



VIT[®]

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

CSE3501 – Information Security Analysis and Audit

J Component Report

A project report titled
Authentication using OTP and QR

By

19BEC1308

UTKARSH MAURYA

BACHELOR OF TECHNOLOGY
IN
ELECTRONICS AND COMPUTER ENGINEERING

Submitted to

Vijaya Kumar P

School of Computer Science and Engineering

November 2021

TABLE OF CONTENTS

Ch. No	Chapter	Page Number
1	Introduction	7
2	Related Work	8
3	TRADITIONAL AUTHENTICATION SYSTEM AND ITS LIMITATIONS	9
4	PROPOSED ALGORITHM	10-11
5	Conclusion	12
6	Appendix	
	1.Code	13-39
	2.Implementation	40-43
7	References	44

ABSTRACT

- Ensuring the user authentication and the verification of online transactions that are performed on an untrusted computer or device is an important and challenging problem.
- Most people are unaware that scanning an unknown QR code offers serious security concerns. While the QR code itself isn't dangerous, there is no opportunity to evaluate the site it will lead you to such as the case with an email or website
- To help combat security issues around QR code. QR code reader **Norton Snap** verifies the safety of websites before they are allowed to load on your mobile device.
- As we are performing the authentication using QR. The QR only directs to particular URL.
- Before the data encoding process begin, an error correction level is selected to create error correction code words based on the encoded data. By this, error correction code words able to correct the error if QR code reader did not read the data correctly
- Users could generate the QR-Code image for authentication easily through mobile application on their smart devices.
- we analyze the security of our scheme and discuss the mechanisms in the scheme for circumventing a variety of security threats including password stealing, man-in-the-middle and man-in-the-browser attacks.

KEYWORDS: artificial intelligence algorithm; QR image code; image recognition; backpropagation neural networks; two-dimensional code distortion

INTRODUCTION

IMPORTANCE OF TECHNOLOGY

- Smart Phones, greatly expanding in the recent mobile market, are equipped with various features compared to existing feature phones and provide the conveniences to in several ways. The camera, one of the features of a smartphone, creates the digital contents, such as photos and videos, and plays a role for the media which transmits information, such as video calls and bar code reader.
- QR-Code recognition is also one of camera features. It contains a variety of information in two-dimensional bar code type in matrix format, and makes it possible to obtain the information by using smart phones.

PROPOSED WORK.

- The user logs in or registers into a website. Now the user is required to open an application on a smartphone which is protected by a pin number and scan the one time QR code that is displayed using the phone's primary camera. The application then communicates with the server through an out of band channel and provides a proof of possession of the device. Here the user acts as the conduit between the authorized device and the authentication entity.

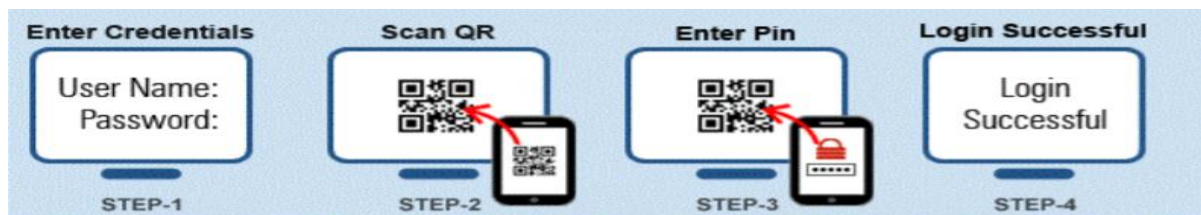


Fig.1:

Block diagram of procedure

- The primary aim of this chapter is to verify the effectiveness of the algorithm proposed through the preprocessing of QR code images. The preprocessing process includes grayscale, filtering, binarization, distortion correction, and perspective projection inverse transformation.
- The filtering uses an improved adaptive median filter algorithm to filter the image, and the distortion correction uses a distortion correction method based on the BP neural network.

RELATED WORK

- Hongyu L, Hui C, Ying W, Yong C, Wei Y. Prediction of two-dimensional topography of laser cladding based on neural network. Int J Mod Phys B.2019;19:2–25.
- Rathee G, Sharma A, Saini H, Kumar R, Iqbal R. A hybrid framework for multimedia data processing in IoT-healthcare using blockchain technology. Multimed Tools Appl. 2019;19:1–23.
- Rathee G, Sharma A, Kumar R, Iqbal R. A secure communicating things network framework for industrial IoT using blockchain technology. Ad Hoc Netw.
- Frankovský P, Pástor M, Dominik L, Kicko M, Trebuňa P, Hroncová D, et al. Wheeled mobile robot in structured environment. In 2018 ELEKTRO. IEEE; 2018 May. p. 1–5.

TRADITIONAL AUTHENTICATION SYSTEM AND ITS LIMITATIONS

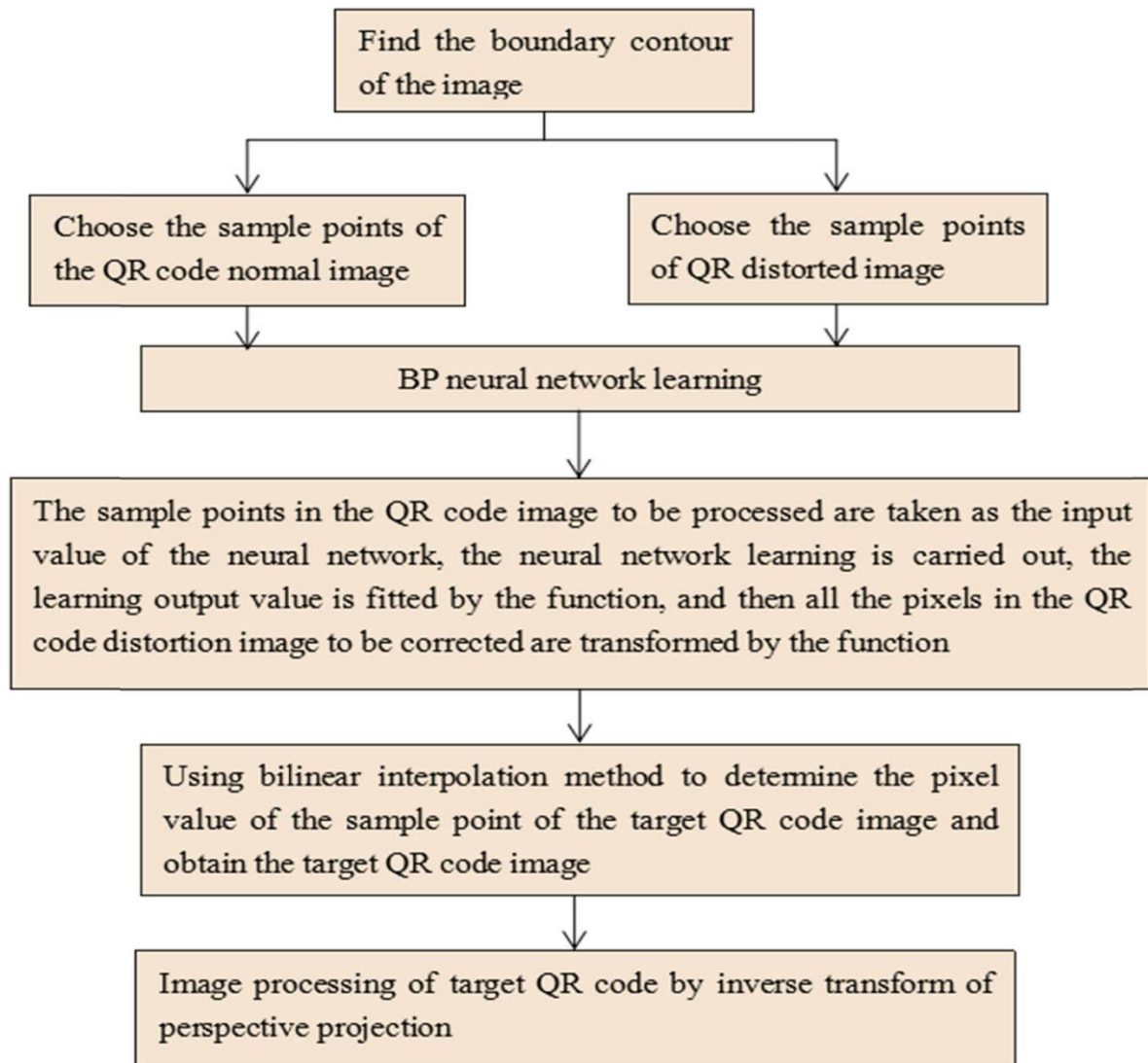
- Password-based authentication.
 - ❖ Passwords are prone to **phishing** attacks and bad hygiene that weakens effectiveness. An average person has about 25 different online accounts, but only **54%** of users use different passwords across their accounts.
- Multi-factor authentication.
 - ❖ MFA may be a good defense against **most account hacks**, but it has its own pitfalls. People may lose their phones or SIM cards and not be able to generate an authentication code.
- Certificate-based authentication.
 - ❖ While the idea of digital certificates is to block outsiders from intercepting your messages, the system is not an infallible one. In 2011, for example, a Dutch digital certificate authority called **DigiNotar** was compromised by hackers.

PROPOSED ALGORITHM

- The basic idea of using the BP neural network to process QR code images is to find the polynomial relationship between these pixel coordinates through the self-learning ability of the BP neural network
- First, consider the point coordinate set on the distorted QR code image as the input layer node of the neural network, and the corresponding output layer node is the point coordinate set on the standard image corresponding to the distorted QR code image.
- The same operation is taken for multiple images, thus forming a set of learning sample data sets. Through the learning of the learning sample set, the neural network forms a distortion mode from point coordinates to point coordinates.
- Snap2pass, and its extension Snap2pay, is a QR code-based approach which requires a system to have an active connection.

To login to a website, the server sends a QR code, which encodes a crypto-graphic challenge, to the browser. The user is to take a picture of the QR code.

Flow Chart



CONCLUSION

This idea can be implemented to achieve security to a great extent in applications such as Net Banking, Online Shopping, detecting Counterfeit products etc. The usage of true random numbers in the generation of QR codes and OTP itself makes it very unique and secure.

The highlight of this technique is that there is no necessity for carrying external hardware such as tokens and smartcards. These features make it a very attractive option for second level authentication in future projects.

APPENDIX

Code / Implementation

1. Index.js

```
import React from "react"
import ReactDOM from "react-dom"
import App from "../components/App"
import "bootstrap/dist/css/bootstrap.min.css"
```

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById("root")
)
```

2. Login.js

```
import React, { useRef, useState } from "react"
import { Form, Button, Card, Alert } from "react-bootstrap"
import { useAuth } from "../contexts/AuthContext"
import { Link, useHistory } from "react-router-dom"

export default function Login() {
  const emailRef = useRef()
  const passwordRef = useRef()
  const { login } = useAuth()
  const [error, setError] = useState("")
  const [loading, setLoading] = useState(false)
  const history = useHistory()

  async function handleSubmit(e) {
    e.preventDefault()

    try {
      setError("")
      setLoading(true)
      await login(emailRef.current.value, passwordRef.current.value)
      history.push("/")
    } catch {
      setError("Failed to log in")
    }

    setLoading(false)
  }
}
```

```

return (
  <>
    <Card>
      <Card.Body>
        <h2 className="text-center mb-4">Log In</h2>
        {error && <Alert variant="danger">{error}</Alert>}
        <Form onSubmit={handleSubmit}>
          <Form.Group id="email">
            <Form.Label>Email</Form.Label>
            <Form.Control type="email" ref={emailRef} required />
          </Form.Group>
          <Form.Group id="password">
            <Form.Label>Password</Form.Label>
            <Form.Control type="password" ref={passwordRef} required />
          </Form.Group>
          <Button disabled={loading} className="w-100" type="submit">
            Log In
          </Button>
        </Form>
        <div className="w-100 text-center mt-3">
          <Link to="/forgot-password">Forgot Password?</Link>
        </div>
      </Card.Body>
    </Card>
    <div className="w-100 text-center mt-2">
      Need an account? <Link to="/signup">Sign Up</Link>
    </div>
  </>
)
}

```

3. Signup.js

```
import React, { useRef, useState } from "react"
import { Form, Button, Card, Alert } from "react-bootstrap"
import { useAuth } from "../contexts/AuthContext"
import { Link, useHistory } from "react-router-dom"

export default function Signup() {
  const emailRef = useRef()
  const passwordRef = useRef()
  const passwordConfirmRef = useRef()
  const { signup } = useAuth()
  const [error, setError] = useState("")
  const [loading, setLoading] = useState(false)
  const history = useHistory()

  async function handleSubmit(e) {
    e.preventDefault()

    if (passwordRef.current.value !== passwordConfirmRef.current.value) {
      return setError("Passwords do not match")
    }

    try {
      setError("")
      setLoading(true)
      await signup(emailRef.current.value, passwordRef.current.value)
      history.push("/")
    } catch {
      setError("Failed to create an account")
    }

    setLoading(false)
  }

  return (
    <>
      <Card>
        <Card.Body>
          <h2 className="text-center mb-4">Sign Up</h2>
          {error && <Alert variant="danger">{error}</Alert>}
          <Form onSubmit={handleSubmit}>
            <Form.Group id="email">
              <Form.Label>Email</Form.Label>
              <Form.Control type="email" ref={emailRef} required />
            </Form.Group>
            <Form.Group id="password">
              <Form.Label>Password</Form.Label>
              <Form.Control type="password" ref={passwordRef} required />
            </Form.Group>
            <Form.Group id="password-confirm">
              <Form.Label>Password Confirmation</Form.Label>
```

```

        <Form.Control type="password" ref={passwordConfirmRef} required />
      </Form.Group>
      <Button disabled={loading} className="w-100" type="submit">
        Sign Up
      </Button>
    </Form>
  </Card.Body>
</Card>
<div className="w-100 text-center mt-2">
  Already have an account? <Link to="/login">Log In</Link>
</div>
</>
)
}

```

4. UpdateProfile.js

```

import React, { useRef, useState } from "react"
import { Form, Button, Card, Alert } from "react-bootstrap"
import { useAuth } from "../contexts/AuthContext"
import { Link, useHistory } from "react-router-dom"

export default function UpdateProfile() {
  const emailRef = useRef()
  const passwordRef = useRef()
  const passwordConfirmRef = useRef()
  const { currentUser, updatePassword, updateEmail } = useAuth()
  const [error, setError] = useState("")
  const [loading, setLoading] = useState(false)
  const history = useHistory()

  function handleSubmit(e) {
    e.preventDefault()
    if (passwordRef.current.value !== passwordConfirmRef.current.value) {
      return setError("Passwords do not match")
    }

    const promises = []
    setLoading(true)
    setError("")

    if (emailRef.current.value !== currentUser.email) {
      promises.push(updateEmail(emailRef.current.value))
    }
    if (passwordRef.current.value) {
      promises.push(updatePassword(passwordRef.current.value))
    }
  }

```

```

Promise.all(promises)
  .then(() => {
    history.push("/")
  })
  .catch(() => {
    setError("Failed to update account")
  })
  .finally(() => {
    setLoading(false)
  })
}

return (
  <>
    <Card>
      <Card.Body>
        <h2 className="text-center mb-4">Update Profile</h2>
        {error && <Alert variant="danger">{error}</Alert>}
        <Form onSubmit={handleSubmit}>
          <Form.Group id="email">
            <Form.Label>Email</Form.Label>
            <Form.Control
              type="email"
              ref={emailRef}
              required
              defaultValue={currentUser.email}
            />
          </Form.Group>
          <Form.Group id="password">
            <Form.Label>Password</Form.Label>
            <Form.Control
              type="password"
              ref={passwordRef}
              placeholder="Leave blank to keep the same"
            />
          </Form.Group>
          <Form.Group id="password-confirm">
            <Form.Label>Password Confirmation</Form.Label>
            <Form.Control
              type="password"
              ref={passwordConfirmRef}
              placeholder="Leave blank to keep the same"
            />
          </Form.Group>
          <Button disabled={loading} className="w-100" type="submit">
            Update
          </Button>
        </Form>
      </Card.Body>
    </Card>
    <div className="w-100 text-center mt-2">

```

```

        <Link to="/">Cancel</Link>
      </div>
    </>
  )
}

```

5. PrivateRoute.js

```

import React from "react"
import { Route, Redirect } from "react-router-dom"
import { useAuth } from "../contexts/AuthContext"

export default function PrivateRoute({ component: Component, ...rest }) {
  const { currentUser } = useAuth()

  return (
    <Route
      {...rest}
      render={props => {
        return currentUser ? <Component {...props} /> : <Redirect to="/login" />
      }}
    ></Route>
  )
}

```

6. ForgotPassword.js

```
import React, { useRef, useState } from "react"
import { Form, Button, Card, Alert } from "react-bootstrap"
import { useAuth } from "../contexts/AuthContext"
import { Link } from "react-router-dom"

export default function ForgotPassword() {
  const emailRef = useRef()
  const { resetPassword } = useAuth()
  const [error, setError] = useState("")
  const [message, setMessage] = useState("")
  const [loading, setLoading] = useState(false)

  async function handleSubmit(e) {
    e.preventDefault()

    try {
      setMessage("")
      setError("")
      setLoading(true)
      await resetPassword(emailRef.current.value)
      setMessage("Check your inbox for further instructions")
    } catch {
      setError("Failed to reset password")
    }

    setLoading(false)
  }

  return (
    <>
      <Card>
        <Card.Body>
          <h2 className="text-center mb-4">Password Reset</h2>
          {error && <Alert variant="danger">{error}</Alert>}
          {message && <Alert variant="success">{message}</Alert>}
          <Form onSubmit={handleSubmit}>
            <Form.Group id="email">
              <Form.Label>Email</Form.Label>
              <Form.Control type="email" ref={emailRef} required />
            </Form.Group>
            <Button disabled={loading} className="w-100" type="submit">
              Reset Password
            </Button>
          </Form>
          <div className="w-100 text-center mt-3">
            <Link to="/login">Login</Link>
          </div>
        </Card.Body>
      </Card>
      <div className="w-100 text-center mt-2">
```



```

        Need an account? <Link to="/signup">Sign Up</Link>
      </div>
    </>
  )
}

```

7. Dashboard.js

```

import React, { useState } from "react"
import { Card, Button, Alert } from "react-bootstrap"
import { useAuth } from "../contexts/AuthContext"
import { Link, useHistory } from "react-router-dom"

export default function Dashboard() {
  const [error, setError] = useState("")
  const { currentUser, logout } = useAuth()
  const history = useHistory()

  async function handleLogout() {
    setError("")

    try {
      await logout()
      history.push("/login")
    } catch {
      setError("Failed to log out")
    }
  }

  return (
    <>
      <Card>
        <Card.Body>
          <h2 className="text-center mb-4">Profile</h2>
          {error && <Alert variant="danger">{error}</Alert>}
          <strong>Email:</strong> {currentUser.email}
          <Link to="/update-profile" className="btn btn-primary w-100 mt-3">
            Update Profile
          </Link>
        </Card.Body>
      </Card>
      <div className="w-100 text-center mt-2">
        <Button variant="link" onClick={handleLogout}>
          Log Out
        </Button>
      </div>
    </>
  )
}

```

8. App.js

```
import React from "react"
import Signup from "../Signup"
import { Container } from "react-bootstrap"
import { AuthProvider } from "../contexts/AuthContext"
import { BrowserRouter as Router, Switch, Route } from "react-router-dom"
import Dashboard from "../Dashboard"
import Login from "../Login"
import PrivateRoute from "../PrivateRoute"
import ForgotPassword from "../ForgotPassword"
import UpdateProfile from "../UpdateProfile"

function App() {
  return (
    <Container
      className="d-flex align-items-center justify-content-center"
      style={{ minHeight: "100vh" }}
    >
      <div className="w-100" style={{ maxWidth: "400px" }}>
        <Router>
          <AuthProvider>
            <Switch>
              <PrivateRoute exact path="/" component={Dashboard} />
              <PrivateRoute path="/update-profile" component={UpdateProfile} />
              <Route path="/signup" component={Signup} />
              <Route path="/login" component={Login} />
              <Route path="/forgot-password" component={ForgotPassword} />
            </Switch>
          </AuthProvider>
        </Router>
      </div>
    </Container>
  )
}

export default App
```

1.1 Implementation

The image shows a login form titled "Log In". It contains two input fields: "Email" and "Password". Below the password field is a blue button labeled "Log In". Underneath the button is a link "Forgot Password?". At the bottom of the form is a link "Need an account? Sign Up".

Log In

Email

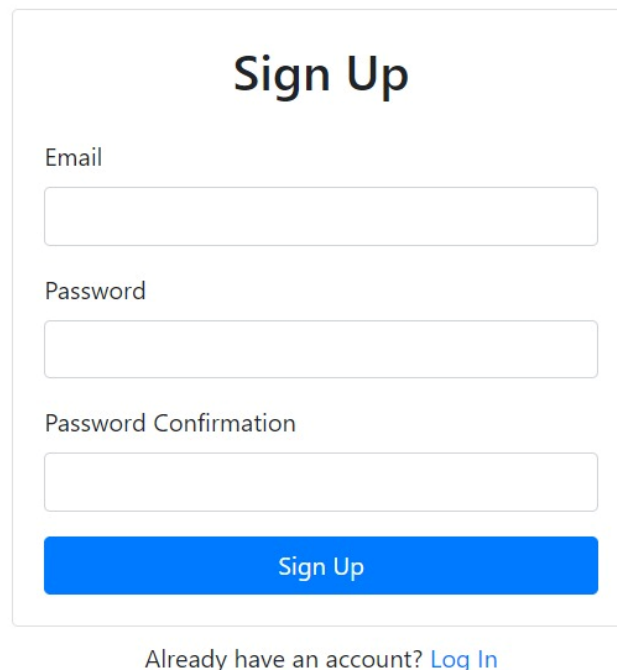
Password

Log In

[Forgot Password?](#)

Need an account? [Sign Up](#)

Fig1.1: Login with the registered credentials using email-id



The Sign Up form is a vertical rectangle with a light gray border. At the top, the title "Sign Up" is centered in a bold, black, sans-serif font. Below the title are three input fields, each with a label to its left: "Email", "Password", and "Password Confirmation". The input fields are white with a thin gray border. At the bottom of the form is a solid blue button with the text "Sign Up" in white, centered. Below the button, the text "Already have an account? [Log In](#)" is centered, with "Log In" in blue.

Sign Up

Email

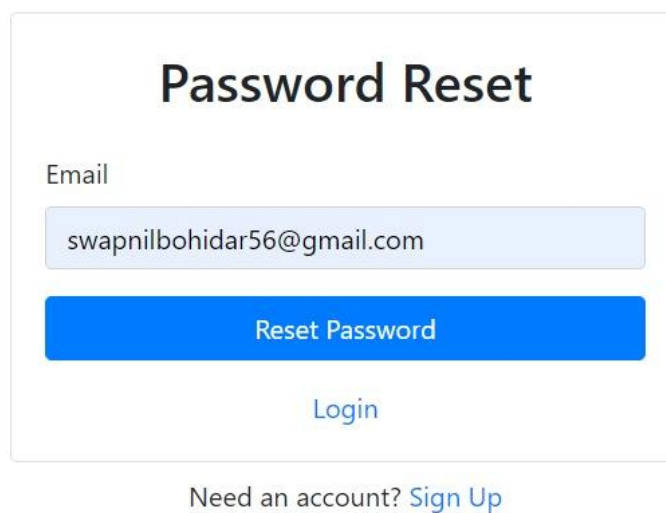
Password

Password Confirmation

Sign Up

Already have an account? [Log In](#)

Fig 1.2: Use your credentials to signup



The Password Reset form is a vertical rectangle with a light gray border. At the top, the title "Password Reset" is centered in a bold, black, sans-serif font. Below the title is an input field with the label "Email" to its left. The input field contains the text "swapnilbohidar56@gmail.com". Below the input field is a solid blue button with the text "Reset Password" in white, centered. Below the button, the text "Login" is centered in blue. At the bottom of the form, the text "Need an account? [Sign Up](#)" is centered, with "Sign Up" in blue.

Password Reset

Email

swapnilbohidar56@gmail.com

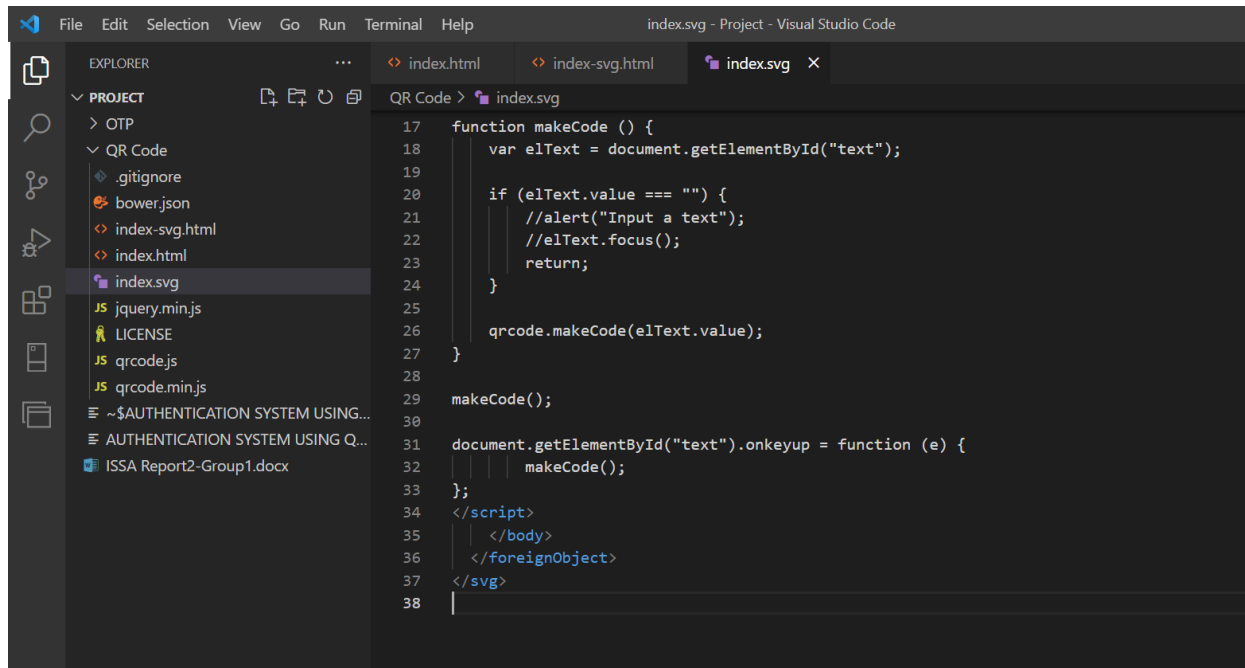
Reset Password

Login

Need an account? [Sign Up](#)

Fig 1.3 Password Reset Window

QR AUTHENTICATION



```
17 function makeCode () {
18     var elText = document.getElementById("text");
19
20     if (elText.value === "") {
21         //alert("Input a text");
22         //elText.focus();
23         return;
24     }
25
26     qrcode.makeCode(elText.value);
27 }
28
29 makeCode();
30
31 document.getElementById("text").onkeyup = function (e) {
32     makeCode();
33 };
34 </script>
35 </body>
36 </foreignObject>
37 </svg>
38
```



REFERENCES

- [1] E. Barkan and E. Biham. Conditional Estimators: An Effective Attack on A5/1, pages 1–19. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006.
- [2] 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 IEEE Symposium on Security and Privacy, pages 553–567. IEEE Computer Society, 2012.
- [3] 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. Technical Report 817, University of Cambridge Computer Laboratory, 2012.
- [4] Y. Chow, W. Susilo, M. H. Au, and A. M. Barmawi. A visual one-time password authentication scheme using mobile devices. In L. C. K. Hui, S. H. Qing, E. Shi, and S. Yiu, editors, ICICS 2014, volume 8958 of Lecture Notes in Computer Science, pages 243–257. Springer, 2014.
- [5] Y. Chow, W. Susilo, G. Yang, J. G. Phillips, I. Pranata, and A. M. Barmawi. Exploiting the error correction mechanism in QR codes for secret sharing. In J. K. Liu and R. Steinfeld, editors, ACISP 2016, volume 9722 of Lecture Notes in Computer Science, pages 1–17. Springer, 2016.
- [6] D. S., Kim, B. H., & Lee, J. K., (2011) “A study on authentication system using QR code for mobile cloud computing environment”, Future Information Technology, pp500-507.
- [7] Kao, Y. W., Luo, G. H., Lin, H. T., Huang, Y. K., & Yuan, S. M., (2011) “Physical access control based on QR code” Cyber-enabled distributed computing and knowledge discovery pp285-288.
- [8] H. Sun, "Cryptanalysis of password authentication schemes with smart cards," Information Security Conference 2001, pp. 221-223, May 2001.
- [9] Financial security Agency, "Guide for End-to-End Encryption", 2007.10
- [10] M.L. Das, A. Saxena, and V.P. Gulati, "A Dynamic ID-based Remote User Authentication Scheme," IEEE Transactions on Consumer Electronics, vol.50, no.2, pp. 629-631, May 2004.