

## 1. INTRODUCTION

The design of secure authentication protocols is quite challenging, considering that various kinds of root kits reside in PCs (Personal Computers) to observe user's behavior and to make PC's untrusted devices. Involving human in authentication protocols, while promising, is not easy because of their limited capability of computation and memorization. Therefore, relying on users to enhance security necessarily degrades the usability. To that end, we propose two visual authentication protocols: one is a one-time-password protocol, and the other is a password-based authentication protocol.

Our approach to solving the problem is to introduce an intermediate device that bridges a human user and a terminal. Then, instead of the user directly invoking the regular authentication protocol, she invokes a more sophisticated but user-friendly protocol via the intermediate helping device. Every interaction between the user and an intermediate helping device is visualized using a Quick Response (QR) code.

The visual involvement of users in a security protocol boosts both the security of the protocol and is re-assuring to the user because she feels that she plays a role in the process. To securely implement visual security protocols, a Smartphone with a camera is used. Instead of executing the entire security protocol on the personal computer, part of security protocol is moved to the Smartphone. This visualization of some part of security protocols enhances security greatly and offers protection against hard-to-defend against attacks such as malware and key logging attack, while not degrading the usability. However, we note that our goal is not securing the authentication process against the shoulder surfing attacker who can see or compromise simultaneously both devices over the shoulder, but rather to make it hard for the adversary to launch the attack.

Common password stealing attempts can be prevented using two step verification. Two step verification can be done using one time passwords or secondary passwords. One time passwords are delivered to users using SMS, Time based One Password Generators like Google Authenticator or using dedicated devices like RSA SecureID. However these methods have many disadvantages. SMS delivery mechanisms need continuous network availability and its implementation is costly considering elements like International Roaming. Time based One Time Password Applications are

efficient in this case however they need frequent synchronization with the server and complex algorithms.

Sequoro solves this problem by using QR codes for encoding One Time Passwords in them. Thus its implementation provides users with a highly secure but economical solution. Sequoro is currently developed for Android platform. Thus it targets millions of users. Sequoro has been extended to create an Application Program Interface that website owners could easily implement in their sites.

## 2. PROBLEM DEFINITION

Common password stealing attempts can be prevented using two step verification. Two step verification can be done using one time passwords or secondary passwords. One time passwords are delivered to users using SMS, Time based One Time Password Generators like Google Authenticator or using dedicated devices like RSA SecureID. However these methods have many disadvantages. SMS delivery mechanisms need continuous network availability and its implementation is costly considering elements like International Roaming. Time based One Time Password Applications are efficient in this case however they need frequent synchronization with the server and complex algorithms.

Sequoro solves this problem by using QR codes for encoding One Time Passwords in them. Thus its implementation provides users with a highly secure but economical solution. Sequoro is currently developed for Android platform. Thus it targets millions of users. Sequoro has been extended to form an Application Program Interface that website owners could easily implement in their sites.

Sequoro aims at providing an economical way to provide secure login to websites. It is secure against common password stealing attacks including key loggers. We aim at realizations of protocols that not only improve the user experience but also resist challenging attacks, such as the key logger and malware attacks. Our protocols utilize simple technologies available in most out-of-the-box Smartphone devices. Our work indeed opens the door for several other directions that we would like to investigate as a future work. In future, we plan to implement our protocol on the smart glasses such as the Google glass, and conduct the user study.

## **3. SOFTWARE REQUIREMENT SPECIFICATION**

### **3.1. INTRODUCTION**

#### **3.1.1 PURPOSE OF THIS DOCUMENT**

This SRS describes the function and the performance allocated to our product. It provides a reference for the validation of the final product. SRS provides an overview of the product including functional and nonfunctional requirements, abbreviations used, product and functions etc.

#### **3.1.2 SCOPE OF THE DEVELOPMENT PROJECT**

The name of our product is “**Sequoro**”. This application is mainly based on security. It provides better protection for user’s accounts on all websites by using QR code encoded OTP’s.

#### **3.1.3 OVERVIEW OF DOCUMENT**

This document provides a description of the requirements of the product. Section 2 of the SRS gives detailed description of the product including the data requirements. Section 3 provides specific functional requirements of the different components of the product and the performance criteria. Section 4 provides idea about the software and hardware requirements.

### **3.2 GENERAL DESCRIPTION**

#### **3.2.1 USER CHARACTERISTICS**

The user is the owner of an account on Sequoro. Sequoro protection covers all their accounts on every website. The user may also be website owners, they have the option to extend Sequoro protection to their sites.

#### **3.2.2 PRODUCT PERSPECTIVE**

The software can be installed on any android system which has a camera. The server side website needs to be run on JSP and MySQL.

#### **3.2.3 GENERAL CONSTRAINTS, ASSUMPTIONS, DEPENDENCIES AND GUIDELINES**

The system will work only if there is a properly configured network connection as users need to complete their registration, there after network is not necessary. First-off, it’s the

infrastructure, which includes the type of the internet connection (Wi-Fi, 2G or 3G), the transfer bandwidth as well as the storage capacity of our (client and server) databases that determine transmission speed and the amount of data we obtain. Another crucial aspect is the resource reservoir, battery consumption, computing power and financial restrictions.

### **3.3 SPECIAL REQUIREMENTS**

#### **3.3.1 USER INTERFACE**

The user can interact with the system using graphical user interface. Using the screen, user can customize all settings the application. They could login to their account and perform operations.

#### **3.3.2 DETAILED DESCRIPTION OF FUNCTIONAL REQUIREMENTS**

This section provides a requirement overview of the product. The project will be developed in java, and will run on android operating system and server side on JSP.

#### **3.3.3 FUNCTIONAL REQUIREMENTS**

- Login only for valid user.
- User can change the password after login to application.
- User can customize the security settings.
- User could login to any website that uses Sequoro Authentication

#### **3.3.4 USER INPUT VALIDATION**

If the user leaves a mandatory field blank, he will be prompted to enter valid data in that particular field. User would be allowed to register only by entering valid email.

#### **3.3.5 PERFORMANCE REQUIREMENTS**

The performance of our product is at its best if accessed via internet. The user registration involves RSA 2048 bit key generation which is highly resource intensive.

## **3.4 REQUIREMENTS**

### **3.4.1 SOFTWARE REQUIREMENTS**

Operating system : Android

Client Side : Android

Server Side : JSP

### **3.4.2 HARDWARE REQUIREMENTS**

Processor : 1 GHz or faster processor.

Memory size : 1GB RAM or higher

Storage : 10 GB Hard Disk.

Device : Android Mobile with 2MP or higher camera.

## **4. SYSTEM STUDY/REVIEW OF LITERATURE**

### **4.1 SURVEY OF LITERATURE/ EXISTING SYSTEM**

Common password stealing attempts can be prevented using two step verification. Two step verification can be done using one time passwords or secondary passwords. One time passwords are delivered to users using SMS, Time based One Password Generators like Google Authenticator or using dedicated devices like RSA SecureID. However these methods have many disadvantages. SMS delivery mechanisms need continuous network availability and its implementation is costly considering elements like International Roaming. Time based One Time Password Applications are efficient in this case however they need frequent synchronization with the server and complex algorithms.

Several websites like Google, Facebook, Github etc. support two step authentication. Google is the most prominent among its competitors. Google uses an SMS delivered 6 digit number as its basic two step verification. Later they developed a Time Based One Time Passwords Application called Google Authentication. Google Authenticator needs to be synchronized with server frequently. Most companies use the SMS delivery of One Time Passwords as TOTP. Applications are proprietary and are difficult to develop. If companies developed TOTP apps for their own, users would have to install them individually. Sequoro is targeting these problems by providing an OAuth API that inherently uses advanced two step authentication mechanisms.

## 4.2 PROPOSED SYSTEM

Proposed system not only provide a secure two step authentication mechanism but we extended it to form an Application Program Interface that enables the owner of any website to include Sequoro protection to their website.

Sequoro uses a pair of new two-step authentication protocols that uses QR code to enable two step authentications. The first protocol is based on one time password. It is used to create an OTP and it is encrypted and encoded into a QR code that could only be decrypted by the user. The second protocol is based on password. It needs the user to specify a secondary password. The layout for entering the secondary password would be scrambled every time the user logins.

Sequoro has then been extended over an API that the other websites could be implemented. The website owners could register in the Sequoro and add the website details. The website owners would be given a Client ID and client secret key. The users need to download the appropriate files from our site and set the Client ID and client secret key and create the Sequoro sign in button. Thus Sequoro protection would be added to their sites.



### **4.3 FEASIBILITY STUDY**

The main objective of this study is to determine whether the proposed system is feasible or not. Mainly there are three types of feasibility study to which the proposed system is subjected as described below.

Three key considerations involved in the feasibility are:

- Economic Feasibility
- Technical Feasibility
- Behavioural Feasibility.

The proposed system must be evaluated from a technical viewpoint first, and if technically feasible, their impact on the organization must be assessed. If compatible, the behavioural system can be devised. Then those must be tested for economic feasibility.

#### **4.3.1 ECONOMIC FEASIBILITY**

Economic analysis is used for evaluating the cost effectiveness of a proposed system. In existing system companies uses SMS which is costly. But in proposed system, we use cost free QR Codes. This will make the user more comfortable.

#### **4.3.2 TECHNICAL FEASIBILITY**

The technical requirements of the proposed system are highly affordable. There is no difficulty in migrating from the existing system to the proposed system. Since J2EE is platform independent the application will run smoothly on any kernel without causing much trouble to the end user.

#### **4.3.3 BEHAVIOURAL FEASIBILITY**

The system is intended for use by web administrator and user. Since the software is simple and provides good Graphical User Interface, it provides high level user convenience.

## 5 SYSTEM DESIGN

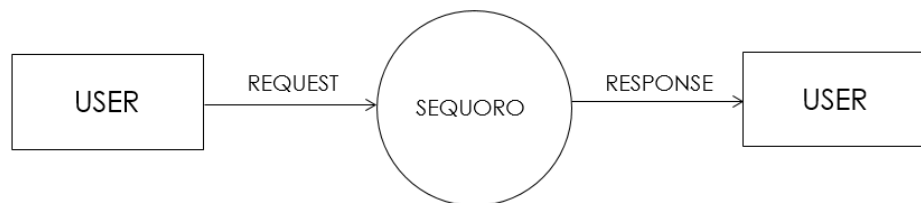
### 5.1 MODULES

There are 5 modules in this project.

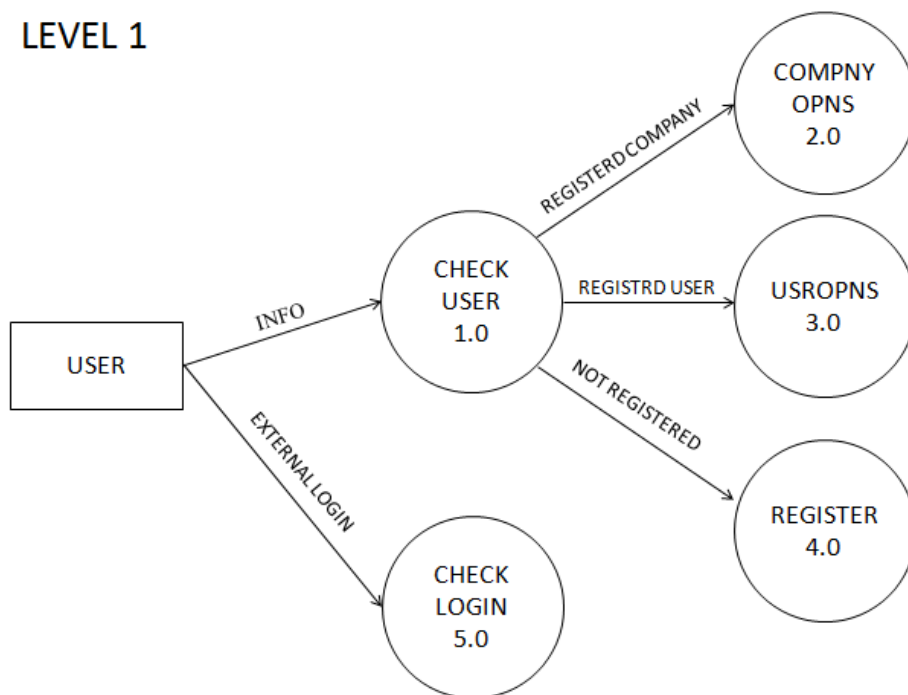
- **REGISTRATION**
  - Email address and password are collected from users and companies
  - Companies can enter their website details
  - User can select protocol and specify secondary password
- **APPLICATION CONFIGURATION**
  - Verified user can only use the system
  - RSA 2048 bit key pair generation
  - Public key is uploaded to server
  - Private key is saved to local storage
- **QR CODE GENERATION**
  - User is verified and protocol selected
  - For protocol 1 OTP is generated, encrypted and encoded to QR code
  - For protocol 2 random keyboard layout is created ,encrypted and encoded to QR code
- **LOGIN PROCEDURE**
  - User is verified , QR code is generated and displayed
  - User scans the QR Code
  - For protocol 1 OTP is decoded from QR code, decrypted and displayed
  - For protocol 2 Keyboard layout is decoded from QR code ,decrypted and displayed
- **OPEN AUTHENTICATION**
  - Users of a website are shown option to sign in using Sequoro
  - User would be prompted to enter his credentials
  - Two Step Authentication would be performed
  - Authenticated users would be redirected to website along with request code
  - Website authenticates the request code to obtain access code
  - Websites could access user details using access code

### 5.1.1 DATA FLOW DIAGRAMS

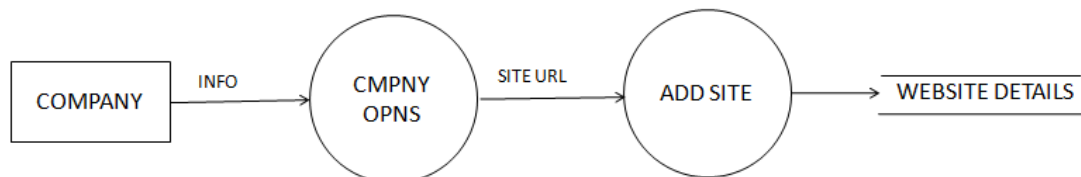
#### LEVEL 0



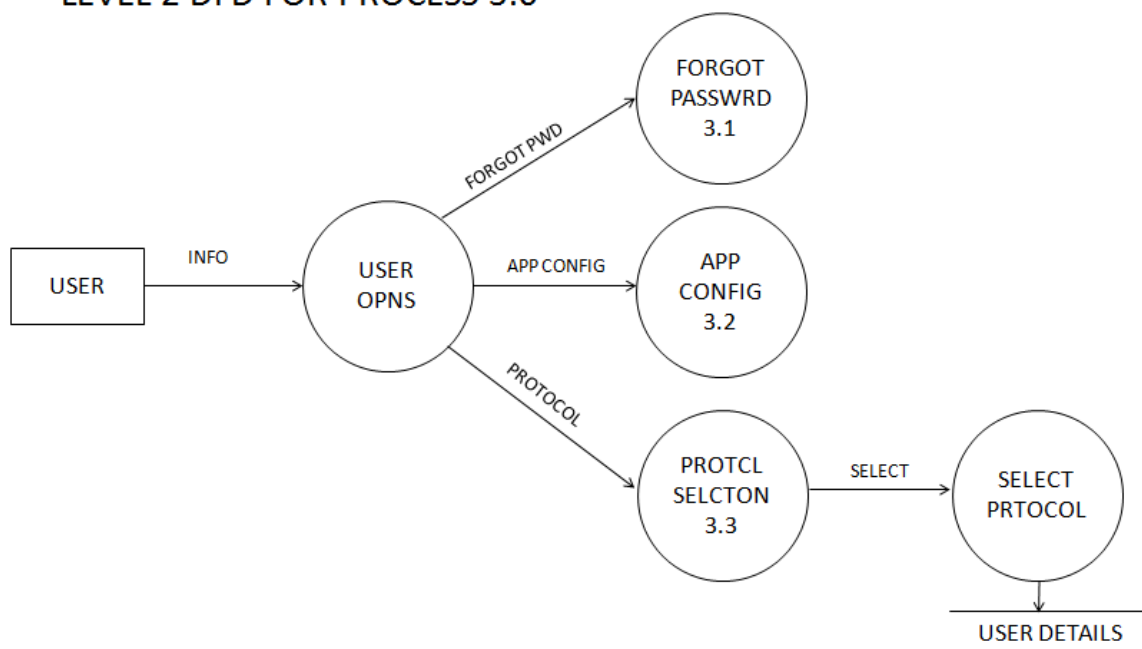
#### LEVEL 1



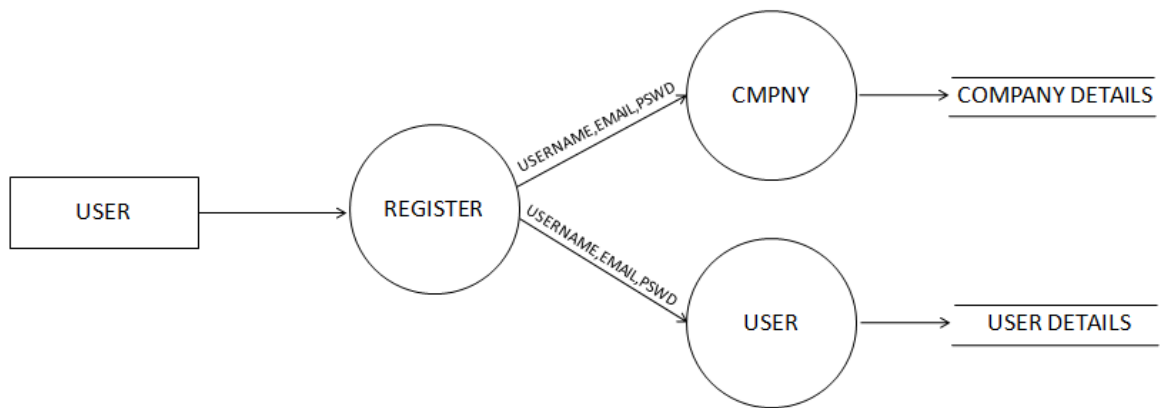
## LEVEL 2 DFD FOR PROCESS 2.0



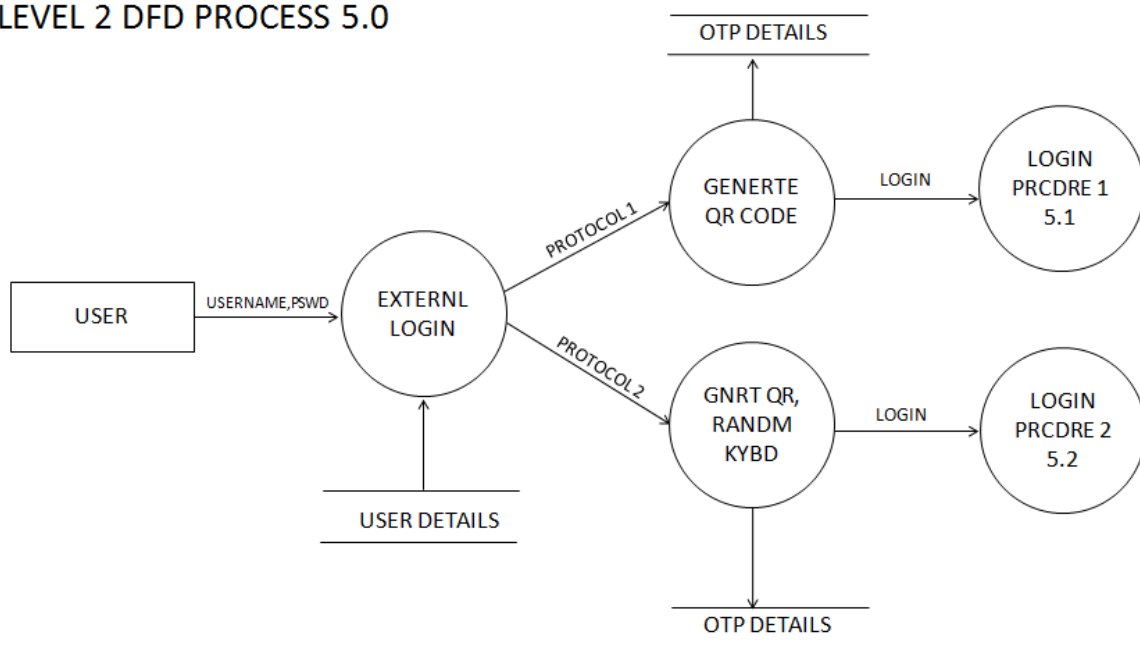
## LEVEL 2 DFD FOR PROCESS 3.0



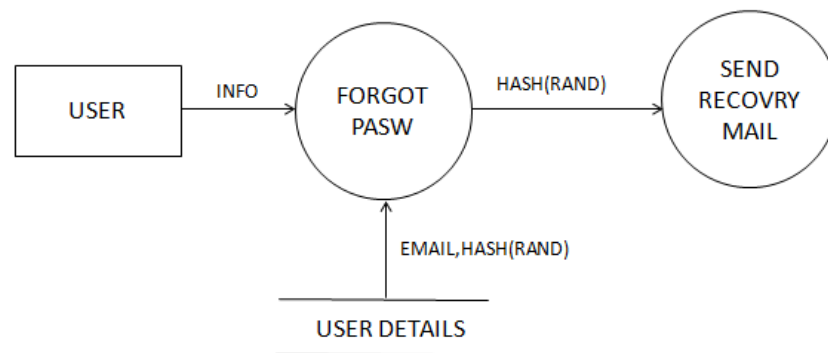
## LEVEL 2 DFD PROCESS 4.0



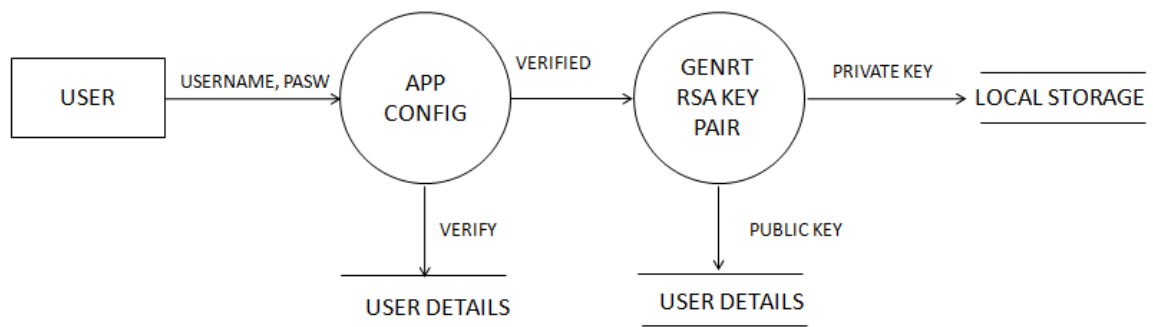
## LEVEL 2 DFD PROCESS 5.0



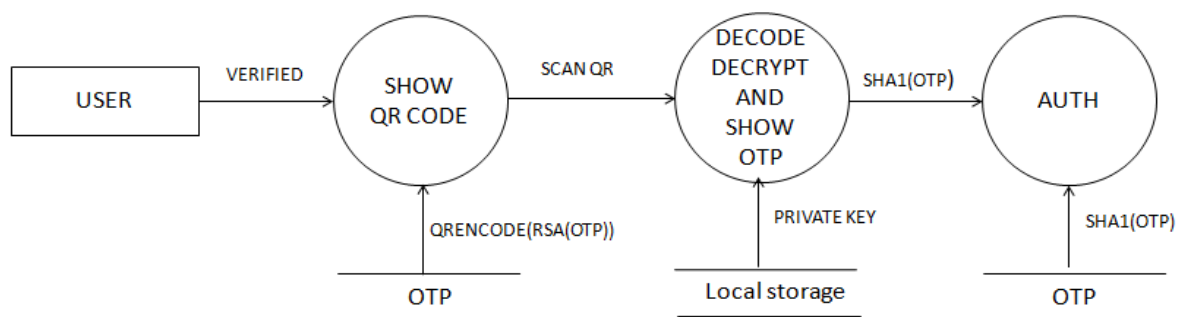
## LEVEL 3 DFD FOR PROCESS 3.1



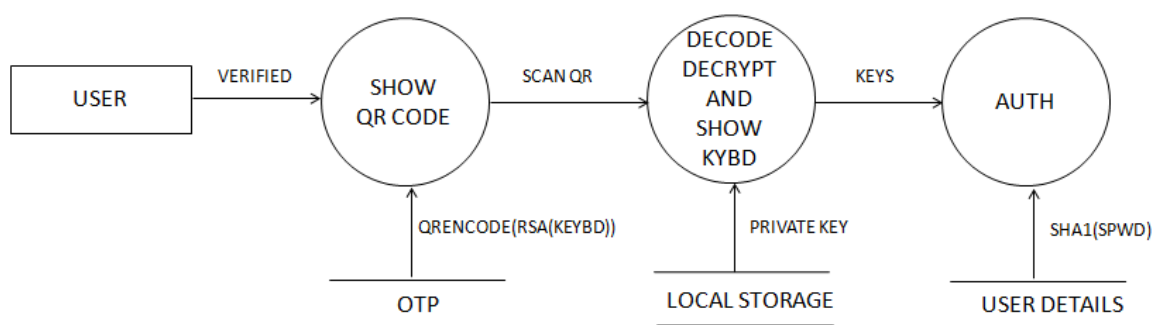
## LEVEL 3 DFD FOR PROCESS 3.2



## LEVEL 3 DFD FOR PROCESS 5.1



## LEVEL 3 DFD FOR PROCESS 5.2



## 6. USE CASE DESIGN

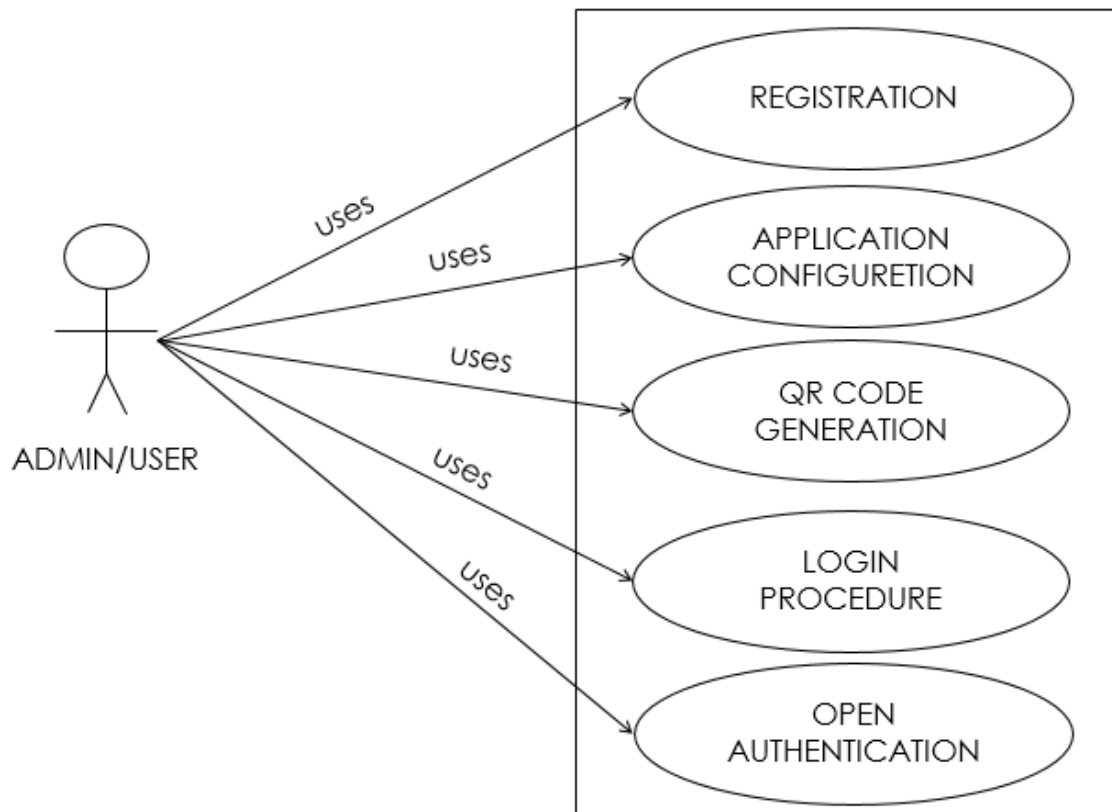
Use case is a set of scenarios that describing an interaction between a user and a system. The two main component of use case diagram are use cases and actors. They are helpful in exposing requirements and planning the projects. During the initial stage of project, most use cases should be defined. But as the project continues more might become visible, the use case diagram for showing the project is that it measures the size of the software. It reduces users need for training and operating the system.

In use case diagram, we all know, the main two things are the actors and the actions. Here the actors is the user. Apart from the actors there are actions to be performed by these actors. These actions are shown in oval.

Use case diagrams are relatively easy UML diagram. They are helpful in exposing requirements and planning of the project. During the initial stage of a project most use cases should be defined, but as the project continues more might become visible. A use case is an external view of the system that represents some action the user might perform in order to complete a task. A use case diagram displays the relationship among actors and use cases.



## 6.1 USE CASE DIAGRAM



## **7. DETAILED DESIGN**

### **7.1 DESIGN**

Design of the system includes mainly two steps: System design and detailed design. In system design a structural framework for the entire system is created. It is done in such a way that related part come under particular groups. Thus after the system design, a network of different groups is obtained. It is the high-level strategy for solving the problem and building a solution. It includes the decision about the organization of the system into subsystems, the allocation of subsystems to hardware and software components, and major conceptual and policy decisions that form the framework for the detailed design. System design is the first stage in which the basic approach to solving problem is selected. During system design, the overall style and structure are decided.

In detailed design, each group is studied in detail and the internal operations are decided. Based on this, the data structures and the programming language to be used are decided. Apart from detailed design, the system design can be grouped into physical design and structural design. The physical design maps out the details of the physical system and plans the system implementation and specifies the hardware and software requirements. Structured design is an attempt to minimize the complexity and make a problem manageable by subdividing into smaller segments, which is called modularization or decomposition.

### **7.2 SPECIFICATION**

The following are the specifications that are used to develop the proposed system: based upon the levels of the product, the project had been divided to 5 modules which are as follows.

#### **7.2.1 REGISTRATION**

The user needs to register in the Sequoro website. The email address and password are collected from users and companies. Companies can add Sequoro protection to their website by adding their website details like Website Name, Website URL, Website Redirect URL, Website Logo and Website description. The website owners would then be given Client ID and client secret keys.

### **7.2.2 APPLICATION CONFIGURATION**

Verified users could download and login to the Sequoro Application .The users those who have not created keys on their first login the security keys would be generated and uploaded to server. The RSA 2048 bit key generation is highly processor intensive that requires a few seconds. The public key would be uploaded to server and the private key would be saved in the phone.

### **7.2.3 QR CODE GENERATION**

User on login would be authenticated and One Time Password would be generated if the user has opted protocol 1.The OTP would then be encrypted using the users public key and encoded to form a QR code. The QR code would then be sent to the user along with a nonce. If the user has selected for protocol 2 a scrambled keyboard layout would be generated, encrypted and encoded to a QR code. It would be sent to the user along with a nonce.

### **7.2.4 LOGIN PROCEDURE**

The users would be authenticated using two step verification that uses QR code. After the user credentials are validated a QR code would be generated and displayed. The user needs to decrypt the QR code and enter the OTP or enter the secondary password. In case of OAuth the user would be logged in to the target website.

### **7.2.5 OPEN AUTHENTICATION**

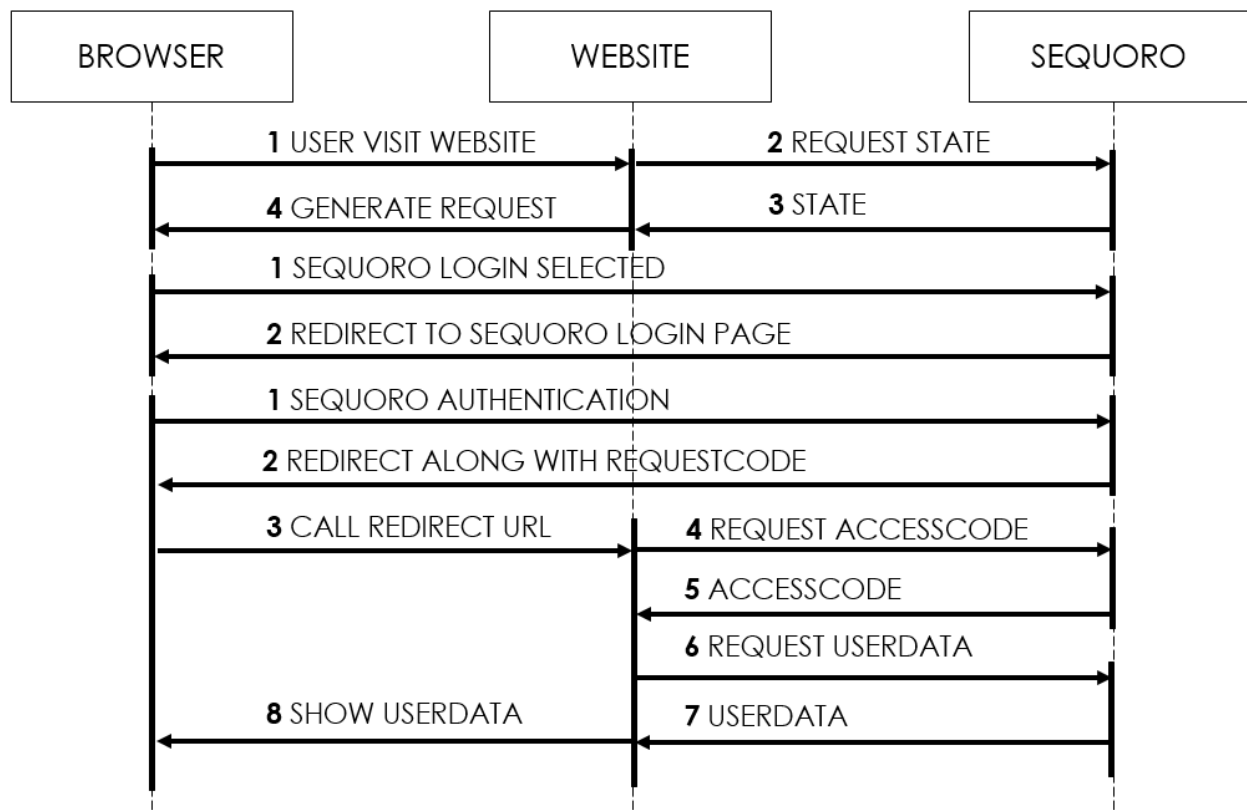
Users of a website are shown option to sign in Sequoro. User's needs to enter their Sequoro credentials and two step authentication would be performed. Authenticated users would be redirected to website redirect URL along with a request code. Website authenticates the request code and if authenticated an access code would be sent back to the server. The server would use this access code for further use.

## 7.3 SEQUENCE DIAGRAM

### 7.3.1 PURPOSE

Sequence diagram tell the manner in which interaction takes place and the style in which it is executed. They generally show the sequence in which the diagram demonstrates how objects are statically connected. It demonstrates the behaviour of an object in a use case by describing the actions they performed. The object is represented in rectangle and existence of objects is shown by vertical lines. The main purpose of a sequence diagram is to define event sequences that results in some desired outcome. The focus is less on events themselves and more on the order in which events occur; nevertheless, most sequence diagrams will communicate what information are sent between a system's objects is information along the horizontal well as the order in which they occur. The diagram conveys this information along the horizontal and vertical dimensions: the vertical dimensions shows, top down, the time sequence of instructions as they occur, and the horizontal dimension shows, left to right, the object instances that the instructions are sent to. Here in this there are some rules to draw the diagram. They are explained as follows: When drawing a sequence diagram, lifeline notation elements are placed across the top of the diagram. Lifelines represent either roles or object instances that participate in the sequence being modelled. Lifelines are drawn as a box with a dashed line descending from the centre of the bottom edge. The lifeline's name is placed inside the box. It shows the way the different components of project interact with each other from starting to end .The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behaviour of a system.

### 7.3.2 SEQUENCE DIAGRAM



## 8 DATABASE DESIGN

### USER DETAILS

ATTRIBUTE	TYPE	CONSTRAINTS
Id	int	Auto increment
name	varchar(60)	Not null
password	varchar(50)	Not null
<u>email</u>	varchar(150)	Not null
verified	int	Default 0
public_key	varchar(500)	Null
keyhash	varchar(50)	Null
protocol	int	Default 0
secondary_password	varchar(30)	Null
isSiteOwner	int	Default 0

### OTP DETAILS

ATTRIBUTE	TYPE	CONSTRAINTS
<u>id</u>	int	Auto increment
email	varchar(150)	Not null
client_id	varchar(50)	Not null
otp_hash	varchar(50)	Null
time	timestamp	Default CURRENT_TIMESTAMP
used	int	Default 0
accesscode	varchar(150)	Not null
secure_otp_hash_id	varchar(150)	Null
protocol	int	Default 0
keyboard_layout	varchar(200)	Null

### WEBSITE DETAILS

ATTRIBUTE	TYPE	CONSTRAINTS
<u>id</u>	int	Auto increment
owner_email	varchar(150)	Not null
site_name	varchar(100)	Not null
site_url	varchar(100)	Not null
site_redirecturl	varchar(200)	Not null
site_logo	varchar(300)	Not null
site_desc	varchar(500)	Not null
client_secretkey	varchar(150)	Not null
client_id	varchar(150)	Not null

### API REQUEST DETAILS

ATTRIBUTE	TYPE	CONSTRAINTS
<u>id</u>	int	Auto increment
state	varchar(100)	Not null
requestcode	varchar(100)	Not null
accesscode	varchar(100)	Not null
clientid	varchar(100)	Not null
time	timestamp	Default CURRENT_TIMESTAMP
used	int	Default 0

## 9 SYSTEM IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on new systems for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over an evolution of change over methods.

Apart from planning, major task of preparing the implementation are education and training of users. The more complex the system being implemented, the more involved will be the system analysis and design effort is requested just for implementation.

Implementation phase of the software developed is concerned with translating design specification into source code. Implementation is used here to mean the processes of converting a new revised system design into an optional one. Conversation is one aspect of implementation. The primary goal of implementation is to write source code so its specification can be easily verified, and so that debugging, testing and modification are erased.

Simplicity, clarity and elegance are the hallmarks of good programs. Obscurity, cleverness and complexity are the indications of inadequate design and misdirected thinking. The implementation team should be provided with a well-defined set of software requirements, an architectural design description. Also each team member must understand the objectives of implementation. There are two types of implementation:

Implementation of a computer system replaces a manual system. The problems encountered are converting files, training users, creating accurate files and verifying printouts for integrity. Implementation of a new computer system to replace an existing one is an extremely tedious task. If these are not handled properly they can cause many a problem.



## 9.1 SYSTEM REQUIREMENTS

### 9.1.1 Software Requirements

Operating system	: Android
Client Side	: Android
Server Side	: JSP

### 9.1.2 Hardware Requirements

Processor	: 1 GHz or faster processor.
Memory size	: 1GB RAM or higher
Storage	: 10 GB Hard Disk.
Device	: Android Mobile with 2MP or higher camera.

## 9.2 DEVELOPMENT ENVIRONMENT

### Windows

Microsoft Windows (or simply Windows) is a family of graphical operating systems developed, marketed, and sold by Microsoft. It consists of several families of operating systems, each of which cater to a certain sector of the computing industry. Active Windows families include Windows NT, Windows Embedded and Windows Phone; these may encompass subfamilies, e.g. Windows Embedded Compact (Windows CE) or Windows Server. Defunct Windows families include Windows 9x and Windows Mobile.

Microsoft introduced an operating environment named Windows on November 20, 1985 as a graphical operating system shell for MS-DOS in response to the growing interest in graphical user interfaces (GUIs).<sup>[6]</sup> Microsoft Windows came to dominate the world's personal computer market with over 90% market share, overtaking Mac OS, which had been introduced in 1984. However, since 2012, it sells less than Android, which became the most popular operating system in 2014, when counting all of the computing platforms Windows runs on (same as Android); in 2014, the number of Windows device sold were less than 25% of Android devices sold.

As of April 2014, the most recent versions of Windows for personal computers, smart phones, server computers and embedded devices are respectively Windows 8.1, Windows Phone

8.1, Windows Server 2012 R2 and Windows Embedded 8. A specialized version of Windows runs on the Xbox One game console.

The next version of Windows is Windows 10 and is currently available as a technical preview; it is set for release for phones, tablets, laptops, and PCs in late 2015.

## **9.3 FAMILIARIZATION WITH TOOLS/TECHNIQUES/API'S**

### **JAVA**

Java was conceived by James Gosling, Patrick Naughton, Chris Wrath, Ed Frank and Mike Sheridan at Sun Microsystems Inc. in 1991. The primary motivation was the need for a platform-independent language that could be used to create software to be embedded in various consumer electronic devices, such as microwave ovens and remote controls. Java can be used to create two types of programs: applications and applets. An applet is an application designed to be transmitted over the Internet and executed by a Java-enabled web browser. An applet is actually a tiny Java program, dynamically downloaded across the network, just like an image, sound files, or video clip.

#### **Java Features**

- Simple: - java was designed to be easy for the professional programmers to learn and use effectively.
- Source: - Java provides firewall security.
- Portable: - Java is portable language.
- Object-oriented: - Object oriented programming is the core of java.
- Robust: - Provide multiplatform environment.
- Multithreaded: - Programs that do many things simultaneously.
- Interpreted: - Java enables the creation of cross platform programs by compiling into an intermediate representation called java byte code.
- Distributed: - Java is designed for distributed environment of internet, because it handles TCP/IP protocol.
- Dynamic: - Java programs carry with them substantial amounts of run time type information that is used to verify and resolve access to object at runtime. This makes it possible to dynamic link code in an expedient manner.

## **JAVA SERVER PAGES (JSP)**

JSP is a java based technology that simplifies the process of developing dynamic web sites. With JSP, web designers and developers can quickly incorporate dynamic elements into web pages using embedded java and simple mark-up tags. These tags provide the HTML designer with a way to access data and business logic stored inside java objects. Java Server Pages are text files with the extension .JSP, which take the place of traditional HTML pages. JSP files contain traditional HTML along with embedded code that allows the developer to access data from the java code running on the server.

JSP offers several benefits for dynamic content generation. As a Java-based technology, it enjoys all of the advantages that the Java language provides with respect to development and deployment. As an object-oriented language with strong typing, encapsulation, exception handling, and automatic memory management, use of Java leads to increased programmer productivity and more robust code. Because compiled Java byte code is portable across all platforms that support a JVM, use of JSP does not lock us into using a specific hardware platform, operating system, or server software. If a switch in any of these components becomes necessary, all JSP pages and associated Java classes can be migrated over as is. Because JSP is vendor-neutral, developers and system architects can select best of breed solutions at all stages of JSP deployment .JSP technology is the Java platform technology for building applications containing dynamic web content such as HTML, DHTML, XHTML, and XML. The Java Server Pages technology enables the authoring of web pages that create dynamic content easily but with maximum power and flexibility. The Java Server Pages technology offers a number of advantages:

### **Features:-**

- Write Once, Run Anywhere properties
- High quality tool support
- Reuse of components and tag libraries
- Separation of dynamic and static content
- Support for scripting and actions
- Web access layer for N-tier enterprise application architecture

## JAVA SCRIPT

JavaScript is use for validation purposes usually at the client-side, which do not require the server. It is a programming language integrated with HTML. JavaScript facilitates the developer with properties related to document windows, frames, forms, loaded documents and links. This scripting language also traps user events so programs can be developed for such events. This is an interpreter-based language and source code files are directly executed at runtime. JavaScript includes built-in objects related to the current windows and documents as well as objects such as Math, String, Date functions respectively. Since JavaScript is an object-based language, it supports instances, methods and properties.

The browsers support JavaScript. In JavaScript, the document object refers to ‘whatever web page the reader is currently looking through’ – which is also the document that contains the JavaScript code. The links are objects within the document object. Date, button, checkbox, elements array from password are the other objects available. JavaScript statements are used to build loops into the script so that commands can be executed several times.

Event handlers are those parts of language that tell JavaScript to send or to carry out some actions. The event handlers go into regular HTML tags.

- Can display custom dialog boxes on the screen, i.e. alert (), confirm () and prompt ().
- Have two data types – Numbers and Strings.
- JavaScript has custom functions and allows user to write code to have user-defined functions.

## MySQL

MySQL database has become the world's most popular Open source database because of its consistency, fast performance, high reliability and ease of use. It has also become the database of choice for a new generation of applications built on the LAMP stack (Linux, Apache, MySQL, PHP / Perl / Python). MySQL runs on more than 20 platforms including Linux, Windows, OS/X, HP-UX, AIX, Netware, giving you the kind of flexibility that puts you in control. MySQL offers a comprehensive range of certified software, support, training and consulting.

MySQL is a multithreaded, multi-user SQL Database Management System. My SQL's implementation of a relational database is an abstraction on top of a computer's file system. The relational database abstraction allows collection of data items to be organized as a set of formally

described tables. Data can be accessed or reassembled from these tables in many different ways, which do not require any reorganization of the database tables themselves.

MySQL is popular for web applications such as MediaWiki or Drupal and act as the database component of the LAMP ,MAMP and WAMP platforms(Linux/Mac, Windows-Apache MySQL-PHP/Perl/Python), and for open -source bug tracking tools like Bugzilla.

Its popularity as a web application is closely tied with the popularity of PHP, which is often combined with MySQL and nicknamed Dynamic Duo .It is easy to find many references that combines the two in websites and books (PHP and MySQL for Dummies, PHP and MySQL Bible, Beginning PHP and MySQL etc.).

Relational database speak SQL (Structured Query Language). SQL is a standard interactive programming language for getting information from and updating a relational database. Although SQL itself is both an ANSI and an ISO standard, many database products support SQL with proprietary extensions to the standard language. SQL queries take the form of a command language that lets you select, insert, update, find out the location of data, and so forth.

**MySQL Features:**

- Cross platform support
- Stored procedures
- Triggers
- Cursors
- True VARCHAR support
- Updated views
- Very fast and much reliable for any type of application.
- Very lightweight application.
- Command line tool is very powerful and can be used to run SQL queries against database.
- Supports indexing and binary objects.
- Allows changing the structure of table while server is running.
- It has a wide user base.
- It is a very fast thread-based memory allocation system.

## **RSA**

RSA is one of the first practicable public-key cryptosystems and is widely used for secure data transmission. In such a cryptosystem, the encryption key is public and differs from the decryption key which is kept secret. In RSA, this asymmetry is based on the practical difficulty of factoring the product of two large prime numbers, the factoring problem. RSA stands for Ron Rivest, Adi Shamir and Leonard, who first publicly described the algorithm in 1977. Clifford Cocks, an English mathematician, had developed an equivalent system in 1973, but it was not declassified until 1999.

A user of RSA creates and then publishes a public key based on the two large prime numbers, along with an auxiliary value. The prime numbers must be kept secret. Anyone can use the public key to encrypt a message, but with currently published methods, if the public key is large enough, only someone with knowledge of the prime numbers can feasibly decode the message. Breaking RSA encryption is known as the RSA problem; whether it is as hard as the factoring problem remains an open question.

## 10 SOURCE CODE

### RSA ENCRYPTION, DECRYPTION AND KEY GENERATION

```
import java.io.BufferedOutputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.KeyPair;
import java.security.KeyPairGenerator;
import java.security.NoSuchAlgorithmException;
import java.security.PrivateKey;
import java.security.PublicKey;
import java.security.spec.InvalidKeySpecException;
import java.security.spec.RSAPrivateKeySpec;
import java.security.spec.RSAPublicKeySpec;
import javax.crypto.Cipher;
import org.apache.commons.codec.binary.Base64;

public class RSAEncryptionDescription {

    private static final String PUBLIC_KEY_FILE = "Public.key";
    private static final String PRIVATE_KEY_FILE = "Private.key";
    private static final String CIPHER_INSTANCE = "RSA/ECB/PKCS1Padding";
    RSAPublicKeySpec rsaPubKeySpec;
    RSAPrivateKeySpec rsaPrivKeySpec;
    public boolean keygen(String prikeypath, String pubkeypath) {
        try {
```

```

    KeyPairGeneratorkeyPairGenerator = KeyPairGenerator.getInstance("RSA");
    keyPairGenerator.initialize(2048);
    KeyPairkeyPair = keyPairGenerator.generateKeyPair();
    PublicKeypublicKey = keyPair.getPublic();
    PrivateKeyprivateKey = keyPair.getPrivate();
    KeyFactorykeyFactory = KeyFactory.getInstance("RSA");
    rsaPubKeySpec = keyFactory.getKeySpec(publicKey, RSAPublicKeySpec.class);
    rsaPrivKeySpec = keyFactory.getKeySpec(privateKey, RSAPrivateKeySpec.class);
    saveKeys(pubkeypath,rsaPubKeySpec.getModulus(), rsaPubKeySpec.getPublicExponent());
    (prikeypath, rsaPrivKeySpec.getModulus(), rsaPrivKeySpec.getPrivateExponent());
} catch (NoSuchAlgorithmException e) {
    e.printStackTrace();
    return false;
} catch (InvalidKeySpecException e) {
    e.printStackTrace();
    return false;
} catch (Exception e) {
    e.printStackTrace();
    return false;
} finally {
    return true;
}
}

public void saveKeys(String fileName, BigInteger mod, BigIntegerexp) throws IOException {
    FileOutputStreamfos = null;
    ObjectOutputStreamoos = null;
    try {
        fos = new FileOutputStream(fileName);
        oos = new ObjectOutputStream(new BufferedOutputStream(fos));
        oos.writeObject(mod);
        oos.writeObject(exp);
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if(oos != null) {

```



```
        oos.close();
    if (fos != null) {
        fos.close();
    }
}
}
}

public String encryptData(String data, String file) throws IOException {
    byte[] dataToEncrypt = data.getBytes("UTF8");
    String encryptedData = null;
    try {
        PublicKey pubKey = readPublicKeyFromFile(file); //PUBLIC_KEY_FILE);
        Cipher cipher = Cipher.getInstance(CIPHER_INSTANCE);
        cipher.init(Cipher.ENCRYPT_MODE, pubKey);
        KeyFactory keyFactory = KeyFactory.getInstance("RSA");
        rsaPubKeySpec = keyFactory.getKeySpec(pubKey, RSAPublicKeySpec.class);
        encryptedData = new String(Base64.encodeBase64(cipher.doFinal(dataToEncrypt)));
    } catch (Exception e) {
        e.printStackTrace();
    }
    return encryptedData;
}

public String decryptData(String data, String file) throws IOException {
    byte[] descryptedData = null;
    try {
        PrivateKey privateKey = readPrivateKeyFromFile(file);
        Cipher cipher = Cipher.getInstance(CIPHER_INSTANCE);
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        descryptedData = cipher.doFinal(Base64.decodeBase64(data.getBytes("UTF8")));
        return new String(descryptedData);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
```

```
public PublicKeyreadPublicKeyFromFile(String fileName) throws IOException {
    FileInputStreamfis = null;
    ObjectInputSteamois = null;
    try {
        fis = new FileInputStream(new File(fileName));
        ois = new ObjectInputStream(fis);
        BigInteger modulus = (BigInteger) ois.readObject();
        BigInteger exponent = (BigInteger) ois.readObject();
        RSAPublicKeySpecrsaPublicKeySpec = new RSAPublicKeySpec(modulus, exponent);
        KeyFactory fact = KeyFactory.getInstance("RSA");
        PublicKeypublicKey = fact.generatePublic(rsaPublicKeySpec);
        Return publicKey;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (ois != null) {
            ois.close();
        }
        if (fis != null) {
            fis.close();
        }
    }
    return null;
}

public PrivateKeyreadPrivateKeyFromFile(String fileName) throws IOException {
    FileInputStreamfis = null;
    ObjectInputSteamois = null;
    try {
        fis = new FileInputStream(new File(fileName));
        ois = new ObjectInputStream(fis);
        BigInteger modulus = (BigInteger) ois.readObject();
        BigInteger exponent = (BigInteger) ois.readObject();
        RSAPrivateKeySpec rsaPrivateKeySpec = new RSAPrivateKeySpec(modulus, exponent);
        KeyFactory fact = KeyFactory.getInstance("RSA");
        PrivateKey privateKey = fact.generatePrivate(rsaPrivateKeySpec);
    }
```

```
        Return privateKey;
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (ois != null) {
            ois.close();
        }
        if (fis != null) {
            fis.close();
        }
    }
}
return null;
}
}
```

## QRCODE GENERATION

```
package bean;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Collections;
import java.util.Iterator;
import java.util.LinkedList;
import org.apache.commons.codec.binary.Base64;
import net.glxn.qrgen.QRCode;
import net.glxn.qrgen.image.ImageType;
import org.json.simple.JSONObject;
import bean.Otp;
import bean.User;
import crypto.RSAEncryptionDescription;
import java.sql.Timestamp;
import java.util.Date;
```

```
import org.apache.commons.codec.digest.DigestUtils;

public class QrGen {

    public String generate_qrcode(String platform, int protocol, Otpotp_obj, User obj, File d, int
    step,Stringcleintid) {
        JSONObjectreplyObj = new JSONObject();
        JSONObjectmatrixObj = new JSONObject();
        JSONObjectjs = new JSONObject();
        Timestamp ts_now = new Timestamp(new Date().getTime());
        String now = ts_now.toString();
        String secure_otp_id = DigestUtils.shaHex(now + "$#@#$%^salt");
        String data = null;
        if (protocol == 0) {
            replyObj.put("protocol", "0");
            js.put("protocol", "0");
            intnum = 0;
            while (num< 1000 || num> 10000) {
                num = (int) (Math.random() * 10000);
            }
            data = "" + num;
        } else { //protocol 2
            replyObj.put("protocol", "1");
            js.put("protocol", "1");
            LinkedListll = new LinkedList();
            ll.add(new Integer(0));
            ll.add(new Integer(1));
            ll.add(new Integer(2));
            ll.add(new Integer(3));
            ll.add(new Integer(4));
            ll.add(new Integer(5));
            ll.add(new Integer(6));
            ll.add(new Integer(7));
            ll.add(new Integer(8));
            ll.add(new Integer(9));
            ll.add(new Integer(-1));
            ll.add(new Integer(-1));
        }
    }
}
```

```
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
ll.add(new Integer(-1));
Collections.shuffle(ll);
inti = 0;
Iterator li = ll.iterator();
while (li.hasNext()) {
    matrixObj.put(i++, li.next());
}
data = matrixObj.toJSONString();
}
try {
    otp_obj.setOtp_hash(data);
    otp_obj.setSecure_otp_hash_id(secure_otp_id);
    if (platform.compareTo("3") == 0) {
        otp_obj.setClient_id(cleintid);
    }
    otp_obj.insert(protocol);
} catch (Exception ex) {
    System.out.println("Excn in entering otp :" + ex.toString());
}
String enc = null;
try {
    String fileName = d.getAbsolutePath() + "/server_keys/ServerPrivate.key";
    RSAPublicKey rsa_obj = new RSAPublicKey();
```

```

enc = rsa_obj.encryptData(data, obj.getKeydetails());
js.put("data", enc);
enc = new String(Base64.encodeBase64(js.toJSONString().getBytes("UTF8")));
if (step == 2) {
    replyObj.put("status", "notok");
} else {
    replyObj.put("status", "ok");
}
String path = "/img/keys/" + (int) (Math.random() * 100000) + ".PNG";
if(ConnectionProvider.baseurl.compareTo("http://localhost:8080/site/")==0)
path = "./img/keys/" + (int) (Math.random() * 100000) + ".PNG";
this.generate_qrcode(enc, d.getAbsolutePath() + path);
replyObj.put("name", obj.getName(obj.getEmail()));
replyObj.put("qrcode", path);
replyObj.put("secureid", secure_otp_id);
returnreplyObj.toJSONString().trim();
} catch (Exception ex) {
    replyObj.put("status", "exception :" + ex.toString());
    return (replyObj.toJSONString().trim());
}
}

public void generate_qrcode(String data, String path) {
try {
    ByteArrayOutputStream out = QRCode.from(data).to(ImageType.PNG).withSize(300,
300).stream();
    try {
        FileOutputStreamfout = new FileOutputStream(new File(path));
        fout.write(out.toByteArray());
        fout.flush();
        fout.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
}

```

```
} catch (Exception exn) {  
    System.out.println("excn in creating qr code : " + exn.toString());  
}  
}  
}
```

## PLUGIN FOR API

```
package bean;  
  
import com.sun.xml.ws.runtime.dev.Session;  
import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.UnsupportedEncodingException;  
import java.net.URL;  
import java.net.URLConnection;  
import java.net.URLEncoder;  
import java.sql.Timestamp;  
import org.apache.commons.codec.binary.Base64;  
import java.util.Date;  
import javax.crypto.Mac;  
import javax.crypto.spec.SecretKeySpec;  
import javax.json.JsonObject;  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
import sun.net.util.URLUtil;  
import org.json.simple.*;  
import org.json.simple.parser.JSONParser;  
import org.json.simple.parser.ParseException;  
  
public class SequoroOAuth {  
    public String clientId, clientSecretKey, redirectUrl, initializeUrl, loginUrl, authUrl, requestUrl,  
        nonce, state, time, email, authcode, requestcode;  
  
    public SequoroOAuth() throws UnsupportedEncodingException, IOException {
```

```

String baseUrl="http://sequoro1.jelasticlw.com.br/";
clientId = ""; // provide your client id
clientSecretKey = ""; // provide your secre key
redirectUrl = ""; // provide your redirect url
authUrl = baseUrl+"api/accesscode_api.jsp?clientid=" + this.clientId + "&clientsecretkey=" +
this.clientSecretKey + "&requestcode=";
initializeUrl = baseUrl+"api/request_initialize.jsp?clientid=" + this.clientId +
"&clientsecretkey=" + this.clientSecretKey;
state = initialize_request().trim();
time = new Timestamp(new Date().getTime()).toString();
requestUrl =baseUrl+ "api/login_api.jsp?clientid=" + clientId + "&redirecturl=" + redirectUrl
+ "&time=" + time + "&state=" + state;
this.nonce = HMACSHA256(this.requestUrl, this.clientSecretKey);
this.requestUrl = this.requestUrl + "&nonce=" + nonce;
}

public String getEmail() {
    return this.email;
}

public String getRedirectUrl() {
    return this.redirectUrl;
}

public String getRequestUrl() {
    return this.requestUrl;
}

private String HMACSHA256(String message, String secret) throws
UnsupportedEncodingException {
String hash = null;
try {
    Mac sha256_HMAC = Mac.getInstance("HmacSHA256");
    SecretKeySpecsecret_key = new SecretKeySpec(secret.getBytes(), "HmacSHA256");
    sha256_HMAC.init(secret_key);
    hash = new String(Base64.encodeBase64(sha256_HMAC.doFinal(message.getBytes())));
} catch (Exception e) {
    System.out.println("Excnhmac :" + e.toString());
}
}

```



```

    }

    Return URLEncoder.encode(hash, "UTF-8");
}

Public boolean authenticate(String requestcode,HttpServletRequest request) throws
ServletException, IOException, ParseException {
    this.requestcode = requestcode;
    URL urldemo = new URL(this.authUrl+requestcode);
    URLConnection yc = urldemo.openConnection();
    BufferedReader in = new BufferedReader(new InputStreamReader(
yc.getInputStream()));
    String inputLine;
    String response = "";
    while ((inputLine = in.readLine()) != null) {
        response += inputLine;
    }
    in.close();
    System.out.println(response);
    JSONParser parser=new JSONParser();
    JSONObjectreplyObj=(JSONObject)parser.parse(response);
    if(replyObj.get("accesscode").toString().compareTo("Error")!=0){
        this.authcode=response;
        request.getSession().setAttribute("accesscode", replyObj.get("accesscode").toString());
        request.getSession().setAttribute("name", replyObj.get("name").toString());
        request.getSession().setAttribute("email", replyObj.get("email").toString());
        return true;
    }
    return false;
}

private String initialize_request() throws IOException {
    this.requestcode = requestcode;
    URL urldemo = new URL(this.initializeUrl);
    URLConnection yc = urldemo.openConnection();
    BufferedReader in = new BufferedReader(new InputStreamReader(
yc.getInputStream()));
    String inputLine;

```

```
String response = "";
while ((inputLine = in.readLine()) != null) {
    response += inputLine;
}
in.close();
System.out.println(response);
return response;
}
```

## KEY SYNCHRONIZATION

```
Package com.jithurjacob.sequoro;
Import android.annotation.TargetApi;
Import android.content.Context;
Import android.content.Intent;
Import android.net.ConnectivityManager;
Import android.net.NetworkInfo;
Import android.os.AsyncTask;
Import android.os.Build;
Import android.os.Bundle;
Import android.os.Handler;
Import android.os.Message;
import android.support.v7.app.ActionBarActivity;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.TextView;
import android.widget.Toast;
import com.squareup.okhttp.FormEncodingBuilder;
import com.squareup.okhttp.Headers;
import com.squareup.okhttp.MediaType;
import com.squareup.okhttp.MultipartBuilder;
import com.squareup.okhttp.OkHttpClient;
import com.squareup.okhttp.Request;
import com.squareup.okhttp.RequestBody;
```

```
import com.squareup.okhttp.Response;
import org.json.JSONObject;
import java.io.File;
import java.io.IOException;
import java.math.BigInteger;
import java.security.KeyFactory;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.security.PublicKey;
import java.security.spec.RSAPublicKeySpec;
public class KeySync extends ActionBarActivity {
    private TextView mStatus;
    public String output;
    public static final MediaType MEDIA_TYPE_MARKDOWN = MediaType.parse("text/x-
markdown; charset=utf-8");
    private static final MediaType MEDIA_TYPE_TEXT = MediaType.parse("text/plain; charset=utf-
8");
    private OkHttpClient client = new OkHttpClient();
    private static final String PRIVATE_KEY_FILE = "Private.key";
    UserManager session;
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_key_sync);
        mStatus = (TextView) findViewById(R.id.status);
        session = new UserManager(getApplicationContext());
        session.checkLogin();
    }
    Thread th; @Override
    protected void onStart() {
        super.onStart();
        th = new Thread() {
            public void run() {
                try {
                    Thread.sleep(5000);
                    File file1 = new File(getFilesDir() + "/" + session.getEmail() + "Private.key");
```

```

File file2 = new File(getFilesDir() + "/" + session.getEmail() + "Public.key");
File file3 = new File(getFilesDir() + "/serverpublic.key");
if ((file1.exists() && file2.exists() && file3.exists())) {
    myHandler.sendMessage(4);
    show_main();
    myHandler.sendMessage(5);
} else {
    myHandler.sendMessage(6);
    startMyTask(new RSA(getApplicationContext()));
}
return;
} catch (Exception ex) {
    System.out.println("\n-----excn1 in create----- " + ex.toString());
}}};
try {
    th.start();
} catch (Exception ex) {
    System.out.println("\n-----excn1 in invk thread----- " + ex.toString());
}
}

public void showtoast(String cont) {
    Toast.makeText(getApplicationContext(), cont, Toast.LENGTH_LONG).show();
}@TargetApi(Build.VERSION_CODES.HONEYCOMB)
static public < T > void startMyTask(AsyncTask< T, ?, ?> task, T...params) {
try {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB) {
        task.executeOnExecutor(AsyncTask.THREAD_POOL_EXECUTOR, params).get();
    } else {
        task.execute(params).get();
    }
} catch (Exception e) {
    System.out.print(" excn in start task : " + e.toString());
}
}
}

```

```
public static String getMD5EncryptedString(String encTarget) {
    MessageDigestmdEnc = null;
    try {
        mdEnc = MessageDigest.getInstance("MD5");
    } catch (NoSuchAlgorithmException e) {
        System.out.println("Exception while encrypting to md5");
        e.printStackTrace();
    } // Encryption algorithm
    mdEnc.update(encTarget.getBytes(), 0, encTarget.length());
    String md5 = new BigInteger(1, mdEnc.digest()).toString(16);
    while (md5.length() < 32) {
        md5 = "0" + md5;
    }
    return md5;
}

public void show_main() {
    // Starting MainActivity
    Intent i = new Intent(getApplicationContext(), MainActivity.class);
    i.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
    // Add new Flag to start new Activity
    i.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
    startActivity(i);
    finish();
}

public class sendPublicKey extends AsyncTask< Void, String, Boolean > {
    private Context context;
    private String hash;
    private String filepath;
    public sendPublicKey(Context context, String hash, String filepath) {
        this.hash = hash;
        this.filepath = filepath;
        this.context = context;
    }

    Private boolean checkNetworkConnection() {
        boolean haveConnectedWifi = false;
```

```

booleanhaveConnectedMobile = false;
ConnectivityManager cm = (ConnectivityManager)
getSystemService(Context.CONNECTIVITY_SERVICE);
NetworkInfo[] netInfo = cm.getAllNetworkInfo();
for (NetworkInfoni: netInfo) {
    if (ni.getTypeName().equalsIgnoreCase("WIFI"))
        if (ni.isConnected())
            haveConnectedWifi = true;
    if (ni.getTypeName().equalsIgnoreCase("MOBILE"))
        if (ni.isConnected())
            haveConnectedMobile = true;
}
Return haveConnectedWifi || haveConnectedMobile;
}

public String check_key(String myurl, String hash) throws IOException {
System.out.println("\n\n\ninside key check\n\n\n");
String a = null;
try {
    Response response;
    RequestBodyformBody = new FormEncodingBuilder()
        .add("email", session.getEmail())
        .add("hash", hash)
        .add("platform", "2")
        .build();
    Request request = new Request.Builder()
        .url(myurl)
        .post(formBody)
        .build();
    response = client.newCall(request).execute();
    a = response.body().string().trim();
return a;
} catch (Exception ex) {
    System.out.println("\n-----excn----- in check_key : " + ex.toString());
} finally {
    return a;
}

```

```

    }
}

private String keyupload(String myurl, String hash, String filepath) throws IOException {
    String a = "ok";
    try {
        Response response;
        String filepath1 = getFilesDir() + "/" + session.getEmail() + "Public.key";
        RSAEncryptionDecryptionobj = new RSAEncryptionDecryption(this.context,
            session.getEmail());
        KeyFactorykeyFactory = KeyFactory.getInstance("RSA");
        PublicKeypublicKey = obj.readPublicKeyFromFile(filepath1);
        RSAPublicKeySpecrsaPubKeySpec = keyFactory.getKeySpec(publicKey,
            RSAPublicKeySpec.class);
        JSONObjectjs = new JSONObject();
        js.put("Modulus", rsaPubKeySpec.getModulus().toString());
        js.put("Exponent", rsaPubKeySpec.getPublicExponent().toString());
        RequestBodyrequestBody = new MultipartBuilder()
            .type(MultipartBuilder.FORM)
            .addPart(Headers.of("Content-Disposition", "form-data; name=\"email\""),
                RequestBody.create(null, session.getEmail()))
            .addPart(Headers.of("Content-Disposition", "form-data; name=\"hash\""),
                RequestBody.create(null, hash))
            .addPart(Headers.of("Content-Disposition", "form-data; name=\"platform\""),
                RequestBody.create(null, "2"))
            .addPart(Headers.of("Content-Disposition", "form-data; name=\"key\""),
                RequestBody.create(null, js.toString()))
            .build();
        Request request = new Request.Builder()
            .url(myurl)
            .post(requestBody)
            .build();
        Response response1 = client.newCall(request).execute();
        if (!response1.isSuccessful())
            a = response1.body().string().trim();
        return a;
    }
}

```

```
} catch (Exception ex) {
    ex.printStackTrace();
}
return a;
}
public void save_serverkeys(String res) {
try {
    RSAEncryptionDecryptionrsa_obj = new RSAEncryptionDecryption(context,
    session.getEmail());
    String fileName = "serverpublic.key";
    JSONObject array = new JSONObject(res);
    BigInteger mod = new BigInteger(array.get("Modulus").toString());
    BigIntegerexp = new BigInteger(array.get("Exponent").toString());
    rsa_obj.saveKeys(fileName, mod, exp);

} catch (Exception e) {
    System.out.print("Excn in saving server keys : " + e.toString());
}
} @Override
protected Boolean doInBackground(Void...params) {
try {
if (checkNetworkConnection()) {
String a = check_key(getString(R.string.baseurl) + "keycheck.jsp", this.hash);
if (a != null) {
    JSONObjectobj = new JSONObject(a);
    if (obj.getString("keystatus").compareTo("ok") == 0) {
        save_serverkeys(a);
        return true;
    } else {
        a = keyupload(getString(R.string.baseurl) + "keyupload.jsp", this.hash, this.filepath);
        obj = new JSONObject(a);
        mStatus.setText(a);
        if (obj.getString("status").compareTo("ok") == 0) {
            save_serverkeys(a);
            output = "Key uploaded successfully";
        }
    }
}
}
```



```
        return true;
    } else {
        output = "Key upload failed";
        System.out.print(output);
        return false;
    }
}
} else {
    output = "Exeception in sending";
    return false;
}
} else {
    output = "no net";
    return false;
}
} catch (Exception e) {
    return false;
}
} @ Override
protected void onProgressUpdate(String...values) {
    mStatus.setText(values[0]);
    if (values[0].compareTo("true") == 0)
        mStatus.setText("key sent");
    else
        mStatus.setText("key sending failed");
}
@ Override
protected void onPostExecute(Boolean result) {
    if (result)
        myHandler.sendEmptyMessage(2);
    else
        myHandler.sendEmptyMessage(3);
}
}
}
public class RSA extends AsyncTask< Void, String, Boolean > {
```

```
private Context context;
public RSA(Context context) {
try {
    this.context = context;
    System.out.println("Completed RSA constructor");
} catch (Exception ex) {
    System.out.print("Excn in costr : " + ex.toString());
}
}@ Override
protected Boolean doInBackground(Void...params) {
try {
    System.out.println("Starting RSA key creation");
    RSAEncryptionDecryptionobj = new RSAEncryptionDecryption(this.context,
    session.getEmail());
    if (!obj.keygen())
        return false;

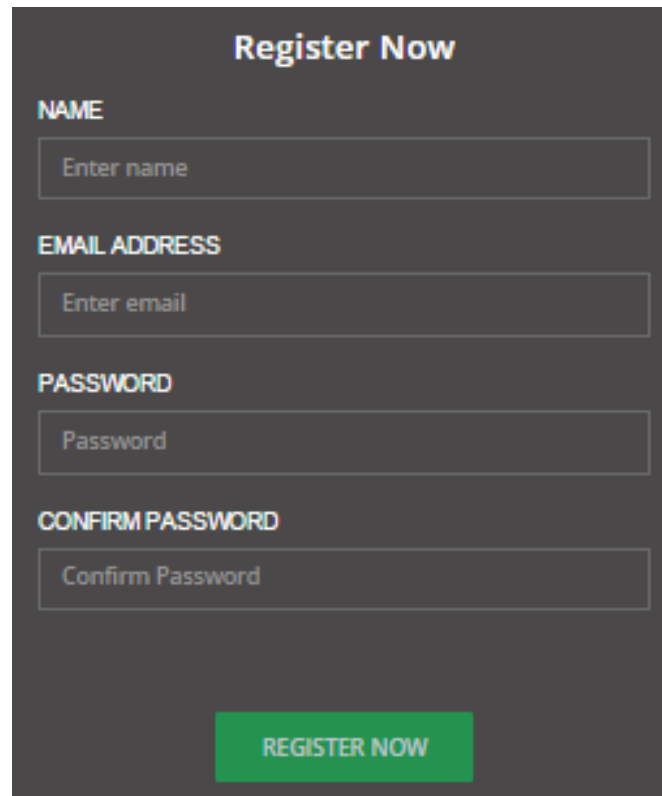
} catch (Exception ex) {
    System.out.print("Excn in costr : " + ex.toString());
}
return true;
}@ Override
protected void onProgressUpdate(String...values) {
mStatus.setText(values[0]);
if (values[0].compareTo("true") == 0)
    mStatus.setText("key generated");
else
    mStatus.setText("keygen fail");
}@ Override
protected void onPostExecute(Boolean result) {
mStatus.setText("Finished generating Key... Uploading Keys");
if (result)
    myHandler.sendMessage(0);
else
    myHandler.sendMessage(1);
```

```
}  
}  
Handler myHandler = new Handler() {  
@ Override  
public void handleMessage(Message msg) {  
switch (msg.what) {  
case 0: //send pub key to db  
    mStatus.setText("key generated successfully");  
    startMyTask(new sendPublicKey(getApplicationContext(),  
getMD5EncryptedString(getFilesDir() + "/" + session.getEmail() + "Public.key"),  
getFilesDir() + "/" + session.getEmail() + "Public.key"));  
break;  
case 1: //send pub key to db  
    mStatus.setText("generating keys failed");  
break;  
case 2: //send ok  
    mStatus.setText("sending key was successful");  
  
    show_main();  
break;  
case 3: //send failed  
    mStatus.setText("sending failed");  
break;  
case 4:  
    mStatus.setText("Key already generated");  
break;  
case 5:  
    mStatus.setText("Syncing with server");  
break;  
case 6:  
    mStatus.setText("Generating keys");  
break;  
default:  
break;  
}  
}
```

```
}  
};  
@ Override  
Public boolean onCreateOptionsMenu(Menu menu) {  
    // Inflate the menu; this adds items to the action bar if it is present.  
    getMenuInflater().inflate(R.menu.menu_key_sync, menu);  
    return true;  
}  
@  
Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    int id = item.getItemId();  
    return super.onOptionsItemSelected(item);  
}  
}
```

## 11. RESULTS AND FINDINGS

### 11.1 SCREEN SHOTS



**Register Now**

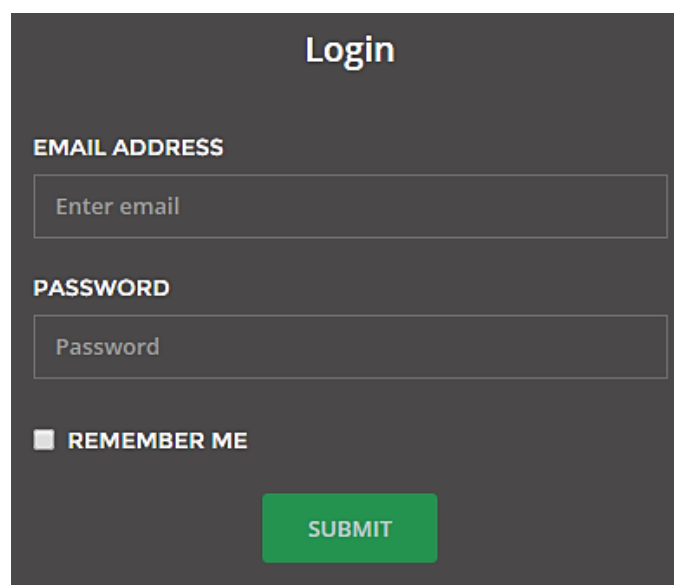
**NAME**

**EMAIL ADDRESS**

**PASSWORD**

**CONFIRM PASSWORD**

**REGISTER NOW**



**Login**

**EMAIL ADDRESS**

**PASSWORD**

☐ **REMEMBER ME**

**SUBMIT**

## Login

Hi Jithu, scan your QR Code




OTP

SUBMIT

## Login

Hi Jithu, scan your QR Code




OTP

SUBMIT

**WEBSITE NAME**

**WEBSITE URL**

**REDIRECT URL**

**WEBSITE LOGO**  

Preview

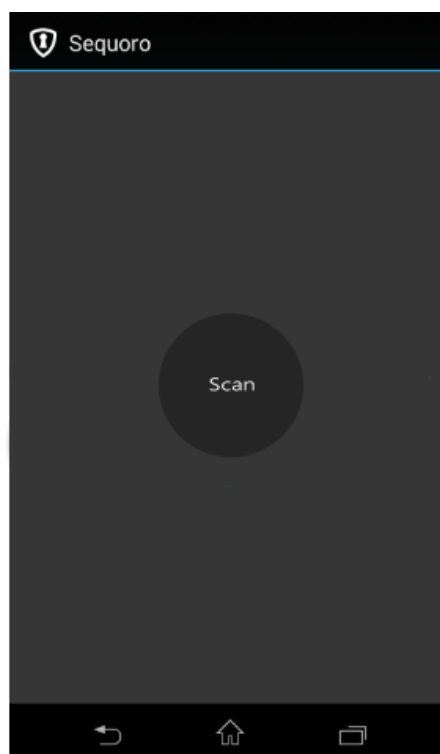
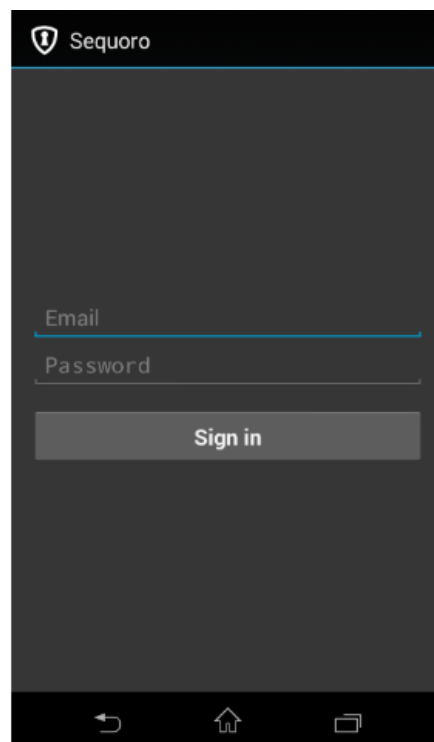
No file chosen

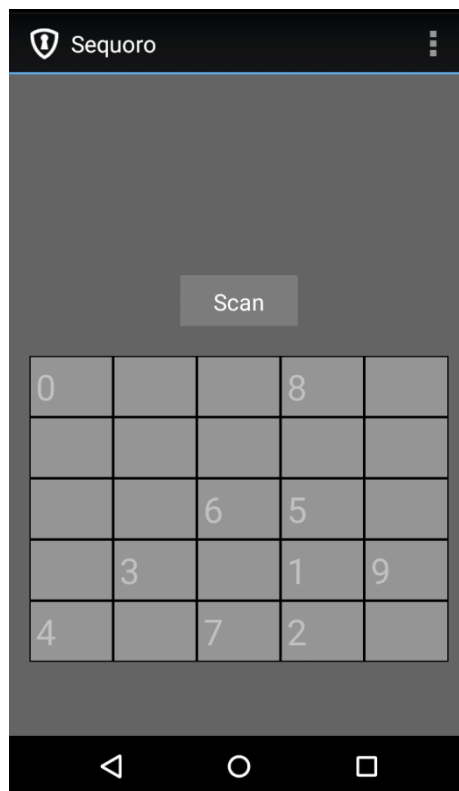
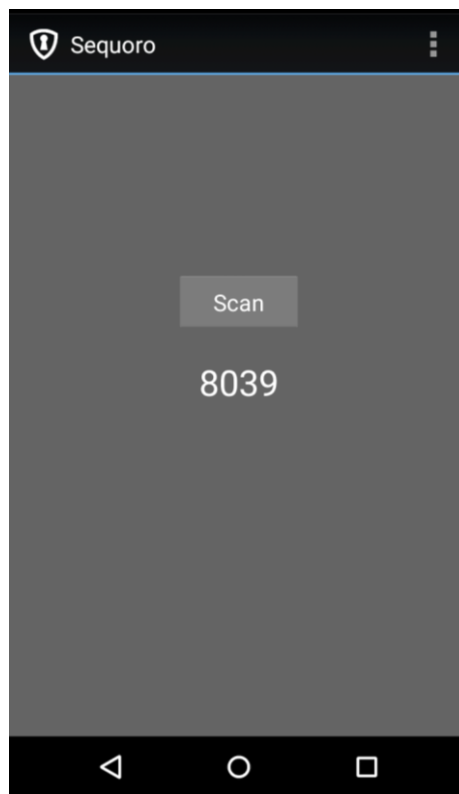
**WEBSITE DESCRIPTION**  

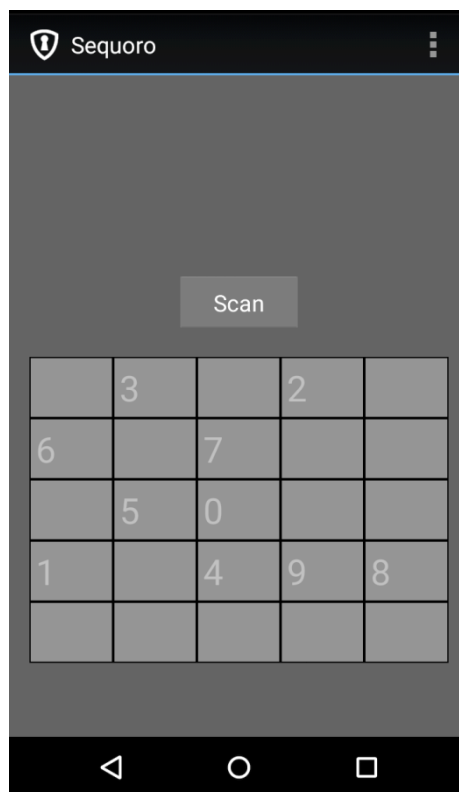
Website Description that user views



## ANDROID APP INTERFACE







## 12. CONCLUSION

The proposed system thus helps us to provide a universal economic secure solution. Sequoro OAuth API is flexible so that it could be extended to any website. Our approach visualizes the security process of authentication using a smartphone aided augmented reality.

Sequoro aims at providing an economical way to provide secure login to websites. It is secure against common password stealing attacks including key loggers. We aim at realizations of protocols that not only improve the user experience but also resist challenging attacks, such as the key logger and malware attacks. Our protocols utilize simple technologies available in most out-of-the-box smartphone devices. Our work indeed opens the door for several other directions that we would like to investigate as a future work. In future, we plan to implement our protocol on the smart glasses such as the Google glass, and conduct the user study.

### 13. REFERENCES

- Google authenticator. <http://code.google.com/p/google-authenticator/>
- Rsa securid. <http://www.emc.com/security/rsa-securid.htm>.
- Cronto. <http://www.cronto.com/>.
- BS ISO/IEC 18004:2006. Information technology. Automatic identification and data capture techniques. ISO/IEC, 2006.
- ZXing. <http://code.google.com/p/zxing/>, 2011.
- D. Boneh and X. Boyen. Short signatures without random oracles. In Proc. of EUROCRYPT, pages 56–73, 2004.
- J. Bonneau, C. Herley, P. C. Van Oorschot, and F. Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In Security and Privacy (SP), 2012 IEEE Symposium on, pages 553–567. IEEE, 2012.
- J. Brown. Zbar bar code reader, zbar android sdk 0.2. <http://zbar.sourceforge.net/>, April 2012.
- C.-H. O. Chen, C.-W. Chen, C. Kuo, Y.-H. Lai, J. M. McCune, A. Studer, A. Perrig, B.-Y. Yang, and T.-C. Wu. Gangs: gather, authenticate group securely. In J. J. Garcia-Luna-Aceves, R. Sivakumar, and P. Steenkiste, editors, MOBICOM, pages 92–103. ACM, 2008.
- S. Chiasson, P. van Oorschot, and R. Biddle. Graphical password authentication using cued click points. In Proc. of ESORICS, 2008.
- D. Crockford. The application/json media type for javascript object notation (json). <http://www.ietf.org/rfc/rfc4627.txt?number=4627>, July 2006.