

**M.S Ramaiah Institute of Technology****Bengaluru-560054****Department of Information Science and Engineering****CERTIFICATE**

Certified that the project work entitled ***PERSON AUTHENTICATION SYSTEM USING BRAIN WAVES AS BIOMETRIC*** was carried out by **Ms. Dharini R**, bearing USN **1MS06IS019**, **Mr. Prakash S Yaji**, bearing USN **1MS06IS046**, **Mr. Rakshath Kashyap M. H.** bearing USN **1MS06IS055** and **Mr. Vinay .K**, bearing USN **1MS06IS111**, bona fide students of **M.S Ramaiah Institute of Technology** in partial fulfilment for the award of Bachelor of Engineering in **Information Science and Engineering** of the Visveswaraiah Technological University, Belgaum during the year 2009-10. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Name &amp; Signature

of the Guide

Name &amp; Signature

of the HOD

Signature of the

Principal

External Viva

Name of the examiners

Signature with date

1.

2.

## ACKNOWLEDGEMENT

The satisfaction of completing our project will not be complete until we thank all the people who have helped us through this.

We would like to thank The Visweswaraiah Technological University for providing us the opportunity to work on a project of our choice.

We would like to thank our director Mr. M.R. Seetharam and our principal Dr. Rajanikanth for obliging us to give such a huge financial grant, without which we would not have been able to complete this project.

We would especially like to convey our enduring gratitude to our HOD, Dr Aswatha Kumar for providing us the confidence and all the necessary support in undertaking this project.

We would like to extend our thanks to our project guide Dr Lingaraju G M for his persistent guidance and support throughout the course of the project. His encouragement and help has been invaluable.

We express our gratitude to the purchase department for helping us in acquiring the EMOTIV EPOC headset for our project.

A large amount of material has been obtained from the Internet and numerous other literary sources. We thank all those unseen faces, whose contribution to our venture has been invaluable.

We thank our parents on whose blessings we live and thrive. It is their prayers that have helped us translate our efforts into fruitful achievements.

**Dharini R [1MS06IS019]**

**Prakash S Yaji[1MS06IS046]**

**Rakshath Kashyap[1MS06IS055]**

**Vinay K [1MS06IS111]**

## ABSTRACT

A Brain Computer Interface (BCI) is a computer device that helps a person operate a computer by using his brain waves. Brain waves are patterns of electrical brain activity. Brain cells can communicate with each other by electricity and chemicals called neurotransmitters. The electrical activity of the brain is measured by a test called the electroencephalogram (EEG). For this test, tiny metal discs called electrodes are painlessly stuck to the scalp. These electrodes can pick up the electrical impulses of the brain, which pass through the skull. Brain waves measured by EEG do not represent a particular thought or movement, but rather represent a summary of brain electrical activity at a recording point on the scalp. Brain waves can be altered by thoughts of planned action (such as thinking of moving a hand), without actually moving the hand itself. This ability to think of a movement can be used to actually move a computer cursor or a prosthetic arm.

In this project, we investigate the use of brain activity for person authentication. It has been shown in previous studies that the brain-wave pattern of every individual is unique and that the electroencephalogram (EEG) can be used for biometric authentication. Person authentication aims to accept or reject a person claiming an identity. We perform EEG recording, feature extraction and matching of the feature vector with the stored feature vector all in real time. We also show that there are some mental tasks that are more appropriate for person authentication than others.

<b>Contents</b>	<b>Page No.</b>
1. Introduction	8
1.1 Problem statement	8
1.2 What is a Brain Computer Interface?	8
1.2.1 Invasive BCIs	9
1.2.2 Partially-invasive BCIs	9
1.2.3 Non-invasive BCIs	9
1.2.4 The Emotiv Education Edition SDK	10
1.3 Background on EEG	12
1.4 Present Solution	14
1.5 Proposed Solution	15
1.5.1 Experimental Protocol	17
2. Literature Survey	19
2.1 Review on the previous work in EEG based Authentication/Identification	
2.1.1 EEG-based Person Authentication as proposed by Marcel	19
2.1.2 EEG-based Person Identification as proposed by Paranjap	19
2.1.3 EEG-based Person Identification as proposed by Poulos	20
2.1.4 EEG-based Person Authentication as proposed by R Palaniappan	20
2.2 EEG Features	
2.2.1 Power Spectral Density	22
2.2.2 Channel Spectral Power and	23

---

Inter Hemispherical Power Difference	
2.2.3 Auto- regressive coefficients	23
3. Software requirement specification	24
3.1 Purpose	24
3.2 Definitions and Abbreviations	
3.1 Definitions	24
3.2 Abbreviations	24
3.3 Specific Requirements	
3.3.1 Functional requirements	25
3.3.2 Non-functional Requirements	25
3.3.3 Input/output Requirements	26
3.3.4 User interface Requirements	26
3.3.5 Software Requirements	27
3.3.6 Hardware Requirements	27
3.3.7 General Constraints	27
3.3.8 Assumptions and Dependencies	27
4. System design	28
4.1 Identifying the Behaviour of System	28
4.1.1Class Diagram	29
4.1.2 State Chart Diagram	30
4.1.3 Use Case Diagram	31
4.1.4 Activity Diagram	32
5. Implementation	33
5.1Programming Languages and Technologies used	

5.1.1 C#	34
5.1.2 MATLAB	35
5.1.3 MCR	36
5.1.4 Microsoft XNA	36
5.1.5 P/Invoke (Platform Invocation Services)	37
5.2 Libraries and APIs used	38
5.2.1 Libraries and APIs provided by Emotiv	38
5.2.2 Libraries and APIs provided by MathWorks	39
5.3 User Interface Diagram	40
5.4 Detailed Explanation of Implementation	41
6. Testing	66
6.1 Unit Testing	66
6.2 Integration Testing	70
7. Summary, Conclusion and Scope for Future Work	72
8. Bibliography	73

## **Chapter- 1**

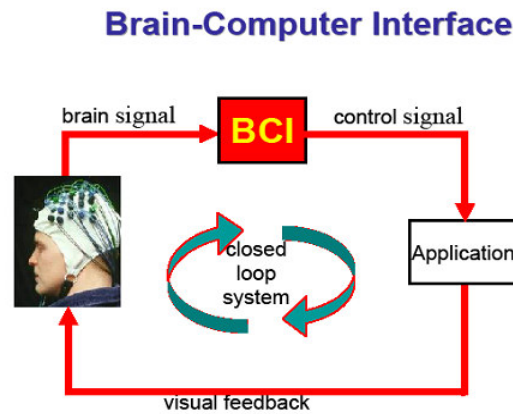
# **INTRODUCTION**

## **1.1 Problem statement**

Existing biometrics like facial patterns, fingerprints and eye irises are prone to forgery. It is required to find alternative biometric traits that can distinguish between individuals. An identity authentication system has to deal with two kinds of events: either the person claiming a given identity is the one who he claims to be (in which case, he is called a *client*), or he is not (in which case, he is called an *impostor*). Moreover, the system may generally take two decisions: either *accept* the *client* or *reject* him and decide he is an *impostor*. The main aim is to keep the False Acceptance Error (FAE) and the False Rejection Error (FRE) close to zero.

## **1.2 What is a Brain Computer Interface?**

A brain–computer interface (BCI), sometimes called a direct neural interface or a brain–machine interface, is a direct communication pathway between a brain and an external device.



**Fig 1.1: Representation of a BCI**

### **1.2.1 Invasive BCIs**

Invasive BCI research has targeted repairing damaged sight and providing new functionality to persons with paralysis. Invasive BCIs are implanted directly into the grey matter of the brain during neurosurgery. As they rest in the grey matter, invasive devices produce the highest quality signals of BCI devices but are prone to scar-tissue build-up, causing the signal to become weaker or even lost as the body reacts to a foreign object in the brain.

### **1.2.2 Partially-invasive BCIs**

Partially invasive BCI devices are implanted inside the skull but rest outside the brain rather than within the grey matter. They produce better resolution signals than non-invasive BCIs where the bone tissue of the cranium deflects and deforms signals and have a lower risk of forming scar-tissue in the brain than fully-invasive BCIs.



### 1.2.3 Non-invasive BCIs

As well as invasive experiments, there have also been experiments in humans using non-invasive neuroimaging technologies as interfaces. Signals recorded in this way have been used to power muscle implants and restore partial movement in an experimental volunteer. Although they are easy to wear, non-invasive implants produce poor signal resolution because the skull dampens signals, dispersing and blurring the electromagnetic waves created by the neurons. Although the waves can still be detected it is more difficult to determine the area of the brain that created them or the actions of individual neurons.

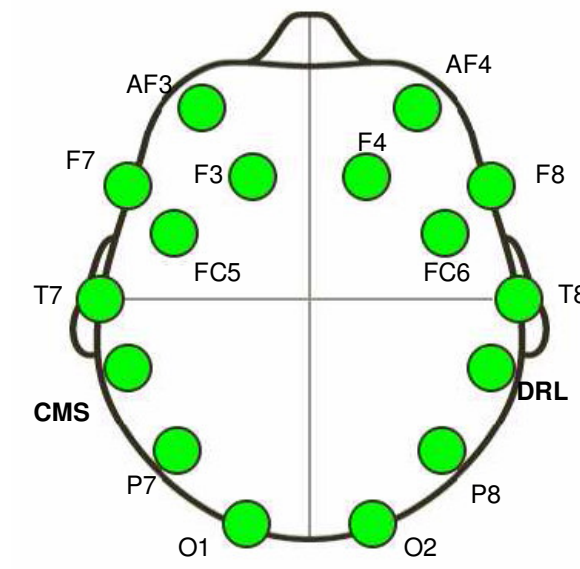
The EEG headset, to extract a person's brain waves to authenticate him, is EPOC headset manufactured by Emotiv Inc. More details about this headset are given below.

### 1.2.4 The Emotiv Education Edition SDK

The Education Edition SDK by Emotiv Systems **includes a research headset**: a 14 channel (plus CMS/DRL references, P3/P4 locations) high resolution, neuro-signal acquisition and processing wireless neuroheadset. Channel names based on the International 10-20 locations are: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4.



**Fig 3.1: Subject wearing the Emotiv Epoc Headset**



**Fig 3.2: Illustration of location of electrodes on the scalp**

The Education Edition SDK also includes a proprietary software toolkit that exposes the APIs and detection libraries. The SDK provides an effective development environment that integrates well with new and existing frameworks. The detection suites that are incorporated into the Education Edition SDK are:

**Affectiv™ Suite**

The Affectiv suite monitors player emotional states in real-time. It provides an extra dimension in game interaction by allowing the game to respond to a player's emotions. Characters can transform in response to the player's feeling. Music, scene lighting and effects can be tailored to heighten the experience for the player in real-time. The Affectiv suite can be used to monitor player state of mind and allow developers to adjust difficulty to suit each situation.

**Cognitiv™ Suite**

The Cognitiv suite reads and interprets a player's conscious thoughts and intent. Gamers can manipulate virtual objects using only the power of their thought! For the first time, the fantasy of magic and supernatural power can be experienced.

**Expressiv™ Suite**

The Expressiv suite uses the signals measured by the neuroheadset to interpret player facial expressions in real-time. It provides a natural enhancement to game interaction by allowing game characters to come to life. When a player smiles, their avatar can mimic the expression even before they are aware of their own feelings. Artificial intelligence can now respond to players naturally, in ways only humans have been able to until now.

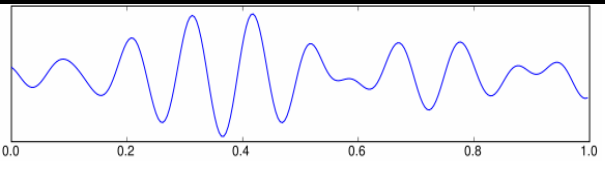
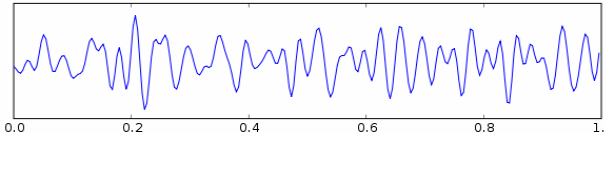
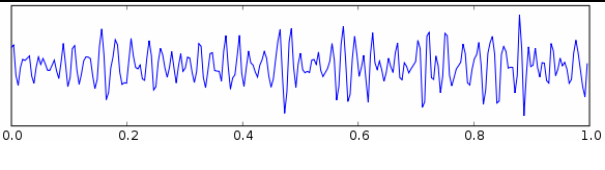
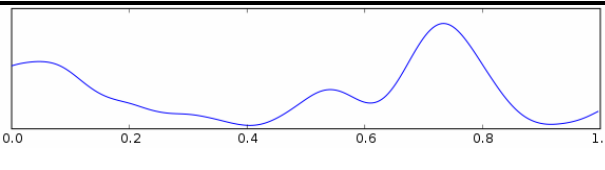
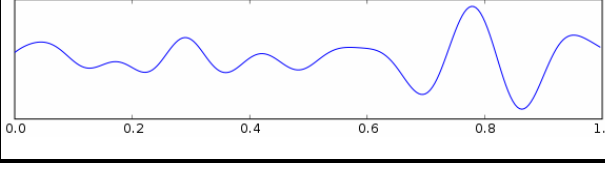
Electroencephalography (EEG) is the most studied potential non-invasive interface, mainly due to its fine temporal resolution, ease of use, portability and low set-up cost. But as well as the technology's susceptibility to noise, another substantial barrier to using EEG as a brain-computer interface is the extensive training required before users can work the technology.

**1.3 Background on EEG**

EEG is the recording of electrical activity along the scalp generated by changes in the electrical charge of thousands of neurons of the cerebral cortex. A synapse is formed by a very small cleft which separates neurons and the specialized membrane between two neurons. The neurons have a resting potential. The resting potential is resulted from the

electrical potential difference between the interior of the cell and the extracellular space. The synapse is a polarized structure which allows the information to travel from one neuron to another. The resting potential fluctuates as a result of impulses or nerve stimulus arriving from other neurons at synapses. The impulses create the local postsynaptic potentials causing electrical current to flow along the membrane of the cell body. Once the changes reduce the membrane potential to a critical level, an action potential is generated. The action potential, which propagates along the neuronal membrane, is a transient disruption of the membrane potential. The synaptic activity gives rise to constantly changing ionic currents across the cell membrane and the electrical potential field. Due to the briefness of the action potentials, the fluctuations in the surface EEG are produced mainly by the temporal and spatial summation of electrical currents caused by the relatively slow postsynaptic potentials. The electrical signals are manifestations of these complex systematic physical and chemical processes which create these potential differences.

EEG has five major wave patterns, which are Delta rhythm, Theta rhythm, Alpha rhythm, Beta rhythm and Gamma rhythm.

Alpha (8-12Hz)	Emitted when we are in a state of physical and mental relaxation, although aware of what is happening around us	
Beta (12-30Hz)	Emitted when we are consciously alert, or we feel agitated, tense, afraid	
Gamma(25-100Hz)	These waves may be implicated in creating the unity of conscious perception	
Delta (1-4Hz)	When there is unconsciousness, deep sleep or catalepsy.	
Theta (4-7Hz)	It is a state of somnolence with reduced consciousness	

**Table 1.1: Summary of EEG frequency bands**

## 1.4 Present Solution

Studies have shown that Brain wave pattern for each individual is unique and thus can be used for biometric purpose. EEG-based biometry is an emerging research topic. Very little work has been done in this area, focusing more on person identification than person authentication. Person authentication aims to accept or reject a person claiming an identity, i.e. comparing a biometric data to one template, while the goal of person identification is to match the biometric data against all the records in a database.

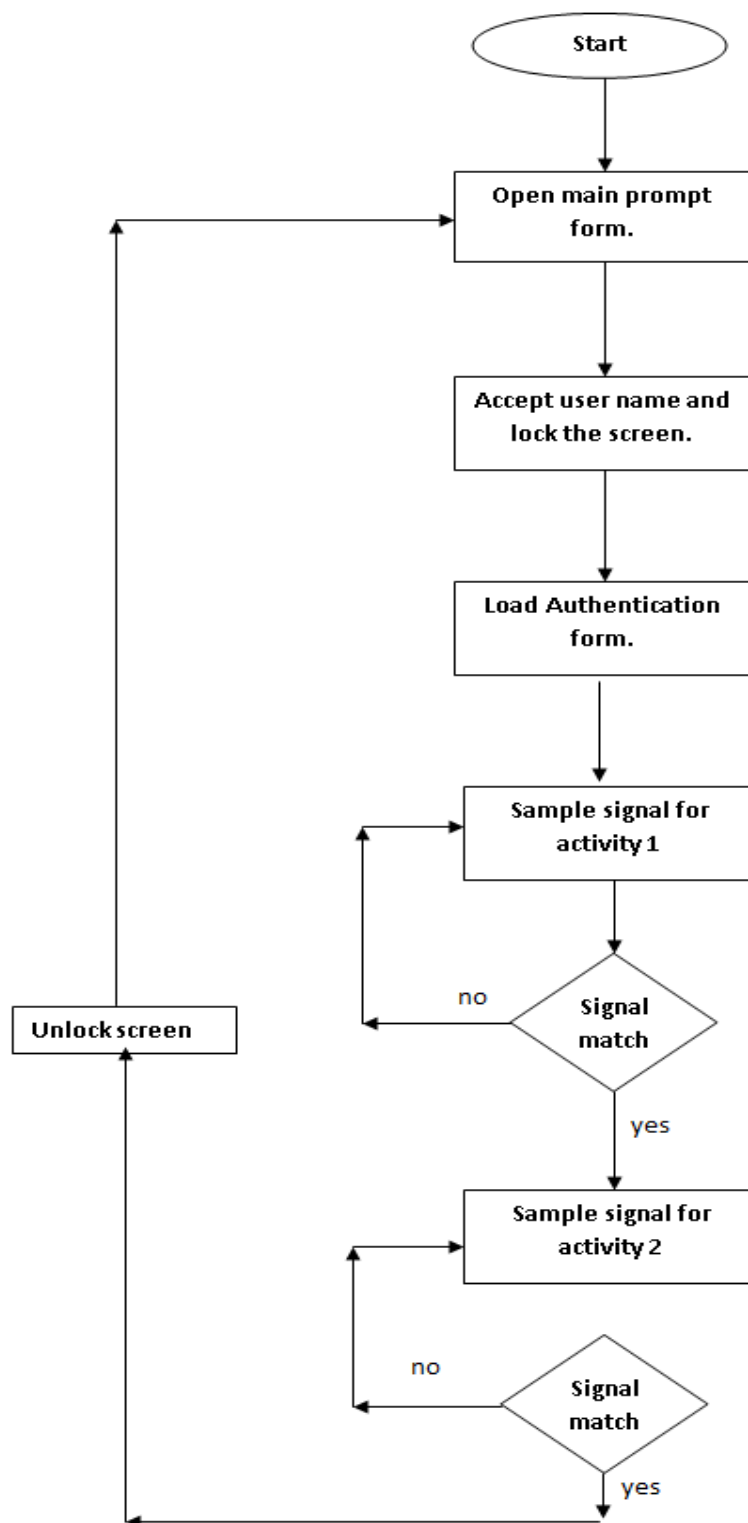
A full-fledged EEG-based authentication system hasn't been developed so far.

## 1.5 Proposed Solution

The motive of our project is to investigate the use of EEG as a new modality for Person Authentication. Our goal is to develop a screen lock application that will lock and unlock the computer screen at the users will. The brain waves of the person, recorded in real time are used as password to unlock the screen. The authentication system consists of two main components: EEG feature extraction and classification.

We've identified Power Spectral Density to be a more reliable feature in comparison to many other features like entropy and auto regression. Hence Power Spectral Density is used as the key feature in our work. After obtaining the features, Principal Component Analysis is performed to obtain relevant features from the high dimensional data. The obtained feature vector is then compared against a previously stored feature vector for that person. The match is considered good if the result of the comparison is greater than the threshold value which has been set after repeated trials keeping in mind the need to satisfy low FAE and FRE.

The User Interface for our project is created in Visual C# (.NET platform) and the Feature Extraction and Matching is performed in Matlab. APIs from the Emotiv Development Kit referred to later in this report have also been used. Our application may be briefly represented using the following flow chart.



### 1.5.1 Experimental Protocol

#### Database

EEG signals are recorded with the Emotiv EPOC headset (referred to later) which uses 14 integrated electrodes located at standard positions of the International 10-20 system. The sampling rate is 128Hz. No artefact rejection or correction is employed.

This dataset contains data from normal subjects for the following tasks:

- **Meditation activity**

The subject is asked to meditate for a fixed period of time while his brain waves are recorded.

- **Math activity**

The subjects are given non-trivial multiplication problems, such as 79 times 56 and are asked to solve them without vocalizing or making any other physical movements.

The EEG data is segmented.

#### Feature Extraction

Channel spectral power (referred to later) for three spectral bands Alpha, Beta and Gamma is computed.  $14 \times 3 = 42$  features are extracted for each segment of the data.

#### Dimensionality Reduction

Method = Principal Component Analysis

Principal component analysis (PCA) involves a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.



We have considered only those components that contribute to 85% (this value has been chosen after repeated trials) of the total variance for signal matching.

**Signal Matching**

Procedure: The proximity of the two principal components (one from the trained feature set and one from the new feature set) is calculated. A proximity value of .78 and above is considered a good match.

## **Chapter- 2**

### **LITERATURE SURVEY**

#### **2. 1 Review on the previous work in EEG-based Authentication/Identification**

##### **2.1.1 EEG-based Person Authentication as proposed by Marcel**

EEG based person authentication was first proposed by Marcel. He proposed the use of Power Spectral Density as the feature, and a statistical framework based on Gaussian Mixture Models (GMM) and Maximum A Posteriori Model (MAP) Adaptation.

In this framework, one first needs a probabilistic model of anybody's biometric data, often called a world model and trained on a large collection of recordings of several people. From this generic model, a more specific, client-dependent model, is then derived using adaptation techniques, built on data from a particular client. One can then estimate the ratio of the likelihood of the data corresponding to some access with respect to the model of the claimed client identity, with the likelihood of the same data with respect to the world model. The access is accepted or rejected if the likelihood ratio is higher or lower than a given threshold, selected in order to optimize either a low rejection rate, a low acceptance rate, or a combination of both [1].

##### **2.1.2 EEG-based Person Identification as proposed by Paranjap**

He employed autoregressive (AR) modeling of EEG from a single channel and applied discriminant functions to the AR coefficients for identification. Their work focused on normal subjects in the mental states of eyes open and eyes closed [2].

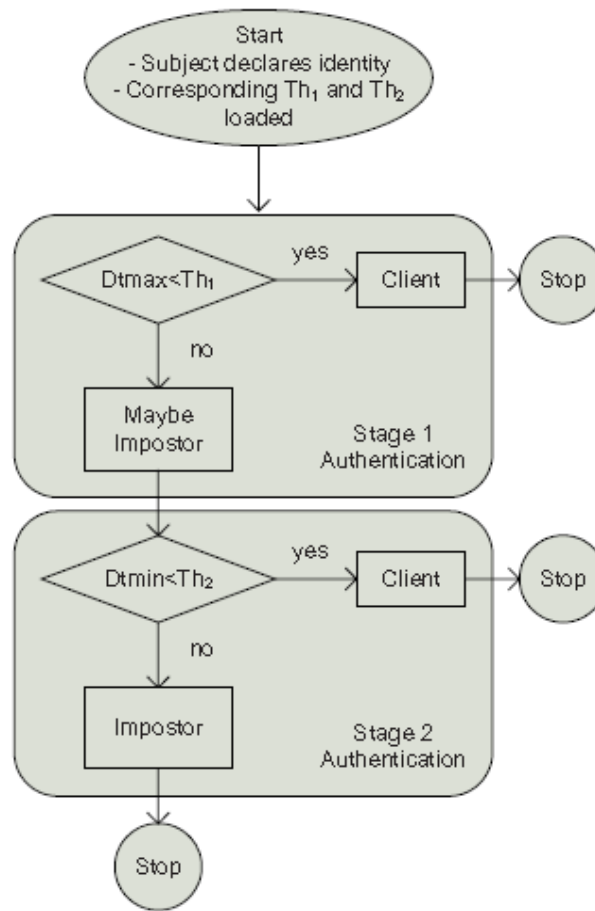
### **2.1.3 EEG-based Person Identification as proposed by Poulos**

Poulos introduced a method based on spectral information extracted from the EEG non-parametrically via the FFT and employed a neural network to classify unknown EEGs as belonging to one of a finite number of individuals. His work focused on healthy individuals and aimed to establish a one to one correspondence between genetic information of the individual [3].

### **2.1.4 EEG-based Person Authentication as proposed by R Palaniappan**

R Palaniappan proposed a novel two stage authentication procedure which gives improved accuracy as compared to existing authentication methods that adjusts a single threshold to balance false accept error (FAE) and false reject error (FRE). He used a 10 sec EEG data from five imagined activities for five different subjects: Baseline Activity, Math Activity, Letter Composing Activity, Trace Activity, and Cube Rotation Activity. He then divided the EEG signal into 20 segments. He investigated features based on the spectral power, autoregressive coefficients and entropy for each of the segments of the signal together with a fuzzy Neural Network for the classification. He divided the obtained feature vectors from the EEG data of the 5 subjects into training patterns, validation patterns and test patterns. The Manhattan (city block) distances  $D$ , were computed between validation patterns and training patterns. Next,  $D_{max}$  and  $D_{min}$ , the maximum and minimum of these validation-training distances for each validation pattern were computed. Thresholds,  $Th1$  and  $Th2$  were obtained using  $Th1 = \min(D_{min})$ ,  $Th2 = \max(D_{max})$ .

The  $D_t$ , Manhattan distances of the test patterns from the training patterns were computed. Next, maximum and minimum of these  $D_t$ ,  $D_{tmax}$  and  $D_{tmin}$  were computed. The Authentication Procedure used by Dr. Ramaswamy Palaniappan is shown in the figure below [4].



**Fig 2.1: Authentication Procedure used by Dr. Ramaswamy Paniappan**

## 2.2 EEG Features

In this section we summarize common EEG features that may be used in EEG-based biometrics.

### 2.2.1 Power Spectral Density

Power spectral density is a positive real function of a frequency variable associated with a stationary stochastic process. It is the measure of the power strength at each frequency. In other words, it shows at which frequencies variations are strong and at which frequencies variations are weak. The unit of PSD is energy per frequency (width). Computation of PSD can be done directly by the method of FFT or computing auto-correlation function and then transforming it.

The Discrete Fourier transform is given by [5]

$$X(f) = \sum_{i=1}^N x(i)w_N(i-1),$$

where,

$w_N = \exp(2\pi i/N)$ , is the Nth root of unity. Power spectral density is given by

$$S_X(f) = \frac{1}{N} \sum_{i=1}^N |X(f)|^2.$$

### 2.2.2 Channel Spectral Powers and Inter-hemispheric Channel Spectral Power Differences

The channel spectral power is the measure of the total power between two frequencies and is given by [5]:

$$P_{f_1, f_2} = \int_{f_1}^{f_2} S_X(f) df,$$

where,  $(f_1, f_2)$  is the frequency band and  $S_X(f)$  is the power spectral density.

The inter-hemispheric channel spectral power differences in each spectral band is given by [2]

$$P_{diff} = \frac{P_1 - P_2}{P_1 + P_2},$$

where  $P_1$  and  $P_2$  are the powers in different channels in the same spectral

band but in the opposite hemispheres.

### 2.2.3 Autoregressive coefficients

The autoregressive (AR) models are used in time series analysis to describe stationary time\_series . These models represent time series that are generated by passing the white noise\_through a recursive linear filter\_.

The mathematical formulation of AR model is given by [2]:

$$x(n) = - \sum_{k=1}^p a_k x(n-k) + e(n),$$

where,  $p$  is the model order,  $x(n)$  is the signal at the sampled point  $n$ ,  $a_k$  are the real valued AR coefficients and  $e(n)$  represents the error term independent of past samples.

## **Chapter- 3**

### **SOFTWARE REQUIREMENT SPECIFICATION**

The SRS describes the resources and methodology used in the development of our system. This SRS provides details regarding the functional and performance related requirements of the system.

#### **3.1 Purpose**

The purpose of this application is to let a user lock his computer screen while he is not at his terminal and unlock the same by recording his brain activity (EEG signals) for a certain period of time.

#### **3.2 Definitions and Abbreviations**

##### **3.2.1 Abbreviations**

- FAE – False Accept Error
- FRE – False Reject Error
- PPM – Percentage Probabilistic Match
- GUI – Graphical User Interface
- EEG – ElectroEncephaloGram
- EDF – European Data Format
- CSV – Comma Separated Values

##### **3.2.2 Definitions**

- **EPOC** – An EEG headset used to record brain activity of the user, developed by Emotiv<sup>TM</sup> Inc.
- **Application Database** – A flat file database which stores the EEG samples of a user along with an index file.
- **User** – Is the system user who has an account in the application database.

- **Super user / Admin** – A user account with the highest privileges.
- **Recorded Brain activity** – The EEG signals of an individual recorded during one/more thought activities.
- **Stored EEG samples** – The filtered and sampled EEG signals of an individual which are stored as EDF or CSV files and which act as reference for authentication of a person.

### 3.3 Specific Requirements

#### 3.3.1 Functional requirements:

System should perform the following functions:-

- It should be able to correctly identify a client and an impostor keeping the FAE and the FRE very low.
- EEG Recording, Feature Extraction and Classification should all be done in real time.
- EEG training data (in the form of .csv files) for different mental tasks should be stored in an application database, with an index file.
- The application should allow the admin / super user to add and delete user profiles.
- The application should also maintain a backup of the user profiles in a suitable location easily accessible to the super user.

#### 3.3.2 Non-functional Requirements:

- Reliability, Availability and Maintainability (RAM) are 3 important requirements in Real Time Scenarios.



- Security (should be able to resist unauthorized attempts at usage or behaviour modification, while still providing service to legitimate users.)
- Performance (The response time should very less)
- The user interface should be simple and unambiguous.
- Flexibility of the program ( should be able to update it later/add more functionalities)

### **3.3.3 Input/output Requirements:**

#### **Input**

- User Name
- Training EEG Data
- Newly recorded EEG data

#### **Output**

- Authentication result - (client/impostor)

### **3.3.4 User interface Requirements**

Users should be provided with simple and easy to use interfaces which includes an initial form to accept the user name and also a Control panel like interface which allows the user to train the system for a particular mental task (such as rotating a cube left/right, etc). The panel should also display a virtual 3D cube to show an animated representation of the Cognitiv detection output and also to assist the user in visualizing the intended action during the training process.

### **3.3.5 Software Requirements**

- Operating system : Microsoft Windows XP/Windows Vista
- Execution Environment: .Net Framework 3.5 or lower version, MCR
- Software Development Kit: Emotiv Education Edition SDK

### **3.3.6 Hardware Requirements**

- 2.4 GHz Intel Pentium 4 processor (or higher).
- 1GB RAM.
- One or two unused USB 2.0 ports (depending on the number of neuro headsets you wish to use simultaneously).

### **3.3.7 General Constraints**

Since the matching of the brain signals is delegated to MCR the time taken to match the signals is slightly more because of the extra time involved in invoking MCR itself. Since the brain activity (EEG signals) of the user is recorded to authenticate a user, it demands for a high level of concentration from the user.

### **3.3.8 Assumptions and Dependencies**

The application is only meant to be a screen lock application and not a replacement of the login screen of the OS. The application is compatible with Windows XP/ Windows Vista/ Windows 7. It does not support any UNIX / Linux OS. The application also needs a pre-installed .NET framework and pre-installed MCR. In order to record the brain activity of a person, an EEG EPOC headset, a proprietary product of Emotiv<sup>TM</sup> is used

## **Chapter- 4**

### **SYSTEM DESIGN**

#### **4.1 Identifying the Behavior of System**

Any complex system is best understood by creating diagrams or pictures. These diagrams have a better impact on our understanding. A single diagram is not enough to cover all aspects of the system. So UML defines various kinds of diagrams to cover most of the aspects of a system.

There are two broad categories of diagrams which are again divided into sub-categories:

**Structural Diagrams:** - The *structural diagrams* represent the static aspect of the system.

Ex: Class Diagram

Object Diagram

Component Diagram

Deployment Diagram

**Behavioural Diagrams:** - *Behavioural diagrams* basically capture the dynamic aspect of a system.

Ex: State Diagram

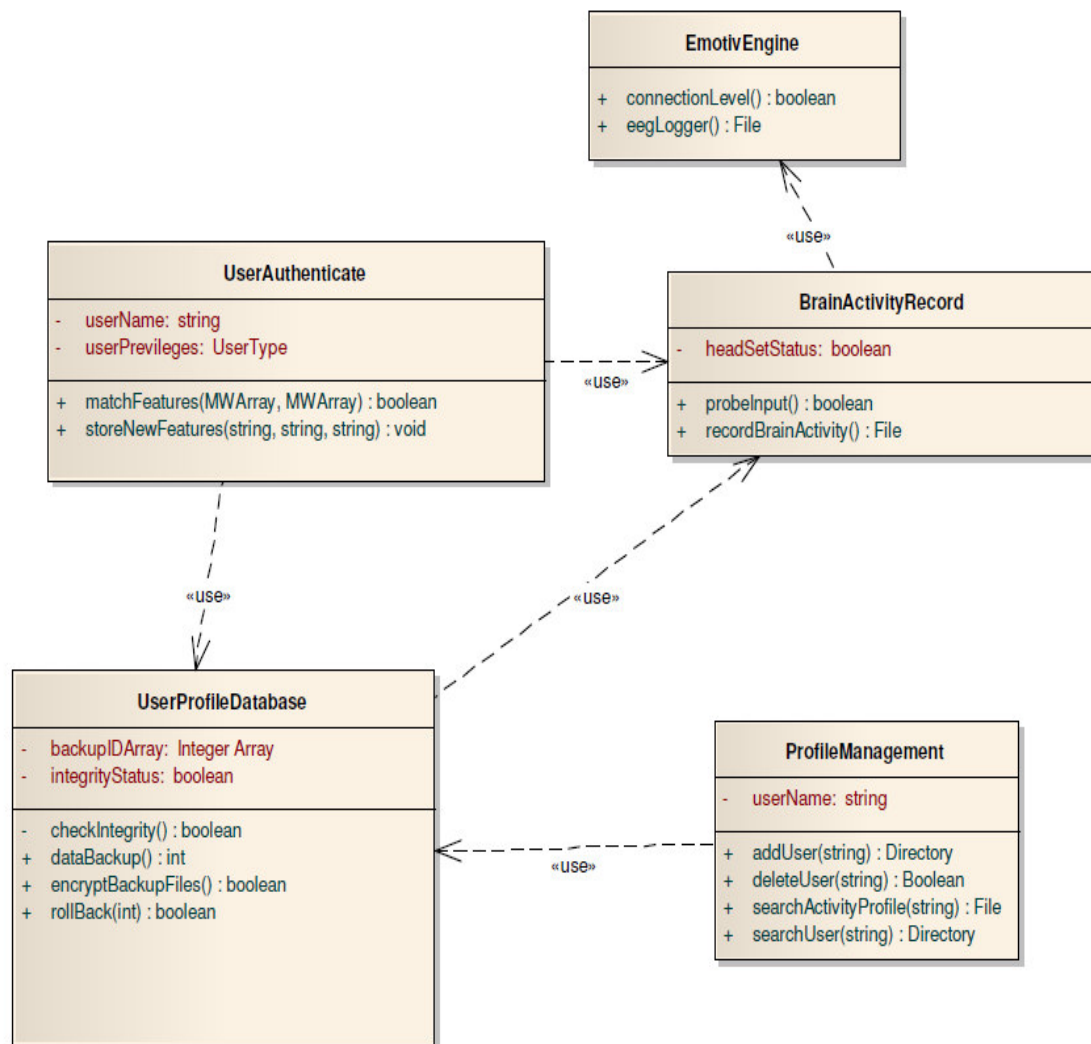
Activity Diagram

Sequence Diagram

Use Case Diagram

### 4.2.1 Class Diagram

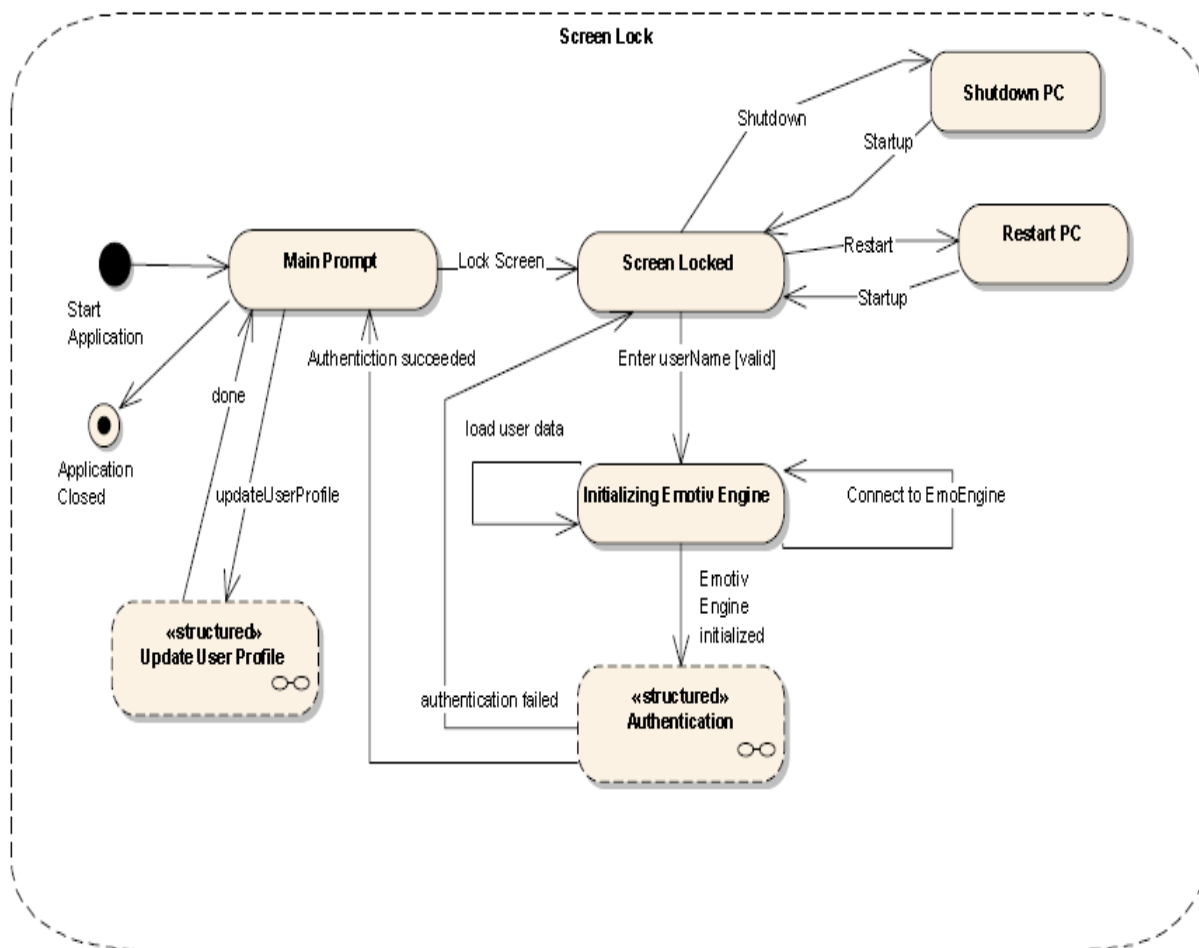
A class diagram is a static representation of the system. It shows the types of classes, and how these classes are linked to each other. Figure 4.2.1 shows the static structure of the system in the form of a class diagram.



### 4.1 Class Diagram of Authentication System

#### 4.2.2 State Chart Diagram

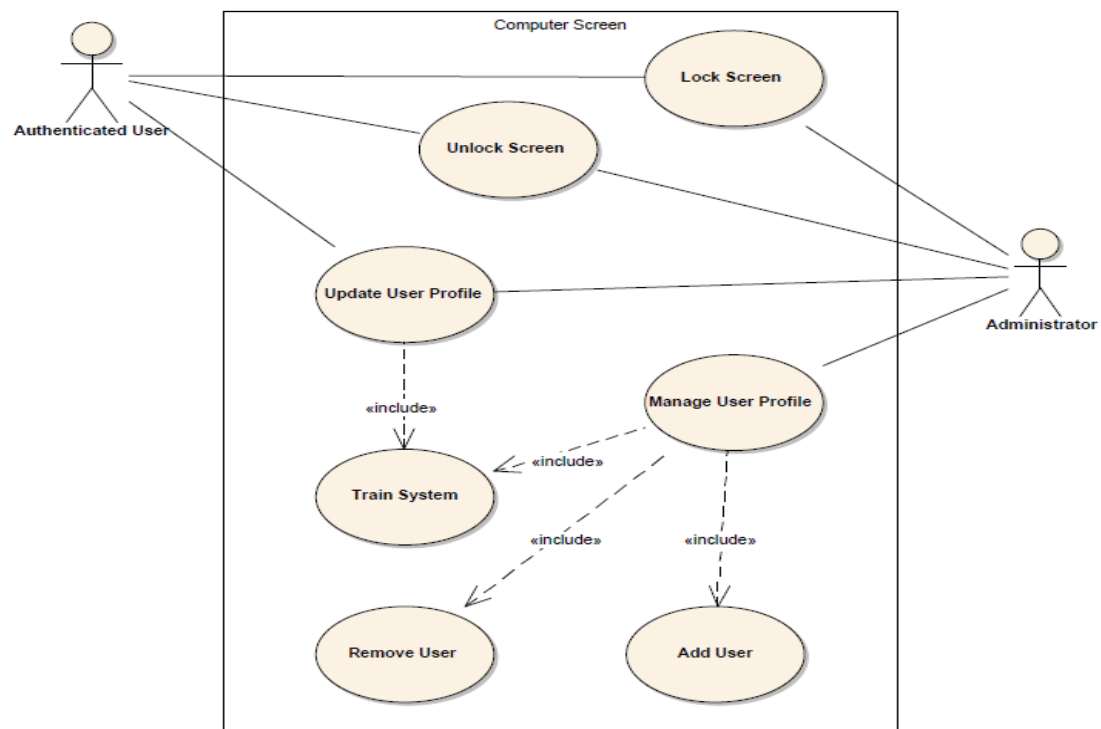
Any real time system is expected to be reacted by some kind of internal/external events. These events are responsible for state change of the system. State chart diagram is used to represent the event driven state change of a system. It basically describes the state change of a class, interface etc. It is used to visualize the reaction of a system by internal/external factors.



**Fig 4.2 State Chart Diagram Of Authentication System**

### 4.2.3 Use Case Diagram

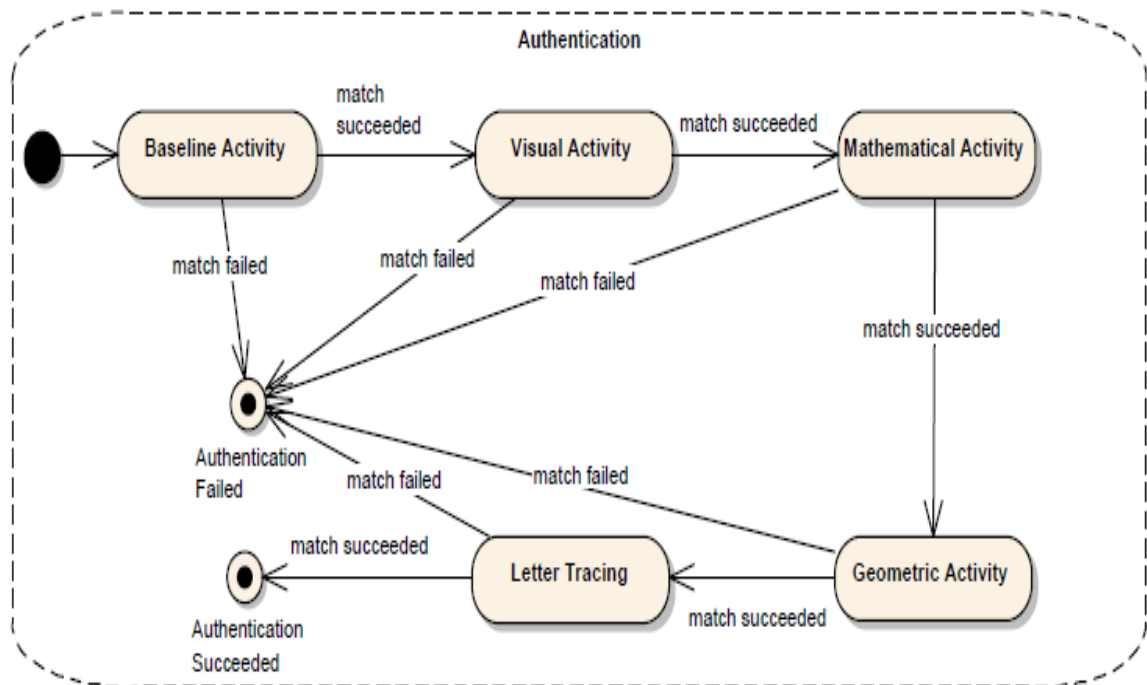
Use case diagrams are a set of use cases, actors and their relationships. They represent the use case view of a system. A use case represents a particular functionality of a system. So use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.



**Fig. 4.3 Use Case Diagram Of Authentication System**

#### 4.2.4 Activity Diagram

Activity diagram describes the flow of control in a system. So it consists of activities and links. The flow can be sequential, concurrent or branched. It is used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.



**Fig. 4.4 Activity Diagram of Authentication System**

## **Chapter- 5**

### **IMPLEMENTATION**

Two versions of the project have been implemented in parallel. Although most of the UI components remain the same for both the versions, they employ different methods to sample a particular subject and authenticate him. The following points highlight the working of these 2 methods:

1. Method 1 of the project involves using the Emotiv APIs' bundled with the Emotiv Education Edition SDK.
  - **Training of the system:** The brain waves of the user are recorded when he performs certain tasks such as mentally visualizing the pushing/rotation of a cube.
  - **Creating user profile:** The sampled brain waves are stored as the user's profile.
  - **Authenticating:** The user is made to perform the same set of actions as in the training session and the newly recorded brain waves are matched with the stored samples.

The entire version of this project is coded in C#.

2. Method 2 of the project involves the following stages:
  - **Training of the system:** The brain waves of the user are recorded when he performs certain mental tasks such as meditation and math activity.
  - **Feature extraction:** The channel spectral power in the alpha, beta and gamma spectral bands is extracted.
  - **Creating user profile:** These features are stored in a separate file as the user's profile.
  - **Authenticating:** The brain waves of the person are recorded in real time for the same set of activities as in the training. Features are extracted from these recorded waves. Feature reduction is performed using Principal Component Analysis and these features are matched with the previously stored features.



The feature extraction and matching part are coded in MATLAB, while the UI part is designed and coded in C#. The MATLAB codes are converted to CLR compliant library (\*.dll file). These files are then referenced in C# using the *using* statement and adding an appropriate reference in the project.

## 5.1 Programming languages and technologies used:

Before delving into the core implementation of the project, a brief introduction about the technologies and the programming languages used in this project is provided below:

### 5.1.1 C#

C# is a multi-paradigm programming language encompassing imperative, functional, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft within the .NET initiative and later approved as a standard by Ecma and ISO (International Organization for Standardization). One of the most important features of the C# programming language is the ability to port the executable system to any type of underlying computer architecture, be it 32-bit or 64-bit systems. This is achieved by the C# compiler which compiles the code to an *Intermediate Language* (IL) code. This IL code is then converted to machine specific codes by CLR (Common Language Runtime) depending on the architecture of the system at hand. Also since C# is an object oriented programming language; it provides the power of objected oriented development for a given software. The version of the C# language for the development of this project was C# 3.0 with the support of .NET framework 3.5. For both the versions of the project, the UI part was done using C#.

The reasons for choosing C# to develop this project are:

- Rich developer Experience
- Automatic memory management through garbage collection
- Ability to seamlessly integrate with MATLAB
- Easier UI component design
- Portability of code to other systems.

Windows Forms were used to design and develop the UI part. Windows Forms is the name given to the graphical Application Programming Interface (API) included as a part of Microsoft's .NET Framework, providing access to the native Microsoft Windows interface elements by wrapping the existing Windows API in managed code. Just like AWT, the equivalent Java API, Windows Forms was an early and easy way to provide GUI components to the .NET Framework.

### **5.1.2 MATLAB**

MATLAB stands for "MATrix LABoratory" and is a numerical computing environment and fourth-generation programming language. Developed by The MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Fortran and C#. MATLAB was created in the late 1970s by Cleve Moler, then chairman of the computer science department at the University of New Mexico. MATLAB, the application, is built around the MATLAB language. The simplest way to execute MATLAB code is to type it in at the prompt, ">>", in the *command window*, one of the elements of the MATLAB desktop. In this way, MATLAB can be used as an interactive mathematical shell. Sequences of commands can be saved in a text file, typically using the MATLAB Editor, as a script or encapsulated into a function, extending the commands available. Like Python and Perl languages, MATLAB has its own scripting language. MATLAB is a weakly dynamically typed programming language. It is a weakly typed language because types are implicitly converted. It is a dynamically typed language because variables can be assigned without declaring their type, except if they are to be treated as symbolic objects, and that their type can change. In the second version of the project, the feature extraction and matching part was developed using MATLAB. The reasons for using MATLAB for the development of the aforementioned modules are:

- Ability to handle and manipulate voluminous amount of data.
- Inbuilt support for certain mathematical functions such as Principal Component Analysis (PCA).

- Ability to generate an assembly file, which could be easily referred in any of the .NET aware languages.

### **5.1.3 MCR (MATLAB Compiler Runtime)**

The feature extraction and matching modules developed in MATLAB, have to work even on a machine that does not have pre-installed MATLAB software. This is where MATLAB Compiler Runtime (MCR) comes into the picture. MATLAB Compiler uses the MCR, a standalone set of shared libraries that enables the execution of MATLAB files on computers without an installed version of MATLAB.

The MCR differs from MATLAB in several important ways:

- In the MCR, MATLAB files are securely encrypted for portability and integrity.
- MATLAB has a desktop graphical interface. The MCR has all of MATLAB's functionality without the graphical interface.
- The MATLAB and Java paths in an MCR instance are fixed and cannot be changed. To change them, MATLAB has to be customized first.

When the program is run on a system which does not have a pre-installed MATLAB, the C# component initializes the MCR object in order to execute the functions developed in MATLAB.

### **5.1.4 Microsoft XNA**

Microsoft XNA ('XNA's Not Acronymed') is a set of tools with a managed runtime environment provided by Microsoft that facilitates computer game development and management. XNA attempts to free game developers from writing "repetitive boilerplate code" and to bring different aspects of game production into a single system. The XNA Framework is based on the native implementation of .NET Framework 2.0 on Windows. XNA Game Studio is an integrated development environment (IDE) for development of games. XNA Game Studio was used in the development of the first version of the project. XNA Game Studio was used to design and develop the cube

animation part. Animations of the cube involve pushing the cube, rotating the cube to the left etc. XNA significantly reduced the development effort required to do this task.

### 5.1.5 P/Invoke (Platform Invocation Services)

Platform Invocation Services, commonly referred to as P/Invoke, is a feature of Common Language Infrastructure (CLI) implementations, like Microsoft's Common Language Runtime (CLR), that enables managed code to call native code. When using P/Invoke, the CLR handles DLL loading and conversion of the unmanaged previous types to CTS types (also referred to as parameter marshalling). To perform this, the CLR:

- Locates the DLL containing the function.
- Loads the DLL into memory.
- Locates the address of the function in memory and pushes its arguments onto the stack, marshaling data as required.

P/Invoke is useful for using standard (unmanaged) C or C++ DLLs. It can be used when a programmer needs to have access to the extensive Windows API, as many functions provided by the Windows libraries lack available wrappers. When a Win32 API is not exposed by the .NET framework the wrapper to this API must be written manually.

An example of P/Invoke function is given in the following code snippet which was used in the *LoginForm.cs* which is used to lock the screen –

```
[DllImport("user32.dll")]
```

```
static extern IntPtr FindWindow(string lpzClassName, string lpzWindowName);
```

The above code snippet locates the *FindWindow* function, in the *user32.dll* assembly file, which is used to locate a window handle for a particular process.

## 5.2 Libraries and APIs' used

This section briefly discusses about the various libraries and APIs' used in our project.

The libraries used for this project were mainly provided by Emotiv and MathWorks.

### 5.2.1 Libraries and APIs' provided by Emotiv

The following list discusses some of the important APIs' and libraries provided by Emotiv, used in our project:

- *edk.dll* and *edkutils.dll* – These libraries perform the core operations of the Emotiv headset such as profile management and writing the brain waves of a person to .CSV file. Developed using C++.
- *DotNetEmotivSDK.dll* – This library delegates all the core operations to *edk.dll* or *edkutils.dll* (these 2 libraries cannot be used in C# directly and hence *DotNetEmotivSDK.dll* calls the appropriate function from either of these libraries as required).
- *EmoEngine* – *EmoEngine* is a class defined in the Emotiv which provides APIs' to communicate with Emotiv detection engine.
- *CognitivGetCurrentAction()* – This function returns the detected Cognitiv action of the user. Defined in *edk.dll* library.
- *CognitivGetCurrentActionPower()* – This function returns the detected Cognitiv action power of the user. Defined in *edk.dll* library.
- *DataAcquisitionEnable()* – This function controls acquisition of data from EmoEngine (which is off by default). Used in *EEGLogger* class, which is used to record raw EEG data in method 2.
- *ProcessEvent()* – This function processes EmoEngine events until there is no more events. If a parameter is passed indicating maximum processing time. Once a time out occurs, the control returns from the function.
- *UserAddedEventHandler()* – Function pointer of callback functions which will be called when *UserAddedEvent* occurs. *UserAddedEvent* occurs whenever the headset switched on or a new username is created (for a new user).

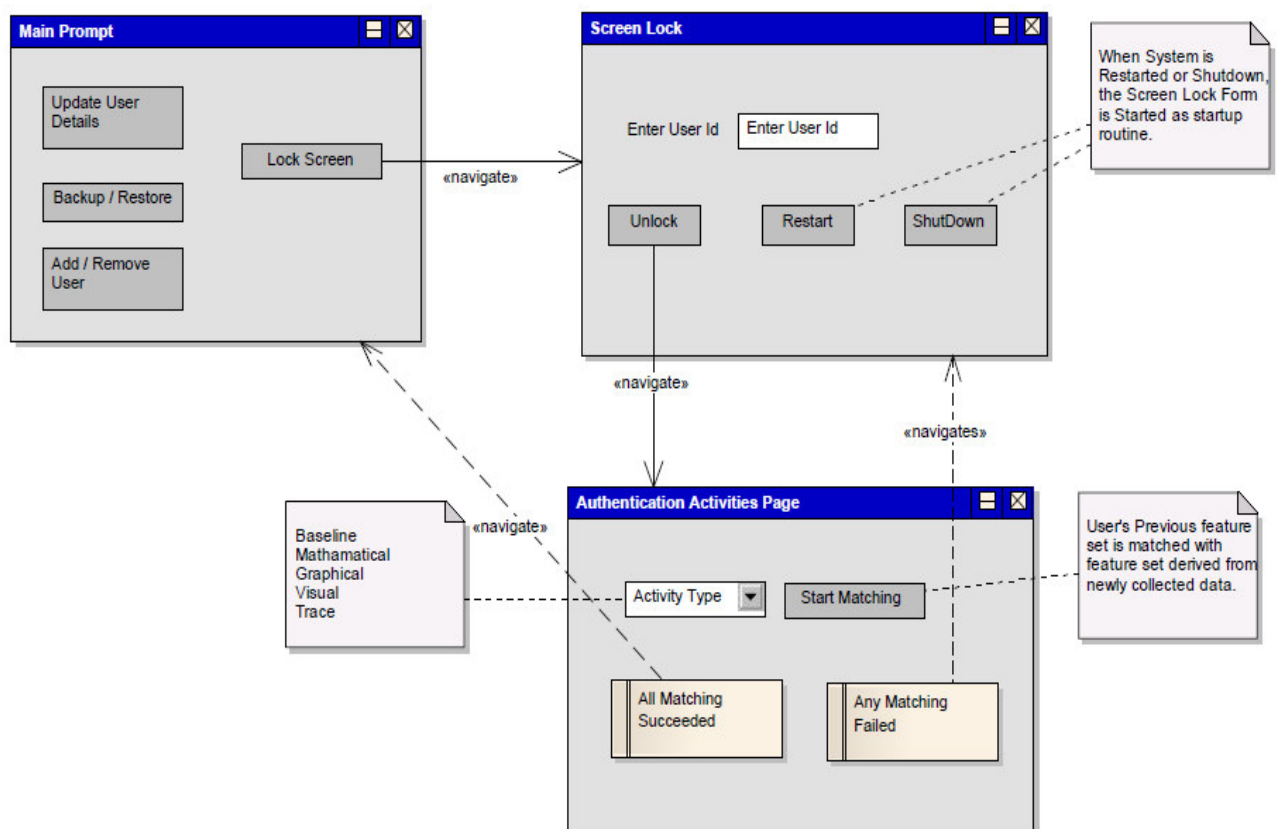
### 5.2.2 Libraries and APIs' provided by MathWorks

This section lists the libraries and APIs' provided by MathWorks which enables instantiation of an MCR object, which executes the MATLAB modules without actually installing MATLAB.

- *MathWorks.MATLAB.NET.Arrays* – The classes in this namespace provide access to MATLAB arrays from any .NET CLS compliant language. These classes support array formatting, type specific indexing, and error handling capabilities. A few of the classes defined in this namespace are: *MWArray*, *MWCellArray*, *MWNumericArray*.
- *MathWorks.MATLAB.NET.Utilities* – The utility classes in this namespace provide general support for the *MWArray* class hierarchy and a managed API for the MATLAB Common Runtime. Two classes are defined in this namespace, they are: *MWMCR* and *NativeGCAttribute*.
- *MWArray* – *MWArray* is an abstract class that serves as the root of the MATLAB array class hierarchy. It encapsulates a native MATLAB *mxArray* and provides a managed API for accessing, formatting, and manipulating the native array. All the parameters passed to function, which are called from C#, be it of any data type are implicitly converted *MWArray* object.
- *MWNumericArray* – *MWNumericArray* is the managed representation of the MATLAB numeric array types. Like its MATLAB equivalent, it is the default array type used by most of the MATLAB math functions.
- *MWArrayComponent* – *MWArrayComponent* is a MATLAB numeric array component enumerator. Members of this particular enumeration are *Real* and *Imaginary*.

### 5.3 User Interface Diagram

The UI diagram explains the various stages and steps involved from the user's perspective. The diagram below depicts the different forms involved in the application for user interface.



**Fig 5.1: User Interface Diagram**

## 5.4 Detailed Explanation of Implementation

The following steps act as a walkthrough for our application.

**Step 1:** The initial screen which is the main prompt screen facilitates the user to perform the following activities.

- Lock screen
- Train system
- Add/remove user
- Create backup for any user
- Change account name
- Restore or resample EEG data

**Step 2:** We add in a new user as there are no existing users initially. The training form opens wherein we train the system for authentication. The training is based on actions performed on a cube. This includes activities such as neutral, cube rotation, lift, drop, etc. The user profile is saved after training.

**Step 3:** Once the training process is complete, the user returns to the main prompt form. The user can now lock the screen by clicking on the lock screen option. The login form appears wherein user name has to be specified for unlocking the screen. There are 3 available options

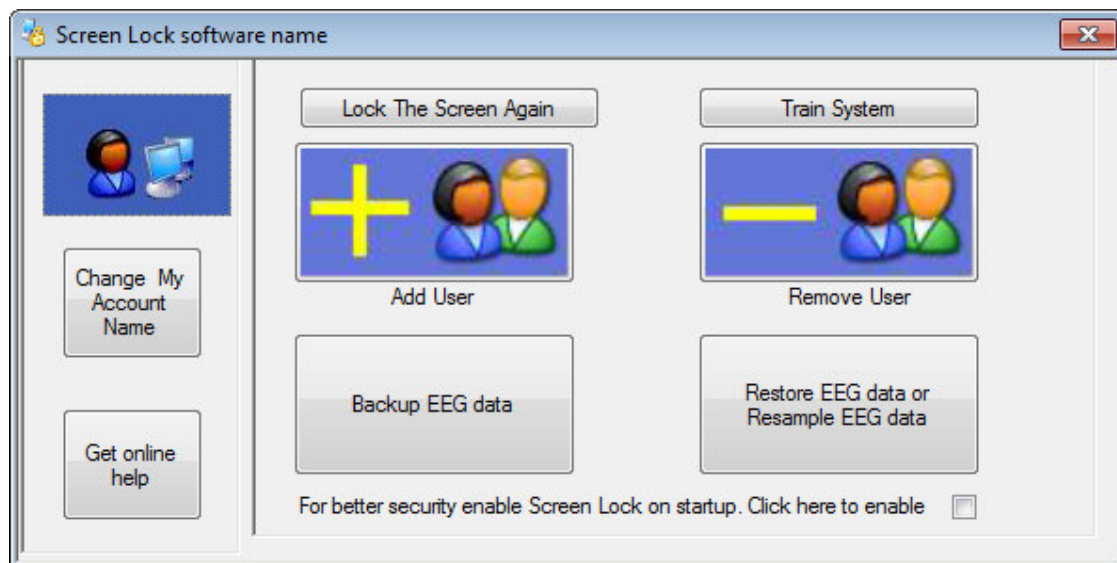
- Unlock
- Restart
- Shutdown

**Step 4:** When the unlock option is pressed by the user an authentication form appears. Three activities, for which the system has been trained earlier, must be performed on the cube for authentication one after the other.



**Step 5:** If the authentication is successful then the main prompt form is displayed and the screen is successfully unlocked, else the authentication fails and the screen remains in the locked state.

## Step 1



**Fig 5.2 Main Prompt Form**

### Change of account name (applicable to both methods):

The following code enables us to change the account name

```
private void accountNameButton_Click(object sender, EventArgs e)
{

    this.Hide();
    if (ScreenLock.AccountUpdate.accountUpdateStaticObject != null)
    {
        ScreenLock.AccountUpdate.accountUpdateStaticObject.Show();
    }
    else
    {
        ScreenLock.AccountUpdate.accountUpdateStaticObject = new
        AccountUpdate();
        ScreenLock.AccountUpdate.accountUpdateStaticObject.Show();
    }
}
```

```
}
```

**Lock screen (applicable to both methods):**

The lock screen option invokes the login form. The details of login form will be discussed in detail in step 3. The code for invoking login form

```
private void lockScreenAgainButton_Click(object sender, EventArgs e)
{
    this.Hide();
    ScreenLock.BackgroundScreen.backgroundScreenStatic = new
    BackgroundScreen();
    ScreenLock.BackgroundScreen.backgroundScreenStatic.Show();
}
```

**Add user****For Method 1:**

The add user option loads the training form with default settings. The new user must provide various training data and save his profile for using the authentication services of the application.

Training form will be discussed in detail in step 3.

Code for add user option

```
private void addUserButton_Click(object sender, EventArgs e)
{
    this.Hide();
    WindowsFormsThreadDemo.TrainingForm.trainingFormStaticObject = new
    WindowsFormsThreadDemo.TrainingForm();
    WindowsFormsThreadDemo.TrainingForm.updateUserProfile = false;
    WindowsFormsThreadDemo.TrainingForm.userNameToBeLoaded = "";
    WindowsFormsThreadDemo.TrainingForm.trainingFormStaticObject.Show();
    this.TopMost = false;
}
```

**For Method 2:**

In method 2 of the project, the same *Add User* button is used to add a new user and extract his brain wave features and store them in a file.

A snapshot of the training form is given in the next page:

**Fig. 5.3: Training Form (Method 2)**

The code snippet for invoking the *TrainingForm* is given below:

```
private void addUserButton_Click(object sender, EventArgs e)
{
    SignalTrainAndMatch.TrainingForm trainingFormObj = new
    SignalTrainAndMatch.TrainingForm();
    trainingFormObj.Show();
}
```

**Remove user****For method 1:**

The remove user option deletes the user profile and all related data from the system.

Code for remove user option

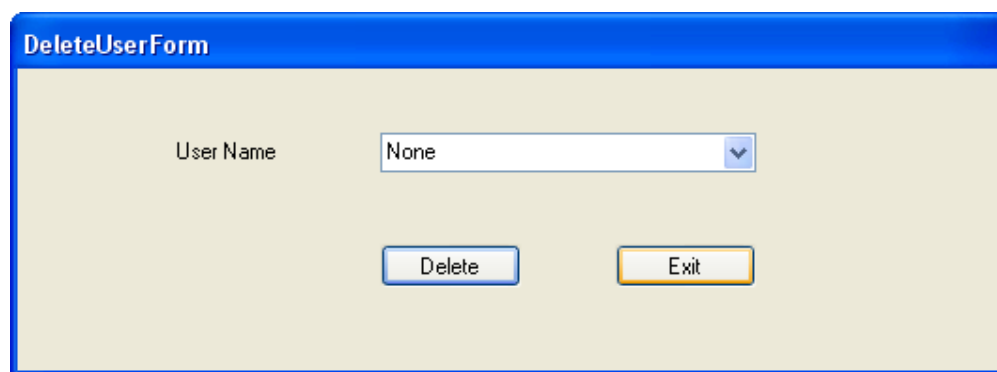
```
private void removeUserButton_Click(object sender, EventArgs e)
{
```

```
ScreenLock.DeleteUser deleteUser = new DeleteUser();  
deleteUser.Show();  
}
```

**For method 2:**

The function of the remove user is same as above, however a separate form, *DeleteUserForm* is displayed to the super user who wants to delete a particular profile.

A snapshot of the *DeleteUserForm* is given below:



**Fig. 5.4: Delete User Form (Method 2)**

Code snippet to delete a user is given below:

```
private void removeUserButton_Click(object sender, EventArgs e)  
{  
    DeleteUserForm deleteUserFormobj = new DeleteUserForm();  
    deleteUserFormobj.Show();  
}
```

**Train system (applicable to method 1 only):**

The train system option loads the previous training data of the logged in user and its enables further training and updating of the system. It loads the training form in update user profile mode. Training form will be discussed in detail in step 3.

Code for train system option

```
private void button_TrainSystem_Click(object sender, EventArgs e)  
{
```

```
this.Hide();  
WindowsFormsThreadDemo.TrainingForm.trainingFormStaticObject = new  
WindowsFormsThreadDemo.TrainingForm();  
WindowsFormsThreadDemo.TrainingForm.updateUserProfile = true;  
WindowsFormsThreadDemo.TrainingForm.userNameToBeLoaded =  
userName;  
WindowsFormsThreadDemo.TrainingForm.trainingFormStaticObject.Show();  
this.TopMost = false;  
}
```

## **Step 2**

For a system with no existing users the add user option is clicked. The training form is loaded with default settings.

The system may be trained in two ways depending on the option selected by the user.

- Authentication based on actions performed on a cube.
- Authentication based on activities such as visual stimulus free meditation data, and math activity.

The screenshot displays a software window titled "Main Form". It contains several input fields and buttons for user management and training. At the top, there is a "New userName" text box followed by a "Save User Profile" button. Below this is a "Current Action" text box with a long horizontal bar underneath it. The "Active Actions" section features a list box labeled "listBox\_ActiveActions" and a "Delete Activity" button. An "Overall Skill Rating" text box is positioned below the list box. The "Add New Action" section includes a dropdown menu and an "Add New Action" button. The "Training Action" section has a dropdown menu, an "Animate cube" checkbox, and "Start Training" and "Erase Training" buttons. An "Exit" button is located at the bottom right of the form.

Main Form

New userName  Save User Profile

Current Action

Active Actions listBox\_ActiveActions Delete Activity

Overall Skill Rating

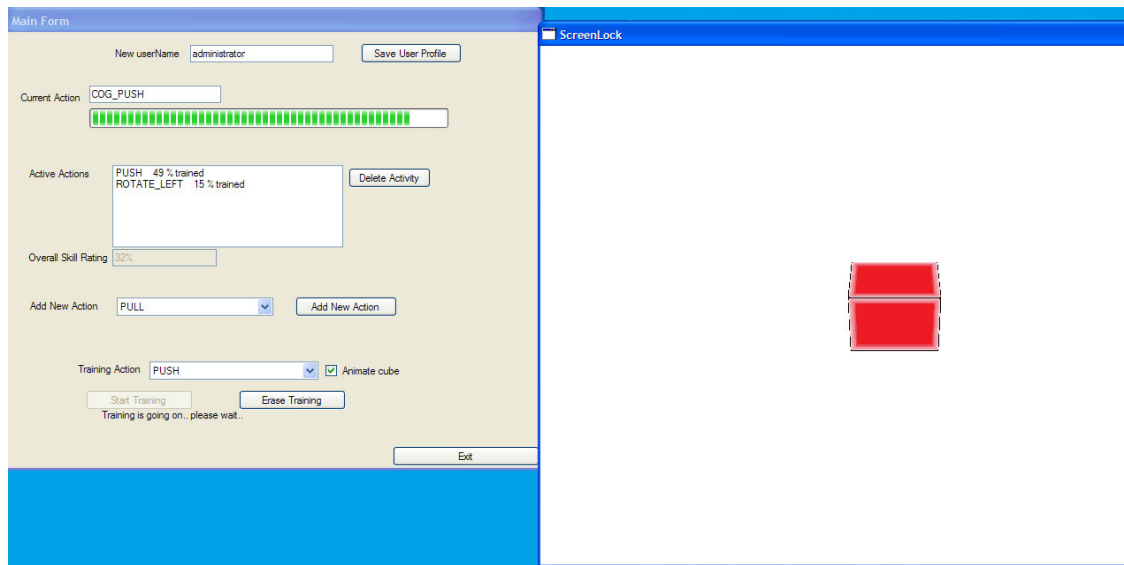
Add New Action  Add New Action

Training Action  ☐ Animate cube

Start Training Erase Training

Exit

**Fig 5.5: Training Form (Method 1)**



**Fig. 5.6: Animation of Cube while training (Method 1)**

### **Add new action**

#### **For method 1:**

Under 'Add New Action' drop down menu we have several activities that may be performed on a cube such as pull, push, rotate left, rotate right, lift, drop, rotate clock wise, rotate anti-clock wise, left, right, rotate reverse and even disappear. It is to be noted that before training any of these activities a neutral signal must be provided by the user. For the purpose of authentication we will be using a maximum of 4 activity signals provided by the user. The recordings are successfully carried out using API's provided by Emotiv SDK.

If a user wants to add a new action such as pushing a cube, the following code snippet would explain its execution

```
private void button_AddNewAction_Click(object sender, EventArgs e)
{
    uint activeActions = 0;
    uint newActiveActions = 0;
    uint newActiveAction;
    activeActions = engine.CognitivGetActiveActions(userId);
```



```
//MessageBox.Show(comboBox_Add_New_Action.SelectedItem.ToString());
if (totalCognitivActiveActions < 4)
{
    totalCognitivActiveActions++;
    switch (comboBox_Add_New_Action.SelectedItem.ToString())
    {
        case "PUSH":
            newActiveAction = uint.Parse(Enum.Format(typeof(EE_CognitivAction_Tt),
            EE_CognitivAction_Tt.COG_PUSH, "D"));
            newActiveActions = activeActions | newActiveAction;
            engine.CognitivSetActiveActions(userId, newActiveActions);
            break;
        }
    }
else
{
    MessageBox.Show("You can have maximum of 4 trained Actions");
    return;
}

newUpdateThread = new Thread(updatelist);
newUpdateThread.Start();

}
```

**For method 2:**

In the second method of the project, there is no such option to add a new action as of now. The number of activities are fixed to 2 viz. Meditation and Math.

**Save user profile****For Method 1:**

Saves the profile under the user name entered in the text box. All the training data is correlated to the user name and stored in a new directory.

Code for save user profile

```
private void button_SaveUserProfile_Click(object sender, EventArgs e)
{

    if (textBox_NewUserName.Text.Length <= 5)
    {
        MessageBox.Show("User name must contain atleast 5 characters");
        return;
    }
    else
    {
        bool uniqueFlag = true;
        DialogResult result = System.Windows.Forms.DialogResult.No;
        DirectoryInfo dInfo = new DirectoryInfo("profiles");
        FileInfo[] fInfos = dInfo.GetFiles();
        foreach (FileInfo finfo in fInfos)
        {
            if (finfo.Name.Equals(textBox_NewUserName.Text + ".emu"))
            {
                uniqueFlag = false;
            }
        }
        if (!uniqueFlag)
        {

```

---

```

        result = MessageBox.Show("User name exists! Do you want to
        overwrite the file?", "WARNING", MessageBoxButtons.YesNo,
        MessageBoxIcon.Warning, MessageBoxDefaultButton.Button2);

        if (result.ToString().Equals("Yes"))
        {
            engine.EE_SaveUserProfile(userId,
            "profiles\\"+textBox_NewUserName.Text + ".emu");
        }
    }
    else
    {
        engine.EE_SaveUserProfile(userId,
        "profiles\\"+textBox_NewUserName.Text + ".emu");
    }
}
}
}

```

Once the training is complete user is brought back to the main prompt form.

#### **For Method 2:**

This particular module, for method 2, is implemented in MATLAB. The code creates a new directory for the user with the username as the directory name, and the features of the user for a particular activity are stored in .CSV format as \*.txt files.

```

function storeNewFeature(userName , activityFile , activityType)
x = featureExtract(activityFile);
if(x == 0)
    error('No feature contributed to 90% of covariance. Please train the activity again');
end
fileName = strcat(activityType , 'Activity.txt');
y = exist(userName , 'dir');
if (y == 0)

```

```
    mkdir(userName);  
end  
cd(userName);  
csvwrite(fileName , x);  
cd ..;  
end
```

Feature extraction module is also implemented in MATLAB. Code definition for it is given below:

```
function feature_set = featureExtract(file,segno)  
data=loadCSV(file);  
segmented_data=data(:,(512*(segno-1)+1):512*segno);  
channel_spec_pow = channel_spectral_power(file,segmented_data);  
feature_set=channel_spec_pow ;  
end
```

The *storeNewFeature* module is called from the C# code. The code snippet is given below:

```
void TrainStartButtonClick(object sender, EventArgs e)  
{  
    /*Error checking and intialization part*/  
    try  
    {  
        userAuthObj.storeNewFeature(userName , storedFile , activityType);  
        MessageBox.Show("Training Completed successfully!!");  
        mediTrainedlabel.Text = "Completed";  
        trainMediButton.Visible = false;  
        trainMathButton.Visible = true;  
    }  
}
```

```
catch (Exception ex)
{
    MessageBox.Show("There was some error while extracting features");
    MessageBox.Show(ex.Message);
    MessageBox.Show(ex.StackTrace);
    trainMediButton.Enabled = true;
}
```

### **Step 3**

Once the training process is complete, the user returns to the main prompt form. The user can now lock the screen by clicking on the lock screen option. The login form appears wherein user name has to be specified for unlocking the screen. There are 3 available options

- Unlock
- Restart
- Shutdown



**Fig. 5.7: Login Form**

The form does the following:

- All special key combinations such as Alt+Tab , Alt+F4 and Windows key are disabled, thereby giving an essence of 'locked screen' to the user.
- hides the taskbar
- minimizes the task manager window hence pressing "Ctrl+Alt+Del" will only create new instance of minimized window.
- minimizes all the active/open windows.

All the above functions were implemented using PInvoke functions.

### **Unlock**

#### **For Method 1:**

The unlock option invokes an authentication form object and the authentication form is displayed for the user name entered in the text box.

Code for unlock option

```
private void UnlockButton_Click(object sender, EventArgs e)
{

    userName=this.userNameTextbox.Text;
    string userNameFile=userName+".emu";
    bool flag = false;

    if (this.userNameTextbox.TextLength == 0)
    {
        MessageBox.Show("No username entered!!", "ERROR",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        this.Close();
        LoginForm loginFormObj = new LoginForm();
        BackgroundScreen.loginFormReload(loginFormObj);
        return;
    }
}
```

```
try
{
    try
    {
        DirectoryInfo dInfo = new DirectoryInfo("profiles");
        FileInfo[] fInfos = dInfo.GetFiles();

        foreach (FileInfo fInfo in fInfos)
        {
            if (userNameFile.Equals(fInfo.Name))
            {
                flag = true;

                this.Close();

                WindowsGameSimpleCube1.AuthenticateForm.
                authenticationFormStaticObject = new
                WindowsGameSimpleCube1.AuthenticateForm();
                WindowsGameSimpleCube1.AuthenticateForm.aut
                henticationFormStaticObject.Show();
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.StackTrace);
        MessageBox.Show("Exception Username: " + userName + " is
        invalid! ", "ERROR", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
        this.Close();
    }
}
```

```
        LoginForm loginFormObj = new LoginForm();
        BackgroundScreen.loginFormReload(loginFormObj);
        flag = true;

    }
    if (!flag)
    {
        MessageBox.Show("Username: " + userName + " is invalid! ",
            "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        this.Close();
        LoginForm loginFormObj = new LoginForm();
        BackgroundScreen.loginFormReload(loginFormObj);

    }
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
```

**For Method 2:**

When a user clicks the unlock button, a new form called match form is displayed. The code that is executed when the user clicks the *Unlock* button is given below:

```
private void UnlockButton_Click(object sender, EventArgs e)
{
    userName=this.userNameTextbox.Text;
    bool flag = true;
    try
    {
```



```
        flag = true;
        if (this.userNameTextbox.TextLength == 0)
        {
            MessageBox.Show("No username entered!!",
                "ERROR", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            this.Hide();
        }

        screenLockObj.removeFromStartup();
        this.Close();
        SignalTrainAndMatch.MatchForm matchFormObj = new
        SignalTrainAndMatch.MatchForm();
        matchFormObj.Show();
        flag = false;
        if (flag)
        {
            LoginForm loginFormObj = new LoginForm();
            BackgroundScreen.loginFormReload(loginFormObj);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

**Restart (applicable to both methods):**

The restart button is coded to restart the computer.

```
private void RestartButton_Click(object sender, EventArgs e)
{
    Process.Start("shutdown", " /r /t 00");
}
```

**Shut down (applicable to both methods):**

The shutdown button is coded to shut down the computer.

```
private void ShutdownButton_Click(object sender, EventArgs e)
{
    Process.Start("shutdown", " /s /t 00");
}
```

**Step 4****For Method 1:**

When the unlock option is pressed by the user an authentication form appears. Three activities, for which the system has been trained earlier, must be performed on the cube for authentication one after the other.

Action	Status	Action
Action1	Not Authenticated	Authenticate
Action2	Not Authenticated	Authenticate
Action3	Not Authenticated	Authenticate

Action Power: 
 Current Action:

**Fig. 5.8: Authentication Form (Method 1)**

Action 1, action 2 and action 3 are loaded according to the actions trained by the user.

Each action must be authenticated one after the other. Every action must be performed for a certain minimum period of time with the power of the action greater than a certain

threshold value. A main timer times the whole recording and matching activity for 30 seconds. At the end of 30 seconds if the activity is performed by the user according to the above constraints then that action is authenticated. This process is repeated for all 3 actions. The actions are identified using the API's provided by EMOTIV SDK.

Sample code for authentication is given below (it is assumed that the user is performing the cube push activity for authentication purposes).

```
private void button_AuthenticateAction1_Click(object sender, EventArgs e)
{
    actionNumber = 1;
    label1.Text = "";
    label2.Text = "";
    label_Action1State.Text = "Not Authenticated";
    label3.Text = "";
    differentActionCount = 0;
    CurrentAuthenticationAction = label_Action1.Text;
    timer = new System.Windows.Forms.Timer();
    timer.Enabled = true;
    timer.Interval = 30000;
    timer.Tick += new EventHandler(timer_Tick);
    MinerTimer = new System.Windows.Forms.Timer();
    MinerTimer.Interval = 200;
    MinerTimer.Enabled = false;
    MinerTimer.Tick += new EventHandler(MinerTimer_Tick);
    MainTimer = new System.Windows.Forms.Timer();
    MainTimer.Enabled = false;
    MainTimer.Interval = 3000;
    MainTimer.Tick += new EventHandler(MainTimer_Tick);
    failedCount = 0;
    cognitivAction1HandlerDisabled = false;
```

```
engine.CognitivEmoStateUpdated += new
EmoEngine.CognitivEmoStateUpdatedEventHandler(engine_CognitivEmoStateUp
datedForAction1);
CurrentCognitivActivity = " COG_PUSH ";
cognitivActivityPower = 0.0f;
animateCube();
newThreadStart = new ThreadStart(RunCube);
newBoxUpdateThread = new Thread(newThreadStart);
newBoxUpdateThread.Start();
label_ActionPower.Visible = true;
label_CurrentAction.Visible = true;
progressBar_CognitivActivity.Visible = true;
}
void animateCube()
{
    switch (CurrentCognitivActivity)
    {
        case "COG_PUSH":
            DeltaY1 = 0f;
            DeltaX1 = 0f;
            moveForward = false;
            animateDepth = true;
            liftCube = false;
            dropCube = false;
            right = left = false;
            rotateLeft = rotateRight = false;
            depth = 5;
            rotateClockwise = rotateAntiClockwise = false;
            break;
    }
}
```

```
}

```

### For Method 2:

When a user has to be authenticated, *MatchForm* is displayed. A snapshot of the form is shown below:

MatchForm		
Activity		Done
Meditation	Authenticate Meditation	Not authenticated
Math		Not authenticated

**Fig. 5.9: Authentication Form (Method 2)**

The MATLAB module to match 2 features is given below:

```
function authparam = matchFeatures(userName , activityFile , activityType)
file1 = strcat(activityType , 'Activity.txt');
count1=0;
count2=0;
cd(userName)
x = loadtxt(file1);
x = cell2mat(x);
cd ..;
y = featureExtract(activityFile);
authparam = 0;
[eigvalue1,eigvector1]=pca(x);
[eigvalue2,eigvector2]=pca(y);
covper1=eigvalue1(1)/sum(eigvalue1);
```

---

```

covper2=eigvalue2(1)/sum(eigvalue2);
if(covper1 > .85 && covper2 > .85)
    princomp1 = eigvector1(:,1);
    princomp2 = eigvector2(:,1);
    S=(abs(princomp1'*princomp2)^2);
    disp(S);
else
    count2=count2+1;
end
if(S>=.78)
    if(count1~=0)
        data1=horzcat(prev1,temp1(3:16,(512*count1)+1:512*(count1+1)));
        data2=horzcat(prev2,temp2(3:16,(512*count1)+1:512*(count1+1)));
    elseif(count1==0)
        data1=temp1(3:16,(512*count1)+1:512*(count1+1));
        data2=temp2(3:16,(512*count1)+1:512*(count1+1));
    end
    prev1=data1;
    prev2=data2;
    count1=count1+1;
end
end
if(count1>=3)
    x = channel_spectral_power1(data1,count1);
    y = channel_spectral_power1(data2,count1);
    [eigvalue1,eigvector1]=pca(x,1);
    [eigvalue2,eigvector2]=pca(y,1);
    S=(abs(eigvector1'*eigvector2)^2);
    disp(S);
    if(S>=.8)

```

```

        authparam = 1;
    else
        authparam = 0;
    end
elseif(count2>=3)
    authparam = -1;
else
    authparam = 0;
end

```

In the above MATLAB module a return value of 1 indicates that the user is a client, while a value of 0 indicates that he is an impostor and a value of -1 indicates that the data is too noisy and no meaningful inference can be made from this data.

The code to call this module from C# is given below:

```

private void signalMatchButton_Click(object sender, EventArgs e)
{
    /*Error checking and intialization part*/
    try
    {
        res = userAuthenticateObj.matchFeatures(userName, storedFile, activityType);
        ans = ((MWNumericArray)res).ToVector(MWArrayComponent.Real);
        authParam = ans.GetValue(0).ToString();
        if (authParam.Equals("1"))
        {
            mediMatchButton.Visible = false;
            mathMatchButton.Visible = true;
            matchMediLabel.Text = "Authenticated";
        }
        else if(authParam.Equals("0"))
        {

```

```
        authParam = null;
        MessageBox.Show("Authentication Failure!!", "FAILURE",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
        mediMatchButton.Enabled = true;
        mathMatchButton.Visible = false;
        return;
    }
    else
    {
        authParam = null;
        MessageBox.Show("Recorded data was noisy or inconsistent!!Please try
        again..");
        mediMatchButton.Enabled = true;
        mathMatchButton.Visible = false;
        return;
    }
}
catch (Exception ex)
{
    MessageBox.Show("There was some error while matching the features!!");
    MessageBox.Show(ex.Message);
    MessageBox.Show(ex.StackTrace);
}
}
```

### **Step 5**

If the authentication is successful then the main prompt form is displayed and the screen is successfully unlocked, else the authentication fails and screen remains in the locked state.



## **Chapter-6**

### **TESTING**

Software Testing forms an important activity of software development and maintenance. It is the process used to help identify the correctness, completeness, security, and quality of developed computer software. Testing is a process of technical investigation, performed on behalf of stakeholders, that is intended to reveal quality-related information about the product with respect to the context in which it is intended to operate. This includes, but is not limited to, the process of executing a program or application with the intent of finding errors.

#### **6.1 Unit Testing**

Unit testing is a software verification and validation method in which a programmer tests if individual units of source code are fit for use. A unit is the smallest testable part of an application. The test cases and their expected behaviour are depicted in the form of tables in the following set of pages.

**6.1.1 Unit: *TrainingForm.cs***

Sl. No.	Test Case	Expected Result	Remarks
1	Username=NULL	Prompt message saying "User name must contain atleast 5 characters"	Result as expected
2	Existing user name	Warning message saying "User name exists! Do you want to overwrite the file?"	Result as expected
3	Attempt to delete all activities	Message box saying "Sorry at least one activity should be trained"	Result as expected

***Table 6.1: Training Form (Method 1)***

**6.1.2 Unit: *LoginForm.cs***

Sl. No.	Test Case	Expected Result	Remarks
1	Username=NULL	Prompt error message saying "No user name entered"	Result as expected
2	Input of invalid user name	Error message saying "Invalid User name"	Result as expected

**Table 6.2: Login Form (Applicable to both methods)****6.1.3 Unit: *AuthenticationForm.cs***

Sl. No.	Test Case	Expected Result	Remarks
1	No input for 30 seconds	Authentication failure	Result as expected
2	Input of invalid user name	Error message saying "Invalid User name"	Result as expected
3	Action power is less than threshold value for a span of 30 seconds	Authentication failure	Result as expected
4	All the actions are not authenticated	User Authentication failure	Result as expected

**Table 6.3: Authentication Form (Method 1)**

**6.1.4 Unit: *TrainingForm.cs***

Sl. no.	Test Case	Expected Result	Remarks
1	Noisy or inconsistent data recorded	Throw an exception stating that the data was too noisy and hence may require re-recording	Result as expected

**Table 6.4: Training Form (Method 1)****6.1.4 Unit: *MatchForm.cs***

Sl. No.	Test Case	Expected Result	Remarks
1	Match percentage of meditation data is less than the threshold value.	Meditation Authentication failure	Result as expected
2	Match percentage of Math activity data is less than the threshold value.	Math activity Authentication failure	Result as expected
3	Noisy or inconsistent data recorded (any activity)	Authentication Failure	Result as expected

**Table 6.5: Match Form (Method 2)**

## 6.2 Integration testing

Integration testing is a phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before system testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

### 6.2.1 Interaction of LoginForm.cs with other components:

Sl. no.	Test Case	Expected Results	Remarks
1	User feeds in the username and presses unlock button	MatchForm is shown to authenticate the user	Result as expected

**Table 6.6: Login Form Interaction**

### 6.2.2 Interaction of MatchForm.cs with other components:

Sl. No.	Test Case	Expected Result	Remarks
1	The user is completely authenticated	MainPromptForm is displayed for advanced user options	Result as expected

**Table 6.7: Match Form Interaction**

**6.2.3 Interaction of MainPromptForm.cs with other components:**

Sl. no.	Test Case	Expected Result	Remarks
1	A superuser decides to add a new user and clicks the “Add User” button	TrainingForm is displayed and the brain waves are recorded	Result as expected
2	A superuser decides to delete an existing user and clicks the “Remove User” button	DeleteUserForm is displayed and the super user can delete the target user	Result as expected
3	A user decides to modify his user name and clicks on “Update User Details” button	AccountUpdateForm is shown so that the user can modify only his username	Result as expected
4	The user decides to lock his screen again	LoginForm is displayed on top of the BackGroundScreen	Result as expected (the application is restarted)
5	The user needs help clicks on the “Online Help” button	A browser instance is created, so that the user can browse through the web to get online help	Result as expected, however the browser home page is currently set to <a href="http://www.google.com">www.google.com</a>

**Table 6.8: Main Prompt Form Interaction**

## **Chapter 7**

### **SUMMARY, CONCLUSION AND SCOPE FOR FUTURE WORK**

Two different versions of this project were implemented. The first version was developed entirely using the Emotiv APIs' to train and authenticate the user. The second version of the project was developed using MATLAB for feature extraction and matching, while the user interface was created in C#. We observed that the first version is not very suitable for implementing a full fledged authentication system. The FAE is high when only a single activity is used. With the use of two or more activities, the FRE increases considerably. The second version is, however more authentic, but has its own drawbacks:

- Recording must be done in clinical conditions where there are no external interferences (noise free environment).
- Training the users to perform the various mental tasks with full concentration.
- Handling high dimensional data.

Future work in this regard includes:

- Exploring other approaches which can increase the reliability of scalp EEG recordings.
- Exploring other dimension reduction algorithms that can reduce the size of the EEG features.
- Devising a more or less perfect matching algorithm that gives 0 FAE and 0 FRE.
- Developing a multimodal biometric authentication system.

## **Chapter -8**

### **BIBLIOGRAPHY**

Our references for the project include the following research papers:

**[1] PERSON AUTHENTICATION USING BRAINWAVES (EEG) AND MAXIMUM A POSTERIORI MODEL ADAPTATION** by S'ébastien Marcel and Jos'e del R. Mill'an.

**[2] THE ELECTROENCEPHALOGRAM AS A BIOMETRIC** by R. Paranjape, J. Mahovsky, L. Benedicenti and Z. Koles.

**[3] NEURAL NETWORK BASED PERSON IDENTIFICATION USING EEG FEATURES** by M. Poulos, M. Rangoussi. and N. Alexandris.

**[4] TWO-STAGE BIOMETRIC AUTHENTICATION METHOD USING THOUGHT ACTIVITY BRAIN WAVES** by RAMASWAMY PALANIAPPAN

**[5] PERSON AUTHENTICATION USING EEG BRAINWAVE SIGNALS** by Chen He

To develop the UI component using C#, the following book helped us a lot:

- **Pro C# with .NET 3.0** by **Andrew Troelsen**, Apress.

The Internet has been a Huge Source of Information, where we fetched a great number of ideas for this project:

<http://www.emotiv.com/>

<http://www.mathworks.com/>

<http://www.msdn.microsoft.com/>

<http://www.sccn.ucsd.edu/eeglab/>

<http://www.pinvoke.net/>

<http://www.creators.xna.com/en-US/>