A Project Review Report

on

OTP Grabber

Submitted in partial fulfillment of

the requirement for the award of

the degree of

B.Tech in Computer Science and Engineering

Under The
Supervision of
Dr. A Daniel

Submitted By:
Anchal Sharma
18SCSE1010752
Ishan Adhikari
Bairagi
18SCSE1120018

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
OCT,2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project entitled **"OTP Hands Free"** in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of July 2021 to December 2021 under the supervision of Dr. A. Daniel, AP, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by me/us for the award of any other degree of this or any other places.

Ishan Adhikari Bairagi (18SCSE1120018)

Anchal Sharma(18SCSE1010752)

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. A. Daniel

AP

CERTIFICATE

The Final Project Viva-Voce examination of Ishan Adhikari Bairagi(18SCSE1120018) and Anchal Sharma(18SCSE1010752) has been held on _____ and their work is recommended for the award of Bachelor of Technology

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: December,2021

Place: Greater Noida

ACKNOWLEDGEMENT

We would like to express our special thanks to our project guide Dr. A Daniel for motivating us and continuously guiding us towards the completion of this project. We owe this great opportunity to work on the project to Galgotias University. Our timely reviewers Mr. C Ramesh Kumar and Mr. S Prakash for throughout evaluation of our work and navigating us with proper steering from mistake during the course of project is also to be highly appreciated. All the researchers and authors who have already done commendable work in this field and provide generous knowledge to others are also to be thanked. The open source world of knowledge for programming, documentation and text formatting is a prestigious gift for all and the various websites like “Geeks for Geeks”, “W3 Schools”, “Tutorials Point” are doing great for the cause of technology computer programming. In the end we extend gratitude to our friends, colleagues and parents with whose support this project was spun into reality.

ABSTRACT

Since the advent of secure transactions over the internet, OTPSs have become an efficient and secure method to verify the transfer of money from one place to another. This generation that tirelessly keeps working on laptops or personal computers is often left wandering searching and submitting OTPs via phone i.e. physically see the OTP on phone and then enter it on the laptop to complete the transaction.

To save this hassle we intend to create a system which would link the phone to the PCs/Laptops for submission of the OTP without the need of holding phones altogether. The system created shall be a secure method of message transfer between two devices which shall provide much convenience to the users.

Latest technologies for development shall be incorporated including Python and a web framework like Django.

A chrome extension for facilitating text messages on laptops or PCs exists but its scope is limited and there are concerns of security.

Initially the roll out shall be for windows but in future the scope of this shall not be limited and it can be extended to multiple operating systems.

CONTENTS

List of Figures

Acronyms

OTP	One Time Password
API	Application Program Inteface
GUI	Graphical User Interface
2FA	Two Factor Authentication
MITM	Man in The Middle
ZITMO	Zeus in the Mobile

CHAPTER 1

INTRODUCTION

Secure Transactions have found new heights since incorporating the system of OTPs i.e. One Time Passwords. A transaction is verified by generating a unique code for each instance and successfully processed only when the code generated and sent to the user matches the one entered by user. This has significantly reduced the number of fraudulent transactions occurring due to multiple attacks on password/pin systems which involve brute force approach, phishing etc.

But with OTPs transactions can be carefully verified as you need the physical device in which the previously entered mobile number or email is logged in.

A very secure method like OTP has its own benefits but it comes with several cons too. We are often left perplexed searching for OTPs on mobile phones which can be time consuming and if we do not have the device near us then the OTP may expire as most of them are valid for only 2-3 minutes.

The system we propose can be used to view the OTP received on mobile on the PC or Laptop a user is working on. This can be very helpful for those who continuously work on their professional devices and want to remove the middleman between them and their transaction i.e. mobile phones.

Static OTP is a password that is valid for only one login session or transaction, on a computer system or other digital device. A one-time PIN code is a code

that is valid for only one login session or transaction using a mobile phone. It is often used in two factor authentication or 2FA to provide an extra layer of security for the user when he uses an ATM machine or tries to login to a service from a different computer. Since the one-time pin is valid for only a single use, they are not vulnerable as static passwords. We expect a variety of different transport mechanisms to enable OTPs to be received, most notably via SMS, email and hardware devices.

Each of these transport mechanisms will need their own conventions on how to provide OTPs to the browser.

In this draft, we leave the API surface to be extensible to any number of transports.

Use of one time passwords (OTPs) as a second step to logging in seems to be getting more popular recently. There are two main approaches to OTPs, the first being delivery of the OTP over a channel like SMS, and the other being a code that changes every time you use one to log in or on a predefined time schedule, based on a predefined algorithm. To use the first type, one must have a device with network connectivity and a phone number to receive SMS.

With devices like RSA Secure ID tokens or Google Authenticator, a person can generate the second type of OTP manually generate the OTP on your mobile handset instead of receiving it through SMS. Let's say a user wants to verify their phone number with a website. The website sends a text message to the user

over SMS and the user enters the OTP from the message to verify the ownership of the phone number.

With the WebOTP API, these steps are as easy as one tap for the user, as demonstrated in the video. When the text message arrives, a bottom sheet pops up and prompts the user to verify their phone number. After clicking the Verify button on the bottom sheet, the browser pastes the OTP into the form and the form is submitted without the user needing to press Continue.

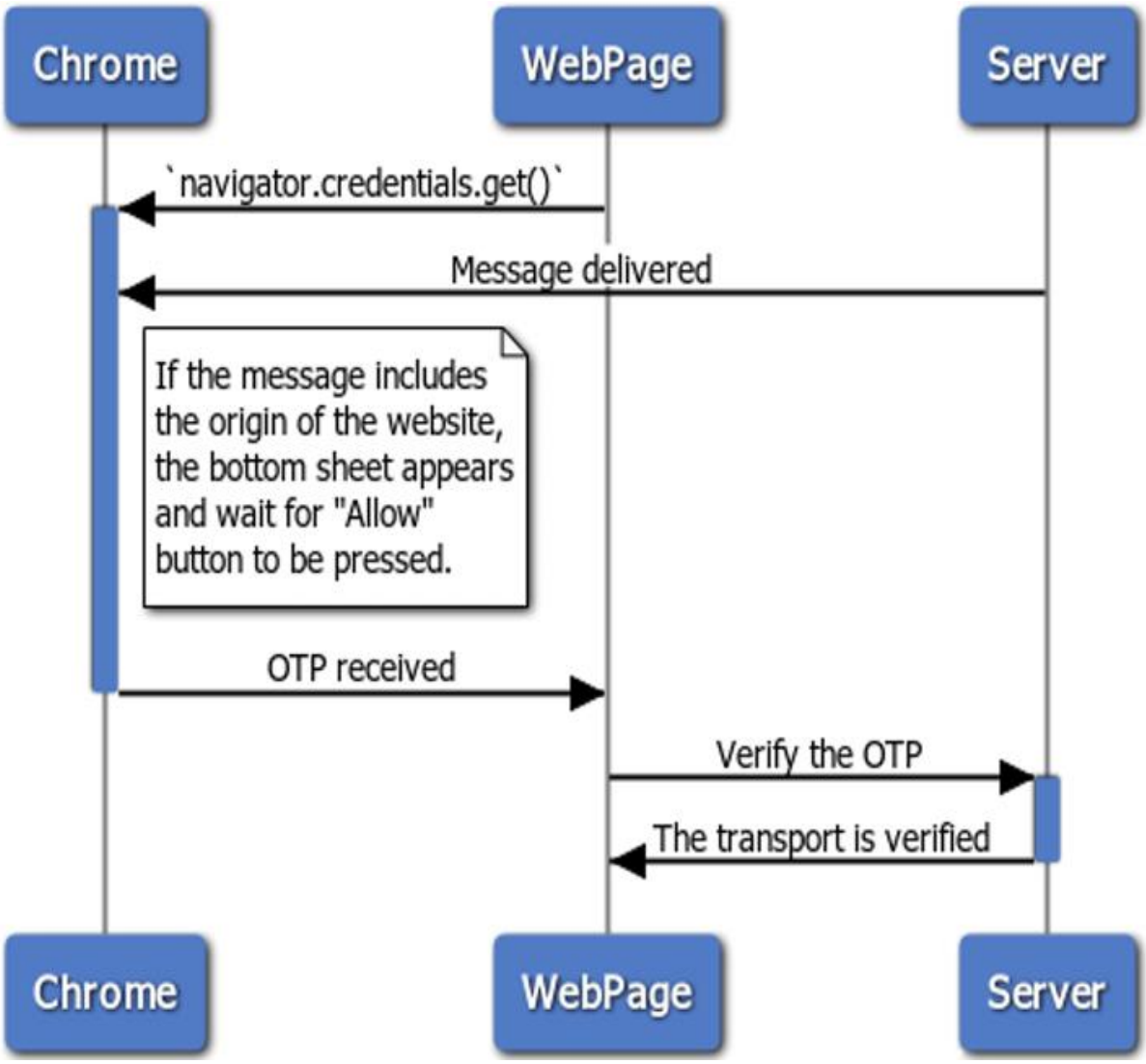

Fig1 Sequence Diagram for the complete process

The whole process is diagrammed in the image above:

We intend to use the following technologies in our project:

1.) Twilio:

Fig2. Twilio's Programmable Wireless Platform benefits

Twilio is a developer platform for communications. Twilio's programmable application program interfaces (APIs) are a set of building blocks developers can use to build the exact customer experiences they want.

The Twilio Customer Engagement Platform can be used to build practically any digital experience, using capabilities like SMS, WhatsApp, Voice, Video, email, and even IoT, across the customer journey. Twilio powers communications for more than 190,000 businesses, and enables nearly 932 billion human

interactions every year. Twilio is the developer platform for communications is reinventing telecom by merging the worlds of cloud computing, web services, and telecommunications. With Twilio, developers and businesses make communications more contextual by embedding voice, video, messaging, and authentication directly into applications.

2.) Tkinter GUI:

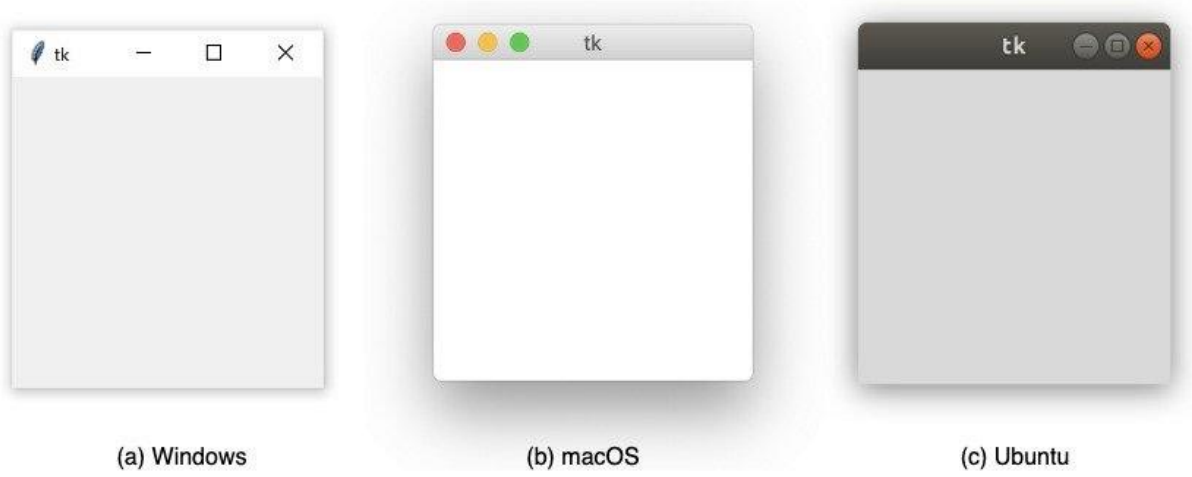

Fig3. Tkinter GUI on various platforms

Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python with tkinter is the fastest and easiest way to create the GUI(Graphical User Interface) applications. Creating a GUI using tkinter is an easy task as: Import the module – tkinter àCreate the main window (container)à Add any number of

widgets to the main windowàApply the event Trigger on the widgets. Tkinter is one of the most popular Python GUI libraries for developing desktop applications. It's a combination of the TK and python standard GUI framework. Tkinter provides diverse widgets such as labels, buttons, text boxes, checkboxes that are used in a graphical user interface application.

3.) WebOTP API:

This is an existing feature available on chrome which will help us and guide us throughout. WebOTP helps users enter a phone number verification code on a mobile website in one tap without switching between apps. The WebOTP API gives websites the ability to programmatically obtain the one-time password from a SMS message and automatically fill the form for users with just one tap without switching apps. The SMS has a specific format and it's bound to the origin, so it mitigates the risk of phishing websites stealing the OTP as well.Many web sites need to verify credentials (e.g. phone numbers and email addresses) as part of their authentication flows. They currently rely on sending one-time-passwords (OTP) to these communication channels to be used as proof of ownership. The one-time-password is manually handed back by the user (typically by copying/pasting) to the web app which is onerous and erroneous.

This a proposal for a client side JavaScript API that enables web sites to request OTPs and a set of transport-specific conventions (we start with SMS while leaving the door open to others) that can be used in coordination with browsers.

3. APIs

i)The client side API:

In this proposal, websites have the ability to call a browser API to request OTPs coming from specific transports (e.g. via SMS).

The browser intermediates the receipt of the SMS and the handing off to the calling website (typically asking for the user's consent), so the API returns a promise asynchronously.Websites call navigator.credentials.get({otp:..., ...}) to retrieve an OTP.The algorithm of navigator.credentials.get() looks through all of the interfaces that inherit from Credential in the Request a Credential abstract operation.

In that operation, it finds OTPCredential which inherits from Credential. It calls OTPCredential.[[CollectFromCredentialStore]]() to collect any credentials that should be available without user mediation, and if it does not find exactly one of those, it then calls OTPCredential.[[DiscoverFromExternalSource]]() to have the user select a credential source and fulfill the request.

Since this specification requires an authorization gesture to create OTP credentials, the OTPCredential.[[CollectFromCredentialStore]]() internal

method inherits the default behavior of Credential.[[CollectFromCredentialStore]](), of returning an empty set.

ii)The server side API:

Once the client side API is called, the website's server can send OTPs to the client via the requested transport mechanisms. For each of these transport mechanism, a server side convention is set in place to guarantee that the OTP is delivered safely and programmatically.

For SMS, for example, servers should send origin-bound one-time code messages to clients. [sms-one-time-codes]

for e.g.: In the following origin-bound one-time code message, the host is "example.com", the code is "123456", and the explanatory text is "Your authentication code is 123456.\n"."Your authentication code is 123456.@example.com #123456

4.) Feature Detection:

Not all user agents necessarily need to implement the WebOTP API at the exact same moment in time, so websites need a mechanism to detect if the API is available. Websites can check for the presence of the OTPCredential global interface.

5.) Web Components:

For the most part, OTP verification largely relies on:

- input, forms and copy/paste, on the client side and
- third party frameworks to send SMS, on the server side.

We expect some of these frameworks to develop declarative versions of this API to facilitate the deployment of their customer's existing code.

6.)Abort API:

Many modern websites handle navigations on the client side. So, if a user navigates away from an OTP flow to another flow, the request needs to be cancelled so that the user isn't bothered with a permission prompt that isn't relevant anymore.

To facilitate that, an abort controller can be passed to abort the request.

7.) The OTPCredential Interface:

The OTPCredential interface extends Credential and contains the attributes that are returned to the caller when a new one time password is retrieved.

OTP Credential's interface object inherits Credential's implementation of [[CollectFromCredentialStore]](origin, options, sameOriginWithAncestors),

and defines its own implementation of [[DiscoverFromExternalSource]](origin, options, sameOriginWithAncestors).

8.) CredentialRequestOptions:

To support obtaining OTPs via navigator.credentials.get(), this document extends the CredentialRequestOptions dictionary as follows:

```
partial dictionary CredentialRequestOptions {
   OTPCredentialRequestOptions otp;
};
```

otp, of type OTPCredentialRequestOptions

This OPTIONAL member is used to make WebOTP requests.

9.) Permissions Policy integration:

This specification defines one policy-controlled feature identified by the feature-identifier token "otp-credentials". Its default allowlist is 'self'.

A Document's permissions policy determines whether any content in that document is allowed to successfully invoke the WebOTP API, i.e., via navigator.credentials.get({otp: { transport: ["sms"]}}). If disabled in any document, no content in the document will be allowed to use the foregoing methods: attempting to do so will return an error.

10.)Addressing:

Each transport mechanism is responsible for guaranteeing that the browser has enough information to route the OTP appropriately to the intended origin.

For example, origin-bound one-time code messages explicitly identify the origin on which the OTP can be used. The addressing scheme must be enforced by the agent to guarantee that it gets routed appropriately.

11) Privacy:

From a privacy perspective, the most notable consideration is for a user agent to enforce the consensual exchange of information between the user and the website.

Specifically, this API allows the programmatic verification of personally identifiable attributes of the user, for example email addresses and phone numbers.

The attack vector that is most frequently raised is a targeted attack: websites trying to find a very specific user accross all of its user base. In this attack, if left unattended, a website can use this API to try to find a specific user that owns a specific phone number by sending all / some (depending on the confidence level) of its users an origin-bound one-time code message and detecting when one is received.

Notably, this API doesn’t help with the acquisition of the personal information, but rather with its verification. That is, this API helps verifying whether the user owns a specific phone number, but doesn’t help acquiring the phone number in the first place (it assumes that the website already has access to it).

Nonetheless, the verification of the possession of these attributes is extra information about the user and should be handled responsibly by a user agent, typically via permission prompts before handing back the OTP to the website.

CHAPTER 2

LITERATURE REVIEW

Though not many systems exist in this particular field but a chrome extension to look on text messages exists and is available on the chrome store. Already existing research in the field of OTPs and mobile PC interaction are there but scarce. A study of few of them brings important and classic ideas in to th perspective.

We learned about OTP generation and encryption and the comparison of 7 different algorithms for OTP generation including the S/Key and hotp:hmac based OTP algorithm etc. helped as get an overview of the scenario. [1]

We also looked on how a large number of notifications affect the system and we assess the impact of notifications on the system and it recommends that only relevant notifications are shown on the desktop.[2]

Then we studied OTP Authentication methods and the proposed Challenge Response Authentication Method for tackling the MITM or man in the middle attack for a much safer OTP transaction.[3]

We also had a brief overview of how fruitful the integration of mobile and PC could be and several limitations and problems faced were discussed.[4]

The importance of the end user was highly advocated by Kuo-Ying Huang[5].

We read about an SMS Threat Model and the various Mobile Trojans like ZITMO Trojan.[6]

An in-depth systematic analysis of the security of SMS authentication usage in

modern mobile platforms was learned by us. Various concerns were raised by researchers over SMS authentication and some ways to prevent malicious activities were presented in the research paper. [7]

We also came to know about the importance of two factor authentication or 2FA and how they help in achieving secure transactions [8].

The use of OTPs or similar texts as a means of early warning in case of disasters was researched and we found pre-emptive texts on mobiles as well as computers very necessary for the mass working continuously on computers in metropolitans. [9]

Since everything is moving to cloud so the usefulness of OTPs in cloud applications using RSA Encryption etc seems a very secure method of safeguarding cloud services. [10]

CHAPTER 3

PROJECT DESIGN

Throughout the project various useful diagrams were created so there is an in-depth understanding of all the modules, entities etc.
The diagrams are:

- **ER DIAGRAM**

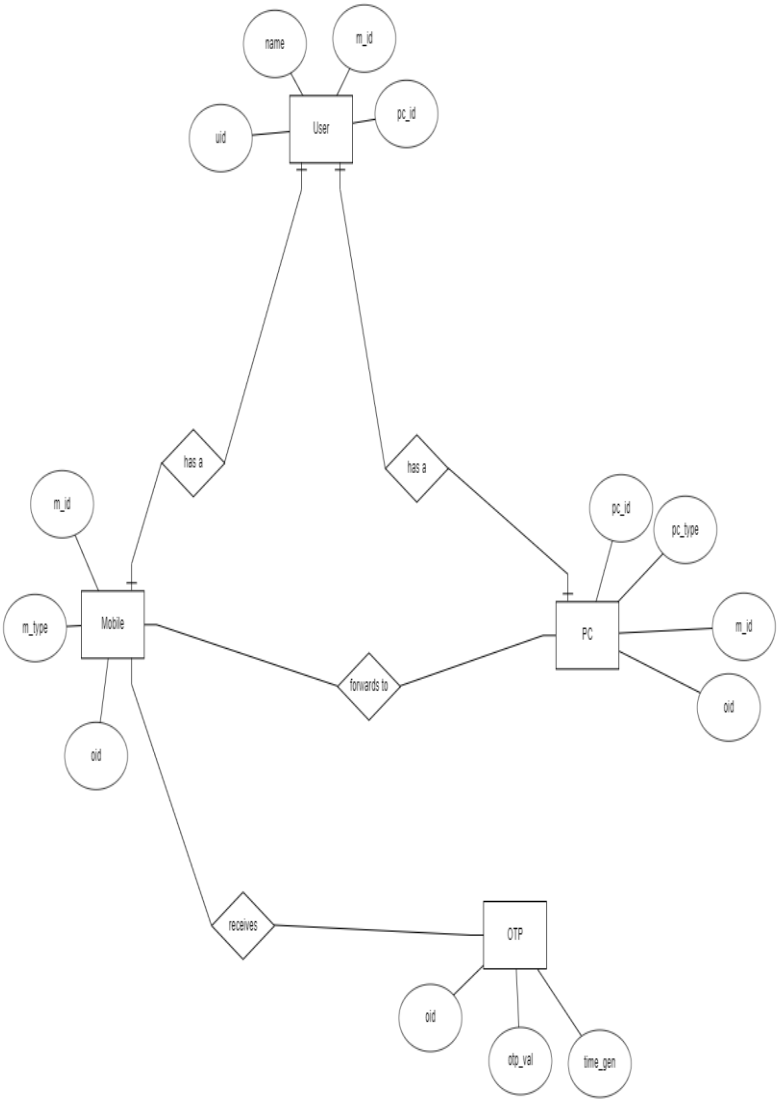

Fig4. ER Diagram

1.) User:

The user will have an id , a mobile and a PC for the system to work.

2.) Mobile:

Each mobile has a unique id, type i.e. Android/iOS and each mobile when receiving an OTP will generate a simultaneous id for every OTP.

3.) PC:

Each PC or Laptop will also have an id and type i.e. whether Windows/MAC/Linux. It also shares the same id for the OTP received on the mobile.

4.) OTP:

Each OTP generated has an id, an OTP Code the time of its generation and the sender name.

- **USE CASE Diagram:**

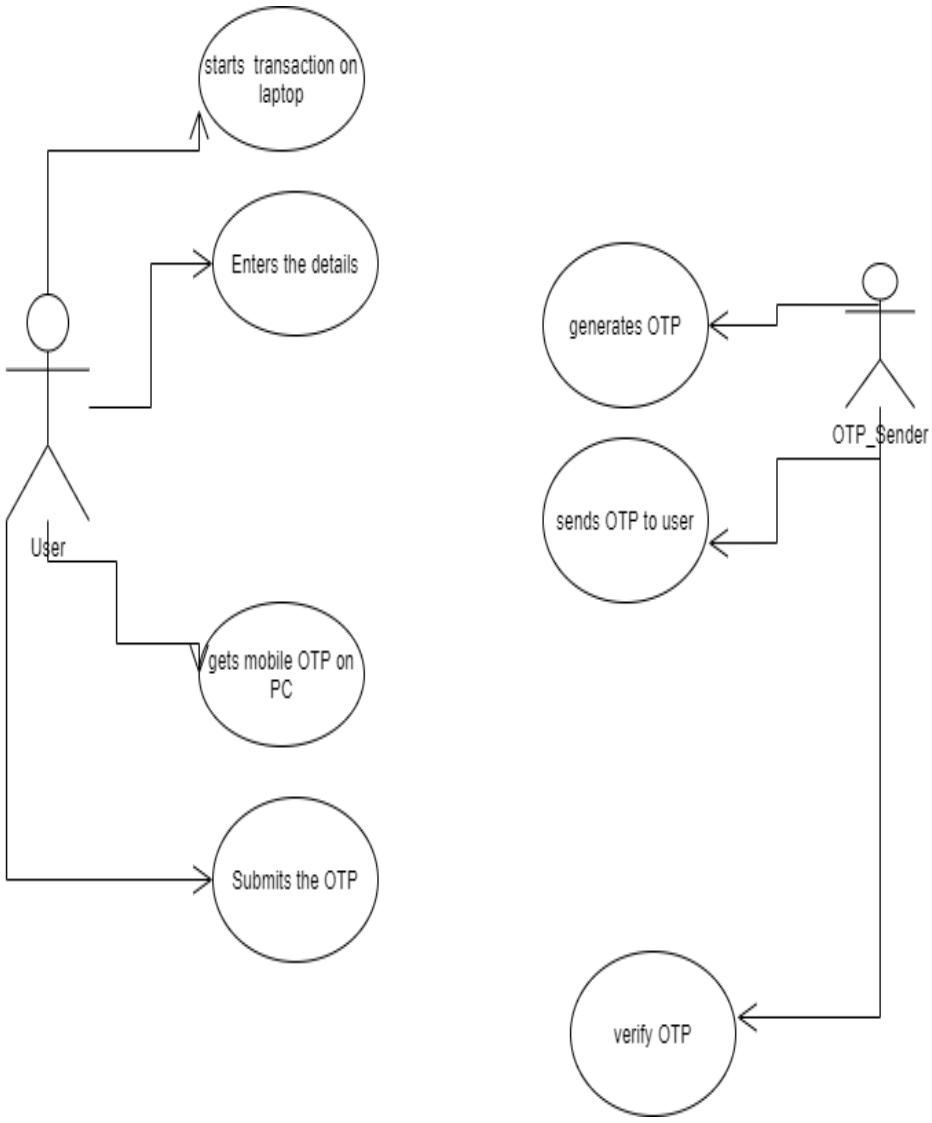

Fig5. UML Diagram

The user starts the transaction on laptop and then enters the relevant details. The OTP Sender now generates a unique OTP and sends it to user.

The user gets the OTP on PC through automated system and submits the OTP which is then verified for successful completion of transaction.

- **Flow Diagram:**

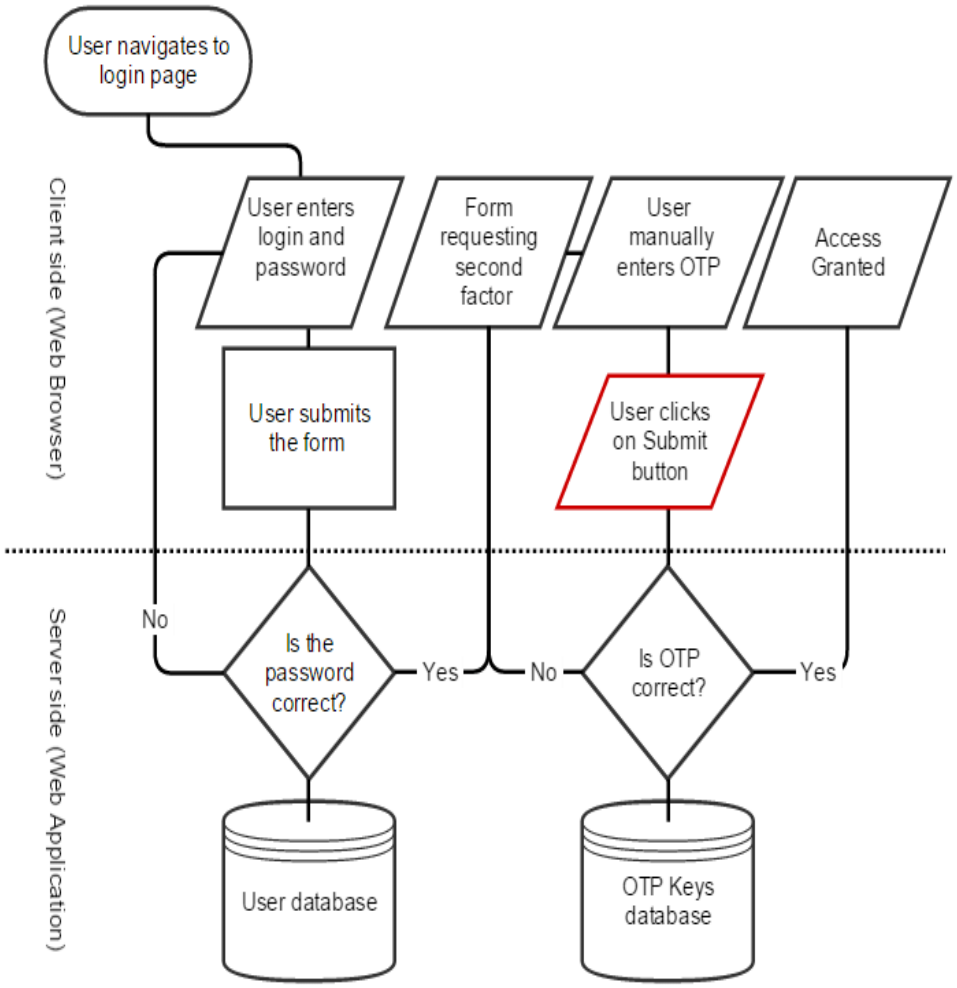

Fig6.Flow Diagram for earlier systems

The user initially logs in to the page and enters the credentials which are verified by the respective database. Now if it is correct it goes to the two factor authentication method or 2nd factor i.e. OTP system which once generated, user has to manually enter.

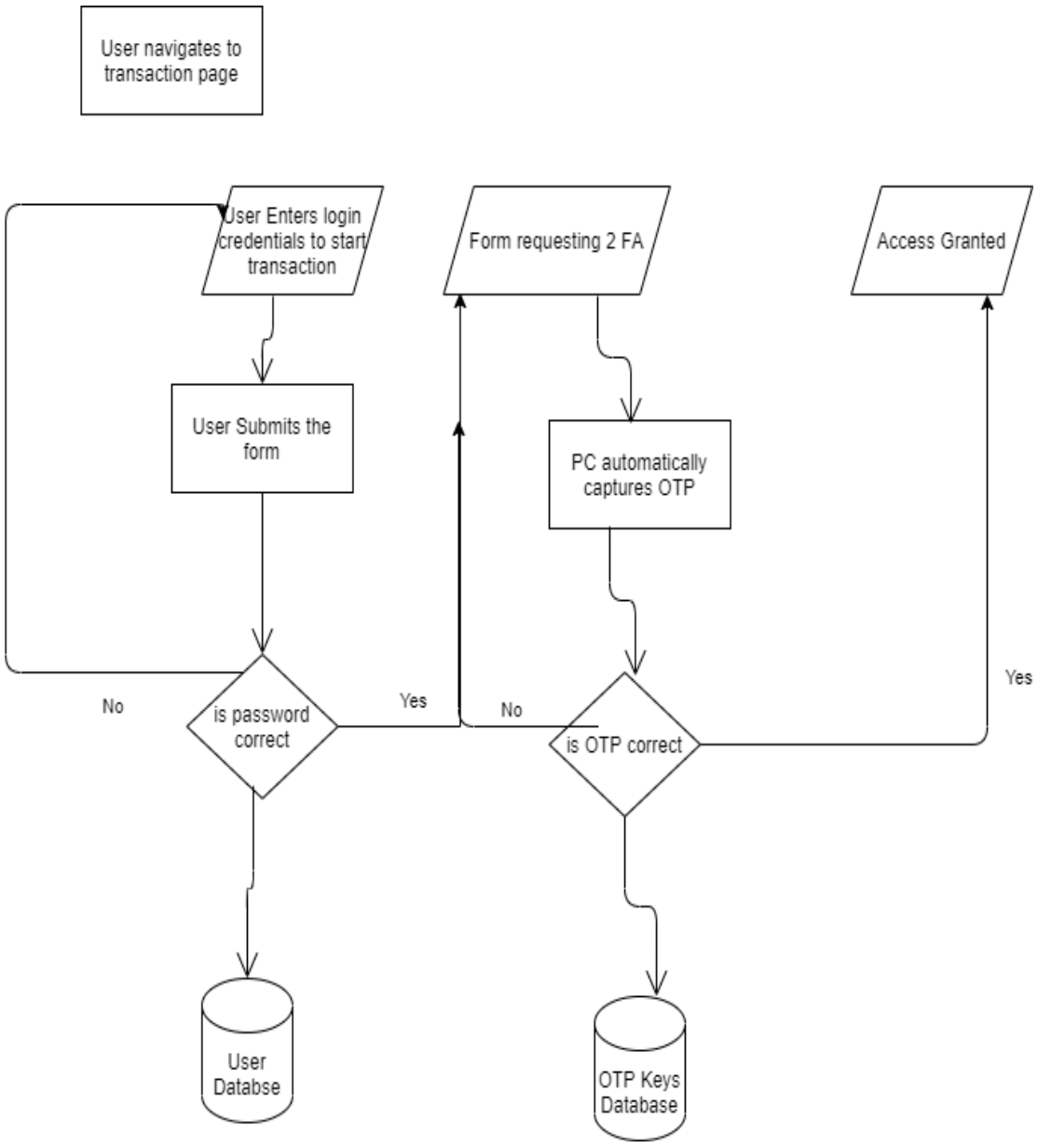

Fig 7. Flow Diagram for proposed system

But with the proposed system we can bypass the manual enter phase as the PC automatically receives the OTP.

CHAPTER 4
THEORY and METHODOLOGY

OTP:

A one-time password (OTP) is an automatically generated numeric or alphanumeric string of characters that authenticates a user for a single transaction or login session.

An OTP is more secure than a static password, especially a user-created password, which can be weak and/or reused across multiple accounts.

OTPs may replace authentication login information or may be used in addition to it to add another layer of security.

Examples of OTP:

OTP security tokens are microprocessor-based smart cards or pocket-size key fobs that produce a numeric or alphanumeric code to authenticate access to the system or transaction. This secret code changes every 30 or 60 seconds, depending on how the token is configured.

Mobile device apps, such as Google Authenticator, rely on the token device and PIN to generate the one-time password for two-step verification.

OTP security tokens can be implemented using hardware, software or on demand. Unlike traditional passwords that remain static or expire every 30 to 60 days, the one-time password is used for one transaction or login session.

Getting a One Time Password:

When an unauthenticated user attempts to access a system or perform a transaction on a device, an authentication manager on the network server generates a number or shared secret, using one-time password algorithms. The same number and algorithm are used by the security token on the smart card or device to match and validate the one-time password and user.

Many companies use Short Message Service (SMS) to provide a temporary passcode via text for a second authentication factor. The temporary passcode is obtained out of band through cellphone communications after the user enters his username and password on networked information systems and transaction-oriented web applications.

For two-factor authentication (2FA), the user enters his user ID, traditional password and temporary passcode to access the account or system.

Types of Tokens in OTP:

- **Hard tokens**

  Hard tokens (as in hardware) are physical devices that transmit OTPs, helping users gain access to accounts and other resources. Hard tokens broadly include:

  Connected tokens: Users connect these tokens into the system or device they're trying to access. Smart cards and USB drives are inserted into a device's smart card reader and USB port, respectively.

  Disconnected tokens: The most frequently used token for multi-factor authentication (MFA). While users don't have to physically insert these tokens, disconnected tokens typically generate OTPs for users to enter. Pocket-size key fobs, keyless entry systems, mobile phones, and banking security devices are some examples of this in action.

  Contactless tokens: These tokens transmit authentication data to a system, which analyzes the information and determines if the user has access rights. Bluetooth tokens are an example of contactless transmission, with no need for physical connections or manual input.

- **Soft tokens**

  Soft tokens (as in software) aren't physical items that we possess. Rather, they exist as software on a device like a laptop or mobile phone. Soft token authentication usually takes the form of an app that sends push notifications or SMS messages for the user to respond to and verify their identity.
  All of these methods follow the same basic process: the user sends authentication data to a system, the system verifies if the information is correct, and, if so, grants the user authorized access. It's the same idea as using a password, but with an OTP the authentication data doesn't travel or leak beyond the user and target system.

Methods of Getting OTP:

1. SMS-based: In this method, every time the user logs in, they receive a text message to their registered phone number, which contains a One Time Password.
2. TOTP-based: In this method, while enabling 2-factor authentication, the user is asked to scan a QR image using a specific smartphone application. That application then continuously generates the One Time Password for the user.

The working of SMS based OTPs is known to all so we have to know about the TOTP based method.

Working of TOTP-based method

By using the TOTP method, we are creating a one time password on the user side (instead of server side) through a smartphone application.
This means that users always have access to their one time password. So it prevents the server from sending a text message every time user tries to login. Also, the generated password changes after a certain time interval, so it behaves like a one time password..
The following could be a way to implement this solution:
When the user enables two factor authentication:
1. Backend server creates a secret key for that particular user.
2. Server then shares that secret key with the user's phone application.
3. Phone application initializes a counter.
4. Phone application generate a one time password using that secret key and counter.
5. Phone application changes the counter after a certain interval and regenerates the one time password making it dynamic.
But to address problems like generation of a one time password using a secret key and counter, counter update, keeping track of the counter etc.
These all can be addressed using HOTP algorithm.

HOTP:

HOTP stands for "HMAC-Based One-Time Password". This algorithm was published as RFC4226 by the Internet Engineering Task Force (IETF). HOTP defines an algorithm to create a one time password from a secret key and a

counter.
You can use this algorithm in two steps:

1. The first step is to create an HMAC hash from a secret key and counter.

```
// Obtain HMAC hash (using SHA-1 hashing algorithm) by secretKey and
counter
hmacHash = HMAC-SHA-1(secretKey, counter);
```

2. In this code, the output would be a 20 byte long string. That long string is not suitable as a one time password. So we need a way to truncate that string. HOTP defines a way to truncate that string to our desired length.

```
// hmacHash[19] means 19th byte of the string.offset = hmacHash[19] & 0xf;
truncatedHash = (hmacHash[offset++] & 0x7f) << 24 | (hmacHash[offset++] &
0xff) << 16 | (hmacHash[offset++] & 0xff) << 8 | (hmacHashh[offset++] &
0xff);
finalOTP = (truncatedHash % (10 ^ numberOfDigitsRequiredInOTP));
```

It might look scary, but it is not. In this algorithm, we first obtain offset which is the last 4 bits of hmacHash[19]. After that, we concatenate the bytes from hmacHash[offset] to hmacHash[offset+3] and store the last 31 bits to truncatedHash. Finally, using a simple modulo operation, we obtain the one time password that's a reasonable length.
This pretty much defines the HOTP algorithm. The RFA4226 doc explains why this is the most secure way to obtain a one time password from these two values.

Understanding TOTP:

TOTP stands for "Time-Based One-Time Password". This was published as RFC6238 by IETF.
A TOTP uses the HOTP algorithm to obtain the one time password. The only difference is that it uses "Time" in the place of "counter," and that gives the solution to our second problem.
That means that instead of initializing the counter and keeping track of it, we can use time as a counter in the HOTP algorithm to obtain the OTP. As a server and phone both have access to time, neither of them has to keep track of the counter.
Also, to avoid the problem of different time zones of the server and phone, we can use a Unix timestamp, which is independent of time zones.
However the Unix time is defined in seconds, so it changes every second. That means the generated password will change every second which is not good. Instead, we need to add a significant interval before changing the password. For

example, the Google Authenticator App changes the code every 30 seconds.
counter = currentUnixTime / 30
A QR code helps in sharing the key with user.

Using a QR code

Though we can ask the users to type the secret key into their phone application directly, we want to make secret keys quite long for security reasons. Asking the user to type in such a long string would not be a user friendly experience. Since the majority of smartphones are equipped with a camera, we can use it and ask the user to scan a QR code to obtain the secret key from it. So all we need to do is to convert the secret key in the QR code and show it to the user.

Implementing TOTP

There are some free phone applications (like Google Authenticator App, Authy, and so on) available which can generate an OTP for the user. Therefore, in most cases, creating your own phone application is not necessary.
The following pseudo codes explain a way to implement TOTP-based 2-factor authentication in a web application.
When user request to enable 2-factor authentication
// Generate a secret key of length 20.secretKey = generateSecretKey(20);
// Save that secret key in database for this particular
user.saveUserSecretKey(userId, secretKey);
// convert that secret key into qr image.qrCode = convertToQrCode(secretKey);
// send the qr image as responseresponse(qrCode);
The user is asked to scan that QR code. When the phone application scans the QR code, it gets the user's secret key. Using that secret key, the current Unix time, and the HOTP algorithm, the phone application will generate and display the password.
We ask the user to type the generated code after scanning the QR code. This is required, because we want to make sure that the user has successfully scanned the image and the phone application has successfully generated the code.
User types the code displayed in the application.
// Fetch secret key from database.secretKey = getSecretKeyOfUser(userId);
if (codeTypedByUser == getHOTP(secretKey, currentUnixTime / 30)) {
enableTwoFactorAuthentication(userId);}
Here we use the HOTP algorithm on the server side to get the OTP-based authentication on the secret key and current unix time. If that OTP is the same

as the one typed by the user, then we can enable 2-factor authentication for that user.
Now after every login operation, we need to check if this particular user has 2-factor authentication enabled. If it is enabled, then we ask for the one time password displayed in the phone application. And if that typed code is correct, only then is the user authenticated.
User types the code displayed in the phone application to login
// Fetch secret key from database.secretKey = getSecretKeyOfUser(userId);
if (codeTypedByUser == getHOTP(secretKey, currentUnixTime)) {
signIn(userId);}

Problems with SMS Authentication

SMS authentication might be more convenient, but is less secure
We know from our day-to-day lives just how easy it is to communicate through SMS. It makes sense, then, that many companies and service providers have implemented SMS OTP as a second form of identity verification.
Unfortunately, SMS OTP is open to several types of attack including:

- SIM swapping and hacking: Your SIM card tells your phone which carrier to connect to, and what phone number to connect with. In a SIM swap attack, a threat actor convinces your carrier to switch your number to a SIM that they own. As a result, they can access all the SMS OTP messages synced to your accounts.
- Account takeover: Many wireless providers let users view text messages within their web portal. If your online account for the web portal is protected only by a weak or common password, an attacker can breach this account and access any SMS OTP messages.
- Lost and synced devices: In theory, losing your phone means you shouldn’t be able to receive SMS OTP messages. However, we can now sync messages between different devices, allowing us to authenticate via SMS OTP and access accounts even without the phone. Forwarding sensitive messages like this isn’t a strong security practice—especially not when your email may have a guessable password.
- Phishing: In a social engineering attack, a threat actor impersonating an employee from a trustworthy service deceives you into handing over your

account credentials, and your SMS OTP. Phishing attacks hinge on hackers exploiting users' emotions or lack of knowledge, and can result in SMS OTPs leaking in the same way as a password.

Keeping Safe

Authenticator apps are a strong alternative

- Mobile authenticators like Okta Verify Authy, and Google Authenticator verify users by sending OTPs and push notifications to the user's app. Authentication apps are more secure than the above methods for a number of reasons:
- Mobile OTPs don't depend on internet access, your location, or the security of your wireless carrier. OTP and push notifications are tied to your device, rather than your number, and they generally work without network service or data.
- Mobile OTP is typically a free feature built into many authenticator apps, meaning it's easy to use in enterprise and individual contexts.
- Push notifications and mobile OTP codes expire quickly, reducing the risk of exploitation as compared to SMS OTP.
- Some authenticator apps support biometrics such as face and fingerprint identification. This offers a stronger layer of protection—even if your phone is stolen, no one else but you can accept push notifications to the device.
- Using WebAuthn which is a browser-based API that uses registered devices (desktop or mobile) as authentication factors. Biometric authenticators built into devices (e.g., Windows Hello, Fingerprint on Android, Touch ID on iOS) all enable WebAuthn, as can portable devices such as Yubikey 5Ci.

Benefits of OTP:

- **Resistance to replay attacks:** OTP authentication provides distinct advantages over using static passwords alone. Unlike traditional passwords, OTPs aren't vulnerable to replay attacks—where a hacker intercepts a transmission of data (like a user submitting their password), records it, and uses it to gain access to the system or account themselves. When a user gains access to their account using an OTP, the code becomes invalid, and therefore can't be repurposed by attackers.
- **Difficult to guess:** OTPs are often generated with algorithms that make use of randomness. This makes it difficult for attackers to successfully guess and use them. OTPs may be valid only for short periods of time, require the user to have knowledge of a previous OTP, or provide the user with a challenge (e.g., "please enter the second and fifth number"). All of these measures further reduce an environment's attack surface when compared to password-only authentication.
- **Reduced risk when passwords are compromised:** Users that don't adopt strong security practices tend to recycle the same credentials across different accounts. If these credentials are leaked or otherwise fall into the wrong hands, stolen data and fraud are significant threats to the user on every front. OTP security helps to prevent access breaches, even if an attacker has obtained a valid set of login credentials.
- **Easy adoption:** One-time passcodes are also easy for organizations to integrate into their authentication strategies. While the cryptic nature of these codes makes them difficult for people to memorize, phones, tokens, and other technologies are widely accessible for security teams to use and distribute to their employees.

CHAPTER 5

RESULTS

The Outputs of the implemented project are as follows:

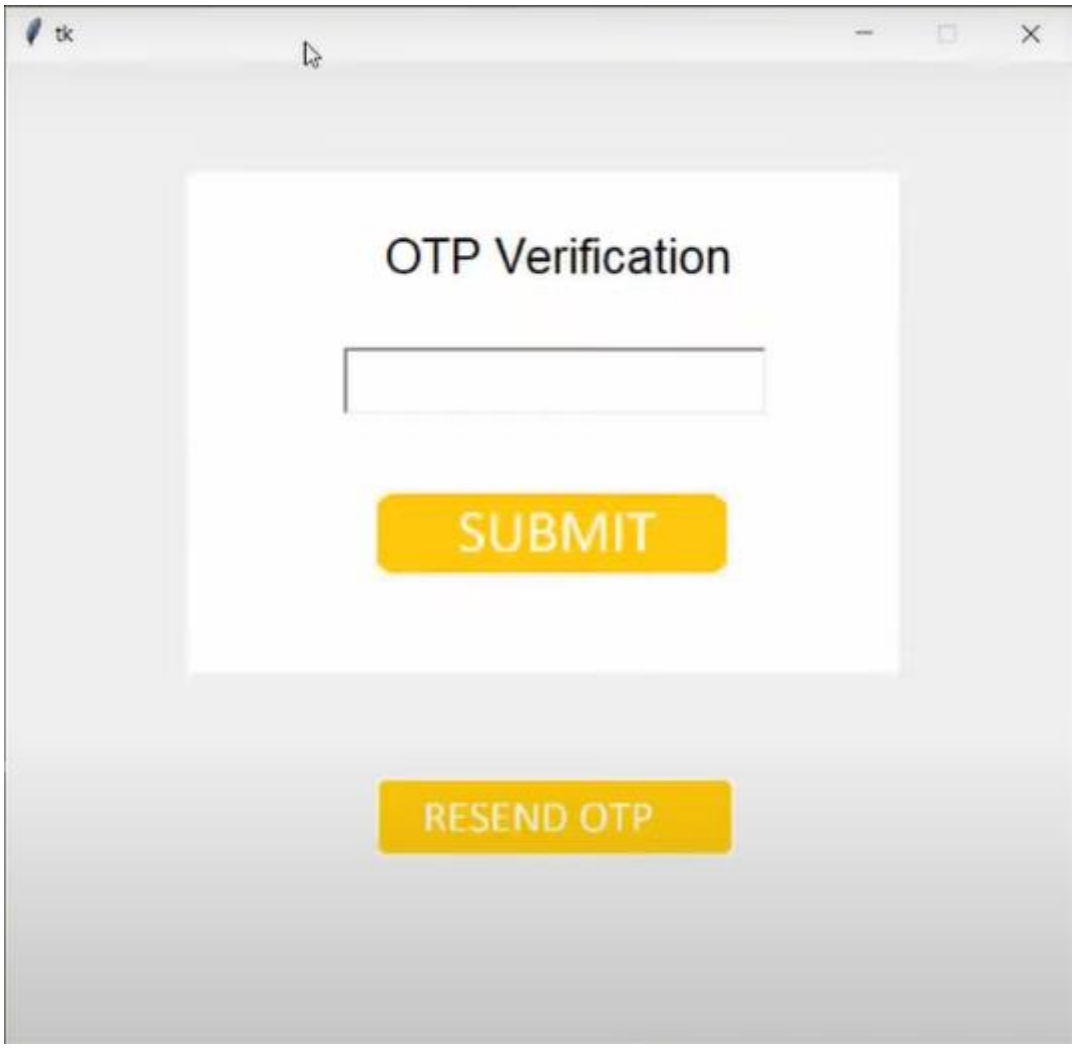

Fig. 8 Enter OTP Home Screen

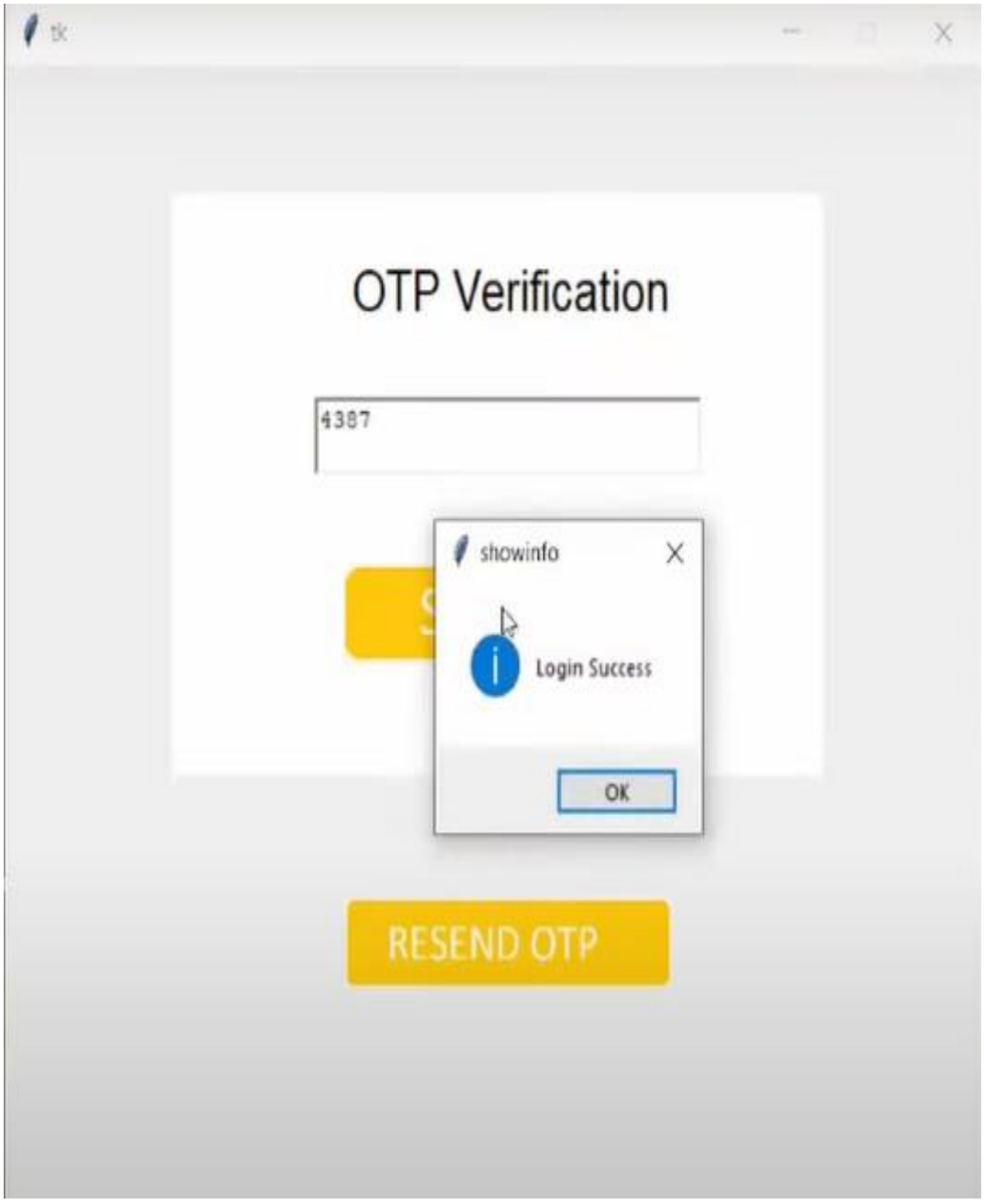

Fig. 9 Success Message After Logging in

CHAPTER 6
FUTURE SCOPE

The system has multiple advantages and has a great extent to be expanded in future,

1.) The system can be trained with data to smartly identify the OTPs and display only OTPs rather than all the texts.
2.) It can be extended to multiple platforms and has unlimited scope for its applications. It can find its uses in multiple human computer interaction devices.
3.) It can be used to automate several other mobile PC interaction instances too.
4.) A Fraud Detection Mechanism may also be applied which will automatically if the OTP isn't from verified sources.
5.) It can be applied to filter emails and extract verification codes when logging into multiple websites which require you to enter a code every time you login.

REFERENCES

1.) Comparative Study on One Time Password Algorithms, Dr. K. Mohan Kumar[1], G. BalaMurugan[2], IJCSMC, Vol. 7, Issue. 8, August 2018, pg.37 – 52

2.) Large-Scale Assessment of Mobile Notifications, Alireza Sahami Shirazi, Niels Henze, Tilman Dingler, Martin Pielot† , Dominik Weber, Albrecht Schmidt

3.) A Secure OTP Algorithm using Smartphone Application, Sonal N. Pannase, Prof. P. R. Pardhi, International Journal of Latest Trends in Engineering and Technology (IJLTET) ISSN: 2278-621X

4.) Integrating Mobile Devices into the Computer Science Curriculum, Qusay H. Mahmoud, Conference Paper in Proceedings - Frontiers in Education Conference · November 2008 DOI: 10.1109/FIE.2008.4720686

5.) Challenges in Human-Computer Interaction Design for Mobile Devices, Kuo-Ying Huang, Proceedings of the World Congress on Engineering and Computer Science 2009 Vol I WCECS 2009, October 20-22, 2009, San Francisco, USA

6.) A review of ONE TIME PASSWORD Mobile Verification, Shally & Gagangeet Singh Aujla, International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR) ISSN(P): 2249-6831; ISSN(E): 2249-7943 Vol. 4, Issue 3, Jun 2014, 113-118

7.) On the Insecurity of SMS One-Time Password Messages against Local Attackers in Modern Mobile Devices, Zeyu Lei , Yuhong Nan , Yanick Fratantonio , and Antonio Bianchi,NDSSSymposium, https://www.ndss-symposium.org/wp-content/uploads/ndss2021_3B-4_24212_paper.pdf

8.) OTP-Based Two-Factor Authentication Using Mobile Phones, Mohamed Hamdy Eldefrawy; Khaled Alghathbar; Muhammad Khurram Khan, 2011 Eighth International Conference on Information Technology: New Generations, DOI: 10.1109/ITNG.2011.64

9.) An Improved OTP Grid Authentication Scheme Email-based using Middle-square for Disaster Management System, B. B. Jr. Balilo , B. D. Gerardo , R. P. Medina1 and Y. Byun, International Journal of Grid and Distributed Computing Vol. 10, No. 11 (2017), pp.43-56 http://dx.doi.org/10.14257/ijgdc.2017.10.11.05 ISSN: 2005-4262

10.) Secure One Time Password OTP Generation for user Authentication in Cloud Environment, K.Aung. International Journal of Trend in Scientific Research and Development 3 (6): 89-92 (October 2019)