE International Conference on Communications, 2021, pp. 1-6, doi: 10.1109/ICC42927.2021.9500741.
19. M. Younis, M. Youssef and K. Arisha, "Energy-aware routing in cluster-based sensor networks," Proceedings. 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002, pp. 129-136, doi: 10.1109/MASCOT.2002.1167069.