

CHAPTER 1

INTRODUCTION

CHAPTER 1

INTRODUCTION

Bio-Me is a biometric solution for the identification of lost children and senior citizens, runaways and incapacitated or delirious individuals. It uses Fingerprint Authentication technology to do this. It is a desktop application supported by a Bio-Me database to store the biometric and demographic data of the individuals. The enBioScan-C1 HFDU08 Fingerprint Scanner is used to capture the fingerprints and using the Nitgen's eNBSP SDK for matching the fingerprints, authentication is performed. The fingerprints are safe from misuse as they are stored as a secure template from which the fingerprints cannot be reproduced.

1.1 Our Name and Logo

The name Bio-Me comes from the word **BIOMETRICS**.

The logo of Bio-Me features a fingerprint and the letter 'O' in the shape of a fingerprint.



Fig. 1 Bio-Me Logo

1.2 Motivation

India suffers from a growing epidemic of missing children. A child goes missing every 8 minutes in our country. Also, the number of elderly suffering from Alzheimer's, Dementia, Amnesia etc. is increasing. "About 50-60 per cent of people with Alzheimer's wander off at some point of time during their illness," notes Dr P.T. Sivakumar, a consultant in adult and geriatric psychiatry, National Institute of Mental Health And Neurosciences (Nimhans), Bangalore. When these individuals are discovered, there is no fixed procedure to identify them. Haphazard processes like publishing their photographs in newspapers or on social media are used.

1.3 Purpose

The target groups of Bio-Me include:

- Lost children or senior citizens
- Runaways
- Incapacitated or Delirious Individuals
- Accident victims

Bio-Me aims at aiding the identification of the above individuals in a fast and efficient manner by using Fingerprint Authentication technology.

1.4 Applicability

Bio-Me can be integrated with India's national ID program, *Aadhar*. The Bio-Me app has an easy to use UI and hence can be comfortably operated by both IT savvy as well as the technologically naïve.

It can be deployed at Police stations, Hospitals , Ambulances, Railway stations, Airports, Bus stations etc. thus eliminating an unstructured and haphazard search process.

CHAPTER 2

LITERATURE SURVEY

CHAPTER 2

LITERATURE SURVEY

2.1 Overview of Biometrics

Biometrics is basically the measurement and analysis of an individual's physical and behavioural characteristics. It mainly finds its application in identification and access control. Biometric authentication uses the uniqueness of an individual's physical or behavioural traits.

The two main types of biometric identifiers are:

- Physiological:
The shape or composition of the body.
(Ex. Fingerprints, Face, Retina etc.)
- Behavioural:
The behaviour of a person.
(Ex. Gait, Gestures, Typing rhythm etc.)

2.1.1 Requirements of Biometrics

- Universality: Every person should have the characteristic
- Uniqueness: No two persons should be the same in terms of the characteristic
- Persistence: The characteristic should be invariant with time
- Collectability: The characteristic can be measured quantitatively.
- Performance: The achievable identification accuracy, the resource requirements to achieve an acceptable identification accuracy and the factors that affect the identification accuracy
- Acceptability: To what extent society is willing to accept the biometric system,
- Circumvention: How easily the system can be fooled

2.1.2 Comparison of Biometric Technologies

Biometric characteristic	Universality	Unicity	Persistence	Collectability	Performance	Acceptability	Circumvention
Face	high	low	medium	high	low	high	low
Fingerprint	medium	high	high	medium	high	medium	high
Hand Geometry	medium	medium	medium	high	medium	medium	medium
Iris	high	high	high	medium	high	low	high
Retinal Scan	high	high	medium	low	high	low	high
Signature	low	low	low	high	low	high	low
Voice	medium	low	low	medium	low	high	low
Thermogram	high	high	low	high	medium	high	high

Table 1. Comparison of Biometrics

Thus, fingerprints in comparison with the other biometrics are highly favored with an added advantage of social acceptability.

2.2 Fingerprints

Fingerprints are graphical flow-like ridges that are present on human fingers. The initial conditions of the embryonic mesoderm of the foetus determine their formation. Fingerprints have been used especially in the field of forensics for a long time.

The biological principles of fingerprints are summarized below:

- Individual epidermal ridges and valleys are different for different fingers.
- The configuration types are variable but they vary within limits that allow for systematic classification.
- The configurations and minute details are permanent and unchanging for a finger.

2.2.1 Types of Fingerprints

Friction ridge patterns are grouped into three distinct types depending on the delta points which are points where lines from three directions come together.

The three types are as follows:

1. Loop : Prints recurve back onto themselves.

Loops have a single delta point.

2. Whorl: Form circular or spiral patterns.

Whorls have two delta points.

3. Arch: Form wave-like patterns.

Arches do not have any deltas.

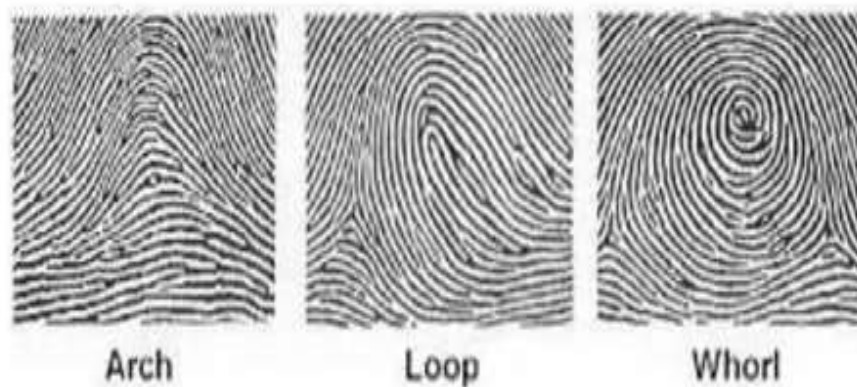


Fig 2. Types of Fingerprints

2.3 Identification and Authentication Systems

There are two types of systems that help automatically establish the identity of a person:

1. Identification Systems (Who am I?).
2. Authentication or Verification Systems (Am I who I claim I am?)

In a verification system, a person desired to be identified submits an identity claim to the system, usually via a magnetic stripe card, login name, smart card, etc., and the system either rejects or accepts the submitted claim of identity.

In an identification system, the system establishes a subject's identity (or fails if the subject is not enrolled in the system data base) without the subject's having to claim an identity.

2.4 Fingerprint Identification Approach

Fingerprint identification systems are based on the minutiae.

Minutiae are location and direction of the **ridge endings and bifurcations** (splits) along a ridge path.

An Automatic Fingerprint Identification System (AFIS) is designed to interpret the structure of the overall ridges in order to perform classification and then extract the minutiae details.



Fig. 3 Minutiae

2.4.1 Fingerprint Scanners

A variety of Sensors are used for collecting the image of a fingerprint. Fingerprint Sensors can be- Optical, Capacitive, Ultrasound and Thermal. The enBioScan-C1 HF DU08 Fingerprint Scanner is an Optical Scanner that makes use of Surface Enhanced Irregular Reflection (SEIR).

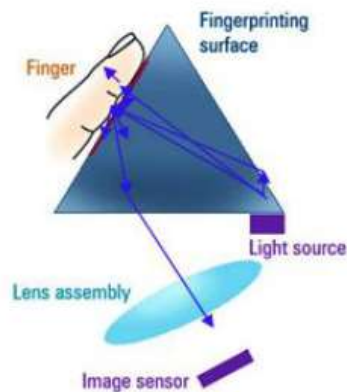


Fig. 4 Concept of SEIR

The SEIR technology uses the principle of Scattering instead of Total Internal Reflection which is used by most of the optical scanners. When light hits the ridge, the light is scattered and reflected. Most of the scattered light is collected which makes the ridge appear as a bright spot in the image. When light hits the valley, it completely passes through and is not scattered. This gives it a dark spot appearance in the image. Thus, SEIR overcomes the problem of low contrast and distortion caused by the traditional Frustrated Total Internal Reflection (FTIR) technology.

2.4.2 Fingerprint Matching Algorithms

Fingerprint Matching algorithms are used to compare previously stored templates of fingerprints against fingerprints of individuals for the purpose of authentication. The original image is directly compared with the candidate image or only certain features must be compared. Thus, they are divided into two types:

- Pattern-based (or Image-based) algorithms

They compare the basic fingerprint patterns (arch, whorl, and loop) of a previously stored template and a candidate fingerprint. However, the images must be aligned in the same orientation. To do this, a central point is found in the fingerprint image.

- Minutia Feature extraction-based algorithms

They use minutiae features of the fingerprint. The major minutia features as shown in Fig.2 of fingerprint ridges are: ridge ending and bifurcation. The ridge ending is where a ridge terminates. A ridge bifurcation is where a single ridge splits into two ridges. They are analysed and extracted to perform the matching.

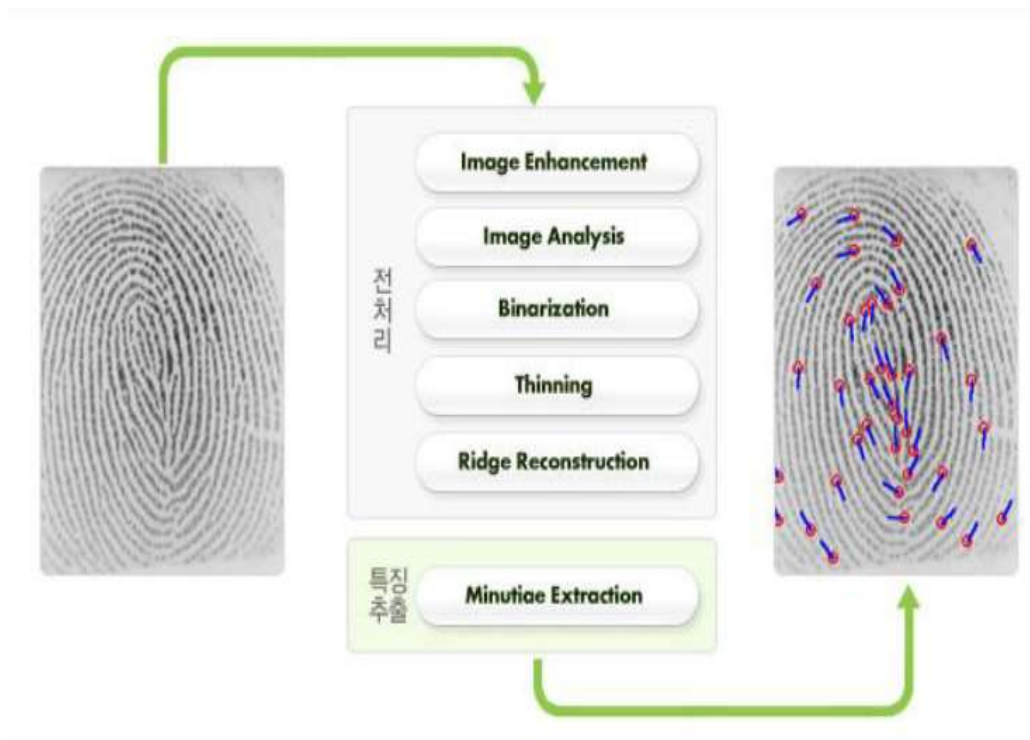


Fig. 5 Minutiae Feature Extraction-based algorithm Flowchart

The NITGEN Algorithm which is used by the eNBSP SDK is also a minutiae feature extraction-based algorithm that follows the steps as shown in Fig.5.

- In the first step, the quality of the image is improved.
- In the second step, the characteristics of the fingerprint are analysed.
- In the third step, the analysed data is binarized.
- The next step consists of thinning considering temporal performance.
- Then the data is stored as a template made up of characteristics.

CHAPTER 3

SYSTEM REQUIREMENTS

CHAPTER 3

SYSTEM REQUIREMENTS

3.1 Hardware Requirements

- USB 1.1/2.0/3.0 port
- 16MB RAM
- 20MB available hard disk space
- enBioScan-C1 HFDU08 Scanner

3.2 Software Requirements

- MS Windows 98(Second Edition or later version) /ME /2000/XP/2003/Vista/7 or higher
- SQL 2005 or higher
- enBioScan-C1 HFDU08 Windows Driver Software v1.000

CHAPTER 4

SYSTEM DESIGN

AND

ANALYSIS

CHAPTER 4

SYSTEM DESIGN AND ANALYSIS

4.1 Overview of Bio-Me



Fig. 6 Overview of Bio-Me

When an individual in distress whose identity is unknown is found, his or her fingerprint is scanned using the enBioScan-C1 scanner. The reason for scanning and description is provided. The fingerprint is matched and the resulting demographic information is displayed. This can be used to rehabilitate the individual within a short period.

4.2 Roles in Bio-Me

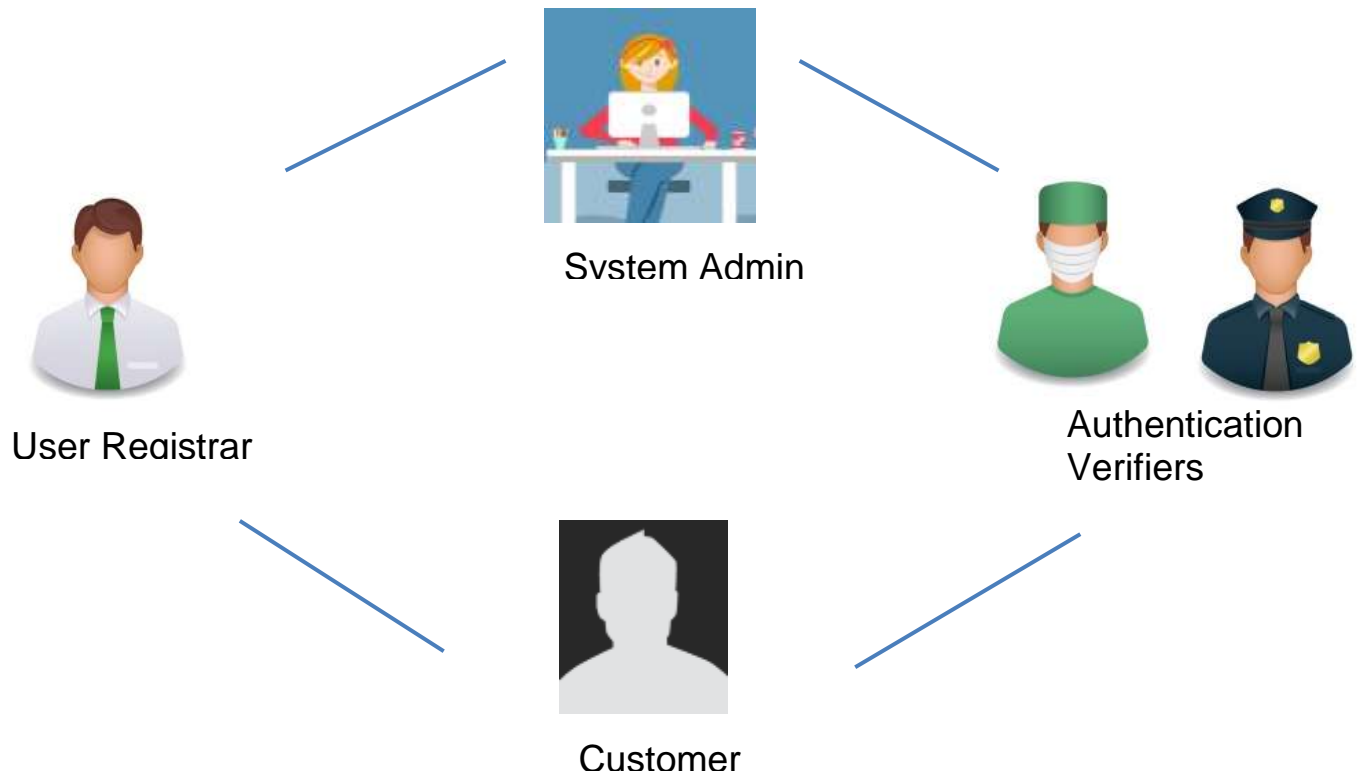


Fig. 7 Roles of Bio-Me

Bio-Me has the following roles:

- System Admin
- User Registrars
- Authentication Verifiers
- Customer

4.3 Modules of Bio-Me

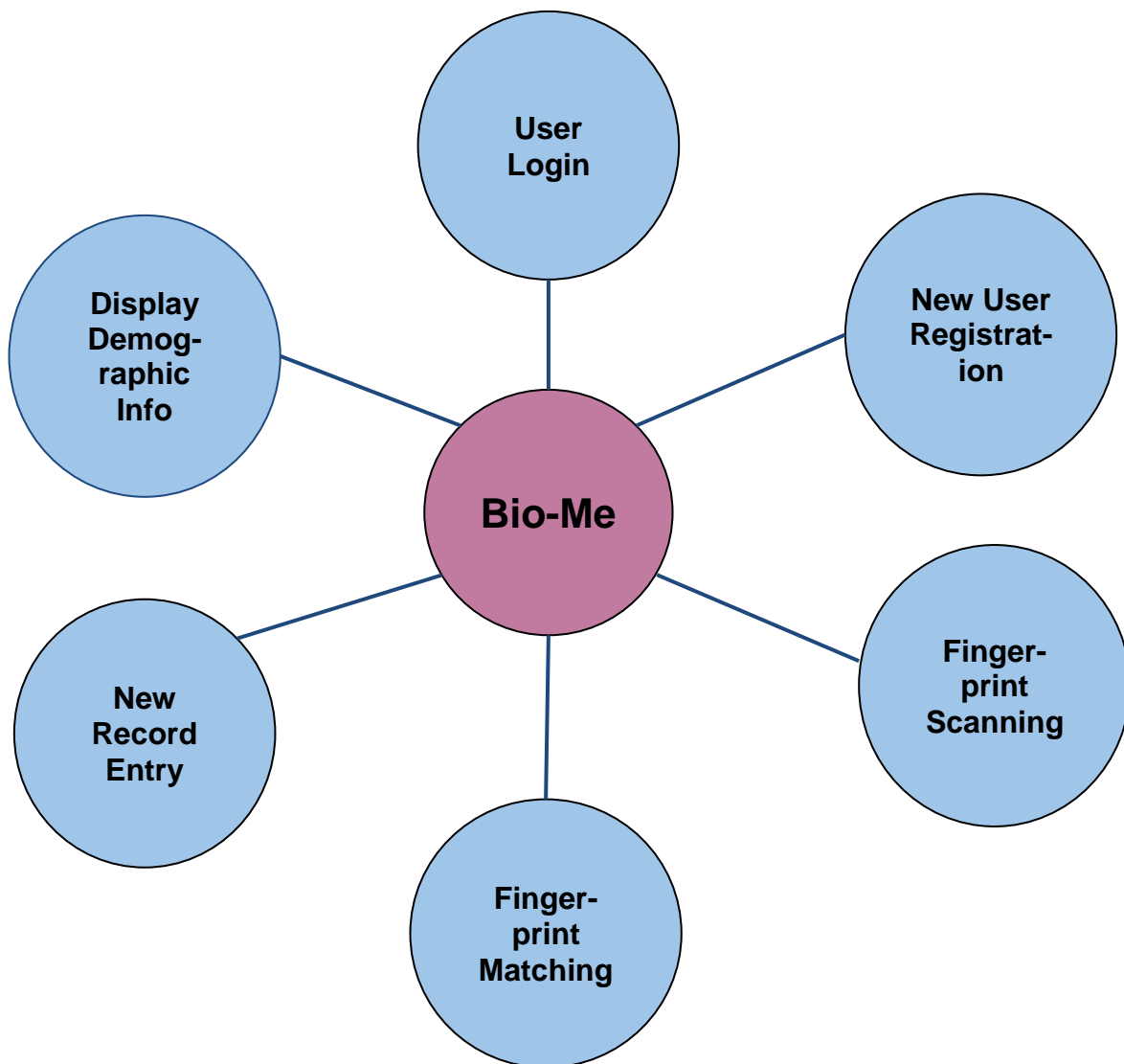


Fig. 8 Modules of Bio-Me

4.4 Use Case Diagram

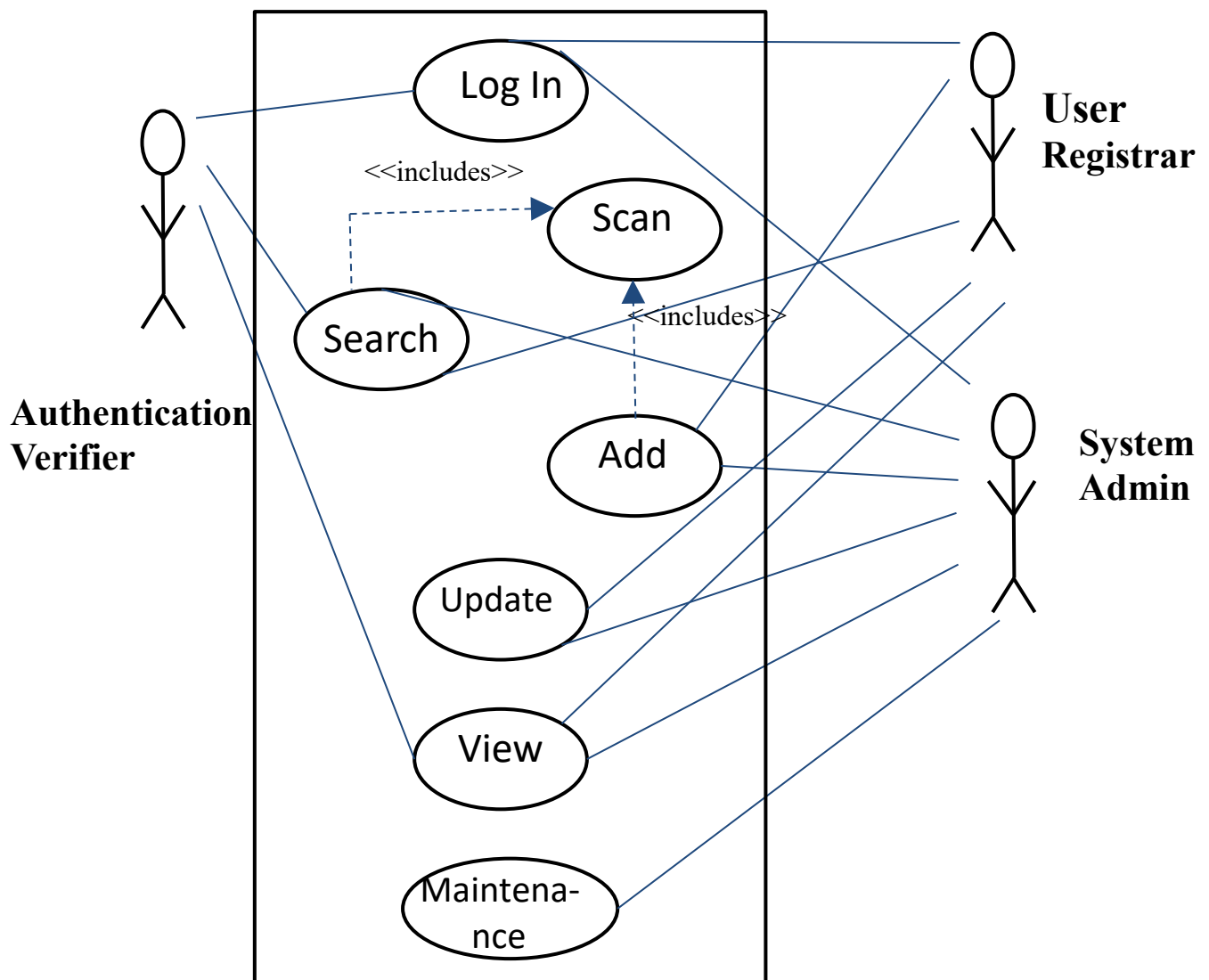


Fig. 9 Use Case Diagram

4.5 Flowchart of Bio-Me

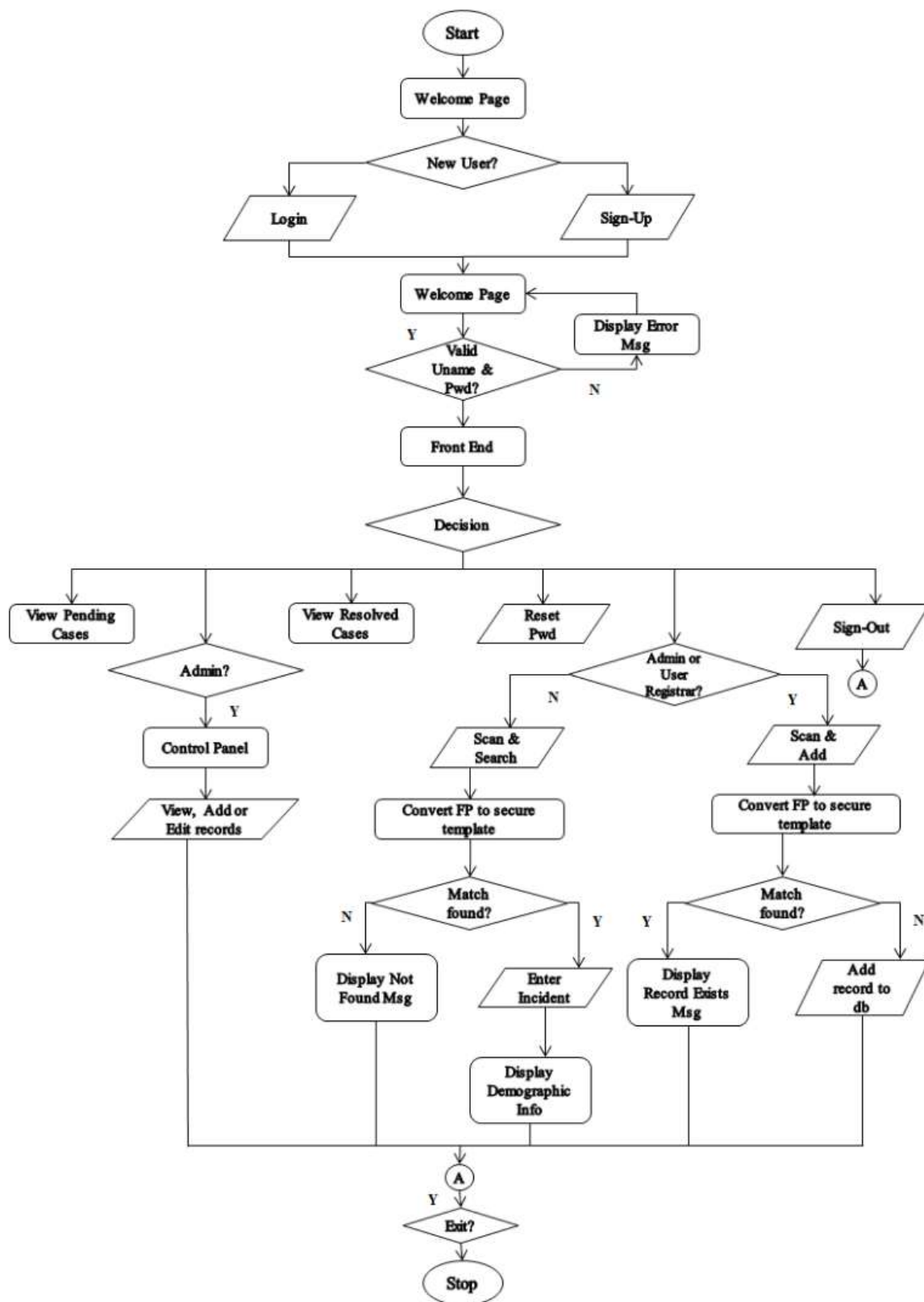


Fig. 10 Flowchart of Bio-Me

4.6 Bio-Me Database Schema

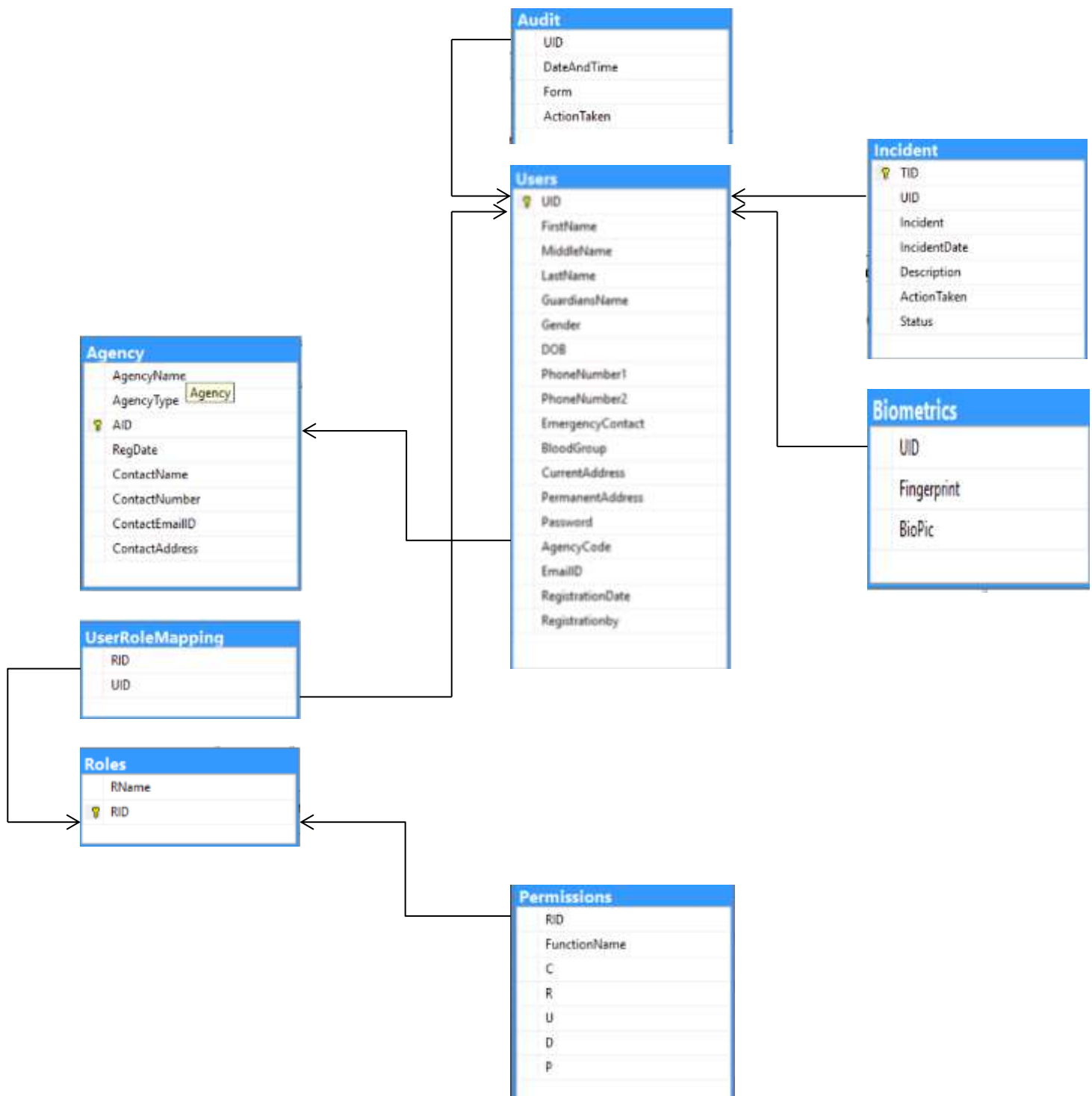


Fig. 11 Database Schema of Bio-Me

4.6 Bio-Me Database

The Bio-Me database is an SQL database, designed to have the following tables:

- Users
 - Stores the demographic information of the user.
- Biometrics
 - Stores the fingerprint and photograph of the user.
- Incident
 - Stores the kind of the kind of incident i.e. Accident, Lost Person, Runaway etc., date and time, description and action taken
- Roles
 - Stores the different roles with their IDs.
- Permissions
 - Stores the different permissions:
C-Create, R-Read, U-Update, D-Delete and P-Print.
- UserRoleMapping
 - Maps the Role IDs to the different User IDs.
- Agency
 - Stores the details of the agency registered with Bio-Me.
- Audit
 - Stores all the transaction details done in Bio-Me once the user logs on.

CHAPTER 5

IMPLEMENTATION

CHAPTER 5

IMPLEMENTATION

5.1 Gantt Chart

A Gantt chart is a graphical depiction of a project schedule. A Gantt chart is a type of bar chart that shows the start and finish dates of several elements of a project that include resources, milestones, tasks and dependencies. Henry Gantt, an American mechanical engineer, designed the Gantt chart.

The Gantt chart is the most widely used chart in project management. These charts are useful in planning a project and defining the sequence of tasks that require completion. In most instances, the chart displays as a horizontal bar chart. Horizontal bars of different lengths represent the project timeline that includes task sequences, duration, and start and end dates for each task. The horizontal bar also shows how much of a task requires completion. The horizontal bar length is proportional to the time necessary for a task's completion. In addition, the project tasks are visible on the vertical bar.

A Gantt chart aids in scheduling, managing, and monitoring specific tasks and resources in a project. The chart shows the project timeline that includes scheduled and completed work over a period. The Gantt chart aids project managers in communicating project status or plans and helps ensure the project remains on track.

The chart identifies tasks that may be executed in parallel and identifies tasks that cannot be started or finished until other tasks are complete. The Gantt chart aids in detecting potential bottlenecks and helps to identify tasks that may have been excluded from the project timeline.

The chart depicts task slack time or additional time for completion of a task that should not delay the project, non-critical activities that may be delayed and critical activities that must be executed on time. For example, if the project is installing new software on a server, the project tasks that require completion are conducting research, selecting a software, testing software and installing software. A milestone is selecting a software. These tasks appear as vertical bars on the chart. Each task takes 10 days to complete, and each task is dependent on the previous task. A critical activity is testing the software in the development and test environments. The task start and end dates, duration, and milestones appear as horizontal bars. The percentage complete for each task also displays on the horizontal bars. The project

duration is 40 days. The project start date is July 23, and the project end date is Sept. 20, which is based on workdays.

Projects of all sizes and types use Gantt charts. These charts are utilized in several industries and for a range of projects, such as building dams, bridges and highways, software development, and development of other goods and services. Project management tools, such as Microsoft Visio, Project, SharePoint or Excel, or specialized software, such as Gantto or Matchware, aid in designing Gantt charts.

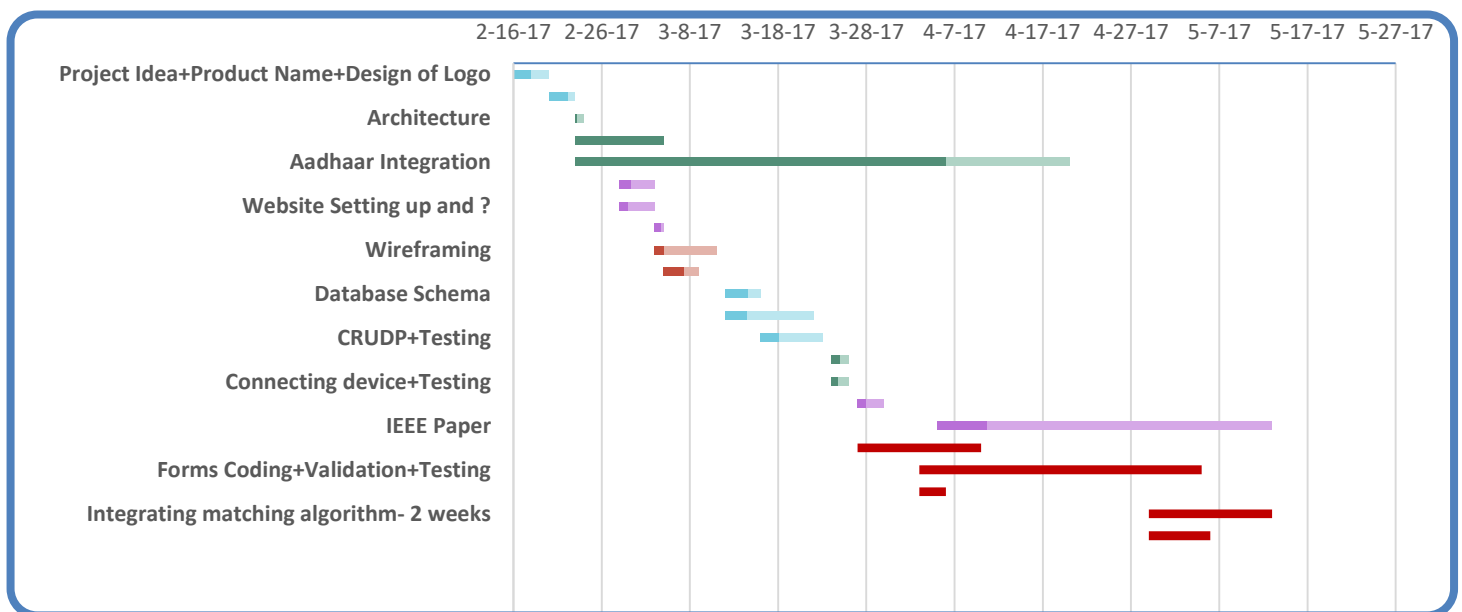


Fig. 12 Gantt chart of Bio-Me

5.2 Language Used

5.2.1 C#

C# is a modern, general-purpose, object-oriented programming language developed by Microsoft and approved by European Computer Manufacturers Association (ECMA) and International Standards Organization (ISO).

C# was developed by Anders Hejlsberg and his team during the development of .Net Framework.

C# is designed for Common Language Infrastructure (CLI), which consists of the executable code and runtime environment that allows use of various high-level languages on different computer platforms and architectures.

The following reasons make C# a widely used professional language:

- It is a modern, general-purpose programming language
- It is object oriented.
- It is component oriented.
- It is easy to learn.
- It is a structured language.
- It produces efficient programs.
- It can be compiled on a variety of computer platforms.
- It is a part of .Net Framework.

5.2.2 Features of C#:

- Portability:

By design, C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of its intrinsic types correspond to value-types implemented by the CLI framework. However, the language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime, or generate Common Intermediate Language (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or Fortran.

- Typing:

C# supports strongly typed implicit variable declarations with the keyword `var`, and implicitly typed arrays with the keyword `new[]` followed by a collection initializer.

C# supports a strict Boolean data type, `bool`. Statements that take conditions, such as `while` and `if`, require an expression of a type that implements the `true` operator, such as the Boolean type. While C++ also has a Boolean type, it can be freely converted to and from integers, and expressions such as `if(a)` require only that `a` is convertible to `bool`, allowing `a` to be an `int`, or a pointer. C# disallows this "integer meaning true or false" approach, on the grounds that forcing programmers to use expressions that return exactly `bool` can prevent certain types of programming mistakes such as `if (a = b)` (use of assignment `=` instead of equality `==`, which while not an error in C or C++, will be caught by the compiler anyway).

C# is more type safe than C++. The only implicit conversions by default are those that are considered safe, such as widening of integers. This is enforced at compile-time, during JIT, and, in some cases, at runtime. No implicit conversions occur between Booleans and integers, nor between enumeration members and integers (except for literal 0, which can be implicitly converted to any enumerated type).

The C# language does not allow for global variables or functions. All methods and members must be declared within classes. Static members of public classes can substitute for global variables and functions.

Local variables cannot shadow variables of the enclosing block, unlike C and C++.

- Meta Programming:

Meta programming via C# attributes is part of the language. Many of these attributes duplicate the functionality of GCC's and Visual C++'s platform-dependent preprocessor directives.

- Methods and functions:

Like C++, and unlike Java, C# programmers must use the keyword `virtual` to allow methods to be overridden by subclasses.

Extension methods in C# allow programmers to use static methods as if they were methods from a class's method table, allowing programmers to add methods to an object that they feel should exist on that object and its derivatives.

The type dynamic allows for run-time method binding, allowing for JavaScript-like method calls and run-time object composition.

C# has support for strongly-typed function pointers via the keyword delegate. Like the Qt framework's pseudo-C++ signal and slot, C# has semantics specifically surrounding publish-subscribe style events, though C# uses delegates to do so.

- Property:

C# provides properties as syntactic sugar for a common pattern in which a pair of methods, accessor (getter) and mutator (setter) encapsulate operations on a single attribute of a class. No redundant method signatures for the getter/setter implementations need be written, and the property may be accessed using attribute syntax rather than more verbose method calls.

- Namespace:

A C# namespace provides the same level of code isolation as a Java package or a C++ namespace, with very similar rules and features to a package.

- Memory access:

In C#, memory address pointers can only be used within blocks specifically marked as unsafe, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references, which always either point to a "live" object or have the well-defined null value; it is impossible to obtain a reference to a "dead" object (one that has been garbage collected), or to a random block of memory. An unsafe pointer can point to an instance of a value-type, array, string, or a block of memory allocated on a stack. Code that is not marked as unsafe can still store and manipulate pointers through the System.IntPtr type, but it cannot dereference them.

Managed memory cannot be explicitly freed; instead, it is automatically garbage collected. Garbage collection addresses the problem of memory leaks by freeing the programmer of responsibility for releasing memory that is no longer needed.

- Exception:

Checked exceptions are not present in C# (in contrast to Java). This has been a conscious decision based on the issues of scalability and version ability.

- Polymorphism:

Unlike C++, C# does not support multiple inheritances, although a class can implement any number of interfaces. This was a design decision by the language's lead architect to avoid complication and simplify architectural requirements throughout CLI. When implementing multiple interfaces that contain a method with the same signature, C# allows implementing each method depending on which interface that method is being called through, or, like Java, allows implementing the method once, and have that be the one invocation on a call through any of the class's interfaces.

However, unlike Java, C# supports operator overloading. Only the most commonly overloaded operators in C++ may be overloaded in C#.

- Functional programming:

Though primarily an imperative language, C# 2.0 offered limited support for functional programming through first-class functions and closures in the form of anonymous delegates. C# 3.0 expanded support for functional programming with the introduction of a lightweight syntax for lambda expressions, extension methods (an affordance for modules), and a list comprehension syntax in the form of a "query comprehension" language.

5.3 Framework

5.3.1 Microsoft .NET

The Microsoft .Net Framework is a platform that provides tools and technologies you need to build Networked Applications as well as Distributed Web Services and Web Applications. The .Net Framework provides the necessary compile time and run-time foundation to build and run any language that conforms to the Common Language Specification (CLS). The main two components of .Net Framework are Common Language Runtime (CLR) and .Net Framework Class Library (FCL).



Fig. 13 The .NET Framework

The Common Language Runtime (CLR) is the runtime environment of the .Net Framework , that executes and manages all running code like a Virtual Machine. The .Net Framework Class Library (FCL) is a huge collection of language-independent and type-safe reusable classes. The .Net Framework Class Libraries (FCL) are arranged into a logical grouping according to their functionality and usability is called Namespaces.

5.3.2 Versions of .NET:

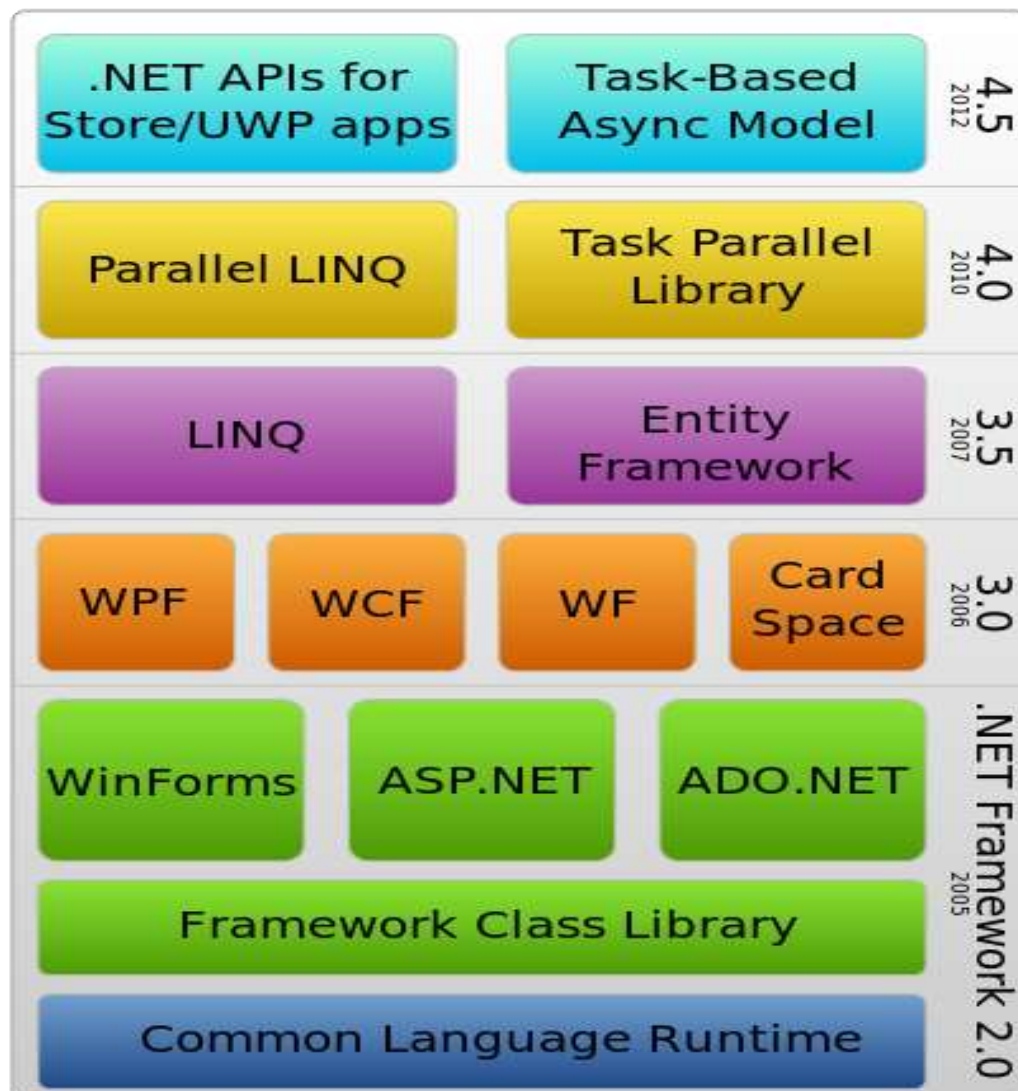


Fig. 14 .NET Versions and their services

5.3.3 Design Principles:

- Interoperability:

Because computer systems commonly require interaction between newer and older applications, .NET Framework provides means to access functions implemented in newer and older programs that execute outside .NET environment.

- Language independence:

.NET Framework introduces a Common Type System (CTS) that defines all possible data types and programming constructs supported by CLR and how they may or may not interact with each other conforming to CLI specification. Because of this feature, .NET Framework supports the exchange of types and object instances between libraries and applications written using any conforming .NET language.

- Type safety:

CTS and the CLR used in .NET Framework also enforce type safety. This prevents ill-defined casts, wrong method invocations, and memory size issues when accessing an object. This also makes most CLI languages statically typed (with or without type inference). However, starting with .NET Framework 4.0, the Dynamic Language Runtime extended the CLR, allowing dynamically typed languages to be implemented atop the CLI.

- Portability:

While Microsoft has never implemented the full framework on any system except Microsoft Windows, it has engineered the framework to be cross-platform, and implementations are available for other operating system. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

- Security:

.NET Framework has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. CAS is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the Internet). CAS uses evidence to determine the permissions granted to the code. Other code can demand that calling code be granted a specified permission. The demand causes CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

- Memory Management:

CLR frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely

freed. Instantiations of .NET types (objects) are allocated from the managed heap; a pool of memory managed by CLR. As long as a reference to an object exists, which may be either direct, or via a graph of objects, the object is considered to be in use. When no reference to an object exists, and it cannot be reached or used, it becomes garbage, eligible for collection.

.NET Framework includes a garbage collector (GC) which runs periodically, on a separate thread from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them. It is a non-deterministic, compacting, mark-and-sweep garbage collector. GC runs only when a set amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, GC runs are non-deterministic.

GC used by .NET Framework is also generational. Objects are assigned a generation. Newly created objects are tagged Generation 0. Objects that survive a garbage collection are tagged Generation 1. Generation 1 objects that survive another collection are Generation 2. The framework uses up to Generation 2 objects. Higher generation objects are garbage collected less often than lower generation objects. This raises the efficiency of garbage collection, as older objects tend to have longer lifetimes than newer objects.[55] Thus, by eliminating older (and thus more likely to survive a collection) objects from the scope of a collection run, fewer objects need checking and compacting.

- Performance:

When an application is first launched, the .NET Framework compiles the CIL code into executable code using its just-in-time compiler, and caches the executable program into the .NET Native Image Cache. Due to caching, the application launches faster for subsequent launches, although the first launch is usually slower. To speed up the first launch, developers may use the Native Image Generator utility to manually ahead-of-time compile and cache any .NET application.

The garbage collection, which is integrated into the environment, can introduce unanticipated delays of execution over which the developer has little direct control. "In large applications, the number of objects that the garbage collector needs to work with can become very large, which means it can take a very long time to visit and rearrange all of them."

5.4 User Interface

5.4.1 Forms

A form is a bit of screen real estate, usually rectangular, that you can use to present information to the user and to accept input from the user. Forms can be standard windows, multiple document interface (MDI) windows, dialog boxes, or display surfaces for graphical routines. The easiest way to define the user interface for a form is to place controls on its surface. Forms are objects that expose properties which define their appearance, methods which define their behavior, and events which define their interaction with the user. By setting the properties of the form and writing code to respond to its events, you customize the object to meet the requirements of your application.

As with all objects in the .NET Framework, forms are instances of classes.

5.4.2 WinForms

Windows Forms is a set of managed libraries in .NET Framework designed to develop rich client applications. It is a graphical API to display data and manage user interactions with easier deployment and better security in client applications. Windows Forms offers an extensive client library providing interface to access native Windows graphical interface elements and graphics from managed code. It is built with event-driven architecture similar to Windows clients and hence, its applications wait for user input for its execution. Windows Forms is similar to Microsoft Foundation Class (MFC) library in developing client applications. It provides a wrapper consisting of a set of C++ classes for development of Windows applications. However, it does not provide a default application framework like the MFC.

Every control in Windows Forms application is a concrete instance of a class. The layout of a control in the GUI and its behavior are managed using methods and accessors. Windows Forms provides a variety of controls, such as text-boxes, buttons, and web pages along with options to create custom controls. It also contains classes for creating brushes, fonts, icons, and other graphic objects (like line and circle). Windows Forms Designer is a tool, in Visual Studio.NET, used to insert controls in a form and arrange them as per desired layout, with provision for adding code to handle their events, which implement user interactions.

Tabular data that is bound to XML, database, etc. can be displayed using DataGrid View control in the form of rows and cells. Application settings is another feature of Windows Forms to create, store, and maintain runtime state information in an XML form that can be used to retrieve the user-preferred settings, such as toolbar positions and most-recently used lists. These settings can be reused in a future application.

5.4.3 Bio-Me Forms

- frmControlPanel
- frmFrontend
- frmIdentityInformation
- frmPendingCases
- frmResolvedCases
- frmResetPassword
- frmSettings
- frmUserRegistration
- frmWelcomePage

5.5 Microsoft Visual Studio

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows, as well as web sites, web apps, web services and mobile apps.

Visual Studio uses Microsoft software development platforms such as Windows API, Windows Forms, Windows Presentation Foundation, Windows Store and Microsoft Silverlight. It can produce both native code and managed code. Visual Studio includes a code editor supporting IntelliSense (the code completion component) as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a code profiler, forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source control systems (like Subversion) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer). Visual Studio supports 36 different programming languages and allows the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C, C++ and C++/CLI (via Visual C++), VB.NET (via Visual Basic .NET), C# (via Visual C#), F# (as of Visual Studio 2010) and TypeScript (as of Visual Studio 2013 Update 2). Support for other languages such as Python,[8] Ruby, Node.js, and M among others is available via language services installed separately.

It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Java (and J#) were supported in the past. Microsoft provides a free version of Visual Studio called the Community edition that supports plugins and is available at no cost.

5.5.1 Designer

Visual Studio includes a host of visual designers to aid in the development of applications. These tools include:

- Windows Forms Designer:

The Windows Forms designer is used to build GUI applications using Windows Forms. Layout can be controlled by housing the controls inside other containers or locking them to the side of the form. Controls that display data (like textbox, list box and grid view) can be bound to data sources like databases or queries. Data-bound controls can be created by dragging items from the Data Sources window onto a design surface.[31] The UI is linked with code using an event-driven programming model. The designer generates either C# or VB.NET code for the application.

- WPF Designer:

The WPF designer, codenamed Cider,[32] was introduced with Visual Studio 2008. Like the Windows Forms designer it supports the drag and drop metaphor. It is used to author user interfaces targeting Windows Presentation Foundation. It supports all WPF functionality including data binding and automatic layout management. It generates XAML code for the UI. The generated XAML file is compatible with Microsoft Expression Design, the designer-oriented product. The XAML code is linked with code using a code-behind model.

- Web designer/development:

Visual Studio also includes a web-site editor and designer that allows web pages to be authored by dragging and dropping widgets. It is used for developing ASP.NET applications and supports HTML, CSS and JavaScript. It uses a code-behind model to link with ASP.NET code. From Visual Studio 2008 onwards, the layout engine used by the web designer is shared with Microsoft Expression Web. There is also ASP.NET MVC support for MVC technology as a separate download[33] and ASP.NET Dynamic Data project available from Microsoft.[34]

- Class designer:

The Class Designer is used to author and edit the classes (including its members and their access) using UML modeling. The Class Designer can generate C# and VB.NET code

outlines for the classes and methods. It can also generate class diagrams from hand-written classes.

- Data designer:

The data designer can be used to graphically edit database schemas, including typed tables, primary and foreign keys and constraints. It can also be used to design queries from the graphical view.

- Mapping designer:

From Visual Studio 2008 onwards, the mapping designer is used by LINQ to SQL to design the mapping between database schemas and the classes that encapsulate the data. The new solution from ORM approach, ADO.NET Entity Framework, replaces and improves the old technology.

5.5.2 Debugger

Visual Studio includes a debugger that works both as a source-level debugger and as a machine-level debugger. It works with both managed code as well as native code and can be used for debugging applications written in any language supported by Visual Studio. In addition, it can also attach to running processes and monitor and debug those processes. If source code for the running process is available, it displays the code as it is being run. If source code is not available, it can show the disassembly. The Visual Studio debugger can also create memory dumps as well as load them later for debugging. Multi-threaded programs are also supported. The debugger can be configured to be launched when an application running outside the Visual Studio environment crashes.

The debugger allows setting breakpoints (which allow execution to be stopped temporarily at a certain position) and watches (which monitor the values of variables as the execution progresses). Breakpoints can be conditional, meaning they get triggered when the condition is met. Code can be stepped over, i.e., run one line (of source code) at a time. It can either step into functions to debug inside it, or step over it, i.e., the execution of the function body isn't available for manual inspection. The debugger supports Edit and Continue, i.e., it allows code to be edited as it is being debugged.

When debugging, if the mouse pointer hovers over any variable, its current value is displayed in a tooltip ("data tooltips"), where it can also be modified if desired. During coding, the Visual Studio debugger lets certain functions be invoked manually from the Immediate tool window. The parameters to the method are supplied at the Immediate window.

5.6 SQL

Structured Query Language (SQL) is a domain-specific language used in programming and designed for managing data held in a relational database management system (RDBMS), or for stream processing in a relational data stream management system (RDSMS).

Originally based upon relational algebra and tuple relational calculus, SQL consists of a data definition language, data manipulation language, and data control language. The scope of SQL includes data insert, query, update and delete, schema creation and modification, and data access control. Although SQL is often described as, and to a great extent is, a declarative language (4GL), it also includes procedural elements.

SQL was one of the first commercial languages for Edgar F. Codd's relational model, as described in his influential 1970 paper, "A Relational Model of Data for Large Shared Data Banks." Despite not entirely adhering to the relational model as described by Codd, it became the most widely used database language.

SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987. Since then, the standard has been revised to include a larger set of features. Despite the existence of such standards, most SQL code is not completely portable among different database systems without adjustments.

5.6.1 Language elements

The SQL language is subdivided into several language elements, including:

- Clauses, which are constituent components of statements and queries. (In some cases, these are optional.
- Expressions, which can produce either scalar values, or tables consisting of columns and rows of data
- Predicates, which specify conditions that can be evaluated to SQL three-valued logic (3VL) (true/false/unknown) or Boolean truth values and are used to limit the effects of statements and queries, or to change program flow.
- Queries, which retrieve the data based on specific criteria. This is an important element of SQL.
- Statements, which may have a persistent effect on schemata and data, or may control transactions, program flow, connections, sessions, or diagnostics.

- SQL statements also include the semicolon (";") statement terminator. Though not required on every platform, it is defined as a standard part of the SQL grammar.

5.6.2 Advantages of SQL

SQL is widely popular because it offers the following advantages –

- Allows users to access data in the relational database management systems.
- Allows users to describe the data.
- Allows users to define the data in a database and manipulate that data.
- Allows to embed within other languages using SQL modules, libraries & pre-compilers.
- Allows users to create and drop databases and tables.
- Allows users to create view, stored procedure, functions in a database.
- Allows users to set permissions on tables, procedures and views.

5.7 Bio-Me Operating Instructions:

1. After installing the Bio-Me app, double click on the icon to open it.
2. The Welcome Page opens.
3. Click on Sign-Up. User Registration Form opens.
4. Fill up the required details. Click on browse to upload a profile picture. (Size of Photograph: Passport Size)
5. Click on Submit.
6. The Welcome Screen reopens.
7. Type your Username (Aadhar Number) and Password and click Login.
8. Your Home Page opens.
9. If you are an Authentication Verifier and want to verify a person, connect the Scanner to your PC and click on Scan and Search.
10. Select the incident and give a brief description of the same.
11. Click on OK, after which the Identity Information of the person is displayed.
12. If you are a User Registrar, to add a new person to the database, click on Add after connecting the scanner.
13. Fill in the user details and scan the fingerprint
14. To view Pending Cases, Click on View Pending Cases.
15. To update a Pending Case 'P' to Resolved 'R', click on the record to be updated and make the change.
16. To view Resolved Cases, Click on View Resolved Cases.
17. To reset your password, click on Reset Password.
18. Fill in your Username and New password and click Reset. A mail will be sent to your email account with your credentials.
19. Click on Sign out, to sign out of the app.

CHAPTER 6

SOURCE CODE

CHAPTER 6

SOURCE CODE

6.1 User Registration:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Globalization;
using System.Text.RegularExpressions;
using System.Collections;
using System.IO;
using NITGEN.SDK.NBioBSP;
using System.Drawing.Imaging;

namespace biometest
{
    public partial class frmUserRegistration1 : Form
    {
        //-----Inputs from user-----

        #region declarations
        NBioAPI m_NBioAPI;
        NBioAPI.Export m_Export;
        string strfName = string.Empty;
        string strmName = string.Empty;
```

```
string strName = string.Empty;
string strGender = string.Empty;
DateTime dtDOB = DateTime.MinValue;
string strBloodGroup = string.Empty;
string strPaOrGuardianName = string.Empty;
string strEmailId = string.Empty;
string strPhoneNum1 = string.Empty;//14 characters with symbols
string strPhoneNum2 = string.Empty;//14 characters with symbols
string strEmerNum = string.Empty;//14 characters with symbols
string strAadhaarNum = string.Empty;//12 characters
string strCurAdd = string.Empty;
string strPermAdd = string.Empty;
string strAgencyCode = string.Empty;
string strUserType = string.Empty;
string strPassword = string.Empty;
string strConfirmPwd = string.Empty;
Image imgContact = null;
string strFingerPrint = string.Empty;
int lengthoffingerprintarr = 0;
byte[] bytearrFingerPrint = null;
Image Compare = null;

#endregion

SqlConnection conn = new SqlConnection("Data Source=DESKTOP-UO0PO1Q;Initial
    Catalog=Biome;Integrated Security=True");

public frmUserRegistration1()
{
    m_NBioAPI = new NBioAPI();
    m_Export = new NBioAPI.Export(m_NBioAPI);
    InitializeComponent();
}
```

```
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    NBioAPI.Type.VERSION version = new NBioAPI.Type.VERSION();
    m_NBioAPI.GetVersion(out version);

    Text = String.Format("Export Demo for C#.NET (BSP Version : v{0}.{1:D4})",
        version.Major, version.Minor);

    SqlCommand cmd = new SqlCommand("SELECT AgencyName FROM Agency",
        conn);

    conn.Open();

    SqlDataReader reader = cmd.ExecuteReader();

    AutoCompleteStringCollection MyCollection = new
        AutoCompleteStringCollection();

    while (reader.Read())
    {
        MyCollection.Add(reader.GetString(0));
    }

    txtAgencyCode.AutoCompleteCustomSource = MyCollection;

    conn.Close();

    conn.Open();

    string SqlString = "Select * From Roles Where RName Is not Null";

    SqlCommand cmdRoles = new SqlCommand(SqlString);

    SqlDataAdapter sda = new SqlDataAdapter();

    cmdRoles.Connection = conn;

    sda.SelectCommand = cmdRoles;

    DataTable dt = new DataTable();

    sda.Fill(dt);

    foreach (DataRow row in dt.Rows)
```

```
{
    clbUserType.Items.Add(row["RName"]);
}
//}

conn.Close();

Compare = picProfilePic.Image;
}

//-----On scan-----

private void btnScan_Click(object sender, EventArgs e)
{
    m_NBioAPI.OpenDevice(NBioAPI.Type.DEVICE_ID.AUTO);
    NBioAPI.Type.HFIR capturedFIR;
    uint ret = m_NBioAPI.Capture(out capturedFIR);
    m_NBioAPI.CloseDevice(NBioAPI.Type.DEVICE_ID.AUTO);
    if (ret != NBioAPI.Error.NONE)
    {
        MessageBox.Show("Failed to capture fingerprint.");
        return;
    }

    NBioAPI.Type.MINCONV_DATA_TYPE exportType;
    exportType = NBioAPI.Type.MINCONV_DATA_TYPE.MINCONV_TYPE_FDU;
    NBioAPI.Export.EXPORT_DATA exportData;
    ret = m_Export.NBioBSPToFDx(capturedFIR, out exportData, exportType);
    if (ret != NBioAPI.Error.NONE)
    {
        MessageBox.Show("Failed to export data.");
        return;
    }
}
```

```
for (int f = 0; f < exportData.FingerNum; f++)
{
    for (int s = 0; s < exportData.SamplesPerFinger; s++)
    {
        bytarrFingerPrint = exportData.FingerData[f].Template[s].Data;
    }
}

lengthoffingerprintarr = bytarrFingerPrint.Length;
}

//-----On submitting the form-----

private void btnSubmit_Click(object sender, EventArgs e)
{
    int count = -1;
    string datetime;
    while (count != 0 )
    {
        datetime = dateTimeDOB.Value.ToString("yyyy-MM-dd");
        dtDOB = Convert.ToDateTime(datetime);
        count = 0;
        //validating DOB
        if (dtDOB == null)
        {
            MessageBox.Show("Please Re-enter the DOB");
            lblStatus.Text = "Invalid DOB";
            dateTimeDOB.Text = "";
            dateTimeDOB.Focus();
            count += -1;
            return;
        }
    }
}
```

```
else
{
    count += 0;
}

if (string.IsNullOrEmpty(txtFName.Text))
{
    //-----Validating First Name-----

    lblStatus.Text = "You must provide your name!";
    MessageBox.Show("You must provide your name!");
    count += -1;
    txtFName.Focus();
    return;
}

else
{
    count += 0;
}

//-----Validating Last Name-----

if (string.IsNullOrEmpty(txtLName.Text))
{
    //This control fails validation: Name cannot be empty.
    lblStatus.Text = "You must provide your last name!";
    MessageBox.Show("You must provide your last name!");
    count += -1;
    txtLName.Focus();
    return;
}

else
{
```



```
        count += 0;
    }

    //Validating UID

    strAadhaarNum=Convert.ToString(txtAdrNum1.Text)+Convert.ToString(txtAadharNum2.Text)+Convert.ToString(txtAadharNum3.Text);

    Regex re3 = new Regex(@"^\d{12}$");
    if (!re3.IsMatch(strAadhaarNum))
    {
        lblStatus.Text = "Please enter in xxxx xxxx xxxx format";
        MessageBox.Show("Please enter in xxxx xxxx xxxx format");
        txtAdrNum1.Clear();
        txtAadharNum3.Clear();
        txtAadharNum2.Clear();
        txtAdrNum1.Focus();
        count +=1;
        return;
    }
    else
    {
        count += 0;
    }

    //-----Validating Mail ID-----

    if (!String.IsNullOrEmpty(Convert.ToString(txtEmailID.Text)))
    {
        Regex re4 = new Regex(@"^s*[\w\-\+_\']+(\.[\w\-\+_\']+)*@[A-Za-z0-9]([\w\.-]*[A-Za-z0-9])?\.?[A-Za-z][A-Za-z\.]*[A-Za-z]$");
        //or Regex re4 = new Regex(@"^\\w+([-\+.']\\w+)*@\\w+([-\+.']\\w+)*\\.\\w+([-\+.']\\w+)*");
        //check only if Email ID is entered as it can be null

        if (!re4.IsMatch(txtEmailID.Text))
        {

```

```
        lblStatus.Text = "Please enter valid email ID";
        MessageBox.Show("Please enter valid email ID");
        txtEmailID.Text = "";
        txtEmailID.Focus();
        count += -1;
        return;
    }

    else
    {
        count += 0;
    }

}

else
{
    count += 0;
}

//-----Validating Gender-----

if (chkMale.Checked == true)
{
    strGender = "Male";
    count += 0;
}

else if (chkFemale.Checked == true)
{
    strGender = "Female";
    count += 0;
}
```

```
else if (chkOthers.Checked == true)
{
    strGender = "Others";
    count += 0;
}
else
{
    lblStatus.Text = "You must provide your gender!";
    MessageBox.Show("You must provide your gender!");
    count += -1;
    return;
}

//-----Validating BloodGroup-----

if((String.IsNullOrEmpty(Convert.ToString(txtNegative.Text))) &&
    (String.IsNullOrEmpty(Convert.ToString(txtPositive.Text))))
{
    lblStatus.Text = "You must provide Valid Blood Group!";
    MessageBox.Show("You must provide Valid Blood Group!");
    count += -1;
    return;
}
else
{
    count += 0;
}

Regex re = new Regex(@"^\d{10}$");

//-----Validating Phone Number 1-----

if (!re.IsMatch(txtPhNum1.Text))
{
    lblStatus.Text = "Please enter 10 digit Phone Number ";
```

```
        MessageBox.Show("Please enter valid 10 digit Phone Number");

        txtPhNum1.Text = "";

        txtPhNum1.Focus();

        count += -1;

        return;
    }

    else

    {

        count += 0;

    }

    //-----Validating Phone Number2-----

    Regex re1 = new Regex(@"^\d{10}$");

    if (!String.IsNullOrEmpty(Convert.ToString(txtPhNum2.Text)))

    {

        if (!re1.IsMatch(txtPhNum2.Text))

        {

            lblStatus.Text = "Please enter 10 digit Phone Number";

            MessageBox.Show("Please enter valid 10 digit Phone Number");

            txtPhNum2.Text = "";

            txtPhNum2.Focus();

            count += -1;

            return;

        }

        else

        {

            count += 0;

        }

    }

    //-----Valdiating Emergency Contact-----
```

```
if (!String.IsNullOrEmpty(Convert.ToString(txtEmCon.Text)))
{
    if (!re1.IsMatch(txtEmCon.Text))
    {
        lblStatus.Text = "Please enter 10 digit Phone Number";
        MessageBox.Show("Please enter valid 10 digit Phone Number");
        txtEmCon.Text = "";
        txtEmCon.Focus();
        count += -1;
        return;
    }
    else
    {
        count += 0;
    }
}

//-----Validating Current Address-----
if (string.IsNullOrEmpty(txtCurAdd.Text))
{
    //This control fails validation: Name cannot be empty.
    lblStatus.Text = "You must provide your current address!";
    MessageBox.Show("You must provide your current address!");
    txtCurAdd.Text = "";
    txtCurAdd.Focus();
    count += -1;
    return;
}
else
{

```

```
        count += 0;
    }

    //-----Validating User Type-----

    if(clbUserType.SelectedItem == null)
    {
        lblStatus.Text = "You must provide your UserType!";
        MessageBox.Show("You must provide your UserType!");
        count += -1;
        return;
    }
    else
    {
        count += 0;
    }

    if (clbUserType.SelectedItem.ToString() != "Customer")
    {
        //-----Valdiating Password-----

        if (string.IsNullOrEmpty(txtPassword.Text))
        {
            //This control fails validation: Name cannot be empty.
            lblStatus.Text = "You must provide password!";
            MessageBox.Show("You must provide password!");
            txtPassword.Focus();
            count += -1;
            return;
        }
        else
        {

```

```
        count += 0;
    }

    if (string.IsNullOrEmpty(txtConPassword.Text))
    {
        lblStatus.Text = "You must re-enter password!";
        MessageBox.Show("You must re-enter password!");
        txtConPassword.Focus();
        count += -1;
        return;
    }
    else
    {
        count += 0;
    }
    if(txtConPassword.Text == txtPassword.Text)
    {
        count += 0;
    }
    else
    {
        lblStatus.Text = "the password and confirm password field must
            have the same value!";
        MessageBox.Show("the password and confirm password field must
            have the same value!");
        txtConPassword.Clear();
        txtPassword.Clear();
        count += -1;
        return;
    }
}
```

```
}  
else  
{  
    //-----Validate Fingerprint-----  
    if (lengthoffingerprintarr == 0)  
    {  
        lblStatus.Text = "You must Scan fingerprint";  
        MessageBox.Show("You must Scan fingerprint");  
        count += -1;  
        return;  
    }  
    else  
    {  
        count += 0;  
    }  
  
    //-----Validate Photo-----  
    if (picProfilePic.Image.Equals(Compare))  
    {  
        lblStatus.Text = "You must Give the photo";  
        MessageBox.Show("You must Scan The Photo");  
        count += -1;  
        return;  
    }  
    else  
    {  
        count += 0;  
    }  
}
```



```
}

strfName = Convert.ToString(txtFName.Text);
strmName = Convert.ToString(txtMName.Text);
strlName = Convert.ToString(txtLName.Text);
if (chkMale.Checked == true)
{
    strGender = "Male";
}
else if (chkFemale.Checked == true)
{
    strGender = "Female";
}
else if(chkOthers.Checked == true)
{
    strGender = "Others";
}

datetime = dateTimeDOB.Value.ToString("yyyy-MM-dd");
dtDOB = Convert.ToDateTime(datetime);
if (!String.IsNullOrEmpty(Convert.ToString(txtPositive.Text)))
{
    strBloodGroup = Convert.ToString(txtPositive.Text) + "+ve";
}
else if (!String.IsNullOrEmpty(Convert.ToString(txtNegative.Text)))
{
    strBloodGroup = Convert.ToString(txtNegative.Text) + "-ve";
}

strPaOrGuardianName = Convert.ToString(txtParentName.Text);
strEmailId = Convert.ToString(txtEmailID.Text);
strPhoneNum1 = Convert.ToString(txtPhNum1.Text);
```

```
strPhoneNum2 = Convert.ToString(txtPhNum2.Text);

strEmerNum = Convert.ToString(txtEmCon.Text);

strAadhaarNum = Convert.ToString(txtAdrNum1.Text) +
    Convert.ToString(txtAadhaarNum2.Text) + Convert.ToString(txtAadhaarNum3.Text);

strCurAdd = Convert.ToString(txtCurAdd.Text);

strPermAdd = Convert.ToString(txtPermAdd.Text);

strUserType = Convert.ToString(clbUserType.SelectedItem.ToString());

strPassword = Convert.ToString(txtPassword.Text);

strConfirmPwd = Convert.ToString(txtConPassword.Text);

strAgencyCode = Convert.ToString(txtAgencyCode.Text);

string sqlAaadhaar = "SELECT * FROM USERS WHERE UID=@stradrnum";

try
{
    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
        UO0PO1Q;Initial Catalog=Biome;Integrated Security=True"))
    {
        SqlCommand cmd = new SqlCommand(sqlAaadhaar, con);

        con.Open();

        cmd.Parameters.Add("@stradrnum", SqlDbType.NVarChar).Value
            =strAadhaarNum;

        DataTable dt = new DataTable();

        SqlDataAdapter sda = new SqlDataAdapter();

        sda.SelectCommand = cmd;

        cmd.ExecuteNonQuery();

        sda.Fill(dt);

        con.Close();

        int count1 = 0;

        foreach(DataRow row in dt.Rows)
        {
            count1++;
        }
    }
}
```

```
        if(count1>0)
        {
            MessageBox.Show("Aadhhar number already exists");
        }
    }
}

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

try
{
    if (clbUserType.SelectedItem.ToString() == "Customer")
    {
        strAgencyCode = null;
    }

    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
        UO0PO1Q;Initial Catalog=Biome;Integrated Security=True"))
    {
        using (SqlCommand cmd = new SqlCommand("User_profile_Reg1", con))
        {
            cmd.CommandType = CommandType.StoredProcedure;

            cmd.Parameters.Add("@AdhaarNum", SqlDbType.NVarChar).Value =
                strAadhaarNum;

            cmd.Parameters.Add("@firstname", SqlDbType.NVarChar).Value =
                strfName;

            cmd.Parameters.Add("@middlename", SqlDbType.NVarChar).Value =
                strmName;

            cmd.Parameters.Add("@Lastname", SqlDbType.NVarChar).Value =
                strlName;
```

```
cmd.Parameters.Add("@GuardiansName", SqlDbType.NVarChar).Value =
    strPaOrGuardianName;

cmd.Parameters.Add("@Gender", SqlDbType.NVarChar).Value =
    strGender;

cmd.Parameters.Add("@DOB", SqlDbType.DateTime).Value = dtDOB;

cmd.Parameters.Add("@PhoneNumber1", SqlDbType.NVarChar).Value =
    strPhoneNum1;

cmd.Parameters.Add("@PhoneNumber2", SqlDbType.NVarChar).Value =
    strPhoneNum2;

cmd.Parameters.Add("@EmergencyContact", SqlDbType.NVarChar).Value
    = strEmerNum;

cmd.Parameters.Add("@BloodGroup", SqlDbType.NVarChar).Value =
    strBloodGroup;

cmd.Parameters.Add("@CurrentAddress", SqlDbType.NVarChar).Value =
    strCurAdd;

cmd.Parameters.Add("@PermanentAddress", SqlDbType.NVarChar).Value
    = strPermAdd;

cmd.Parameters.Add("@Password", SqlDbType.NVarChar).Value =
    strPassword;

cmd.Parameters.Add("@AgencyCode", SqlDbType.NVarChar).Value =
    txtAgencyID.Text;

cmd.Parameters.Add("@EmailID", SqlDbType.NVarChar).Value =
    strEmailId;

cmd.Parameters.Add("@RegistrationBy", SqlDbType.NVarChar).Value =
    strAgencyCode;

cmd.Parameters.Add("@RName", SqlDbType.NVarChar).Value =
    clbUserType.SelectedItem.ToString();

//cmd.Parameters.Add("@operation", SqlDbType.NVarChar).Value =
    "Insert";

con.Open();

cmd.ExecuteNonQuery();

con.Close();
}
}
```

```
}  
catch (Exception ex)  
{  
  
}  
  
MemoryStream ms2 = new MemoryStream();  
picProfilePic.Image.Save(ms2, ImageFormat.Jpeg);  
byte[] photo_array2 = new byte[ms2.Length];  
  
if (lengthoffingerprintarr != 0 && photo_array2.Length != 0)  
{  
    try  
    {  
        using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-  
            UO0PO1Q;Initial Catalog=Biome;Integrated Security=True"))  
        {  
            using (SqlCommand cmd = new SqlCommand("SP_Reg_On_submit", con))  
            {  
                cmd.CommandType = CommandType.StoredProcedure;  
                cmd.Parameters.Add("@UID", SqlDbType.NVarChar).Value =  
                    strAadhaarNum;  
                cmd.Parameters.Add("@fingerprint", SqlDbType.VarBinary,  
                    lengthoffingerprintarr).Value = bytearrayFingerPrint;  
                // cmd.Parameters.Add("@Biopic", SqlDbType.Image).Value =  
                    imgContact;  
                MemoryStream ms = new MemoryStream();  
                picProfilePic.Image.Save(ms, ImageFormat.Jpeg);  
                byte[] photo_array = new byte[ms.Length];  
                ms.Position = 0;  
                ms.Read(photo_array, 0, photo_array.Length);
```

```
        cmd.Parameters.Add("@Biopic", SqlDbType.Image).Value =
            photo_array;

        con.Open();

        cmd.ExecuteNonQuery();

        con.Close();

    }

}

Form frm1 = new frmWelcomePage();

this.Close();

frm1.Show();

}

catch (Exception ex)

{

    MessageBox.Show(ex.Message);

}

}

frmWelcomePage frmWelcomePage = new frmWelcomePage();

this.Close();

frmWelcomePage.Show();

}

private void checkedListBox1_SelectedIndexChanged(object sender, EventArgs e)

{

    if(clbUserType.SelectedItems.Count>1)

    {

        MessageBox.Show("Only one item can be selected");

        return;

    }

    else if(clbUserType.SelectedItem.ToString()=="Customer")

    {

        txtAgencyCode.Visible = false;
```

```
        txtAgencyAddress.Visible = false;
        txtAgencyID.Visible = false;
        txtAgencyPhNum.Visible = false;
        txtAgencyEmailID.Visible = false;
        txtPassword.Visible = false;
        txtConPassword.Visible = false;
    }
    else
    {
        txtAgencyCode.Visible = true;
        txtAgencyAddress.Visible = true;
        txtAgencyID.Visible = true;
        txtAgencyPhNum.Visible = true;
        txtAgencyEmailID.Visible = true;
        txtPassword.Visible = true;
        txtConPassword.Visible = true;
    }
}

private void btnCancel_Click(object sender, EventArgs e)
{
    Form frm3 = new frmWelcomePage();
    this.Hide();
    frm3.ShowDialog();

}

//-----File dialog to store pictues-----

private void btnBrowse_Click(object sender, EventArgs e)
{

```

```
// open file dialog
OpenFileDialog open = new OpenFileDialog();

// image filters
open.Filter = "Image Files(*.jpg; *.jpeg; *.gif; *.bmp)|*.jpg; *.jpeg; *.gif; *.bmp";

if (open.ShowDialog() == DialogResult.OK)
{
    // display image in picture box
    picProfilePic.Image = new Bitmap(open.FileName);

    // image file path
    // txtFname.Text = open.FileName;
}

imgContact = new Bitmap(open.FileName);
}

private void txtAgencyCode_Leave(object sender, EventArgs e)
{
    string strAgencyCode = null;
    strAgencyCode = txtAgencyCode.Text.ToString();

    SqlConnection conn = new SqlConnection("Data Source=DESKTOP-UO0PO1Q;InitialCatalog=Biome;Integrated Security=True");

    if (clbUserType.SelectedItems.Count > 1)
    {
        MessageBox.Show("Multiple selections not allowed");
        return;
    }

    else if (clbUserType.SelectedItems.Count == 1)
    {
        if (clbUserType.SelectedItem.ToString() != "Customer")
        {

```



```
if (strAgencyCode != null)
{
    string SqlString = "SELECT * FROM Agency WHERE
                        AgencyName=@agencyname";

    try
    {
        conn.Open();

        SqlCommand cmd1 = new SqlCommand(SqlString);

        cmd1.Parameters.Add("@agencyname", SqlDbType.NVarChar).Value
            = strAgencyCode;

        SqlDataAdapter sda = new SqlDataAdapter();

        cmd1.Connection = conn;

        sda.SelectCommand = cmd1;

        DataTable dt = new DataTable();

        cmd1.ExecuteNonQuery();

        sda.Fill(dt);

        conn.Close();

        foreach (DataRow row in dt.Rows)
        {
            txtAgencyID.Text = Convert.ToString(row["AID"]);

            txtAgencyPhNum.Text =
                Convert.ToString(row["ContactNumber"]);

            txtAgencyEmailID.Text =
                Convert.ToString(row["ContactEmailID"]);

            txtAgencyAddress.Text =
                Convert.ToString(row["ContactAddress"]);
        }
    }

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

```
        }
    }
}

}

private void txtFName_Validating(object sender, CancelEventArgs e)
{
    if (string.IsNullOrEmpty(txtFName.Text))
    {
        //This control fails validation: Name cannot be empty.
        lblStatus.Text = "You must provide your name!";
    }
}

}
```

6.2 Front End:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;
using NITGEN.SDK.NBioBSP;

namespace biometest
{
    public partial class frmFrontend : Form
    {
        //-----Initialise-----
        NBioAPI m_NBioAPI;
        NBioAPI.Export m_Export;
        public frmFrontend()
        {
            InitializeComponent();
            m_NBioAPI = new NBioAPI();
            m_Export = new NBioAPI.Export(m_NBioAPI);
        }
        //-----To open Reset Password form
        private void mnuResetPwd_Click(object sender, EventArgs e)
        {
            //this.Hide();
            var form2 = new frmResetPassword();
            //form2.Closed += (s, args) => this.Close();
            form2.Show();
        }
        //-----To open Incident form-----
        private void btnSearch_Click(object sender, EventArgs e)
        {
            //this.Hide();
            //var form2 = new frmReportIncident();
            //form2.Closed += (s, args) => this.Close();
            //form2.Show();
        }
        //-----To open User Reg form-----
        private void btnAdd_Click(object sender, EventArgs e)
        {
            var form2 = new frmUserRegistration1();
            form2.Show();
        }
    }
}
```

```
//----- To open Pending Cases -----
private void mnuViewPending_Click(object sender, EventArgs e)
{
    //this.Hide();
    var form2 = new frmPendingCases();
    // form2.Closed += (s, args) => this.Close();
    form2.Show();
}

//-----To open Resolved cases -----
private void mnuViewResolved_Click(object sender, EventArgs e)
{
    //this.Hide();
    var form2 = new frmResolvedCases();
    //form2.Closed += (s, args) => this.Close();
    form2.Show();
}

//-----On sign out-----

private void mnuSignOut_Click(object sender, EventArgs e)
{
    DialogResult result = MessageBox.Show("Are you sure want to Signout?",
        "Signout", MessageBoxButtons.YesNo);
    if (result == DialogResult.Yes)
    {
        MessageBox.Show("Signing out");
        Globals.passUID = null;
        Globals.passRID = null;

        SqlConnection con = new SqlConnection("Data Source=DESKTOP-
            UO0PO1Q;Initial Catalog=Biome;Integrated Security=True");
        con.Open();
        SqlCommand cmd = new SqlCommand("Insert into
            [Audit](UID,Form,ActionTaken) values(@UID,'User Logged out','Session
            ended') "); //chk query
        cmd.Parameters.Add("@UID", SqlDbType.NVarChar).Value = Globals.passUID;
        cmd.Connection = con;
        cmd.ExecuteNonQuery();
        con.Close();

        this.Hide();
        var form2 = new frmWelcomePage();
        form2.Closed += (s, args) => this.Close();
        form2.Show();
    }
    else if (result == DialogResult.No)
    {
        MessageBox.Show("Signout Cancel");
    }
}
```

```

        return;
    }

}

private void frmFrontend_Load(object sender, EventArgs e)
{
    NBioAPI.Type.VERSION version = new NBioAPI.Type.VERSION();
    m_NBioAPI.GetVersion(out version);

    Text = String.Format("Export Demo for C#.NET (BSP Version : v{0}.{1:D4})",
        version.Major, version.Minor);

    DataTable dt = new DataTable();
    SqlDataAdapter sda = new SqlDataAdapter();
    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
        UO0PO1Q;Initial Catalog=Biome;Integrated Security=True"))
    {
        using (SqlCommand cmd = new SqlCommand("SP_frontend_form_load", con))
        {
            cmd.CommandType = CommandType.StoredProcedure;
            con.Open();
            cmd.Parameters.Add("@uid", SqlDbType.NVarChar).Value =
                Globals.passUID;
            sda.SelectCommand = cmd;
            sda.Fill(dt);
            con.Close();
            // SqlDataReader dr = cmd.ExecuteReader();
            foreach (DataRow row in dt.Rows)
            {
                txtFNname.Text = row["FirstName"].ToString();
                txtMName.Text = row["MiddleName"].ToString();
                txtLName.Text = row["LastName"].ToString();
                txtUsername.Text = row["UID"].ToString();
                txtEmailID.Text = row["EmailID"].ToString();
                txtPhNum.Text = row["PhoneNumber1"].ToString();
                txtAddress.Text = row["CurrentAddress"].ToString();
                lblWelcomeMsg.Text = "Hello " + row["FirstName"].ToString()+"!";
            }
        }
    }
}

```

//-----Function to scan and search fingerprint-----

```
private void btnScan_Click(object sender, EventArgs e)
{
    m_NBioAPI.OpenDevice(NBioAPI.Type.DEVICE_ID.AUTO);

    NBioAPI.Type.HFIR capturedFIR;
    uint ret = m_NBioAPI.Capture(out capturedFIR);

    m_NBioAPI.CloseDevice(NBioAPI.Type.DEVICE_ID.AUTO);

    if (ret != NBioAPI.Error.NONE)
    {
        MessageBox.Show("Failed to capture fingerprint.");
        return;
    }

    NBioAPI.Type.MINCONV_DATA_TYPE exportType;

    exportType = NBioAPI.Type.MINCONV_DATA_TYPE.MINCONV_TYPE_FDU;

    NBioAPI.Type.MINCONV_DATA_TYPE importType;
    importType = exportType;
    NBioAPI.Export.EXPORT_DATA exportData;
    byte[] currentarrayinput = null;
    ret = m_Export.NBioBSPToFDx(capturedFIR, out exportData, exportType);

    if (ret != NBioAPI.Error.NONE)
    {
        MessageBox.Show("Failed to export data.");
        return;
    }

    for (int f = 0; f < exportData.FingerNum; f++)
    {
        for (int s = 0; s < exportData.SamplesPerFinger; s++)
        {
            currentarrayinput = exportData.FingerData[f].Template[s].Data;
        }
    }
    DataTable dt = null;
    string SqlString1 = "Select * From Biometrics Where Fingerprint Is not Null";
    using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
        UO0PO1Q;Initial Catalog=Biome;Integrated Security=True"))
    {
        con.Open();
        SqlCommand cmd1 = new SqlCommand(SqlString1);
        SqlDataAdapter sda1 = new SqlDataAdapter();
        cmd1.Connection = con;
        sda1.SelectCommand = cmd1;
```

```

        dt = new DataTable();
        sda1.Fill(dt);
        con.Close();
    }
    foreach (DataRow row in dt.Rows)
    {

        byte[] dbdata = (byte[])row[1];
        byte[] currentfingerprint = currentarrayinput;

        NBioAPI.Type.HFIR processedFIR;
        NBioAPI.Type.HFIR processedFIR1;
        uint nSize = (uint)currentarrayinput.Length;
        uint nSize1 = (uint)dbdata.Length;
        uint ret1 = m_Export.FDxToNBioBSPEX(currentfingerprint, nSize, exportType,
        NBioAPI.Type.FIR_PURPOSE.VERIFY, out processedFIR);
        uint ret2 = m_Export.FDxToNBioBSPEX(dbdata,
        nSize1,exportType,NBioAPI.Type.FIR_PURPOSE.VERIFY,out processedFIR1);
        Boolean bResult;
        ret1 = m_NBioAPI.VerifyMatch(processedFIR, processedFIR1, out bResult,
            null);
        string row2data = row[2].ToString();
        if(bResult)
        {
            MessageBox.Show("Successful search!");

            Form frm4 = new frmReportIncident();
            frm4.Show();
            break;
        }
        else
        {
            MessageBox.Show("Unsuccessful search!");
        }

    }

}

}
private void picCP_Click(object sender, EventArgs e)
{
    this.Hide();
    var form2 = new frmControlPanel();
    form2.Closed += (s, args) => this.Close();
    form2.Show();
}

}
}

```

6.3 Login

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

    while(string.IsNullOrEmpty(txtPassword.Text)||string.IsNullOrEmpty(txtUsername.Text))
    {
        if (string.IsNullOrEmpty(txtPassword.Text))
        {
            //This control fails validation: Password cannot be empty.
            lblStatus.Text = "You must provide password!";
            MessageBox.Show("Enter the password!");
            txtPassword.Focus();
            return;
        }
        else if (string.IsNullOrEmpty(txtUsername.Text))
        {
            //This control fails validation: Name cannot be empty.
            lblStatus.Text = "You must provide username!";
            MessageBox.Show("Enter the user name!");
            txtUsername.Focus();
            return;
        }
    }
    strUname = Convert.ToString(txtUsername.Text);
    strPassword = Convert.ToString(txtPassword.Text);

    try
    {

        using (SqlConnection con = new SqlConnection("Data Source=DESKTOP-
        UO0PO1Q;Initial Catalog=Biome;Integrated Security=True"))
        {
            using (SqlCommand cmd = new SqlCommand("Sp_welcome_login", con))
            //name of stored proc for login
            {
                cmd.CommandType = CommandType.StoredProcedure;
                object result,result1;
                cmd.Parameters.Add("@UID", SqlDbType.NVarChar).Value =
                    txtUsername.Text;    //variable name of username
                cmd.Parameters.Add("@password", SqlDbType.NVarChar).Value =
                    txtPassword.Text;    //variable name of password

                con.Open();
                result = cmd.ExecuteScalar();
                con.Close();
                if (result == null)
                    MessageBox.Show("Invalid username/password.", "Login Failed",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```



```
        else
        {
            Globals.passUID= Convert.ToString(result);
            SqlCommand cmd1 = new SqlCommand("Select RID from
            UserRoleMapping where UID=@acctUID");
            cmd1.Parameters.Add("@acctUID", SqlDbType.NVarChar).Value =
            Globals.passUID;
            con.Open();
            result1=cmd1.ExecuteScalar();
            Globals.passRID = Convert.ToString(result1);
            con.Close();
            this.Hide();
            var form2 = new frmFrontend();
            form2.Closed += (s, args) => this.Close();
            form2.Show();
        }
    }
}
}
catch (Exception ex)
{
    MessageBox.Show(ex.Message);
}
}
```

6.4 Print the page

```
private void printDocument1_PrintPage(object sender,
    System.Drawing.Printing.PrintPageEventArgs e)
{
    int width = 0;
    int height = 0;
    int i = 0;
    StringFormat str = new StringFormat();
    str.Alignment = StringAlignment.Near;
    str.LineAlignment = StringAlignment.Center;
    str.Trimming = StringTrimming.EllipsisCharacter;
    Pen p = new Pen(Color.Black, 2.5f);

    #region Draw Column 1

    e.Graphics.FillRectangle(Brushes.LightGray, new Rectangle(100, 100,
        dataGridView1.Columns[0].Width, dataGridView1.Rows[0].Height));

    e.Graphics.DrawRectangle(Pens.Black, 100, 100,
        dataGridView1.Columns[0].Width, dataGridView1.Rows[0].Height);

    e.Graphics.DrawString(dataGridView1.Columns[0].HeaderText,
        dataGridView1.Font, Brushes.Black, new RectangleF(100, 100,
            dataGridView1.Columns[0].Width, dataGridView1.Rows[0].Height), str);

    #endregion

    #region Draw column 2

    e.Graphics.FillRectangle(Brushes.LightGray, new Rectangle(100 +
        dataGridView1.Columns[0].Width, 100, dataGridView1.Columns[0].Width,
            dataGridView1.Rows[0].Height));
```

```
e.Graphics.DrawRectangle(Pens.Black, 100 + dataGridView1.Columns[0].Width,
100, dataGridView1.Columns[0].Width, dataGridView1.Rows[0].Height);

e.Graphics.DrawString(dataGridView1.Columns[1].HeaderText,
dataGridView1.Font, Brushes.Black, new RectangleF(100 +
dataGridView1.Columns[0].Width,100,dataGridView1.Columns[0].Width,
dataGridView1.Rows[0].Height),str);

width = 100 + dataGridView1.Columns[0].Width;

height = 100;

while (i < dataGridView1.Rows.Count)
{
    if (height > e.MarginBounds.Height)
    {
        height = 100;

        width = 100;

        e.HasMorePages = true;

        return;
    }

    height += dataGridView1.Rows[i].Height;

    e.Graphics.DrawRectangle(Pens.Black, 100, height,
        dataGridView1.Columns[0].Width, dataGridView1.Rows[0].Height);

    e.Graphics.DrawString(dataGridView1.Rows[i].Cells[0].FormattedValue.ToString(),
dataGridView1.Font, Brushes.Black, new RectangleF(100, height,
dataGridView1.Columns[0].Width, dataGridView1.Rows[0].Height), str);
```

```
e.Graphics.DrawRectangle(Pens.Black, 100 + dataGridView1.Columns[0].Width,
height, dataGridView1.Columns[0].Width, dataGridView1.Rows[0].Height);
```

```
e.Graphics.DrawString(dataGridView1.Rows[i].Cells[1].FormattedValue.ToString(),
dataGridView1.Font, Brushes.Black, new RectangleF(100 +
dataGridView1.Columns[0].Width, height, dataGridView1.Columns[0].Width,
dataGridView1.Rows[0].Height), str);
```

```
width += dataGridView1.Columns[0].Width;
```

```
i++;
```

```
}
```

```
#endregion
```

```
SqlConnection con = new SqlConnection("Data Source=DESKTOP-
UO0PO1Q;Initial Catalog=Biome;Integrated Security=True");
```

```
con.Open();
```

```
SqlCommand cmd = new SqlCommand("Insert into [Audit](UID,Form,ActionTaken)
values(@UID,'frmControlPanel','Printed Users') "); //chk query
```

```
cmd.Parameters.Add("@UID", SqlDbType.NVarChar).Value = "11";
```

```
cmd.Connection = con;
```

```
cmd.ExecuteNonQuery();
```

```
con.Close();
```

```
}
```

6.5 Automated E-mail sending

```
private void mailsend(string strid)
{
    try
    {
        MailAddress mailfrom = new MailAddress("administrator@bio-me.in");
        MailAddress mailto = new MailAddress(strid);
        MailMessage newmsg = new MailMessage(mailfrom, mailto);

        newmsg.Subject = "Bio-Me Account credentials";
        newmsg.Body = "Your Biome Account password has been reset the new password is" + txtPassword.Text;

        SmtpClient smtps = new SmtpClient("mail.bio-me.in", 2525);
        smtps.UseDefaultCredentials = false;
        smtps.Credentials = new NetworkCredential("administrator@bio-me.in", "biome@123");
        smtps.EnableSsl = false;
        smtps.Send(newmsg);
        MessageBox.Show("Mail has been sent with the new credentials");
    }
    catch (Exception ex)
    {
        lblStatus.Text = "The following exception has occurred " + ex.Message + " when trying to send the mail";
    }
}
```

6.6 Stored Procedures

6.6.1 Sp_welcome_login

```
USE [Biome]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
ALTER procedure [dbo].[Sp_welcome_login] @UID nvarchar(50),@password
nvarchar(100)
as
Begin
```

```
Declare @pass nvarchar(100)
set @pass=(Select password from Users where UID=@UID)
```

```
If (@pass=@password)
Begin
Insert into [Audit](UID,
Form,
ActionTaken)
Select
@UID,
'frmWelcomePage',
'SuccessfulLogin'
END
Else
```

```
Begin
Insert into [Audit](UID,
Form,
ActionTaken)
Select
@UID,
'frmWelcomePage',
'UnSuccessfulLogin'
End
```

```
END
```

6.6.2 Sp_Reg_On_submit

USE [Biome]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

ALTER procedure [dbo].[SP_Reg_On_submit]

@UID nvarchar(50),

@fingerprint varbinary(max),

@Biopic Image,

@RID nvarchar(20),

@AgencyName nvarchar(50),

@ContactNumber nvarchar(14),

@ContactEmailID nvarchar(30),

@ContactAddress nvarchar(500),

@Form nvarchar(50),

@ActionTaken nvarchar(50)

as

Begin

Insert into Biometrics (UID,Fingerprint,Biopic)

select

@UID,

@Fingerprint,

@Biopic

Insert into UserRoleMapping (RID,UID)

Select

@RID ,

@UID

Insert into Agency(AgencyName,

ContactNumber,

ContactEmailID,

ContactAddress)

Select

@AgencyName,

@ContactNumber,

@ContactEmailID,

@ContactAddress

```
Insert into [Audit](UID,
Form,
ActionTaken)
Select
@UID,
@Form,
@ActionTaken
end
```

6.6.3 Sp_pending_form_update

```
USE [Biome]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
ALTER PROCEDURE [dbo].[SP_pending_form_update]
    @Description nvarchar(800),
    @Actiontaken nvarchar(800),
    @Status nvarchar(20)
AS

BEGIN
Update Incident
Set
    Description = @Description, Actiontaken = @Actiontaken, Status = @Status
where Status in ('P','p')

--Inserting into Audit table

Insert into [Audit](UID,
Form,
ActionTaken)
Select
UID,
'frmPendingCases',
'Updated'
from incident
Where
Status in ('P','p')

END
```


6.6.4 Sp_Resetpassword

USE [Biome]

GO

SET ANSI_NULLS ON

GO

SET QUOTED_IDENTIFIER ON

GO

ALTER procedure [dbo].[Sp_Resetpassword]@UID nvarchar(50),@password nvarchar(100)
as

Begin

Update users

Set password=@password

where UID=@UID

Insert into [Audit](UID,

Form,

ActionTaken)

Select

@UID,

'frmResetPassword',

'PasswordReset'

End

CHAPTER 7

TESTING

CHAPTER 7

TESTING

7.1 Introduction

Software testing is the most critical phase in the development life cycle. Testing performs a very critical role for quality assurance and ensuring the reliability of the software. Software testing represents the ultimate review of specification design and preview.

During the development of software, errors can be injected at any stage. However requirements and design errors are likely to remain undetected. Those errors will be ultimately reflected in the code. During testing, the program to be tested is executed with a set of test cases and output of program for test cases are evaluated to describe the program performance to expected level. No system design is perfect because of communication problem, programmer's negligence or constraints create errors that must be eliminated before the system is ready for use of acceptance.

The purpose of testing is discovering errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/ or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectation and does not fail in an unacceptable manner.

The common view of testing by user is that it is performed to prove that there are no errors in the programs. However, it is virtually impossible, since analyst cannot prove that software is free and clear of errors. A successful test is the one that finds an error. Reliability is to be designed in to the system.

There are various types of test. Each test type addresses a specific testing requirement. Following sections describes the system testing and test cases.

In order to measure the accuracy of both the modules we downloaded 10k sample input and tested for each of the module.

7.2 Unit testing

This is the first level of testing, where different modules are tested against the requirement specification for the modules. Unit testing is essential for verification of the code produced during the coding phase and hence the goal is to test internal logic of modules.

Unit testing checks two types of errors-syntax and logic. The testing can be performed by bottom up approach, which starts at the smallest and the lowest level of modules and proceeding one at a time ,or Top-Down testing, where testing begins with the upper test of module and moves towards the smaller ones.

7.3 System Testing

System testing is performed on the entire system in the context of a functional requirement specification and system requirement specification .System testing tests not only the design, but also the behavior and the expectations.it is also to test up to and beyond the bounds defined in the software requirement specification.

As a rule, system testing takes input of all the —integrated|| software components that have been integrated .The purpose of software testing is to detect any consistencies between the software units that are integrated together.

7.4 Bio-Me Test Cases

Test Case ID	Form	Test Data	Expected Result	Actual Result	Status
TC_01	Welcome	Valid Uname and Pwd	Open User Front End	Front End Form was opened	PASS
TC_02	Welcome	Valid Uname and Invalid Pwd	Pop error	Error Message Box opens	PASS
TC_03	Welcome	Click of Sign-up	User Reg Form should open	User Reg Form opens	PASS
TC_04	User Reg	Null values	Pop error	Error Message Box opens	PASS
TC_05	User Reg	Invalid values	Pop error	Error Message Box opens	PASS
TC_06	Front End	Click Scan	Initializes scanner	Scanner initialized	PASS
TC_07	Front End	Click Reset Password	Open Reset Password form	Reset Password form opens	PASS
TC_08	Front End	Click Sign Out	Pop confirmation	Confirmation Message Box opens	PASS
TC_09	Front End	Click Scan and Search	Initializes scanner	Scanner initialized	PASS
TC_10	Front End	Click Scan and Add	User Reg Form should open	User Reg Form opens	PASS
TC_11	Pending Cases	Updation of Pending	Update P→R	P is replaced by R	PASS
TC_12	Pending Cases	Click Cancel	Open User Front End	Front End Form was opened	PASS
TC_13	Report Incident	Multiple selects in checklist	Pop error	Error Message Box opens	PASS
TC_14	Report Incident	Click Cancel	Open User Front End	Front End Form was opened	PASS
TC_15	Control Panel	Null values	Pop error	Error Message Box opens	PASS

TC_16	Control Panel	Invalid values	Pop error	Error Message Box opens	PASS
TC_17	Control Panel	Redundant UID	Pop error	Error Message Box opens	PASS
TC_18	Control Panel	Edit uneditable field	Do not update in Db	Db was not updated	PASS
TC_19	Control Panel	Edit uneditable fields in DataGrid	Do not update in Db	Db was not updated	PASS
TC_20	Reset Password	Invalid values	Pop error	Error Message Box opens	PASS
TC_21	Reset Password	Unequal Pwd and Confirm Pwd fields	Pop error	Error Message Box opens	PASS
TC_22	Reset Password	Click Reset	Reset Pwd and send E-mail	E-mail was sent with new pwd	PASS

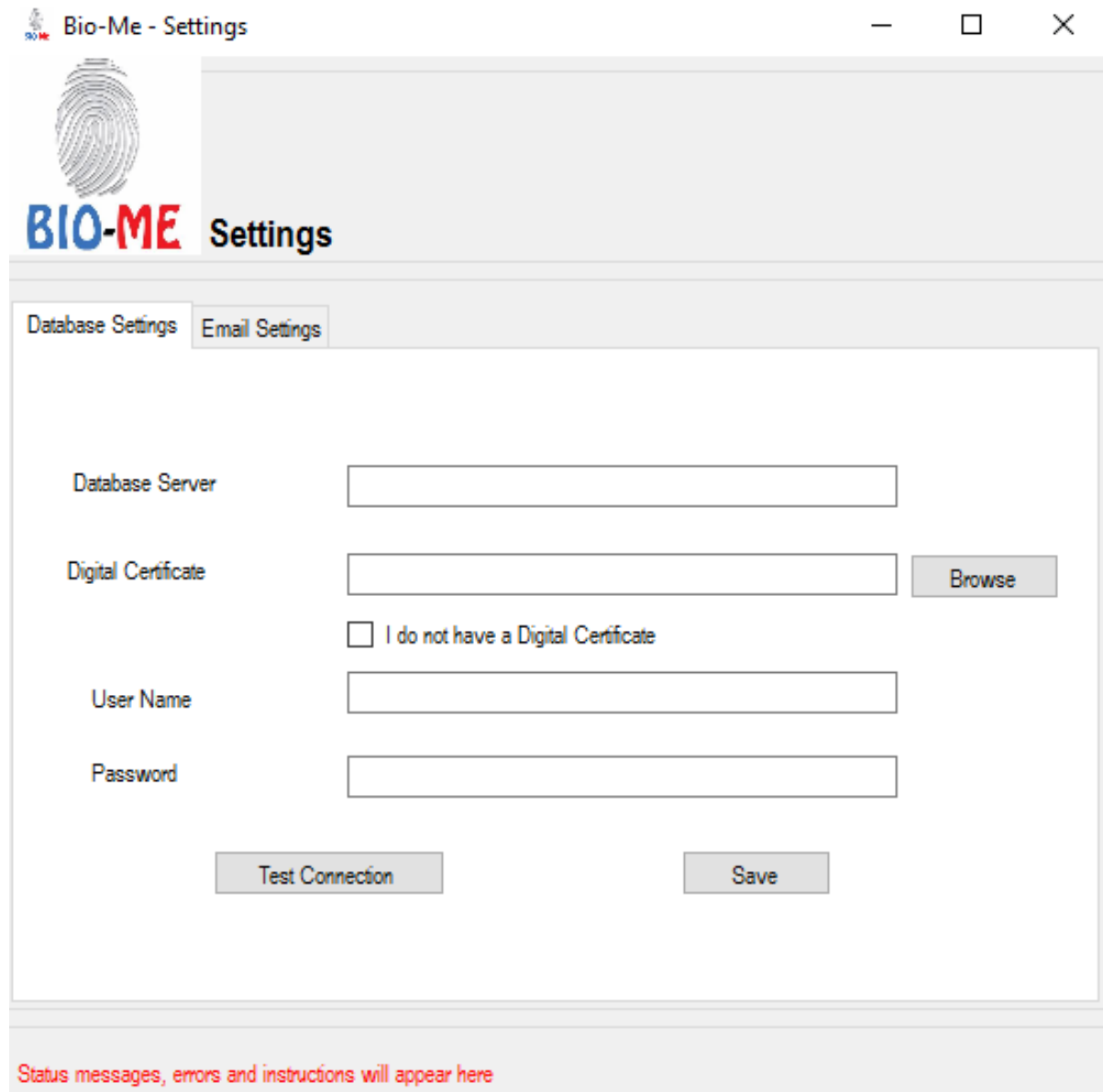
Table 2. Bio-Me Test Cases

CHAPTER 8

SNAPSHOTS

CHAPTER 8

SNAPSHOTS



The screenshot shows a web application window titled "Bio-Me - Settings". The window has a header bar with the "BIO-ME" logo (a fingerprint icon) and the word "Settings". Below the header, there are two tabs: "Database Settings" (selected) and "Email Settings". The "Database Settings" tab contains the following fields and controls:

- Database Server:** A text input field.
- Digital Certificate:** A text input field with a "Browse" button to its right.
- ☐ I do not have a Digital Certificate
- User Name:** A text input field.
- Password:** A text input field.
- Test Connection:** A button.
- Save:** A button.

At the bottom of the window, there is a red text message: "Status messages, errors and instructions will appear here".

Fig. 15 Database Settings Form

Bio-Me - Settings

BIO-ME Settings

Database Settings Email Settings

Email Server

SMTP Ports

Use SSL ☐ Yes ☐ No

Username

Password


Status messages, errors and instructions will appear here

Fig. 16 E-mail Settings Form

Bio-Me - Welcome Page



BIO-ME Welcome to BIO-ME



Login to BIO-ME

Username
(Enter Aadhaar number as username)

Password

Login

— Or —


Are you a new user?

Sign-Up


Status messages, errors and instructions will appear here

Fig. 17 Welcome Page

Export Demo for C#.NET (BSP Version : v4.8610)

 **BIO-ME** Welcome to BIO-ME

[View pending cases](#) [View resolved cases](#) [Reset password](#) [Sign out](#)

 **Scan and Search**
Scan and Add

Hello Arpita!

First Name	<input type="text" value="Arpita"/>
Middle Name	<input type="text" value="M"/>
Last Name	<input type="text" value="Das"/>
Username	<input type="text" value="11"/>
Phone Number	<input type="text" value="9900547621"/>
Address	<input type="text" value="London"/>
Email ID	<input type="text" value="ankitakk5@gmail.com"/>

Status messages, errors and instructions will appear here

Fig. 18 Home Page

BIO-ME Control Panel

Users Roles Permissions User Role Mapping Agency Settings Audit Log

Enter UID: + 🔍 🗑️ Clear Post

UID	First Name	Middle Name	Last Name	Guardiana Name	Gender	DOB	Phone Number 1	Phone Number 2	Emergency Contact	Blood Group	Current Address
11	Anusha	M	Das	roy	Female	19-05-1991	1234534567	9902341222	9738134423	A +ve	London
111123339999	Das		M		Female	22-05-2017	9901840199			Drive	India
12227333444	Anis		K		Female	25-11-2016	9481762441			A +ve	B+
123412344321	as		as		Female	18-05-2017	1234554321				Waldale
12342335	Indra		jadgajal	as	Female	12-05-2017	123	233	234	O +ve	EG Bgh
131113851311	Se		Lakshmi	Abc	Female	28-05-2017	8861262683			B +ve	Rajmangor
131713171317	Anisha	K	Karnad	Karone	Female	05-02-1995	9481762441		9481762531	A +ve	Mahabara
1341123334	DDO		ZZZZZ	BBSDKH	Female	12-05-2017	1234	1345	12334		BSJNDSPK

Status messages, errors and instructions will appear here

Fig. 19 Control Panel-User Settings

BIO-ME Control Panel

Users Roles Permissions User Role Mapping Agency Settings Audit Log

Enter UID: 🔍 Clear

Set Roles

Rule Name: +

UID: 🗑️

Post

Status messages, errors and instructions will appear here

Fig. 20 Control Panel-Roles Settings

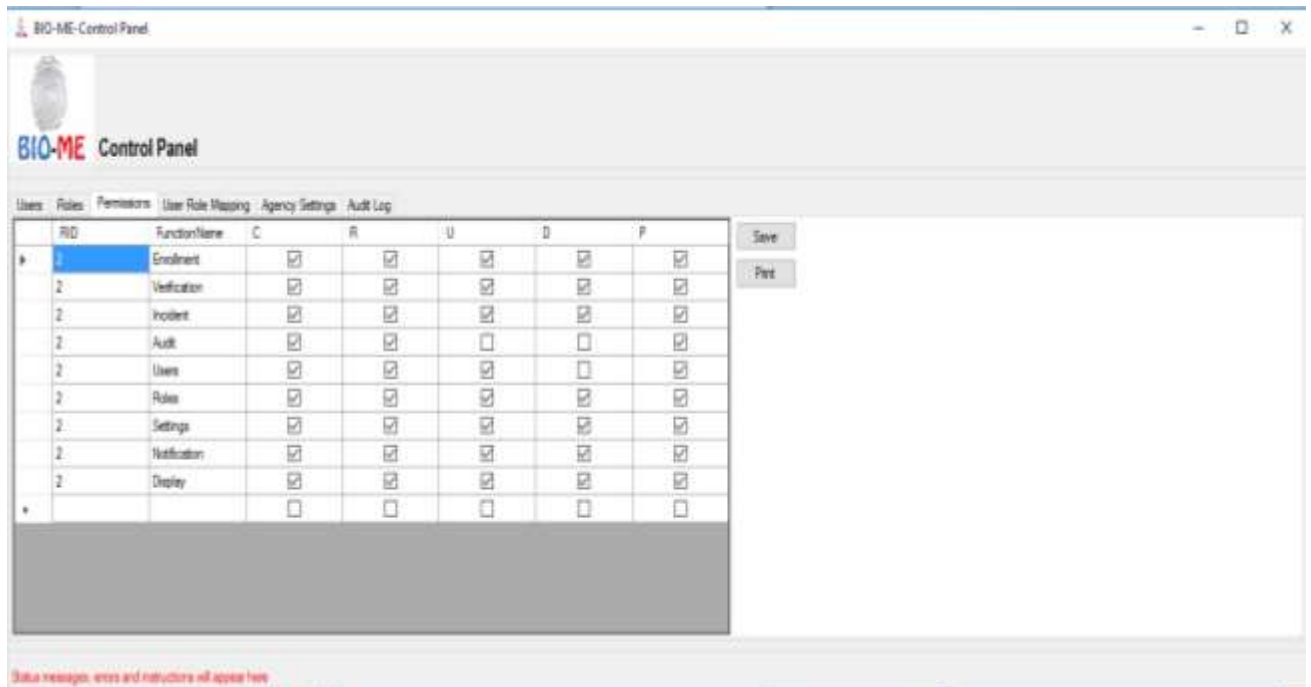


Fig. 21 Control Panel-Permission Settings



Fig. 22 Control Panel-User Role Mapping Settings

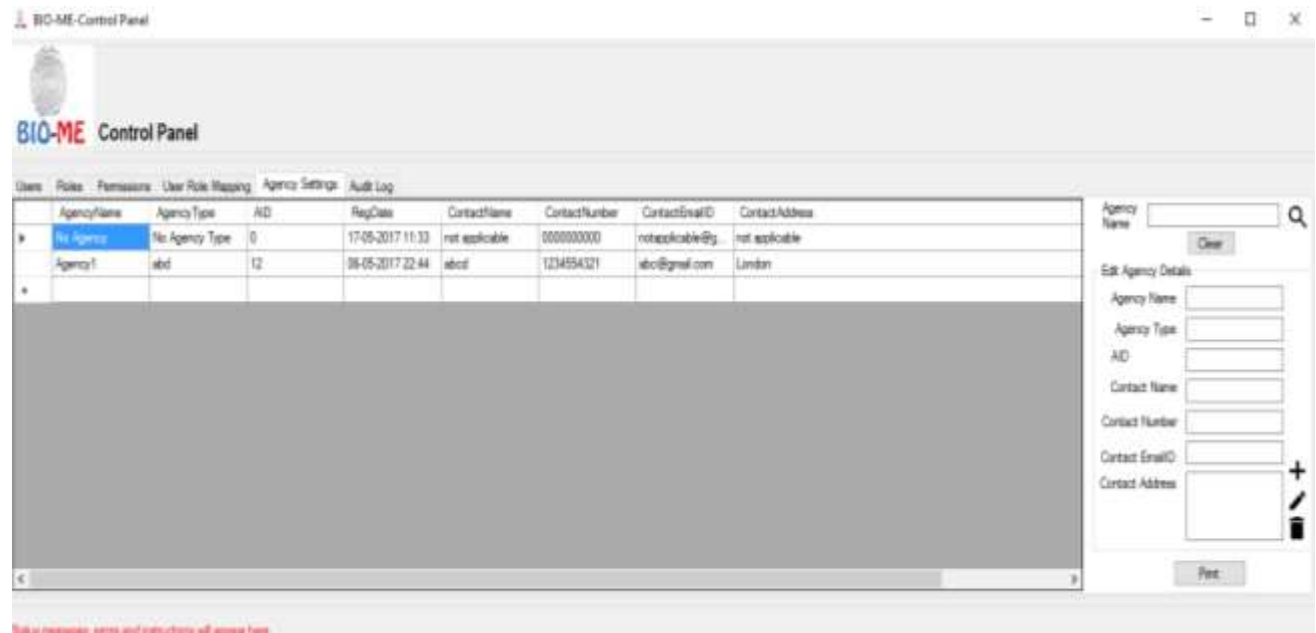


Fig. 23 Control Panel-Agency Settings

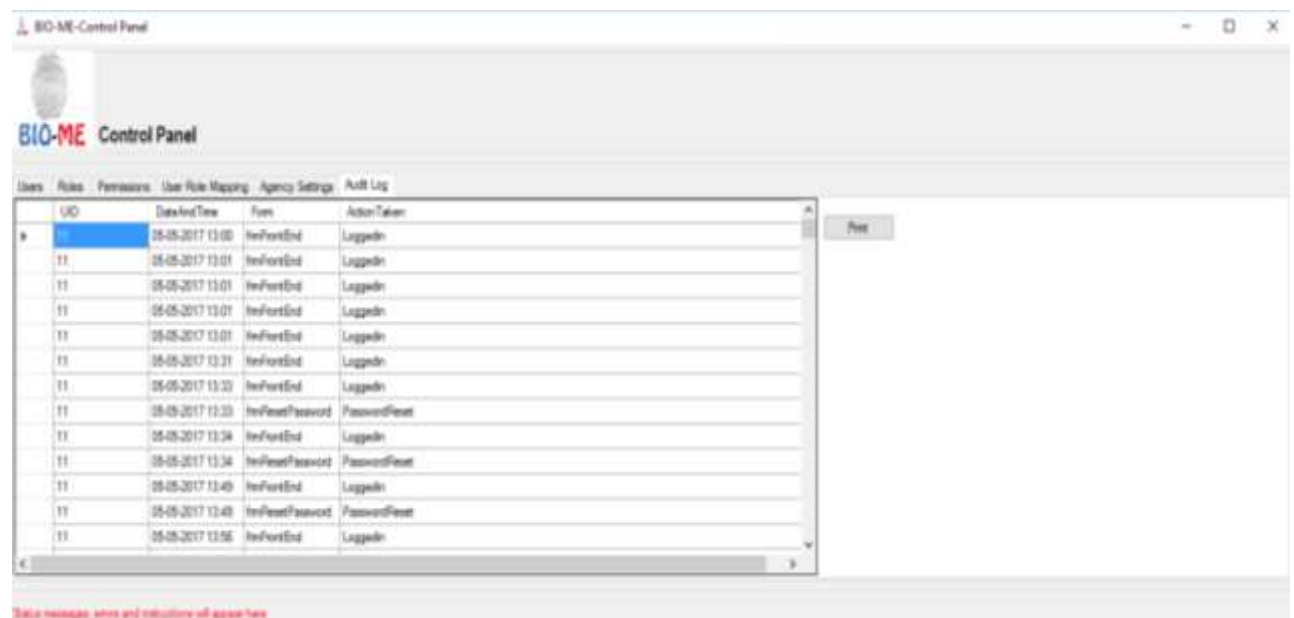


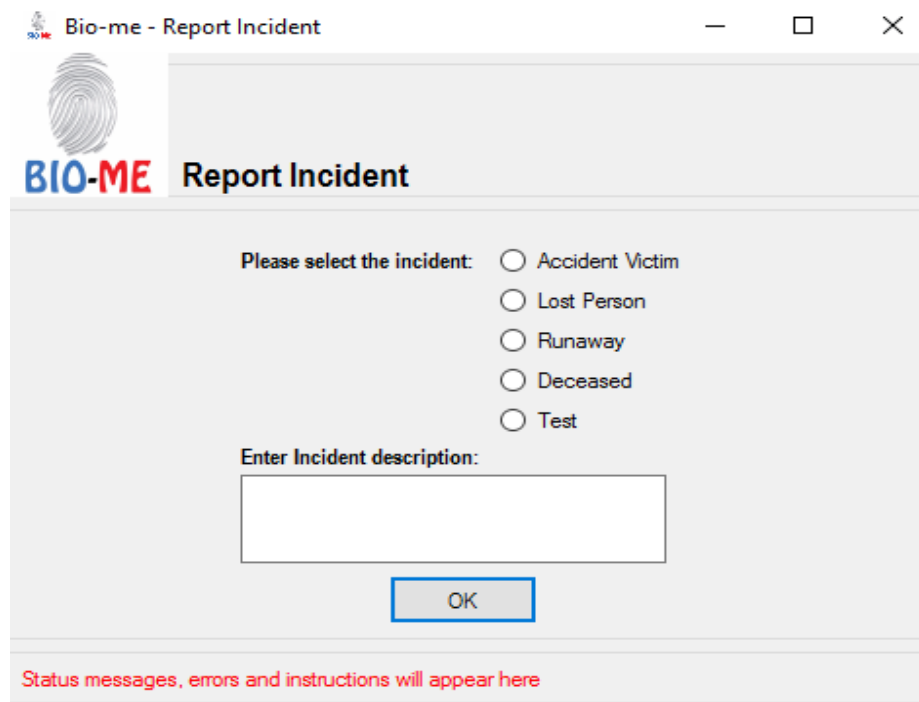
Fig. 24 Control Panel-Audit Log Settings

The screenshot shows a web application window titled "Bio-me - Pending Cases". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. On the left side, there is a logo consisting of a fingerprint icon above the text "BIO-ME". To the right of the logo, the text "Pending Cases" is displayed. The main content area is a large, empty gray rectangle. At the bottom center of the window, there is a "Cancel" button. Below the main content area, there is a red text message: "Status messages, errors and instructions will appear here".

Fig. 25 Pending Cases Form

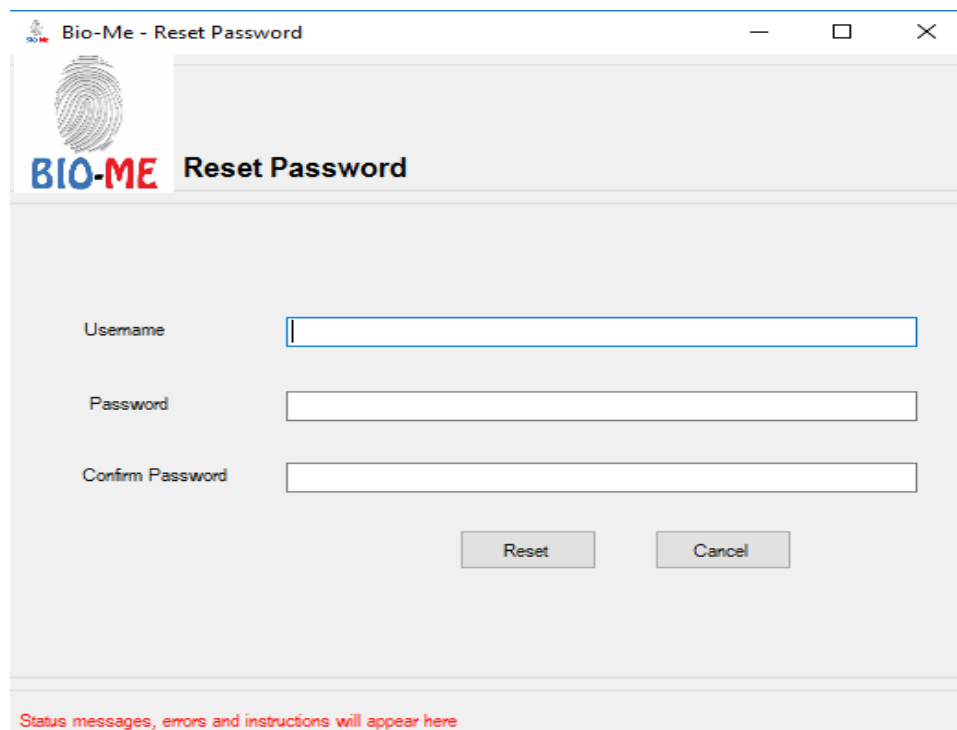
The screenshot shows a web application window titled "Bio-me - Resolved Cases". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. On the left side, there is a logo consisting of a fingerprint icon above the text "BIO-ME". To the right of the logo, the text "Resolved Cases" is displayed. The main content area is a large, empty gray rectangle. At the bottom center of the window, there is a "Cancel" button. Below the main content area, there is a red text message: "Status messages, errors and instructions will appear here".

Fig. 26 Resolved Cases Form



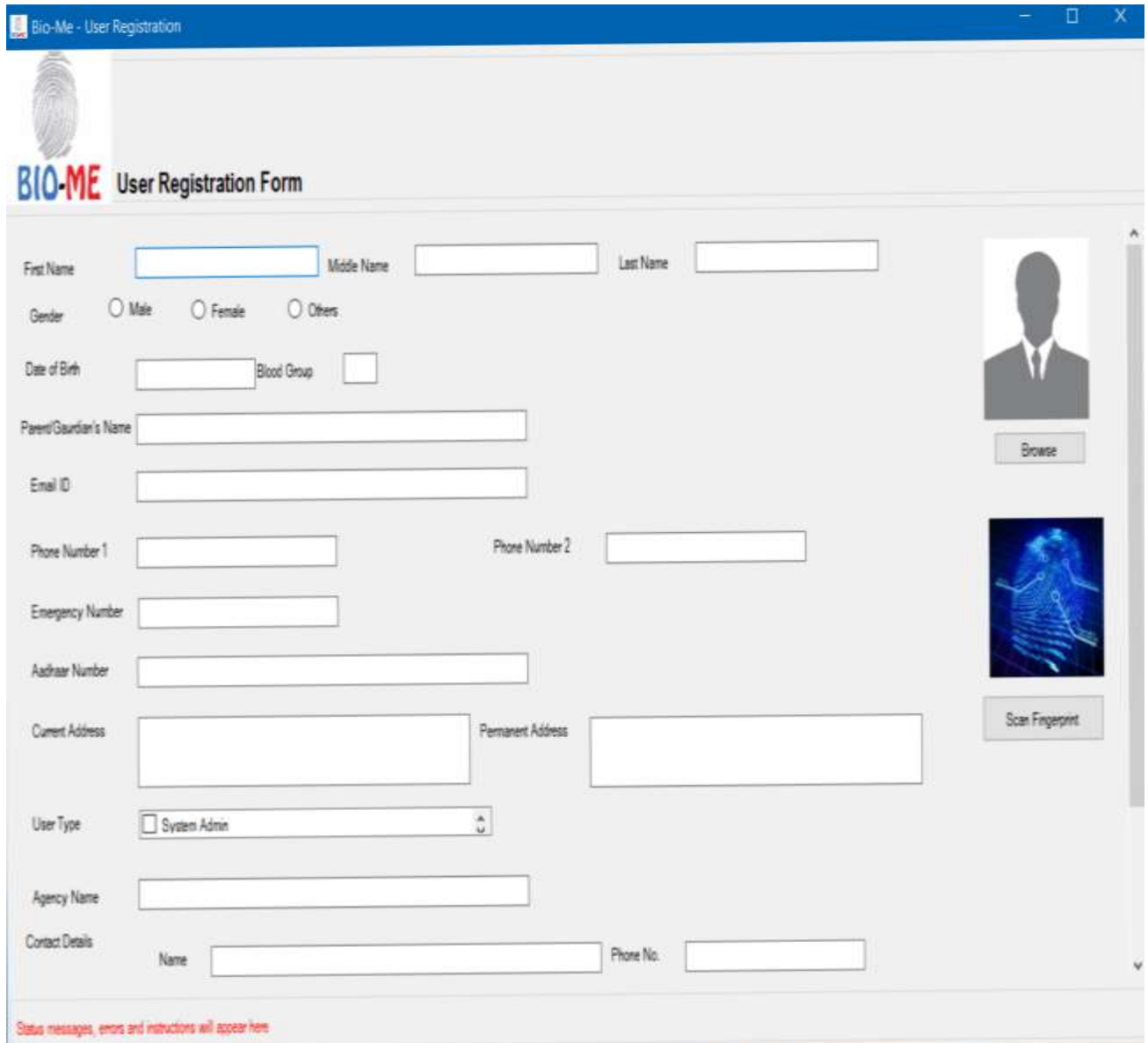
The image shows a web application window titled "Bio-me - Report Incident". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area features the "BIO-ME" logo on the left, which includes a fingerprint icon. To the right of the logo, the text "Report Incident" is displayed. Below this, there is a section titled "Please select the incident:" followed by five radio button options: "Accident Victim", "Lost Person", "Runaway", "Deceased", and "Test". Underneath the radio buttons is a text input field labeled "Enter Incident description:". Below the text field is an "OK" button. At the bottom of the window, there is a red text message: "Status messages, errors and instructions will appear here".

Fig. 27 Report Incident Form



The image shows a web application window titled "Bio-Me - Reset Password". The window has a standard Windows-style title bar with minimize, maximize, and close buttons. The main content area features the "BIO-ME" logo on the left, which includes a fingerprint icon. To the right of the logo, the text "Reset Password" is displayed. Below this, there are three text input fields labeled "Username", "Password", and "Confirm Password". Below the input fields are two buttons: "Reset" and "Cancel". At the bottom of the window, there is a red text message: "Status messages, errors and instructions will appear here".

Fig. 28 Reset Password Form



The screenshot shows a web browser window titled "Bio-Me - User Registration". The page features the "BIO-ME" logo and the title "User Registration Form". The form includes several input fields and sections:

- Name Fields:** First Name, Middle Name, and Last Name.
- Gender:** Radio buttons for Male, Female, and Others.
- Date of Birth:** A date picker and a Blood Group dropdown.
- Parent/Guardian's Name:** A text input field.
- Email ID:** A text input field.
- Phone Numbers:** Phone Number 1, Phone Number 2, and Emergency Number.
- Aadhaar Number:** A text input field.
- Addresses:** Current Address and Permanent Address.
- User Type:** A dropdown menu with "System Admin" selected.
- Agency Name:** A text input field.
- Contact Details:** Name and Phone No.

On the right side, there are two image upload sections:

- A profile picture section with a "Browse" button.
- A fingerprint scan section with a "Scan Fingerprint" button.

At the bottom left, a red text line reads: "Status messages, errors and instructions will appear here".

Fig. 29 User Registration Form

Bio-Me - User Registration

BIO-ME User Registration Form

First Name Middle Name

Gender ☐ Male ☐ Female ☐ Others

Date of Birth Blood Group

Parent/Guardian's Name

Email ID

Phone Number 1

Emergency Number

Aadhaar Number

Current Address

User Type ☐ System Admin

Agency Name

Contact Details
Name Phone No.

Status messages, errors and instructions will appear here

Fingerprint Registration STEP1

Select the finger you wish to enroll by clicking once on the corresponding fingertip.

BACK CANCEL

NITGEN biometric solutions

Browse

Scan Fingerprint

Fig. 30 On clicking Scan

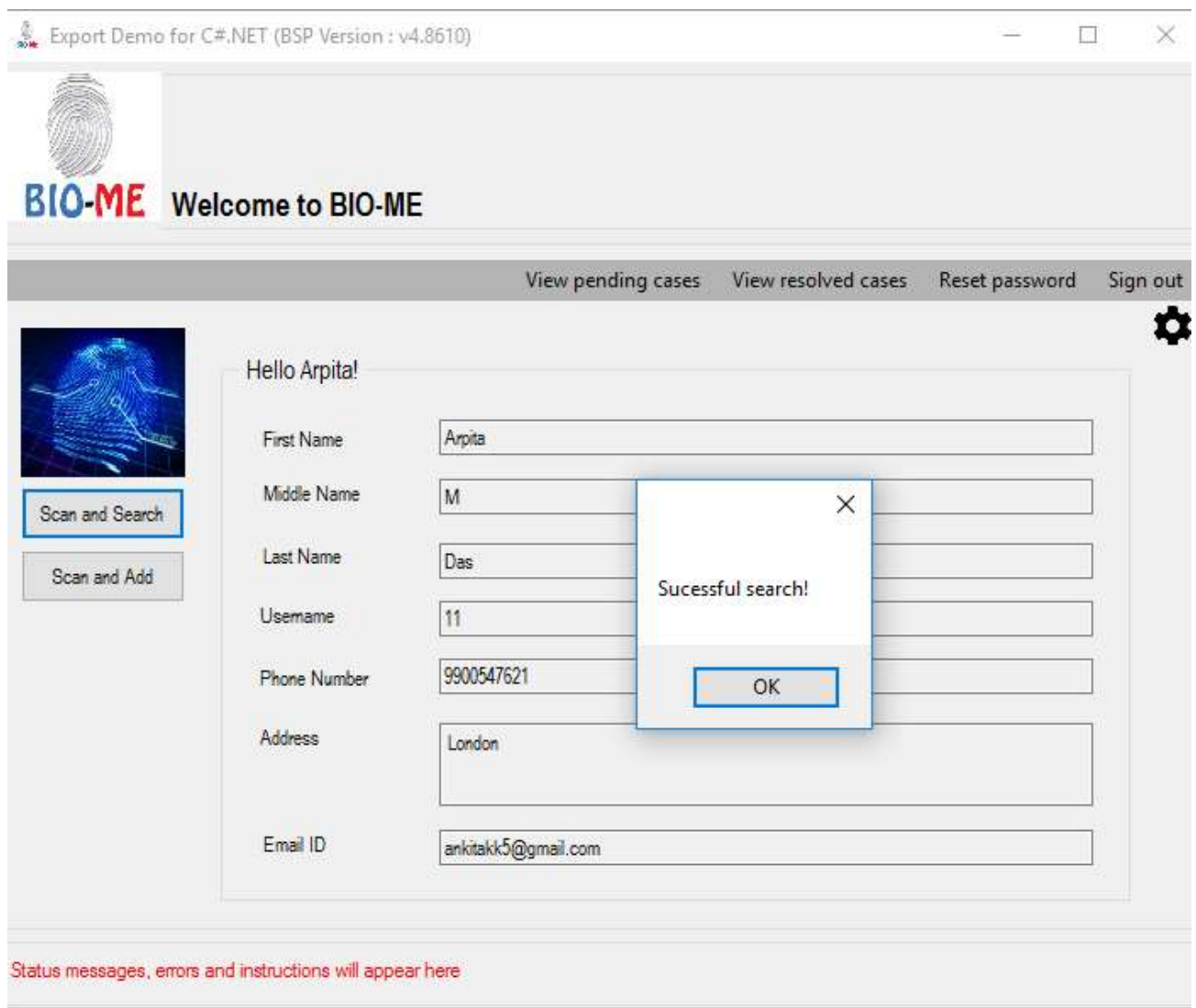


Fig. 31 On Successful Search

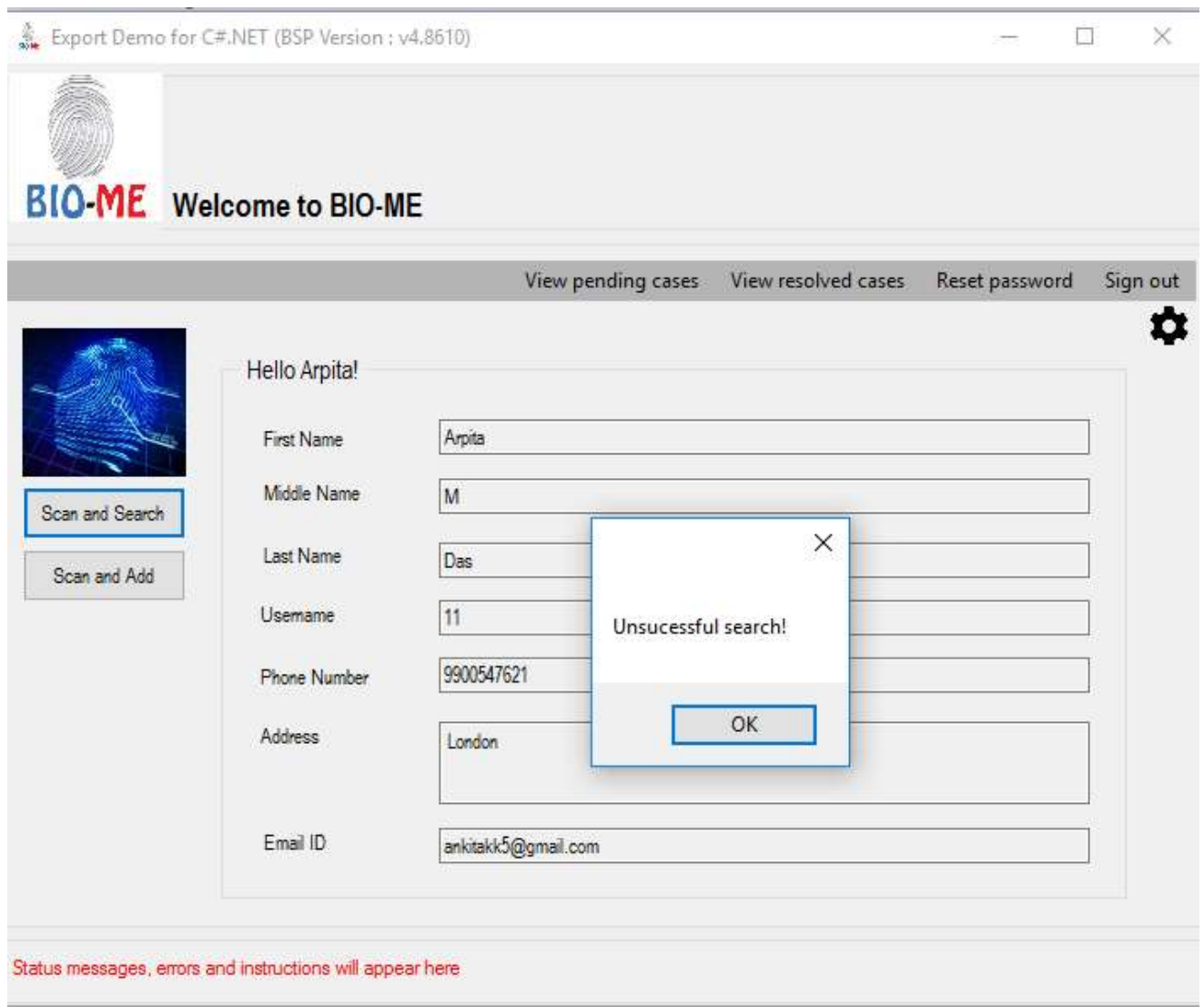


Fig. 32 On Unsuccessful Search

CONCLUSION

Bio-Me uses enBioScan-C1 HFDU08 Fingerprint Scanner to capture fingerprints and the NITGEN Matching Algorithm for matching them to identify people in distress.

Fingerprints are safe from misuse as they are stored as a secure template from which the fingerprints cannot be reproduced.

Has an easy-to-use UI and hence can be comfortably operated by both IT savvy as well as the technologically naïve.

It can be deployed at Police stations, Hospitals etc.

FUTURE ENHANCEMENTS

Bio-Me can be integrated with the Indian ID program-Aadhar by making use of Aadhar KYC policies.

It can be used to identify repeat traffic offenders.

Another application is to record criminal/domestic incidents and provide Clearance Certificates

Further Optimization of Matching process and Real-time handling of multi-user input.

BIBLIOGRAPHY

- <http://www.androidauthority.com>
- <http://www.ravirajtech.com>
- <http://www.nbcphiladelphia.com>
- <https://namus.gov>
- <http://www.hindustantimes.com>
- <http://www.nic.in/projects/missing-persons-information-system>
- <http://www.forensicsciencesimplified.org>
- <https://authportal.uidai.gov.in/web/uidai/developer>
- <http://pubs.sciepub.com>
- <http://web.cs.ucla.edu/honors/UPLOADS/andrew/thesis.pdf>
- Fingerprint Matching, IEEE, 2010
- An Identity-Authentication System Using Fingerprints, Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com Vol. 2, Issue 6, November- December 2012
- Fingerprint Recognition Using Level 3 Features, June 2014,ISSN (Print) : 2319-5940
- Biometric Security Using Finger Print Recognition, Subhra Mazumdar, Venkata Dhulipala, University of California, San Diego
- Personal Authentication using Fingerprint Biometric System, March 2014, ISSN (Print): 2320-9798