

SHIVAJI UNIVERSITY

An Biometric Approach to License Verification

by

Author Name

A project report submitted in partial fulfillment for the
Bachelor of Engineering

in the

Department of Computer Science and Engineering
Sharad Institute of Technology College of Engineering, Yadrav-Ichalkaranji

April 2015

SHIVAJI UNIVERSITY

Abstract

Department of Computer Science and Engineering
Sharad Institute of Technology College of Engineering, Yadrav-Ichalkaranji

Bachelor of Engineering

The term biometrics refers to a users personal characteristics such as fingerprint, iris, or hand geometry and users behavioral such as key stroke. Biometric data belongs to a particular person therefore it verifies the user by means of their personal attributes. This data is unique. It is very rare that the two biometric data are the same if they come from different persons, even twins are not the exception. As the data is the users personal characteristics, it cannot be transferred to others and this data cannot be easily stolen or given away to others even if a particular user wishes to do so. As a result, the biometric authentication process guarantees that whoever presents the biometric data is an authentic user.

Acknowledgements

We take this opportunity to express our profound gratitude and deep regards to our guide Mr. Pudale A. H. sir and Mr. M. I. Mullani sir for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to our Principal, Dr. S. A. Khot and our H.O.D., Mr. O. D. Joshi sir for their cordial support, valuable information and guidance, which helped us in completing this task through various stages. We are obliged to all faculty and staff members of all departments of our Sharad Institute of Technology, College of Engineering, Yadrav for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our assignment. Lastly, we thank almighty, our parents, brothers, sisters, friends and our colleagues for their constant encouragement without which this assignment would not be possible.

Ms. Arati Ashok Gupta

Mr. Ajinkya Sanjeev Mane

Mr. Shreyas Ganesh Mane

Mr. Ajinkya Babasaheb Chougule

Contents

| | |
|--|------------|
| Abstract | i |
| Acknowledgements | ii |
| List of Figures | iii |
| 1 Introduction | 1 |
| 1.1 Introduction about project | 1 |
| 2 Literature Review | 3 |
| 2.1 Literature review of project | 3 |
| 3 Objective and scope | 8 |
| 3.1 Objective | 8 |
| 3.2 Scope | 8 |
| 3.3 Out of scope | 9 |
| 4 Requirement analysis | 10 |
| 4.1 Minimum hardware requirements | 10 |
| 4.2 Minimum software requirement | 11 |
| 5 UML diagram | 17 |
| 5.1 UML diagrams | 17 |
| 5.1.1 Types of UML Diagrams | 18 |
| 5.1.2 Modules | 18 |
| 5.2 Usecase diagram | 19 |
| 5.3 Usecase scenario | 20 |
| 5.4 Sequence diagram | 21 |
| 5.5 Class diagram | 22 |
| 5.6 Deployment diagram | 24 |
| 6 Coding | 25 |
| 6.1 Coding standards | 25 |
| 6.1.1 Indentation | 25 |
| 6.1.2 Inline comments | 25 |
| 6.2 Classes, subroutines, functions, and methods | 26 |
| 6.2.1 Variable names | 26 |

| | | |
|----------|---|-----------|
| 6.2.2 | Use of braces | 26 |
| 6.2.3 | Compiler warnings | 27 |
| 6.3 | Review of code | 27 |
| 7 | Testing | 31 |
| 7.1 | Static vs. Dynamic testing | 31 |
| 7.1.0.1 | The box approach | 31 |
| 7.2 | White-box testing | 31 |
| 7.3 | Black-box testing | 33 |
| 7.4 | Unit testing | 34 |
| 7.5 | Integration testing | 34 |
| 7.6 | Validation testing | 34 |
| 7.7 | System testing | 35 |
| 7.7.0.2 | Why is system testing so important? | 35 |
| 7.7.1 | Snap shots | 37 |
| 8 | Future Scope | 44 |
| 9 | Conclusion | 45 |
| | Bibliography | 46 |

List of Figures

| | | |
|-----|--|----|
| 5.1 | Usecase diagram | 19 |
| 5.2 | Sequence diagram | 21 |
| 5.3 | Sequence diagram | 22 |
| 5.4 | Sequence diagram | 22 |
| 5.5 | Class diagram | 23 |
| 5.6 | Deployment diagram | 24 |
| 6.1 | Use of braces | 26 |
| 7.1 | Front page of application | 37 |
| 7.2 | Administrator login | 38 |
| 7.3 | Checking credentials | 39 |
| 7.4 | License authentication | 40 |
| 7.5 | Verify thumb impression to adhar-based | 41 |
| 7.6 | License details | 42 |
| 7.7 | Adhar details | 43 |

Chapter 1

Introduction

1.1 Introduction about project

User authentication is a method to verify the users identity. User authentication can be accomplished in various different ways: by using what the user knows such as a password, what the user possesses such as a smart card, or what the user is in terms of biometrics. Traditional methods of user authentication such as passwords or smart cards authenticate the user using their knowledge or possessions. A password or a smart card can be easily stolen or given away to others. Thus, we cannot be certain whoever accesses the system is the person who has authorization. This leads to the consideration of using biometrics as an attribute to authenticate the user. The term biometrics refers to a users personal characteristics such as fingerprint, iris, or hand geometry and users behavioral such as key stroke. Biometric data belongs to a particular person therefore it verifies the user by means of their personal attributes. This data is unique. It is very rare that the two biometric data are the same if they come from different persons, even twins are not the exception. As the data is the users personal characteristics, it cannot be transferred to others and this data cannot be easily stolen or given away to others even if a particular user wishes to do so. As a result, the biometric authentication process guarantees that whoever presents the biometric data is an authentic user.

However, in some ways, the risk of using biometric data can be higher than with other methods. The accuracy of the verification is lower than with other methods. This is due to the imperfect imaging conditions (such as the fact that the user does not position his biometric data as exactly the same as when he first presents it to be stored in the system database) or the biometric features themselves are not stable (such as cuts or aging). As the users identity must be presented along with this biometric data in order to perform the user authentication, anonymous authentication is not possible.

The biometric authentication relates the user to his personal attributes therefore if the biometric data is compromised, an intruder can relate this information to the users identity; the users privacy is not guaranteed. Even though biometric objects such as fingers or eyes cannot be used without users consent, the biometric characteristics such as captured fingerprints can be stolen.

A users biometric data is in the public domain. A person will leave his fingerprint on any surface he touches or on any computer he operates. Hence, the user cannot keep his biometric data secret in the same way as he can with a password. Once the biometric data is compromised or stolen, it cannot be replaced or regenerated as other methods of authentication (such as password or smart card authentication) can. Therefore, it is very important to maintain the privacy of biometric data during authentication, as is the case with a credit card number. A credit card number is not secret since we might voluntarily cite it over the telephone or via the internet. However, we want to treat it as though it were private because we do not want such data to be spread around without restriction. Authentication in biometric protocol can be compromised in a number of ways: via an attack on the server storing the biometric code, an interception of the biometric data when read by the biometric reader, or an attack during biometric data transmission. Therefore, security of a biometric authentication protocol is especially important because biometric data are not easily replaceable as passwords or tokens if compromised or lost. Furthermore, it is vital to be able to verify the security property effectively to show what it holds. For example, the Needham Schroeder authentication protocol was developed in 1978 and was believed to satisfy the property of security until 1995, when it was found to be susceptible to attack by Gavin Lowe. Due to the increasing use of biometric data for authentication, both instead of and alongside traditional methods, development of robust biometric authentication protocols increasingly merits consideration in terms of research. One of the major considerations when using biometric data is its nature in which the biometric data is in the public domain. An intruder can capture the biometric data which is left on the surface the user touches and later use it to authenticate to the system as a legitimate user. Hence, the biometric Authentication works well if the verifier can prove that the biometric data comes from the live presentation of the user at the time of verification.

Chapter 2

Literature Review

2.1 Literature review of project

Biometric Identification Systems are widely used for unique identification of humans mainly for verification and identification. Biometrics is used as a form of identity access management and access control. So use of biometrics in student attendance management system is a secure approach. There are many types of biometric systems like fingerprint recognition, face recognition, voice recognition, iris recognition, palm recognition etc. In this project, we used Fingerprint recognition system. [1] A fingerprint is the pattern of ridges and valleys on the surface of a fingertip. The endpoints and crossing points of ridges are called minutiae. It is a widely accepted assumption that the minutiae pattern of each finger is unique and does not change during one's life. Ridge endings are the points where the ridge curve terminates, and bifurcations are where a ridge splits from a single path to two paths at a Y-junction.

When human fingerprint experts determine if two fingerprints are from the same Finger, the matching degree between two minutiae pattern is one of the most important factors. Thanks to the similarity to the way of human fingerprint experts and compactness of templates, the minutiae-based matching method is the most widely studied matching method. [2]

Fingerprints are considered to be the best and fastest method for biometric identification. They are secure to use, unique for every person and does not change in one's lifetime. Besides these, implementation of fingerprint recognition system is cheap, easy and accurate up to satiability. Fingerprint recognition has been widely used in both forensic and civilian applications. Compared with other biometrics features, fingerprint-based biometrics is the most proven technique and has the largest market shares. Not

only it is faster than other techniques but also the energy consumption by such systems is too less. Biometric user authentication can be used in a range of applications, from logging on to a computer locally, to remote user authentication in on-line banking, for example. Biometric user authentication is useful and efficient in supervised situations such as passport border control, but it is significantly more risky from a security point of view if it is used in non-supervised situations such as remote user authentication. For example, an imposter might use a rubber finger that

Replicates the real users fingerprint.

Detection of fake biometric data often relies on measurement of physiological signs such as temperature or pulse, but that is beyond the scope of this thesis. This thesis concerns internal security problems that might occur within the computer system and biometric reader. Can an imposter capture the presented biometric data and later insert it into the communication channel during authentication? This could be overcome by using a biometric sensor contained in the trusted platform module, for example. This would therefore guarantee that the biometric data read by the biometric reader is not stolen since it has been read by the trusted biometric reader.

As stated earlier, biometric authentication can be used instead of or to complement other methods of authentication. Different situations or applications require different types of protocol. Applications are diverse, and include login to a local PC, remote login, or signature creation. Therefore, several different biometric authentication protocols are considered in this thesis, each of which serves a different purpose and has different requirements in order to fulfill the intended purpose. As described earlier it is important to ensure the components involved in the biometric authentication system should be trusted, the is chosen among other authentication protocols as it serves the trustworthiness specification in order to secure the biometric data. The research provides the biometric authentication for general purpose. Therefore, this protocol can be used if any application needs biometric verification to prior accessing a resource or system. We intend to investigate the requirements of verification and analysis of the biometric authentication protocol that uses the trusted computing concept in order to increase the security level for biometric data. The biometric authentication protocol for specific purpose. This protocol uses cryptographic messages for security purpose. The outcome from the verification provides us with the important properties that a specific purposed biometric authentication should hold. A remote biometric authentication is selected in order to gather the verification requirements and considerations when designing and developing a remote biometric authentication.[3]

The Trusted Platform Module (TPM) is used to guarantee the correctness of the components WSE04 Biometric authentication protocol for special purpose i.e. signing a

document A document is shown via the secured viewer to prevent the forgery of the document by an attacker PS06 Biometric authentication protocol for remote login The Trusted Platform Module is used to sign biometricData to guarantee the origin of the data in order to research the verification and analysis of the biometric authentication protocols, we have chosen three different biometric authentication protocols which serve the purposes and security requirements differently among the others. We aim to show how the verification and analysis approach are different for different types of the biometric protocol and we can then conclude security requirements and verification approach in common.

- What is Fingerprint? A fingerprint is the pattern of ridges and valleys on the surface of a fingertip. The endpoints and crossing points of ridges are called minutiae. It is a widely accepted assumption that the minutiae pattern of each finger is unique and does not change during one's life. Ridge endings are the points where the ridge curve terminates, and bifurcations are where a ridge splits from a single path to two paths at a Y-junction.

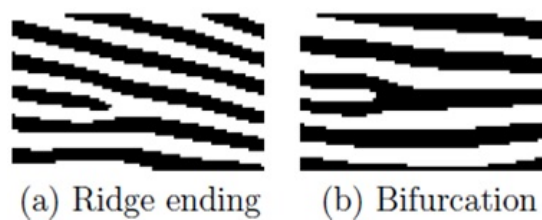


Figure illustrates an example of a ridge ending and a bifurcation. In this example, the black pixels correspond to the ridges, and the white pixels correspond to the valleys.

When human fingerprint experts determine if two fingerprints are from the same finger, the matching degree between two minutiae pattern is one of the most important factors. Thanks to the similarity to the way of human fingerprint experts and compactness of templates, the minutiae-based matching method is the most widely studied matching method.

- Why use fingerprints? Fingerprints are considered to be the best and fastest method for biometric identification. They are secure to use, unique for every person and do not change in one's lifetime. Besides these, implementation of fingerprint recognition system is cheap, easy and accurate up to satisfaction. Fingerprint recognition has been widely used in both forensic and civilian applications. Compared with other biometrics features, fingerprint-based biometrics is the most proven technique and has the largest market shares. Not only it is faster than other techniques but also the energy consumption by such systems is too less. [4]

- How Fingerprint Recognition works? Fingerprint images that are found or scanned are not of optimum quality. So we remove noises and enhance their quality. We extract features like minutiae and others for matching. If the sets of minutiae are matched with those in the database, we call it an identified fingerprint. After matching, we perform post-matching steps which may include showing details of identified candidate, marking attendance etc.
- What is Fingerprint? A fingerprint is the pattern of ridges and valleys on the surface of a fingertip. The endpoints and crossing points of ridges are called minutiae. It is a widely accepted assumption that the minutiae pattern of each finger is unique and does not change during one's life. Ridge endings are the points where the ridge curve terminates, and bifurcations are where a ridge splits from a single path to two paths at a Y-junction.

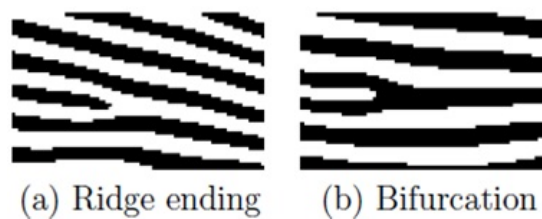


Figure illustrates an example of a ridge ending and a bifurcation. In this example, the black pixels correspond to the ridges, and the white pixels correspond to the valleys.

When human fingerprint experts determine if two fingerprints are from the same finger, the matching degree between two minutiae pattern is one of the most important factors. Thanks to the similarity to the way of human fingerprint experts and compactness of templates, the minutiae-based matching method is the most widely studied matching method.

- Why use fingerprints? Fingerprints are considered to be the best and fastest method for biometric identification. They are secure to use, unique for every person and do not change in one's lifetime. Besides these, implementation of fingerprint recognition system is cheap, easy and accurate up to satisfaction. Fingerprint recognition has been widely used in both forensic and civilian applications. Compared with other biometrics features, fingerprint-based biometrics is the most proven technique and has the largest market shares. Not only it is faster than other techniques but also the energy consumption by such systems is too less. [5]
- How Fingerprint Recognition works? Fingerprint images that are found or scanned are not of optimum quality. So we remove noises and enhance their quality. We extract features like minutiae and others for matching. If the sets of minutiae

are matched with those in the database, we call it an identified fingerprint. After matching, we perform post-matching steps which may include showing details of identified candidate, marking attendance etc.

Chapter 3

Objective and scope

3.1 Objective

An extended approach of digital of image processing is used for extracting of vehicle number plate Information from number plate is extracted can be used in many application like till payment, parking fee payment. The fingerprint matcher will match the pattern of scanning pattern and give the correct result. Using this documents vehicle are verified. [3]

3.2 Scope

1. Network management is a very complicated and important task.
2. Commands in wireless network.
3. Connection of scanner to tablet and accessing data from server.
4. Client application on tablet.
5. Fetching data from central server.
6. Establishing wireless client and server network .
7. Scanner for identify fingerprint.

3.3 Out of scope

1. Data can be accessed: Only the license verification is done.
2. Electricity: The tablet connected with finger scanner so both devices consume the battery of tablet.
3. Wireless connection: the good wireless internet connection is necessary to work with it.
4. It is not Possible to verify the license when damages to finger.
5. But it is capable to verify the license by other way.

Chapter 4

Requirement analysis

In our project, we have designed all the fields by gathering user requirements and needs, We have come to know that, before using this application users need all the details of specified requirements like:-

- Real time application
- Reliable
- Security
- Identification

By considering above points, we have tried to develop all these required points in our project, the main objective behind these points are to provide for users reliability. To achieve the projects implementation there are certain requirements to be fulfilled. so we have to divide these hardware and Software requirements in following parts.

4.1 Minimum hardware requirements

- Server
- Processor:- Minimum:1.60GHz Pentium Processor.
- RAM:- Minimum: 512MB.
- Hard Disk:- Minimum: 200MB.
- Aakash tab
- Fingerprint scanner

4.2 Minimum software requirement

- Operating System:
Windows XP Or Windows 7
- Tools and Technology
 - Android SDK
 - Java Development kit
 - mySql Database
 - Xamp Server

1. ANDROID SDK : Android is a set of software for mobile devices including Operation System, Middleware and Core Application, and a new Mobile Platform of Google. It is a complete mobile platform based on LINUX 2.6 Kernel that provides universal set of powerful Operation System, Comprehensive Library Set, abundant Multimedia User Interface and Phone Application. International Journal of Database Theory and Application Vol. 5, No. 2, June, 2012.[10] Android platform is produced to make new and innovative mobile application program for the developers to make full use of all functions connected to handset internet. The Android platform was developed by Google and later the Open Handset Alliance (OHA). The Open Handset Alliance is a group of mobile and technology companies who come together to accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience for consumers. The Android SDK provides the tools and APIs necessary to begin developing applications on the Android platform using the Java programming language.

Android is a multi-process system in which each application (and parts of the system) runs its own process. Security between applications and the system is enforced at the process level through standard Linux facilities such as user and group IDs that are assigned to applications.

Additional finer-grained security features are provided through a permission mechanism that enforces restrictions on the specific operations that a particular process can perform, and per-URI permissions for granting ad-hoc access to specific pieces of data. Linux Kernel. Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model. The kernel also acts as an abstraction layer between the hardware and the rest of the software stack. Libraries Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework.

- System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices
- Media Libraries - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG
- International Journal of Database Theory and Application Vol. 5, No. 2, June, 2012
- Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications
- LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view
- SGL - the underlying 2D graphics engine
- 3D libraries - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer
- Free Type - bitmap and vector font rendering
- SQLite - a powerful and lightweight relational database engine available to all applications
- Android Runtime. Android includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language. Every Android application runs in its own process, with its own instance of the Dalvik virtual machine. Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint. Application Framework. By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more. Applications. Android will ship with a set of core applications including an email client, SMS program, calendar, maps, browser, contacts, and others. All applications are written using the Java programming language.

2. MYSQL: MySQL ships with many command line tools, from which the main interface is 'mysql' client. Third parties have also developed tools to manage MySQL

servers. MySQL Utilities a set of utilities designed to perform common maintenance and administrative tasks. Originally included as part of the MySQL Workbench, the utilities are now a stand-alone download available from Oracle. Percona Toolkit a cross-platform toolkit for MySQL, developed in Perl. Percona Toolkit can be used to prove replication is working correctly, fix corrupted data, automate repetitive tasks, and speed up servers. Percona Toolkit is included with several Linux distributions such as CentOS and Debian, and packages are available for Fedora and Ubuntu as well. Percona Toolkit was originally developed as Maatkit, but as of late 2011, Maatkit is no longer developed. MySQL Programming MySQL works on many system platforms, including AIX, BSDi, FreeBSD, HP-UX, eCom-Station, i5/OS, IRIX, Linux, OS X, Microsoft Windows, NetBSD, Novell NetWare, OpenBSD, OpenSolaris, OS/2 Warp, QNX, Oracle Solaris, Symbian, SunOS, SCO OpenServer, SCO UnixWare, Sanos and Tru64. A port of MySQL to OpenVMS also exists.[32] MySQL is written in C and C++. Its SQL parser is written in yacc, but it uses a home-brewed lexical analyzer. Many programming languages with language-specific APIs include libraries for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C sharp and VB are most commonly used) and the JDBC driver for Java. In addition, an ODBC interface called MyODBC allows additional programming languages that support the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The HTSQL URL-based query method also ships with a MySQL adapter, allowing direct interaction between a MySQL database and any web client via structured URLs. Features: MySQL is offered under two different editions: the open source MySQL Community Server and the proprietary Enterprise Server.[34] MySQL Enterprise Server is differentiated by a series of proprietary extensions which install as server plugins, but otherwise shares the version numbering system and is built from the same code base. Major features as available in MySQL 5.6:

- A broad subset of ANSI SQL 99, as well as extensions
- Cross-platform support
- Stored procedures, using a procedural language that closely adheres to SQL/PSM[35]
- Triggers
- Cursors
- Updatable views
- Online DDL when using the InnoDB Storage Engine.
- Information schema
- Performance Schema

A set of SQL Mode options to control runtime behavior, including a strict mode to better adhere to SQL standards. X/Open XA distributed transaction processing (DTP) support; two phase commit as part of this, using the default InnoDB storage engine Transactions with savepoints when using the default InnoDB Storage Engine. The NDB Cluster Storage Engine also supports transactions. ACID compliance when using InnoDB and NDB Cluster Storage Engines[36] SSL support Query caching Sub-SELECTs (i.e. nested SELECTs) Built-in Replication support (i.e. Master-Master Replication Master-Slave Replication) with one master per slave, many slaves per master.[37] Multi-master replication is provided in MySQL Cluster,[38] and multi-master support can be added to unclustered configurations using Galera Cluster.[39] Full-text indexing and searching[b] Embedded database library Unicode support[c] Partitioned tables with pruning of partitions in optimizer Shared-nothing clustering through MySQL Cluster Multiple storage engines, allowing one to choose the one that is most effective for each table in the application.[d] Native storage engines InnoDB, MyISAM, Merge, Memory (heap), Federated, Archive, CSV, Blackhole, NDB Cluster. Commit grouping, gathering multiple transactions from multiple connections together to increase the number of commits per second. The developers release minor updates of the MySQL Server approximately every two months. The sources can be obtained from MySQL's website or from MySQL's Bazaar repository, both under the GPL license.Xampp Server: XAMPP requires only one zip, tar, 7z, or exe file to be downloaded and run, and little or no configuration of the various components that make up the web server is required. XAMPP is regularly updated to incorporate the latest releases of Apache, MySQL, PHP and Perl. It also comes with a number of other modules including OpenSSL and phpMyAdmin. Self-contained, multiple instances of XAMPP can exist on a single computer, and any given instance can be copied from one computer to another.It is offered in both a full, standard version and a smaller version.

Officially, XAMPP's designers intended it for use only as a development tool, to allow website designers and programmers to test their work on their own computers without any access to the Internet. To make this as easy as possible, many important security features are disabled by default.[2] In practice, however, XAMPP is sometimes used to actually serve web pages on the World Wide Web[citation needed]. A special tool is provided to password-protect the most important parts of the package.[3] XAMPP also provides support for creating and manipulating databases in MySQL and SQLite among others. Once XAMPP is installed, it is possible to treat a localhost like a remote host by connecting using an FTP client.

Using a program like FileZilla has many advantages when installing a content management system (CMS) like Joomla or WordPress. It is also possible to connect to localhost via FTP with an HTML editor.

The default FTP user is "newuser", the default FTP password is "wampp". The default MySQL user is "root" while there is no default MySQL password.

3. **JAVA DEVELOPMENT KIT** The Java Development Kit[10] (JDK) is an implementation of either one of the Java SE, Java EE or Java ME platforms released by Oracle Corporation in the form of a binary product aimed at Java developers on Solaris, Linux, Mac OS X or Windows. The JDK includes a private JVM and a few other resources to finish the recipe to a Java Application. Since the introduction of the Java platform, it has been by far the most widely used Software Development Kit (SDK). On 17 November 2006, Sun announced that it would be released under the GNU General Public License (GPL), thus making it free software. This happened in large part on 8 May 2007, when Sun contributed the source code to the OpenJDK.

4. **JDK contents** The JDK has as its primary components a collection of programming tools, including:

- **appletviewer** this tool can be used to run and debug Java applets without a web browser
- **apt** the annotation-processing tool
- **extcheck** a utility which can detect JAR-file conflicts
- **Idlj** the IDL-to-Java compiler. This utility generates Java bindings from a given Java IDL file.
- **Jabswitch** the Java Access Bridge. Exposes assistive technologies on Microsoft Windows systems.
- **Java** the loader for Java applications. This tool is an interpreter and can interpret the class files generated by the javac compiler. Now a single launcher is used for both development and deployment. The old deployment launcher, jre, no longer comes with Sun JDK, and instead it has been replaced by this new java loader.
- **javac** the Java compiler, which converts source code into Java bytecode
- **javadoc** the documentation generator, which automatically generates documentation from source code comments
- **jar** the archiver, which packages related class libraries into a single JAR file. This tool also helps manage JAR files.

- `javafxpackager` tool to package and sign JavaFX applications
- `jarsigner` the jar signing and verification tool
- `javah` the C header and stub generator, used to write native methods
- `javap` the class file disassembler
- `javaws` the Java Web Start launcher for JNLP applications
- `JConsole` Java Monitoring and Management Console
- `jdb` the debugger
- `jhat` Java Heap Analysis Tool (experimental)
- `jinfo` This utility gets configuration information from a running Java process or crash dump. (experimental)
- `jmap` This utility outputs the memory map for Java and can print shared object memory maps or heap memory details of a given process or core dump. (experimental)
- `jmc` Java Mission Control
- `jps` Java Virtual Machine Process Status Tool lists the instrumented Hotspot Java Virtual Machines (JVMs) on the target system. (experimental)
- `jrnscrip` Java command-line scriptshell.
- `jstack` utility which prints Java stack traces of Java threads (experimental)
- `jstat` Java Virtual Machine statistics monitoring tool (experimental)
- `jstatd` `jstat` daemon (experimental)
- `keytool` tool for manipulating the keystore
- `pack200` JAR compression tool
- `policytool` the policy creation and management tool, which can determine policy for a Java runtime, specifying which permissions are available for code from various sources
- `VisualVM` visual tool integrating several command-line JDK tools and lightweight[clarification needed] performance and memory profiling capabilities
- `wsimport` generates portable JAX-WS artifacts for invoking a web service.
- `xjc` Part of the Java API for XML Binding (JAXB) API. It accepts an XML schema and generates Java classes

Chapter 5

UML diagram

Designing before one start the actually coding part is primary importance steps, proper designing of project helps one to understand how exactly, the project takes shape, there are two major factors in our project as follows:

- Biometrics
- Image Processing

In this we have made designs for each and every factor, as per the requirement of all factors what we should do proportionally to factor, we have to make design.

- Biometrics: Finger Scanner: Biometric Identification Systems are widely used for unique identification of humans mainly for verification and identification. Biometrics is used as a form of identity access management and access control. So use of biometrics in this system is a secure approach. There are many types of biometric systems like fingerprint recognition, face recognition, voice recognition, iris recognition, palm recognition etc. In this project, we used fingerprint recognition system.
- Image Processing: Image processor :
The purpose of this requirement analysis is to establish the major requirements necessary to develop this project.

5.1 UML diagrams

Goals of UML The primary goals in the design of the UML were:

- Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models. Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language
- Encourage the growth of the OO tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices

5.1.1 Types of UML Diagrams

There are following types of UML diagrams:

- Use case diagram
- Sequence diagram
- Class diagram
- Deployment diagram

5.1.2 Modules

- User
- Media Device
- Webcam

5.2 Usecase diagram

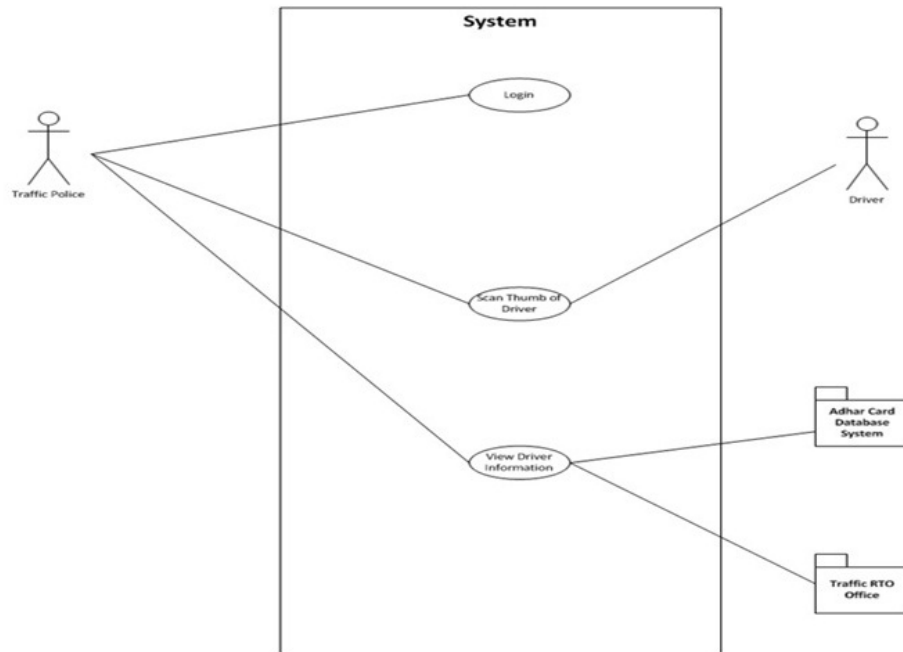


FIGURE 5.1: Usecase diagram

5.3 Usecase scenario

| Use case | Description |
|-------------------------|--|
| Login | <ol style="list-style-type: none"> 1. Traffic police will Enter user name and Password to start device. 2. Device will Reply as successfil login. 3. If user name or password is wrong then device will reply as unsuccessful login 4. Device will work after successful login. will move forward. |
| Scan Thumb of driver | <ol style="list-style-type: none"> 1. Driver gives his thumb impression to scan. 2. Driver waits for result. |
| View Driver information | <ol style="list-style-type: none"> 1. After scanning of drivers thumb. 2. Device will send request to adhar card database for drivers personal information. 3. When information is avialable in the adhar card database then it will send to device and device will send requset to the RTO database for license verification. 4. After license verification traffic police will take decision. 5. If information is not avialable in adharcad database. then traffic police will not get information from RTO database then traffic police will take decision. |

5.4 Sequence diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows the specification of simple runtime scenarios in a graphical manner. If the lifeline is that of an object, it demonstrates a role. Note that leaving the instance name blank can represent anonymous and unnamed instances. Messages, written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages.[1] If a caller sends a synchronous message, it must wait until the message is done, such as invoking a subroutine. If a caller sends an asynchronous message, it can continue processing and doesn't have to wait for a response. Asynchronous calls are present in multithreaded applications and in message-oriented middleware. Activation boxes, or method-call boxes, are opaque rectangles drawn on top of lifelines to represent that processes are being performed in response to the message (Execution Specifications in UML).

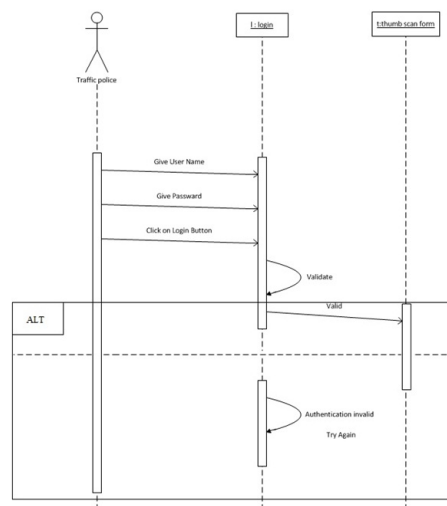


FIGURE 5.2: Sequence diagram

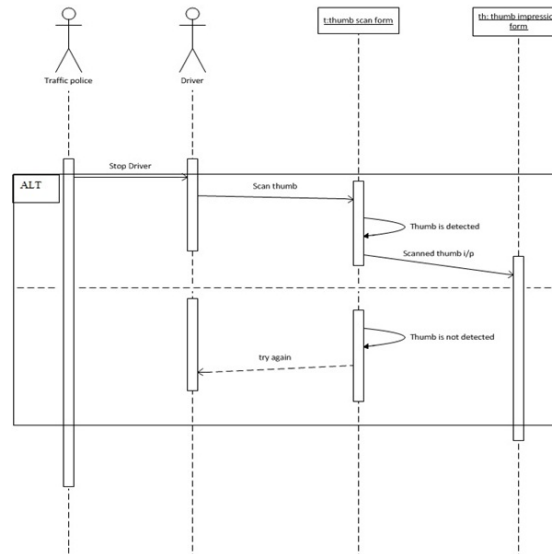


FIGURE 5.3: Sequence diagram

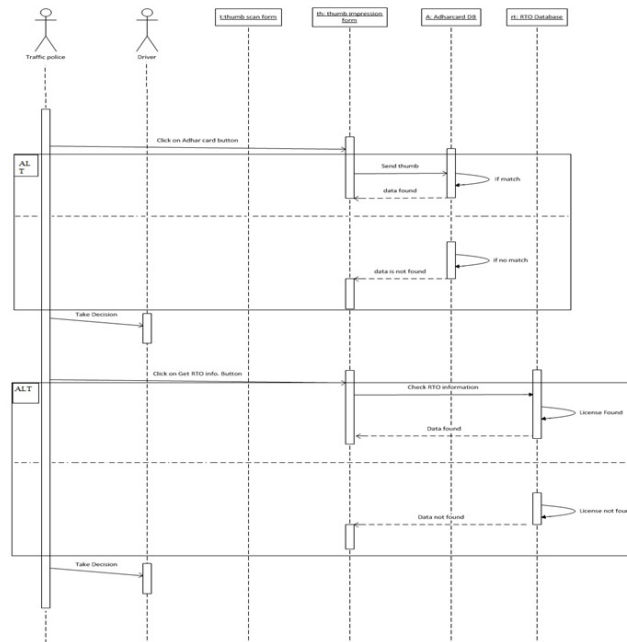


FIGURE 5.4: Sequence diagram

5.5 Class diagram

A UML class diagram describes the object and information structures used by your application both internally and in communication with its users. It describes the information without reference to any particular implementation. Its classes and relationships can be implemented in many ways, such as database tables, XML nodes, or compositions of software objects. The class diagram is the main building block of object oriented

modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling.[1] The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. Class diagram, classes are represented with boxes which contain three parts:

- The top part contains the name of the class. It is printed in Bold, centered and the first letter capitalized.
- The middle part contains the attributes of the class. They are left aligned and the first letter is lower case.
- The bottom part gives the methods or operations the class can take or undertake. They are also left aligned and the first letter is lower case.
- Class: A definition of objects that share given structural or behavioral characteristics.
- Attribute: A typed value attached to each instance of a classifier.
- Operation: A method or function that can be performed by instances of a classifier.

Class diagram:

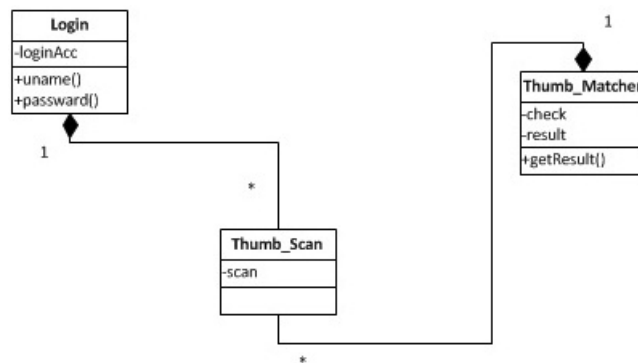


FIGURE 5.5: Class diagram

5.6 Deployment diagram

A deployment diagram in the Unified Modeling Language models the physical deployment of artifacts on nodes. To describe a web site, for example, a deployment diagram would show what hardware components ("nodes") exist (e.g., a web server, an application server, and a database server), what software components ("artifacts") run on each node (e.g., web application, database), and how the different pieces are connected (e.g. JDBC, REST, RMI). The nodes appear as boxes, and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have subnodes, which appear as nested boxes. A single node in a deployment diagram may conceptually represent multiple physical nodes, such as a cluster of database servers. There are two types of Nodes:

- Device Node
- Execution Environment Node Device nodes are physical computing resources with processing memory and services to execute software, such as typical computers or mobile phones. An execution environment node (EEN) is a software computing resource that runs within an outer node and which itself provides a service to host and execute other executable software elements.

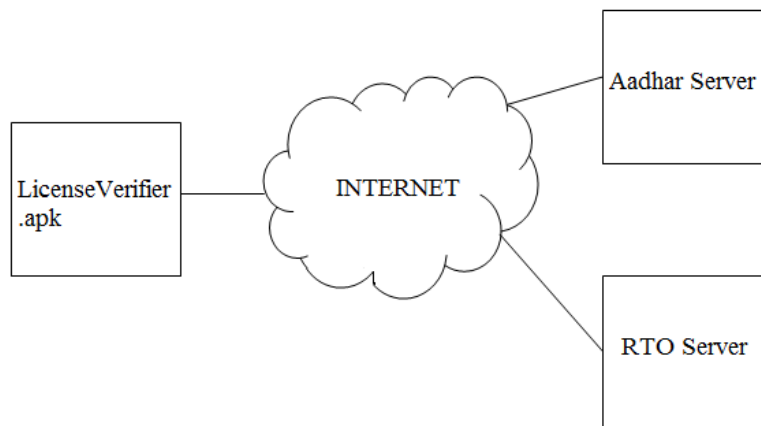


FIGURE 5.6: Deployment diagram

Chapter 6

Coding

6.1 Coding standards

General coding standards pertain to how the developer writes code. The SISEPG has come up with a small set of items it feels should be followed regardless of the programming language being used.

6.1.1 Indentation

Proper and consistent indentation is important in producing easy to read and maintainable programs. Indentation should be used to:

- Emphasize the body of a control statement such as a loop or a select statement
- Emphasize the body of a conditional statement
- Emphasize a new scope block

A minimum of 3 spaces shall be used to indent. Generally, indenting by three or four spaces is considered to be adequate. Once the programmer chooses the number of spaces to indent by, then it is important that this indentation amount be consistently applied throughout the program. Tabs shall not be used for indentation purposes.

6.1.2 Inline comments

Inline comments explaining the functioning of the subroutine or key aspects of the algorithm shall be frequently used. See section 4.0 for guidance on the usage of inline comments.

6.2 Classes, subroutines, functions, and methods

Keep subroutines, functions, and methods reasonably sized. This depends upon the language being used. For guidance on how large to make software modules and methods, see section 4.0. A good rule of thumb for module length is to constrain each module to one function or action (i.e. each module should only do one thing). If a module grows too large, it is usually because the programmer is trying to accomplish too many actions at one time. The names of the classes, subroutines, functions, and methods shall have verbs in them. That is the names shall specify an action,

6.2.1 Variable names

Variable shall have meaningful names that convey to a casual observer, the intent of its use. Variables shall be initialized prior to use.

6.2.2 Use of braces

In some languages, braces are used to delimit the bodies of conditional statements, control constructs, and blocks of scope. Programmers shall use either of the following bracing styles: It is felt that the former brace style is more readable and leads to neater-

```

public class AdministratorLogin extends Activity {
    private ProgressDialog pDialog;

    JSONParser jsonParser = new JSONParser();
    EditText username;
    EditText password;

    private static String url_checkValidAdmin = "http://10.0.2.2/LicenseApplication/check_valid_admin.php";
    private static final String TAG_SUCCESS = "success";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.administrator_login);

        username = (EditText) findViewById(R.id.loginUsername);
        password = (EditText) findViewById(R.id.loginPassword);
        Button btnCheckAdministrator = (Button) findViewById(R.id.btnLogin);
        Button btnNewRegister = (Button) findViewById(R.id.btnLinkToRegisterScreen);
        Button btnForgetPassword = (Button) findViewById(R.id.btnLinkToForgetPasswordScreen);

        btnForgetPassword.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(getApplicationContext(), AdministratorForgetPassword.class);
                startActivity(i);
            }
        });

        btnNewRegister.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent i = new Intent(getApplicationContext(), CreateNewAdministrator.class);
            }
        });
    }
}

```

FIGURE 6.1: Use of braces

looking code than the latter style, but either use is acceptable. Whichever style is used, be sure to be consistent throughout the code. When editing code written by another

author, adopt the style of bracing used. Braces shall be used even when there is only one statement in the control block. For example:

6.2.3 Compiler warnings

Compilers often issue two types of messages: warnings and errors. Compiler warnings normally do not stop the compilation process. However, compiler errors do stop the compilation process, forcing the developer to x the problem and recompile. Compiler and linker warnings shall be treated as errors and xed. Some compilers have options to suppress or enhance compile-time warning messages. Developers shall study the documentation and/or man pages associated with a compiler and choose the options which fully enable the compilers code-checking features.

6.3 Review of code

```

check_valid_admin:
<?php
$response = array();

require_once __DIR__ . '/db_connect.php';

$db = new DB_CONNECT();

if (isset($_POST['username']) && isset($_POST['password'])) {
    $username = $_POST['username'];
    $password = $_POST['password'];

    $result = mysql_query("SELECT *FROM admin WHERE username
= '$username' and password= '$password'");

    if (mysql_num_rows($result) > 0) {
        $response["success"] = 1;
        $response["message"] = "Valid User";
        echo json_encode($response);
    }
    else
    {
        $response["success"] = 0;
        $response["message"] = "Invalid User";
        echo json_encode($response);
    }
}
else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";
}

```

```

        // echoing JSON response
        echo json_encode($response);
    }
    ?>

check_valid_adminKey:
<?php
$response = array();

require_once __DIR__ . '/db_connect.php';

$db = new DB_CONNECT();

if (isset($_POST['adminKey'])) {
    $username = $_POST['adminKey'];

    $result = mysql_query("SELECT *FROM admin WHERE adminKey
= '$adminKey'");

    if (mysql_num_rows($result) > 0) {
        $response["success"] = 1;
        $response["message"] = "Valid User";
        echo json_encode($response);
    }
    else
    {
        $response["success"] = 0;
        $response["message"] = "Invalid User";
        echo json_encode($response);
    }
}
else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    // echoing JSON response
    echo json_encode($response);
}
?>

```

```

check_valid_Image
<?php
$response = array();

require_once __DIR__ . '/db_connect.php';

$db = new DB_CONNECT();

if (isset($_POST['data'])) {
    $data = $_POST['data'];

```

```

$result = mysql_query("SELECT *FROM test WHERE data = '$data'");

    if (mysql_num_rows($result) > 0) {
        $response["success"] = 1;
        $response["message"] = "Valid User";
        echo json_encode($response);
    }
    else
    {
        $response["success"] = 0;
        $response["message"] = "Invalid User";
        echo json_encode($response);
    }
}
else {
// required field is missing
$response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    // echoing JSON response
echo json_encode($response);
}
?>

```

```

create_administrator:

```

```

<?php
$response = array();

// check for required fields
if (isset($_POST['username']) && isset($_POST['password']) &&
    isset($_POST['confirmPassword']) && isset($_POST['securityQuestion'])
    && isset($_POST['securityQuestionAnswer'])
    && isset($_POST['securityKey'])) {

    $name = $_POST['username'];
    $password = $_POST['password'];
    $description = $_POST['confirmPassword'];
    $name = $_POST['securityQuestion'];
    $name = $_POST['securityQuestionAnswer'];
    $name = $_POST['securityKey'];

    // include db connect class
    require_once __DIR__ . '/db_connect.php';

    // connecting to db
    $db = new DB_CONNECT();

    // mysql inserting a new row
    $result = mysql_query("INSERT INTO administratordetails(username, password, confirmPassword,
        VALUES('$username', '$password', '$confirmPassword',
'$securityQuestion','$securityQuestionAnswer','$securityKey')");

    // check if row inserted or not

```

```
if ($result) {
    // successfully inserted into database
    $response["success"] = 1;
    $response["message"] = "Administrator successfully created.";

    // echoing JSON response
    echo json_encode($response);
} else {
    // failed to insert row
    $response["success"] = 0;
    $response["message"] = "Oops! An error occurred.";

    // echoing JSON response
    echo json_encode($response);
}
} else {
    // required field is missing
    $response["success"] = 0;
    $response["message"] = "Required field(s) is missing";

    // echoing JSON response
    echo json_encode($response);
}
?>
```

Chapter 7

Testing

7.1 Static vs. Dynamic testing

There are many approaches to software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100Static testing involves verification, whereas dynamic testing involves validation. Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test-cases will detect errors which are introduced by mutating the source code.

7.1.0.1 The box approach

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

7.2 White-box testing

White-box testing (also known as clear box testing, glass box testing, and transparent box testing and structural testing) tests internal structures or workings of a program,

as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a systemlevel test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements. Techniques used in white-box testing include:

- API testing (application programming interface) testing of the application using public and private APIs
- Code coverage creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
- Fault injection methods intentionally introducing faults to gauge the efficacy of testing strategies
- Mutation testing methods
- Static testing methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for:

- Function coverage, which reports on functions executed
- Statement coverage, which reports on the number of lines executed to complete the test

100 percent statement coverage ensures that all code paths, or branches (in terms of control flow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly

7.3 Black-box testing

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing. Specification-based testing aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional. Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations. One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight." Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested. This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

7.4 Unit testing

Unit testing is the testing of individual hardware or software units or groups of related units. Using white box testing techniques, testers verify that the code does what it is intended to do at a very low structural level. Unit testing focuses verification effort on the smallest unit of software design the Software component or module. Using the component-level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. The relative complexity of tests and uncovered errors is limited by the constrained scope established for unit testing. The unit test is white-box oriented, and the step can be conducted in parallel for multiple components. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

7.5 Integration testing

Integration testing is testing in which software components, hardware components, or both are combined and tested to evaluate the interaction between them. The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top- Down Integration testing.

- Bottom-up integration testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
- Top-Down integration testing, the highest-level modules are tested first and progressively lower-level modules are tested after that. In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing.

7.6 Validation testing

Software validation testing is essentially a task carried out by a software tester. The aim is to check, if the software has been made in lines with the requirements of the client. While verification is a quality control process, quality assurance process carried out before the software is ready for release is known as validation testing. Its goals are to validate and be confident about the software product or system, that it fulfills the requirements given by the customer. The acceptance of the software from the end

customer is also its part. Often the testing activities are introduced early in the software development life cycle. The two major areas when it should take place are in the early stages of software development and towards the end, when the product is ready for release. In other words, it is acceptance testing which is a part of validation testing.

7.7 System testing

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

7.7.0.2 Why is system testing so important?

- System Testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment which is very close to the production environment where the application will be deployed.
- System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.

| Test Case | Test Step | Expected Result | Result |
|---------------------|---|----------------------------|------------|
| 1.Check valid Admin | Shows the user name and password field | Should display same screen | Successful |
| 2.Check valid Admin | Enter Username and Password | Should display same screen | Successful |
| 3.Check valid Admin | Click On Submit Button | AdminSuccess Screen | Successful |
| 4.Check valid Admin | Select registered image | Should display Success | Successful |
| 5.Check Valid User | Match With Adhar Database and Match with License database | Should Display Match | Successful |

7.7.1 Snap shots



FIGURE 7.1: Front page of application

This snap shot shows the front page of our application. click on administrator login for login the admin. when we click on login button we enters into the new form

ADMINISTRATOR LOGIN

ADMINISTRATOR LOGIN

USERNAME

admin

PASSWORD

.....

Login

FIGURE 7.2: Administrator login

here administrator have give user name and password for entering in the main form.

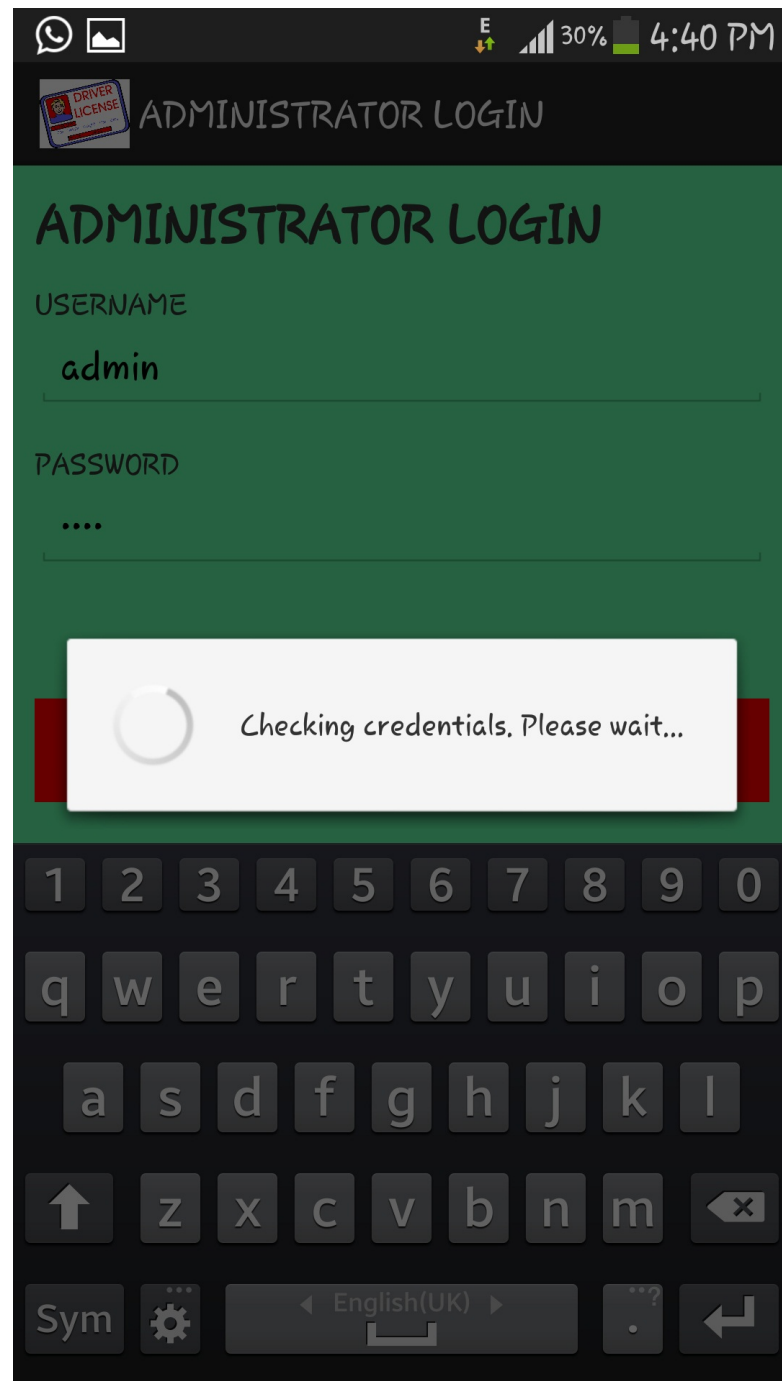


FIGURE 7.3: Checking credentials

This snap shot shows the checking of credentials of administrator.if username and password is correct then we get information otherwise it is not valid user.



FIGURE 7.4: License authentication

This is snapshot of license authentication. here user authenticated and admin get information about user from license server. and will get the result.

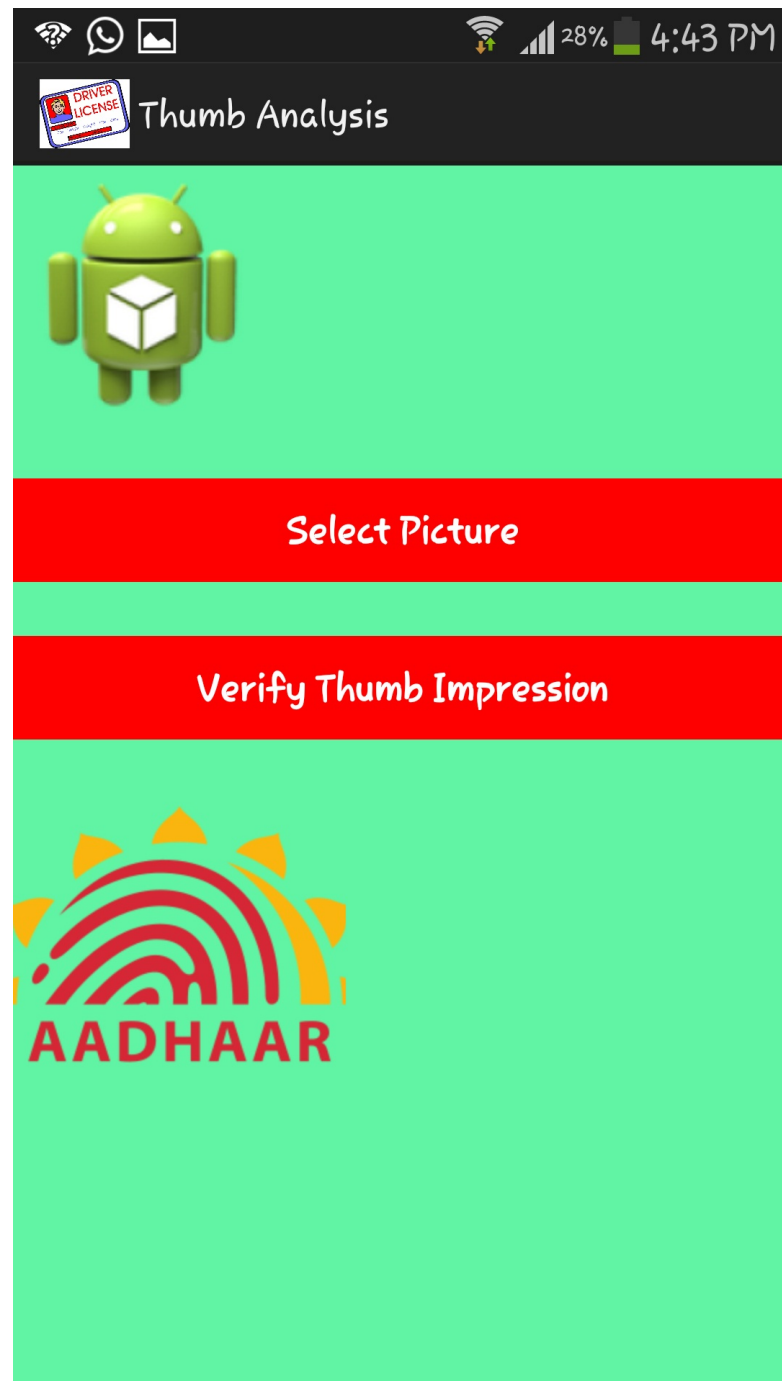
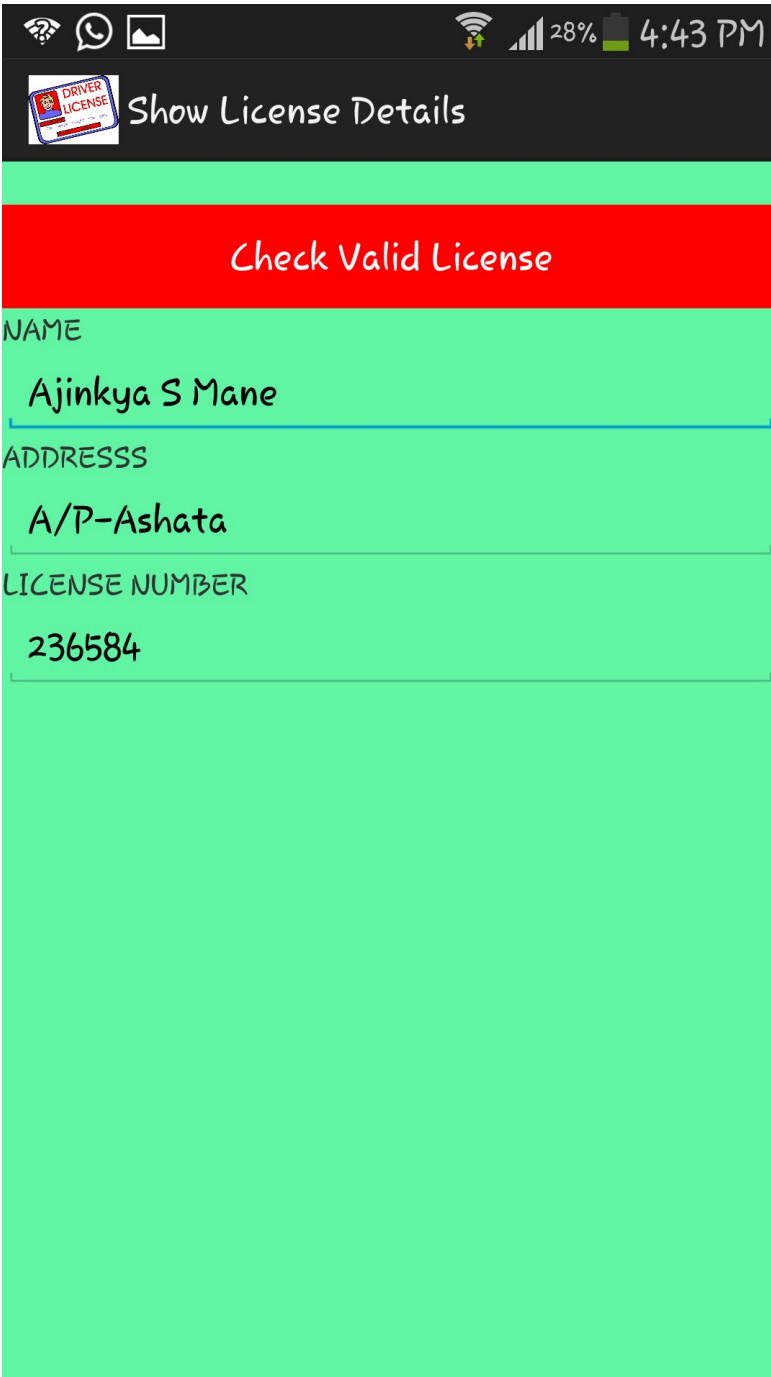


FIGURE 7.5: Verify thumb impression to adhar-based

Here verify the thumb impression of the user and it will check with the help of adhar card server and shows the result.

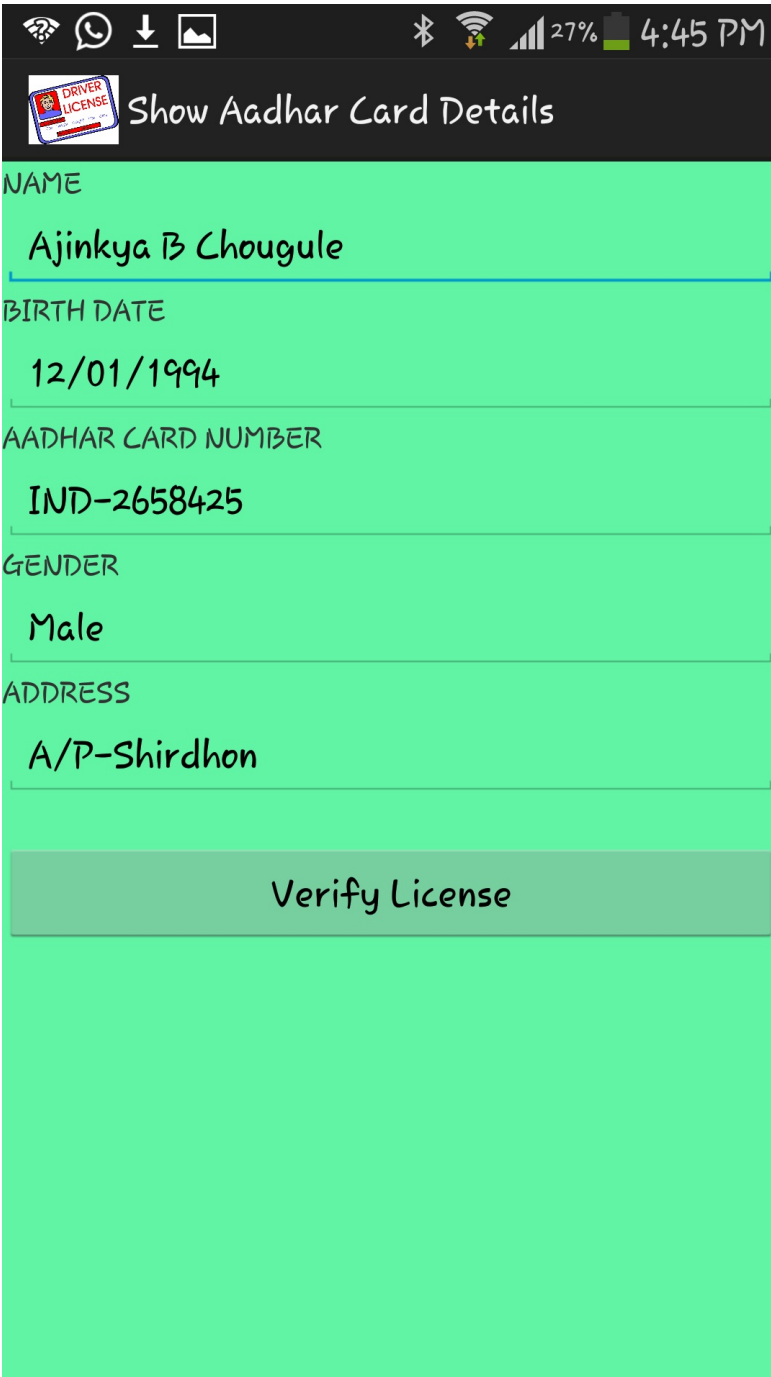


The screenshot shows a mobile application interface for license verification. At the top, there is a status bar with icons for Wi-Fi, WhatsApp, and a gallery, along with signal strength, 28% battery, and the time 4:43 PM. Below the status bar is a dark header with a 'DRIVER LICENSE' icon and the text 'Show License Details'. A red button labeled 'Check Valid License' is positioned below the header. The main content area has a light green background and contains three input fields: 'NAME' with the value 'Ajinkya S Mane', 'ADDRESS' with the value 'A/P-Ashata', and 'LICENSE NUMBER' with the value '236584'.

| Field | Value |
|----------------|----------------|
| NAME | Ajinkya S Mane |
| ADDRESS | A/P-Ashata |
| LICENSE NUMBER | 236584 |

FIGURE 7.6: License details

If license is found then it shows the license number and information about user.



The screenshot shows a mobile application interface with a black header bar. On the left of the header is a small icon of a driver's license with the text 'DRIVER LICENSE'. To the right of the icon, the text 'Show Aadhar Card Details' is displayed in white. Below the header, the background is a solid light green color. The form contains several text input fields with labels on the left and values inside the fields. The labels are in all caps: NAME, BIRTH DATE, AADHAR CARD NUMBER, GENDER, and ADDRESS. The values entered are: Ajinkya B Chougule, 12/01/1994, IND-2658425, Male, and A/P-Shirdhon. At the bottom of the form, there is a wide, light green button with the text 'Verify License' in black.

| Field Label | Value |
|--------------------|--------------------|
| NAME | Ajinkya B Chougule |
| BIRTH DATE | 12/01/1994 |
| AADHAR CARD NUMBER | IND-2658425 |
| GENDER | Male |
| ADDRESS | A/P-Shirdhon |

Verify License

FIGURE 7.7: Adhar details

it shows the adhar number and personal information

Chapter 8

Future Scope

- By using Internet we can fetch Driver License information at any place.
- By using image processing we can improve application to vehical verification.

Chapter 9

Conclusion

The tab is accessible and capable device with several channels for transferring authentication data. Different channels and authentication methods influence the level of security of authentication solutions. This project mainly comprised of development biometrics and image processing system. A general implementable idea is using portable devices along with wireless LAN or 3G networks.

Bibliography

- [1] James L Wayman, Anil K Jain, Davide Maltoni, and Dario Maio. *Biometric systems: technology, design and performance evaluation*. Springer Science & Business Media, 2005.
- [2] Nalini Ratha and Ruud Bolle. *Automatic fingerprint recognition systems*. Springer Science & Business Media, 2003.
- [3] Davide Maltoni, Dario Maio, Anil K Jain, and Salil Prabhakar. *Handbook of fingerprint recognition*. Springer Science & Business Media, 2009.
- [4] John D Woodward, Nicholas M Orlans, and Peter T Higgins. *Biometrics:[identity assurance in the information age]*. McGraw-Hill/Osborne New York, 2003.
- [5] Subhra Mazumdar and Venkata Dhulipala. Biometric security using finger print recognition. *University of California, San Diego*, page 3, 2008.