Identity and Access Management Solutions (IAM)

Major Project report submitted for the
degree of Bachelor of Technology

In

Computer Science and Engineering

By

Angima Anthwal 181464

UNDER THE SUPERVISION OF

Dr. Ruchi Verma

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

CERTIFICATE

This is to certify that the work which is being presented in the project report titled **Identity and Access Management** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering & Information Technology**, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by Angima Anthwal during the period from January 2022 to May 2022 under the supervision of **Dr. Ruchi Verma**, Department of Computer Science and Engineering & Information Technology, Jaypee University of Information Technology, Waknaghat.

Angima Anthwal (181464)

The above statement made is correct to the best of my knowledge.

Supervisor Name: Dr. Ruchi Verma

Designation: Assistant Professor (Grade-II)

Department Name: Computer Science and Engineering

ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for his divine blessing that made it possible to complete the project work successfully.

I am quite grateful to my supervisor, Dr. Ruchi Verma, Asst. Prof. Senior Grade, Department of CSE Jaypee University of Information Technology, Waknaghat, for her assistance. To complete this assignment, my supervisor has extensive knowledge and a deep interest in the subject of Web Development. Her never-ending patience, intellectual direction, constant encouragement, constant and energetic supervision, constructive criticism, good suggestions, and reading many poor versions and fixing them at all stages made it possible to finish this job.

I'd like to thank Dr. Ruchi Verma, Department of CSE, for her invaluable assistance in completing my project.

I would also like to express my gratitude to everyone who has directly or indirectly assisted me in making this project a success. In this unique scenario, I'd want to appreciate the different staff members, both teaching and non-teaching, who have developed their helpful assistance and facilitated my project. Finally, I must express my gratitude for my parents' unwavering support and patience.

Angima Anthwal(181464)

Table of Contents

ABSTRACT

Identity and access management, or IAM, is the security discipline that makes it possible for the right entities (people or things) to use the right resources (applications or data) when they need to, without interference, using the devices they want to use.IAM is comprised of the systems and processes that allow IT administrators to assign single digital identity to each entity, authenticate them when they login, authorize them to access specified resources, and monitor and manage those identities throughout their lifecycle.

User identities and access rights are defined and managed by an Identity and Access Management (IAM) system.Identity and access management ensures that the right people and job responsibilities (identities) in your business have access to the tools they need to execute their tasks. Identity management and access management systems allow your company to manage staff apps without having to login as an administrator to each one.Your organisation can manage a variety of identities, including people, software, and hardware, such as robotics devices, with identity and access management solutions.

The best approach to implementing an IAM solution is to do an audit of existing and legacy systems. Identify gaps and opportunities, and collaborate with stakeholders early and often. Map out all user types and access scenarios, and define a core set of objectives the IAM solution must meet.

Chapter 01-Introduction 1.1 Introduction:

Single sign-on (SSO) is a session and user authentication service that permits a user to use one set of login credentials -- for example, a name and password -- to access multiple applications. SSO can be used by enterprises, smaller organizations and individuals to ease the management of various usernames and passwords.

In a basic web SSO service, an agent module on the application server retrieves the specific authentication credent Single sign-on (SSO) is a session and user authentication service that permits a user to use one set of login credentials -- for example, a name and password to access multiple applications. SSO can be used by enterprises, smaller organizations and individuals to ease the management of various usernames and passwords. In a basic web SSO service, an agent module on the application server retrieves the specific authentication credentials for an individual user from a dedicated SSO policy server, while authenticating the user against a user repository, such as a Lightweight DirectoryaAccessaProtocola(LDAP)adirectory.aTheaserviceaauthenticatesatheaendauseraforaall the applications the user has been given rights to and eliminates future password prompts for individual applications during the same session for an individual user from a dedicated SSO policy server, while authenticating the user against a user repository, such as a Lightweight Directory Access Protocol (LDAP) directory. The service authenticates the end user for all the applications the user has been given rights to and eliminates future password prompts for individual applications during the same session.

Client - Server Architecture

The client server architecture consists of 2 components - a) Client - The user facing front-end application which can make requests to obtain resources from a backend application. Information can be stored on the client side, using - variables of javascript, cookies or local storage. Cookies and local storage serve different purposes. Cookies are primarily for reading server-side, local storage can only be read by the client-side. Example of cookies : document.cookie = "name=manish;email=mvsn98@gmail.com".

Cookies can have an expiry date, after which their data becomes unavailable on the server side. Such an expiry date does not exist in case of data of local storage.

In case of local storage, there's an upper limit of 5 MB imposed upon storage of data, whereas, in case of cookies, the limit if only 4 MB PER COOKIE!

Example of localstorage : localStorage.setItem("state", "firststate"); const state = localStorage.getItem("state"); localStorage.removeItem("state");

b) Server - The backend application which can accept requests from a client and respond to it by returning the requested resources or an error, if something goes wrong while serving the request. Information can be stored on server side using - variables of the backend programming language, cache or databases.

There are 3 layers in a client -server architecture - Client (Presentation Layer), Server (Application / Session Layer), Database (Database Layer).

If all 3 layers are present on same machine, then it's a one- tier architecture (e.g., if using loopback address - localhost / 127.0.0.1, for testing an application). If these 3 layers are distributed among 2 machines, then it's a two- tier architecture and if all 3 layers are present on their own separate machines, then it's a three-tier architecture.

CORS = Cross Origin Resource Sharing, refers to a set of policies that dictate whether a client can access the resources on another server or not, depending upon if the Origin (Domain Name or IP Address) of the client's request is whitelisted by the server for resource access, or not.

HTTP Requests

HTTP = Hyper Text Transfer Protocol

HTTP is called as a stateless protocol because each request is executed independently, without any knowledge of the requests that were executed before it. Default ports on client side and server side for communication using HTTP requests are - 80 and 80. Default Port from which an HTTP request is sent from a client, is : 80 Default Port at which a server receives the HTTP request from a client, is : 80, unless changed.

Components of HTTP Requests :

A) HTTP METHODS

HTTP Request Methods define the basic purpose for which an HTTP request has been sent by the client to the server.

The HTTP methods are -

1. GET - To be used for OBTAINING a resource from the server.

2. POST - To be used for CREATING a new resource at the server.

3. PATCH - Used for UPDATING an existing resource at the server.

4. PUT - Used for REPLACING an existing resource at the server.

5. DELETE - Used for DELETING an existing resource at the server.

B) REQUEST HEADERS

They carry the meta data about the request and indicate the context of the request to the server.

e.g., Origin header defines the domain name of client that sent the request, Authorization header defines the access token that client is using to request a resource from the server etc. Even the cookies which are stored on the client side, are sent to the server, using the 'Cookie header.

C) DESTINATION ADDRESS

<domain_name_of_server>:<port>

e.g., http://localhost: 3001

D) ROUTE

Conveys the name of the application that is supposed to handle the request on the server- side, or it can also be used to directly access the server-side resource.

e.g., http://localhost:3002/app OR http://localhost:3002/user/2

E) ROUTE PARAMETERS

They form a Dynamic part of request route to convey information to the server that is not in a JSON object notation.

e.g., http://localhost:3001/test/1 OR http://localhost:3002/user/1/2

F) QUERY STRING

Used for conveying information in the form of Keypairs , using which server can take decisions on how to serve the request, they're not a part of the route of the request. They are also called as Search Parameters.

{ name: 'Manish', subject: 'computers' }

e.g., http://localhost:3000/test?name=manish&subject=computers

G) REQUEST BODY

Carries the confidential payload of the request, which is to be delivered to the server, usually as part of a POST request.

e.g., CURL -XPOST http://localhost:3000/test/2?student=manish

-H "Password: "mypassword"

-d { id: 45 }

ENCRYPTION

Encryption is the process of securely encoding data in such a way that only authorized users with a key or password can decrypt the data to reveal the original. There are two basic types of encryption; symmetric key and public key. In symmetric key, the same key is used to encrypt and decrypt data, like a password. In public key encryption, one key is use to encrypt data and a different key is used to decrypt the data.

JSON WEB TOKENS (JWTs)

As we know, HTTP is a stateless protocol. So, an HTTP request does not have to know about the context of execution of any other HTTP requests on a server. Since, an inter- relation of HTTP requests cannot be created, it is possible to attach a string that contains authorization related information of a user, in the Authorization header of each request made by the client, so that server always knows whether the request is being made by an authorized user or not.

A JWT is a 3 part string, that has the following structure :

[header].[payload].[signature]

Header defines 2 fields - "typ": "jwt" and "alg" which defines an encryption algorithm.

Payload field carries the list of all the information to be exchanged between client and server.

The header and payload fields of the string are ENCODED using BASE64 algorithm, and can be easily decoded using the same, without the need of a decryption key.

The signature field is ENCRYPTED using a secret key and the encryption algorithm defined by "alg" field of Header of the JWT. Anyone who possesses this secret key will be able to verify the issuer of this JWT. The encryption algorithm can be made more secure by using public- private key encryption strategy rather than symmetric key encryption strategy.

Account Management using JWTs and Hashing

At the time of creation of user, the password of the user is always stored in database using a one way hashing function, such as one provided by Bcrypt library.

When a user logs into the server using a client side, the server issues a JWT signed using an encryption key, to the client. The JWT is issued with an expiry date, so that user can only stay logged in for a limited period of time. This JWT consists of the ID of the user who made this request, and nothing confidential such as 'password' is put inside payload field of JWT, because the payload field is Never encrypted in case of a Signed JWT, and thus can be decoded easily. This JWT is also pushed inside the array of JWTs for the existing user.

If the user needs to access some privately available information, the user must first send this JWT to backend server. The server verifies the issuer of this JWT, decodes and retrieves the ID of the user, and checks the database to find the user who has the same ID and whose array of JWTs also contains the JWT received by the server from the client. If such a user is found then the request is served, otherwise, server returns an error.

If the user needs to logout, they must send a request along with their JWT to the server, and once server verifies the JWT, it will remove this JWT from the array of JWTs of this user and redirect the client to the logout page.

JWTs are of 2 kinds - JWS and JWE.

JSON Web Signing (JWS)

If the payload that is being carried by a JWT is not in an encrypted form, then it's a JWS. We can only verify the signature of the JWS using its Signature Encryption Key to check whether this JWT is valid and whether this has been issued by a specific server or not.

JSON Web Encryption (JWE)

If the payload being carried by the JWT is in an Encrypted from, then it's a JWE. Such a JWT is very secure, because even if the JWT gets stolen by a hacker, they can never decode the payload field of the JWT, since it's encrypted, using a different key.

1.2 Objective:

This project seeks to provide a solution that lets users authenticate themselves once and access different applications without reauthentication. SSO assists users through all the procedures required to access heterogeneous applications. Using applications becomes easier, technical- assistance costs go down, and security improves. However, getting the most of the solution requires understanding related domains such as central user administration, the enterprise directory, and Web single sign- on. SSO is a moving target in a changing context. Many new devices, applications, and authentication methods are on the horizon, and although the general issue remains relatively simple and clearly defined, the solution's integration in an IT environment can become complicated.

The main features of our software /product are as mentioned below:

- SSO (Single Sign-On) Facility
- Using Social Media Accounts to Create Your Own Identity
- Form for User Registration
- Policy on Password Expiration
- Role based Access Management for Authorization
- One Time Passwords with Multi Factor Authentication

Identity and access management ensures that the right people and job responsibilities (identities) in your business have access to the tools they need to execute their task Identity management and access management systems allow your company to manag e staff apps without having to login as an administrator to each one.Your organisatio n can manage a variety of identities,including people, software, and hardware, such a s robotics devices, with identity and access management solutions.

1.3 Problem Statement:

It is usually not practical by asking one user to maintain different pairs of id entity and passwords for different serviceproviders, since this could increase the workload of both users and service providers as well as the communication overhead of networks. So, for this it requires a single sign-on authentication mechanism that is a single login for multiple service providers, which would not increase the workload. There are various attacks and parameters that need to be considered while providing security to authentication system. In this paper, we provide a comprehensive review of existing work done on Single sign-on. Then for security of single sign-on, what parameters and attacks should be covered.

Next, Implementation of Single Sign-on for distributed computing using user-id and password along with biometric verification. Then security analysis is done. Next, the comparison is made and lastly we conclude specifying the future work.

Account management of the users poses another challenge for administrators. Single Sign-On (SSO) can be the solution by providing a service of centralized authentication and user account management. This study applies a token- based SSO architecture and uses Json Web Token (JWT) to grant permission authorities, since JWT can provide a claim process between parties. Additionally, the built-in dashboard lists associated information systems to facilitate accessing for the authenticated users.

1.4 Methodology:

To start with, we present user-id and password, secured user authentication scheme. In this project, we are using One-way hash function and AES Encryption/ Decryption algorithm. This work is divided into 3 parts: 1-Client side, 2- Authentication party, 3- Server side.

Fig 1.

Every SSO protocol consists of three phases :
Phase 1: registration and trust establishment between Service Provider (SP) and Identity Provider (IdP)
Phase 2: End-User authentication on the IdP
Phase 3: End-User authentication on the SP via the authentication token

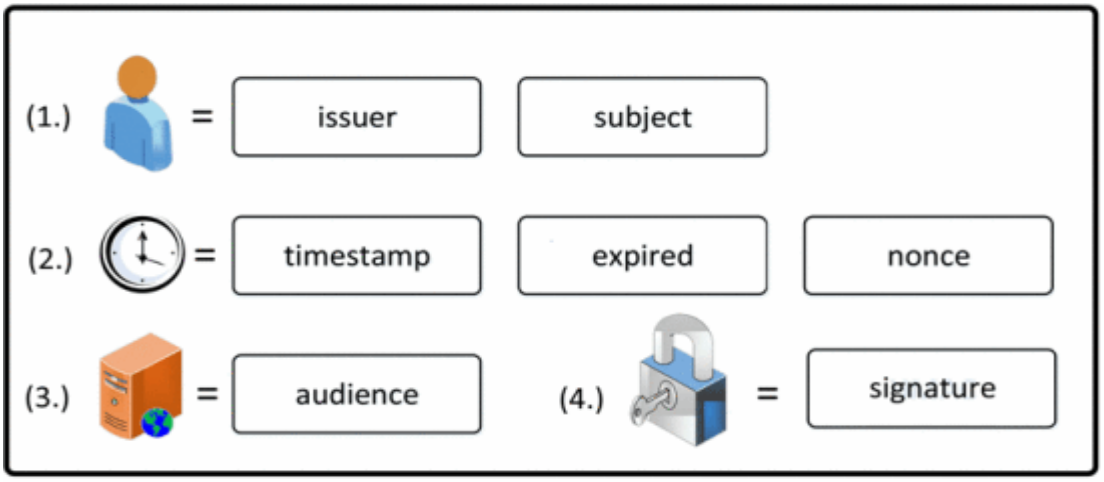


Fig 2.

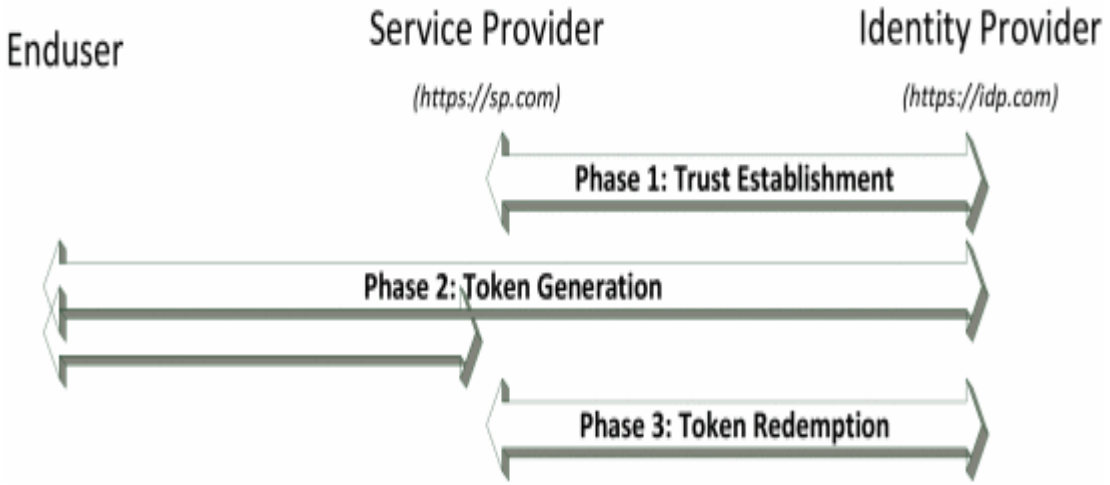


Fig 3.

In the first phase the trust establishment phase between SP and IdP is provided. In classical SSO systems, trust is established by an administrator manually registering a specific IdP on the SP. A typical example is SAML. The administrator visits the IdP and downloads the IdP's metadata, for instance its certificate. Next, he uploads it on the SP and configures further parameters like important URLs of the IdP. We call this full rust establishment since only those authorized people (the administrator) can invoke this manual trust establishment.

Token Generation

In the second phase, the SP typically forwards the End-User to the IdP. This is usually an HTTP redirect to apre-registered URL on the IdP with additional parameters (e.g., the identity of the SP). The End-User then logs in at the IdP, which then generates an SSO token. This token is then submitted to the SP.

Token Redemption

In the final phase, the SP receives the SSO token in order to authenticate the End- User. This is a security critical process since the token contains multiple parameters which must be verified.

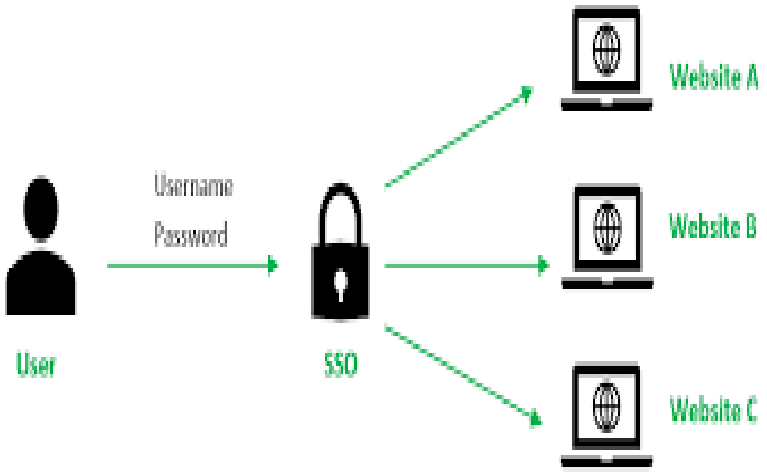


Fig 4.

1.5 SYSTEM OVERVIEW:

Single Sign-On (SSO) is a concept to delegate the authentication of an End- User on a Service Provider (SP) to a third party - the so-called Identity Provider (IdP). Standardized in 2014, OpenID Connect is the latest SSO protocol and is supported by large companies like Amazon , Google , Microsoft and PayPal . In 2015 Google announced that developers should abandon the preceding protocol OpenID 2.0 (OpenID) and recommended switching to its OAuth 2.0 (OAuth) based successor OpenID Connect. The OpenID Connect specification itself offers a list of available libraries supporting OpenID Connect and an additional list of certified libraries. On the one hand, using such a library makes the integration of OpenID Connect into a web application quite easy since the entire authentication (including all security-related operations) can be delegated to it. On the other hand, the security of the web application then depends on the library being used.

To start with, we present
user-id and password, secured user authentication scheme. In this project, we are using One-way hash function and AES Encryption/ Decryption algorithm. This work is divided into 3 parts: 1-Client side, 2- Authentication party, 3- Server side.

Every SSO protocol consists of three phases :

Phase 1: registration and trust establishment between Service Provider (SP) and Identity Provider (IdP)

Phase 2: End-User authentication on the IdP

Phase 3: End-User authentication on the SP via the authentication token

FLOW CHART:

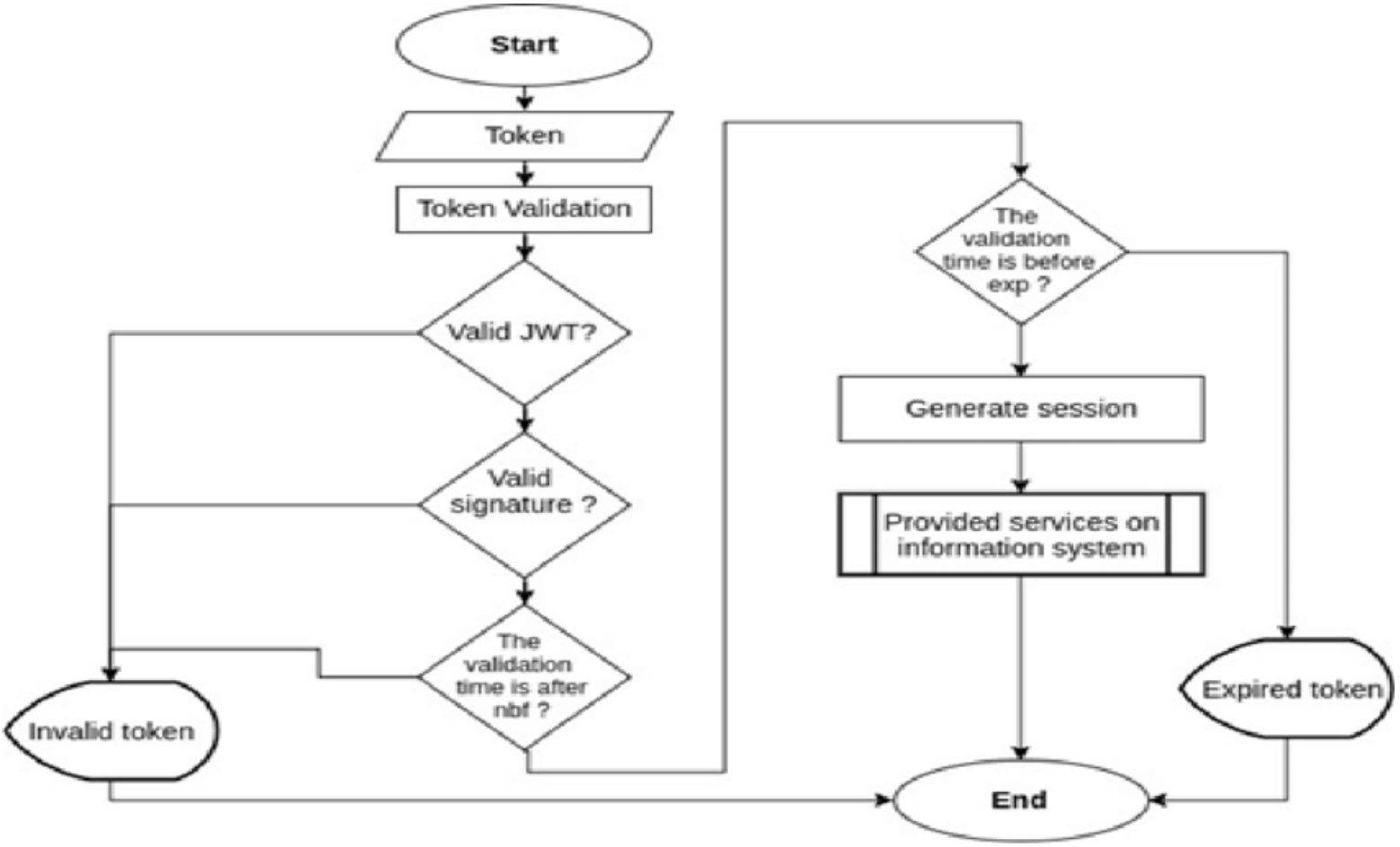


Fig 5.

1.5 Organization:

The project report is broken down into 5 sections. The first chapter covers the background and motivation for the proposed application, the problem statement and aims to answer the issue statement, the recommended technique or research, and the highlighting of successful proposed applications. Chapter 2 illustrates the literature survey of the project from which we took the references. The system development chapter includes the site map, use case diagram, activity diagram, and system wireframe, which is the proposed application's user interface. Software design approach, tools, requirements, system performance specifications, and timescales are discussed in Chapter 4. The fifth chapter concludes the implementation, project evaluation, benefits and future scope of the project.

Chapter 02- Literature Survey

Authors	Year	Description	Outcomes
Alessandro Armando, Carbone, L uc Com-pa gna, Jorge Cuellar	2008	The real advantage of formal analysis relies on finding generic issues in specifications like in SAML, BrowserId and OAuth .	Introduced AuthScan, a penetration testing tool that automatically extracts the authentication protocol based on HTTP traces and JavaScript code. Shows a more convenient approach combining program code analysis with formal analysis
Dominic Scheirli ck and Scott Geary	2016	The application of an attacker IdP can be additionally enforced in some scenarios by the HTTP vulnerability or by settingpecifc HTTP GET parameters in a request	The real advantage of formal analysis relies on finding generic issues in specifications like in SAML , BrowserId and OAuth.

Chapter 3
System Development

Language Used:

React JS
Node JS
HTML
CSS

Platform Used:

Visual Studio Code

Tools Used:

- **MongoDB:**MongoDB is an open-source, cross-platform, and distributed document-based database designed for ease of application development and scaling. It is a NoSQL database developed by MongoDB Inc.

  MongoDB name is derived from the word "Humongous" which means huge, enormous. MongoDB database is built to store a huge amount of data and also perform fast.

  MongoDB is not a Relational Database Management System (RDBMS). It's called a "NoSQL" database. It is opposite to SQL based databases where it does not normalize data under schemas and tables where every table has a fixed structure. Instead, it stores data in the collections as JSON based documents and does not enforce schemas. It does not have tables, rows, and columns as other SQL (RDBMS) databases.

- **PostMan:**Postman began as a REST client and has evolved into today's comprehensive Postman API Platform.

- **MongoDB Atlas:**MongoDB Atlas is a fully-managed cloud database that handles all the complexity of deploying, managing, and healing your deployments on the cloud service provider of your choice (AWS , Azure, and GCP). MongoDB Atlas is the best way to deploy, run, and scale MongoDB in the cloud.
- **Mongoose:**Mongoose is a Node.js-based ObjectaData Modeling (ODM) library for MongoDB . It is akin to n Object Relational Mapper (ORM) such as SQLAlchemy for traditional SQL databases. The problem that Mongoose aims to solve is allowing developers to enforcea specific schema at the application layer. In addition to enforcing a schema, Mongoose also offers a variety of hooks, model validation, and other features aimed at making it easier to work withaMongoDB.

What are Rest APIs?

Application programming interfaces (APIs) are everywhere. They enable software to communicate with other pieces of software—internal or external — which is a key ingredient in scalability, not to mention reusability.

It's quite common nowadays for online services to have public -facing APIs . These enable other developers to easily integrate features like social media logins, credit card payments, and behavior tracking. The *de facto* standard they use for this is called REpresentational State Transfer (REST).

While amultitude of platforms and programming languages can be used for the task—

- **Node.js**

  As a asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications. Upon each connection, the callback is fired, but if there is no work to be done, Node.js will sleep.

  Express, which vastly simplifies building out common web server tasks under Node.js and is standard fare in building a REST API back end

  Express JS is a web application framework that provides you with a simple API to build websites , web apps and back ends. With Express JS, you need not worry about low levelprotocols, processes , etc.

- Mongoose, which will connect our back end to a MongoDB database

Mongoose is a Node.js-based ObjectaData Modeling (ODM) library for MongoDB . It is akin to n Object Relational Mapper (ORM) such as SQLAlchemy for traditional SQL databases. The problem that Mongoose aims to solve is allowing developers to enforcea specific schema at the application layer. In addition to enforcing a schema, Mongoose also offers a variety of hooks, model validation, and other features aimed at making it easier to work with MongoDB.

Following are the APIs I have created in my Login-System

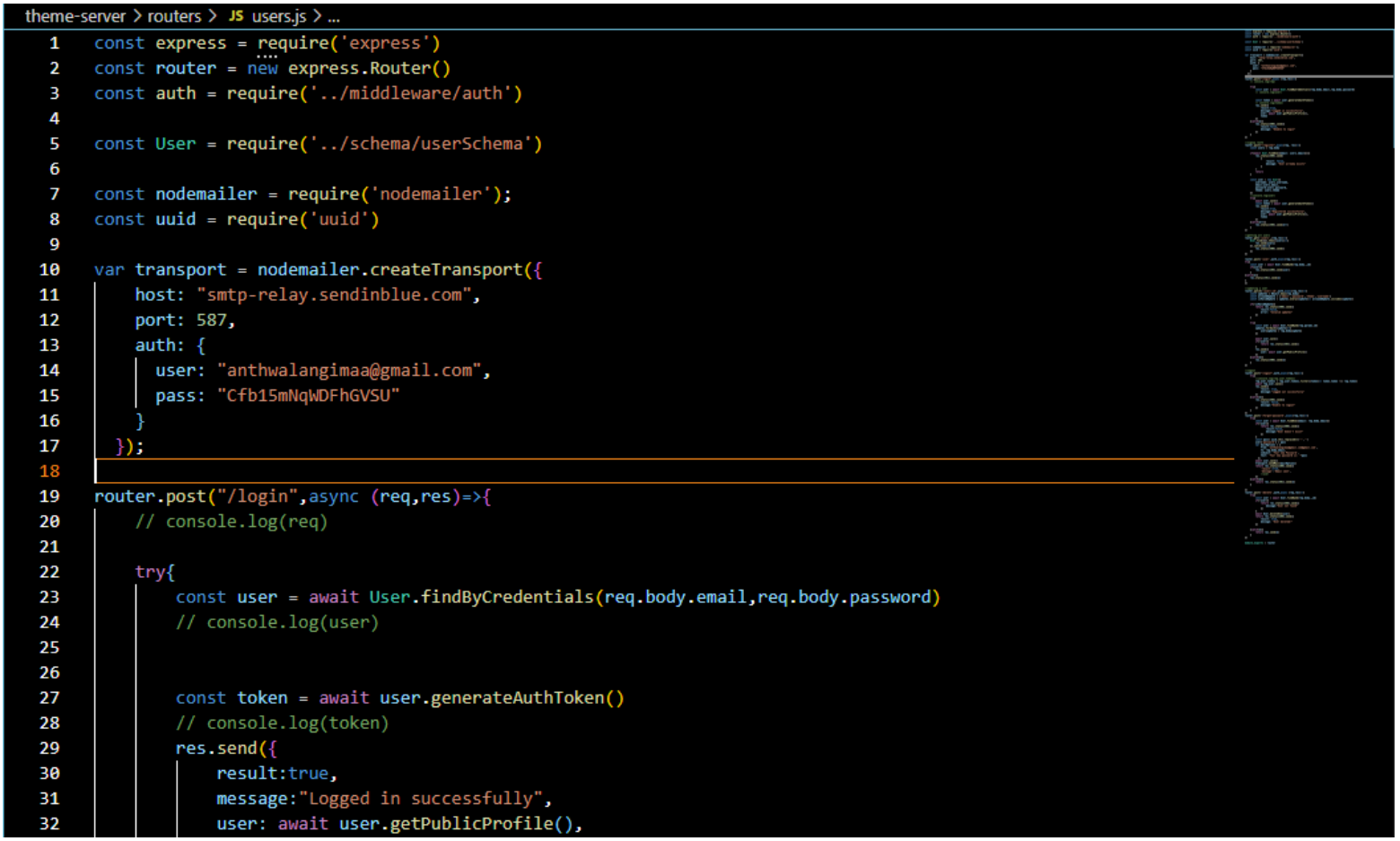


Fig 6.

In the above figure, I have implemented a login API.

The user is logged in successfully, only if he/she is registered prior to it on the backend.

The other APIs have been created in a similar way.

```
    const user = new User({
        username: users.username,
        email:users.email,
        password:users.password,
        theme: users.theme
    })
    //console.log(user)
    try{
        await user.save()
        const token = await user.generateAuthToken()
        res.send({
            result:true,
            message:"Registered successfully",
            user: await user.getPublicProfile(),
            token
        })
    }catch(err){
        res.status(400).send(err)
    }
})

//getting all users
router.get('/users',(req,res)=>{
    User.find({}).then((users)=>{
        res.send(users)
    }).catch((e)=>{
        res.status(500).send()
    })
```

```
    try{
        const user = await User.findByCredentials(req.body.email,req.body.password)
        // console.log(user)


        const token = await user.generateAuthToken()
        // console.log(token)
        res.send({
            result:true,
            message:"Logged in successfully",
            user: await user.getPublicProfile(),
            token
        })
    }catch(e){
        res.status(400).send({
            result:false,
            message: "Unable to login"
        })
    }
})

//signup route
router.post("/register",async(req, res)=>{
    const users = req.body

    if(await User.findOne({email: users.email})){
        res.status(400).send(
            {
                result: false,
                message: "User already exists"
            }
```

Fig 8,9

```
router.post('/delete',auth,async (req,res)=>{
    try{
        const user = await User.findById(req.body._id)
        if(!user){
            return res.status(400).send({
                message:"User not found"
            })
        }
        await User.deleteOne(user)
        return res.status(200).send({
            result: true,
            message: "User deleted!"
        })

    }catch(e){
        return res.send(e)
    }
})

module.exports = router
```

```
        const pass= uuid.v4().replaceAll('-','')
        user['password'] = pass
        var mailOptions = {
            from: 'anthwalangimaa@gmail.com@gmail.com',
            to: req.body.email,
            subject: 'Your new Password',
            text: 'Your new password is: '+pass
          };
        await user.save()
        transport.sendMail(mailOptions)
        return res.status(200).send({
            'result':true,
            'message':'Email sent',
            //user
        })
    }catch(e){
        return res.status(401).send(e)
    }

})
router.post('/delete',auth,async (req,res)=>{
    try{
        const user = await User.findById(req.body._id)
        if(!user){
            return res.status(400).send({
                message:"User not found"
            })
        }
```

Fig 9,10

What is React Router?

React Router is a standard library for routing in React. It enables the navigation among views of various components in a React Application, allows changing the browser URL, and keeps the UI in sync with the URL.

Let us create a simple application to React to understand how the React Router works.

In My Login-System the App.js file contains all the routes to the other files. Following is the list of routes:

- Login
- Edit
- Home
- Register
- Forgot Password

```
import Logout from "./Logout";
import ForgotPassword from "./Forgot-password";
function Theme1(){
    return(
        <>
            <BrowserRouter>
            <div>
            <Routes>
                <Route exact path="/" element={<Login/>}></Route>
                <Route path="/edit" element={<Edit/>}></Route>
                <Route path="/home" element={<Home/>}></Route>
                <Route path="/logout" element={<Logout/>}></Route>
                <Route path="/register" element={<Register/>}></Route>
                <Route path="/forgot-password" element={<ForgotPassword/>}></Route>
              </Routes>

            </div>
            </BrowserRouter>
        </>
    )
    }

export default Theme1;
```

What are React Hooks?

Hooks are the new feature introduced in the React 16.8 version. It allows you to use state and other React features without writing a class. Hooks are the functions which "hook into" React state and lifecycle features from function components. It does not work inside classes.

What is Refactoring?

Refactoring or **Code Refactoring** is defined as systematic process of improving existing computer code, without adding new functionality or changing external behaviour of the code. It is intended to change the implementation, definition, structure of code without changing functionality of software. It improves extensibility, maintainability, and readability of software without changing what it actually does.

Why should we refactor our code when it works fine?

The goal of refactoring is not to add new functionality or remove an existing one. The main goal of refactoring is to make code easier to maintain in future and to fight technical debt. We do refactor because we understand that getting design right in first time is hard and also you get the following benefits from refactoring:

- Code size is often reduced
- Confusing code is restructured into simpler code

Both of the above benefits greatly improve maintainability which is required because requirements always keep changing.

When do we refactor?

- Before you add new features, make sure your design and current code is “good” this will help the new code be easier to write.
- When you need to fix a bug
- When you do a peer review
- During a code review

How to identify code to refactor?

Martin Fowler proposed using “code smells” to identify when and where to refactor. Code smells are bad things done in code, just like bad patterns in the code. Refactoring and Code smells are a few techniques that help us identify problems in design and implementation. It also helps us in applying known solutions to these problems.

React Redux:

Redux is an open-source JavaScript library for managing and centralizing application state. It is most commonly used with libraries such as React or Angular for building user interfaces.

React Redux allows us to manage the state variables globally which allows us to use a common state for the multiple components of the application. Making our application faster and also reducing the cost. The management of central state is done with the help of reducers.

Below is the snippet of the reducers used below:

```
const initialState = {
    token : "",
    email : "",
    id :"",
    theme : "",
    username:""
}

export const reducer = (state=initialState,action)=>{
    if(action.type=="USERNAME"){
        return{
            ...state,
            username:action.payload
        }
    }
    if(action.type=="TOKEN"){
        return{
            ...state,
            token:action.payload
        }
    }
    if(action.type=="EMAIL"){
        return{
            ...state,
            email:action.payload
```

```
    }
    if(action.type=="TOKEN"){
        return{
            ...state,
            token:action.payload
        }
    }
    if(action.type=="EMAIL"){
        return{
            ...state,
            email:action.payload
        }
    }
    if(action.type=="ID"){
        return{
            ...state,
            id:action.payload
        }
    }
    if(action.type=="THEME"){
        return{
            ...state,
            theme:action.payload
        }
    }
```

React Components

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML. Components come in two types, Class components and Function components, in this tutorial we will concentrate on Function components.

Class Component

A class component must include the extends React.Component statement. This statement creates an inheritance to React.Component, and gives your component access to React.Component's functions.

The component also requires a render() method, this method returns HTML.

Function Component

A Function component also returns HTML, and behaves much the same way as a Class component, but Function components can be written using much less code, are easier to understand, and will be preferred in this tutorial. Below is the code of various components I created:

```
<Navbar theme={theme}/>
<form>
    <div className='container-edit'>
        <h1>Edit your details</h1>
        <hr/>
        <label for='username'><strong>Username</strong></label>
        <input type="text" name="username" placeholder="Username" id="user" value={user.username} onChange={(e)=>handleChange
        <label for='email'><strong>Email</strong></label>
        <input type="text" placeholder="Enter email" name="email" id="email" value={user.email} onChange={(e)=>handleChange(e

        <label for='psw'><strong>Password</strong></label>
        <input type={hidden?"password":"text"} placeholder="Enter Password" name="password" id="psw" value={user.password} on
        <i style={{marginLeft: '-30px', cursor: 'pointer', color: 'black'}}
            onClick={()=> showPassword()}
            className = {hidden?"bi bi-eye-slash":"bi bi-eye"}
        ></i>

        <label for='psw-repeat'><strong>Confirm Password</strong></label>
        <input type={hiddenConfirm?"password":"text"} placeholder="Confirm password" name="confirmPassword" value={user.confi
        <i style={{marginLeft: '-30px', cursor: 'pointer', color: 'black'}}
            onClick={()=> showConfirmPassword()}
            className = {hidden?"bi bi-eye-slash":"bi bi-eye"}
        ></i>

        <button type='submit' className={theme==='3'?"updatebtn3":(theme==='2'?"updatebtn2":"updatebtn")} onClick={updateUser
        <button type='submit' className={theme==='3'?"updatebtn3":(theme==='2'?"updatebtn2":"updatebtn")} onClick={deleteUser
    </div>
```

```
    return (
        <>
            <div style={{justifyContent:"center",alignItems:'center', maxWidth:'60%', margin:'auto',marginTop:'8%'}}>
                <h4>Forgot Password</h4>
                <form onSubmit={forgotPassword}>
                    <input type="email" name="email" placeholder="Email" value={user.email} onChange={(e)=>handleChange(e)} re
                    <input type="submit" value="Send mail"  className={theme==='3'?"login3":(theme==='2'?"login2":"login1")}/>
                </form>
            </div>
        </>
    )
}

export default ForgotPassword
```

```
return(
    <>
        <Navbar theme={finalTheme}/>
        <div className="theme">
            <h1>Choose your theme</h1>
            <h5>{finalTheme==='1'?"Theme 1":(finalTheme==='2'?"Theme 2":"Theme 3")}</h5>
            <hr/>
            <form onSubmit={updateUser}>
                <div className="row-home">
                    <div className="column">
                        <input type='radio' value='1' name='theme' checked = {theme==='1'} onChange={()=>setTheme('1')}/>
                        <img src={login1} style={{width:"100%"}}/>
                        <h4>Theme 1</h4>
                    </div>
                    <div className="column">
                        <input type='radio' value='2' name='theme' checked = {theme==='2'} onChange={()=>setTheme('2')}/>
                        <img src={login2} style={{width:"100%"}}/>
                        <h4>Theme 2</h4>
                    </div>
                    <div className="column">
                        <input type='radio' value='3' name='theme' checked = {theme==='3'} onChange={()=>setTheme('3')}/>
                        <img src={login3} style={{width:"100%"}}/>
                        <h4>Theme 3</h4>
                    </div>
                </div>
                <button type="submit" className={finalTheme==='3'?"btn3":(finalTheme==='2'?"btn2":"btn")}>Update Theme</bu
            </form>
        </div>
```

Solutions proposed by our team:

According to Norton Security, more than 800000 online accounts get hacked every year. Even the most secure systems present are prone to the brute force attacks by the hackers. Due to this the username and password are vulnerable and can be stolen by the third parties. Also in today's market there is lack of proper access control, which increases the risk of unauthorized access to the physical and the logical systems.

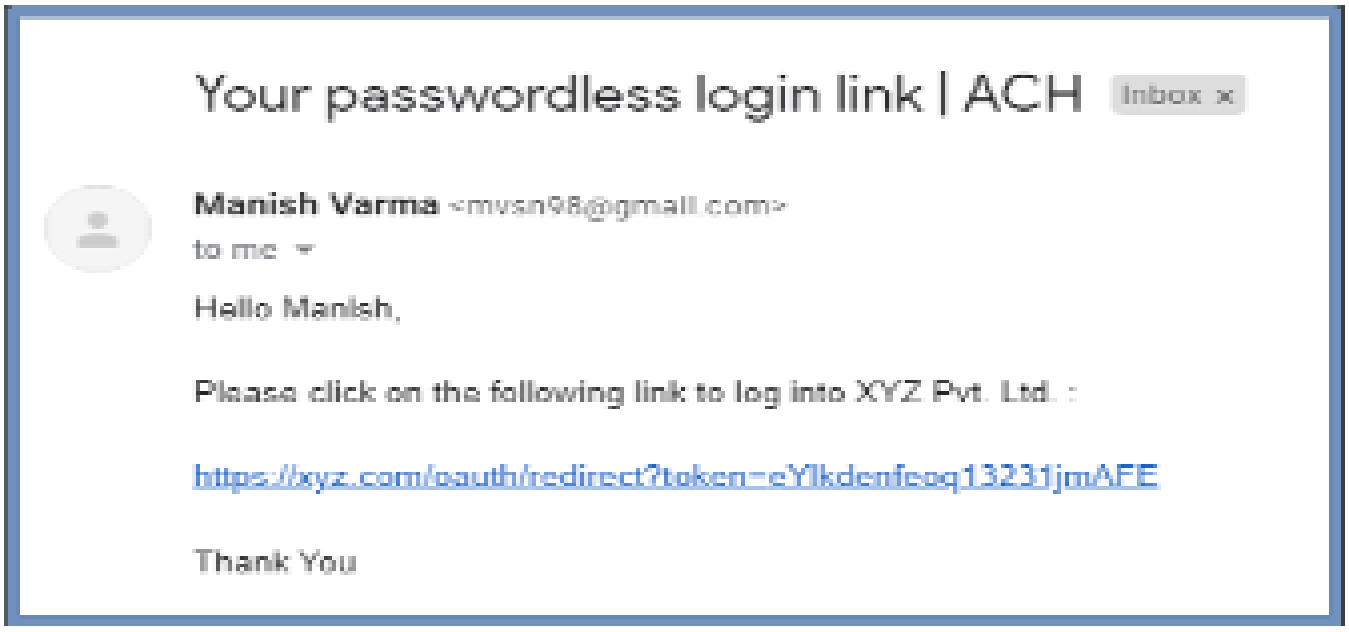
One way organizations can recruit and retain the best talent is to remove the constraints of geographic location and offer a flexible work environment. A remote workforce allows businesses to boost productivity while keeping expenses in check—as well as untethering employees from a traditional office setting. However, with employees scattered all over a country or even the world, enterprise IT teams face a much more daunting challenge: maintaining a consistent experience for employees connecting to corporate resources without sacrificing security. The growth of mobile computing means that IT teams have less visibility into and control over employees' work practices.

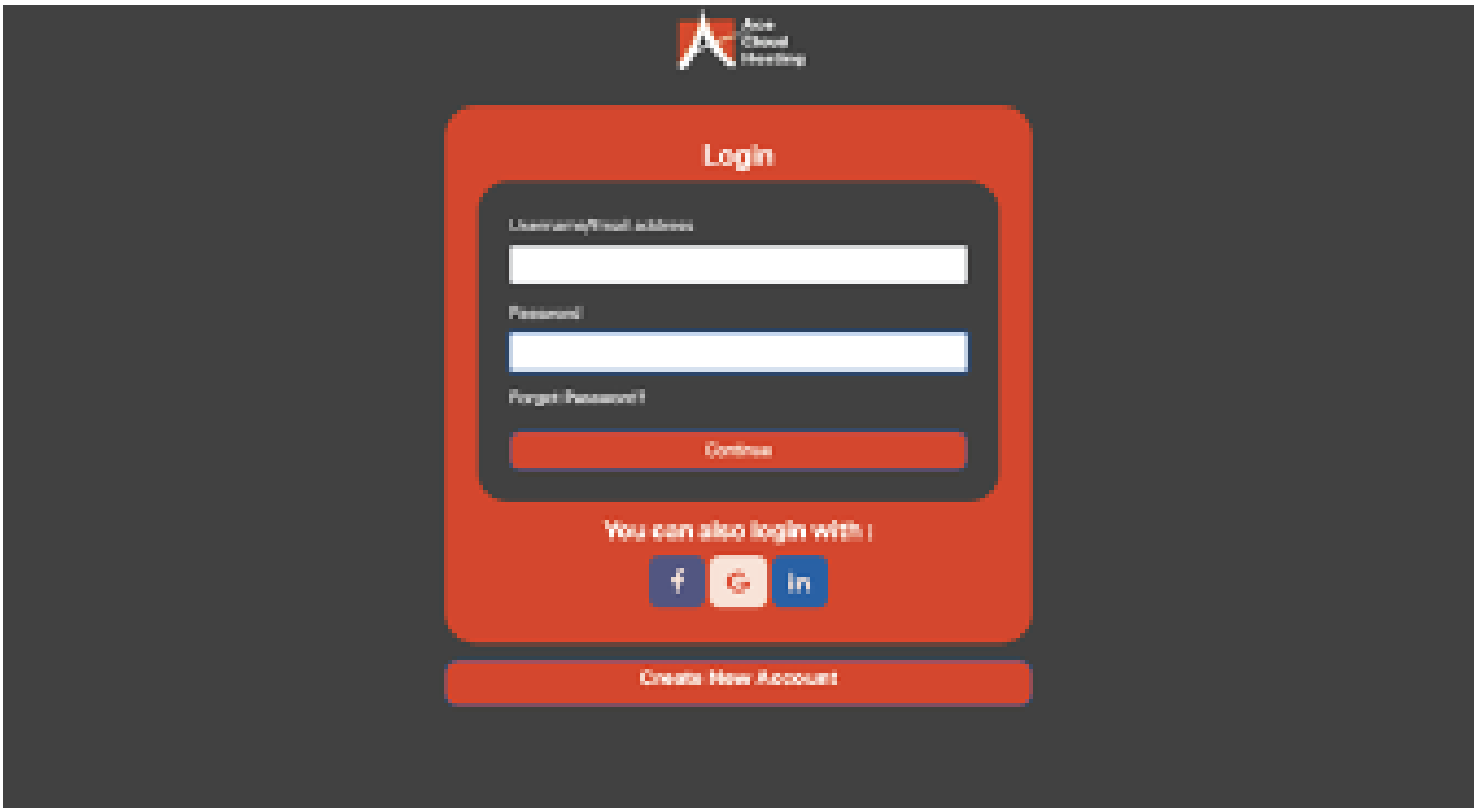
The goal of our organization is to develop a secure, fast, reliable and user friendly system for identity and access management. We are going to enhance our systems User Experience by providing facility for single sign on and we would enhance the security using multifactor authentication. We are also developing customizable user interface that will allow different organizations to update the user interface based on their requirements.

After the above task our motive is to provide the software developed to different organizations as software as a service (SaaS).

We proposed the following solution to overcome the above problems:

- · Secure Authentication with OAuth2 Security framework.
- · Single Sign On (SSO) facility.
- · Authorization with Role based access management.
- · Multi factor authentication.
- · Authentication for mobile apps.
- · Completely Customizable Authentication User Interfaces.
- · Password-less Authentication through Account Access Link.
- · Username and password based standard login system.
- · Authorization with Role based Access Management
- · Password Expiry Policy
- · QR Code based Login
- · Captcha for Bot Detection





Context Switching In React JS

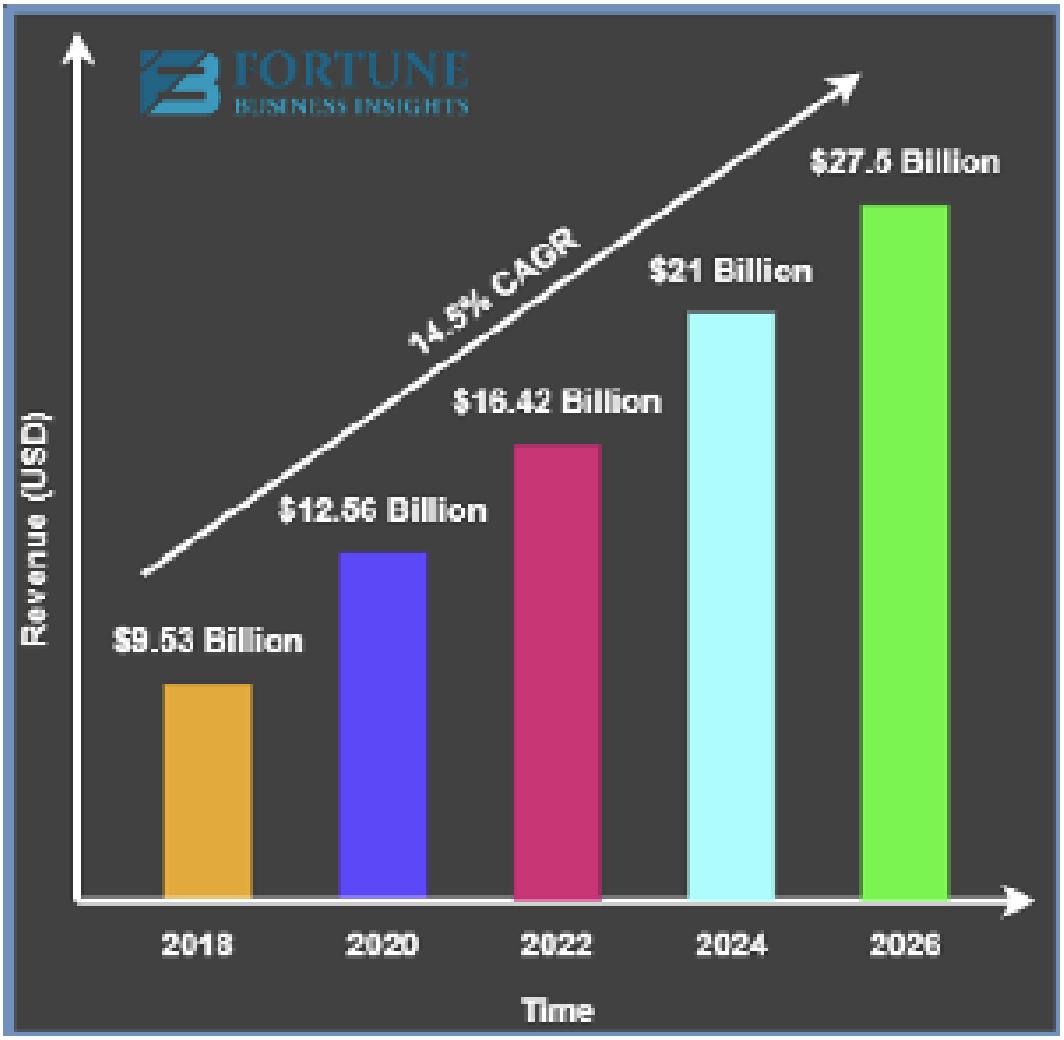
Context provides a way to pass data through the component tree without having to pass props down manually at every level.

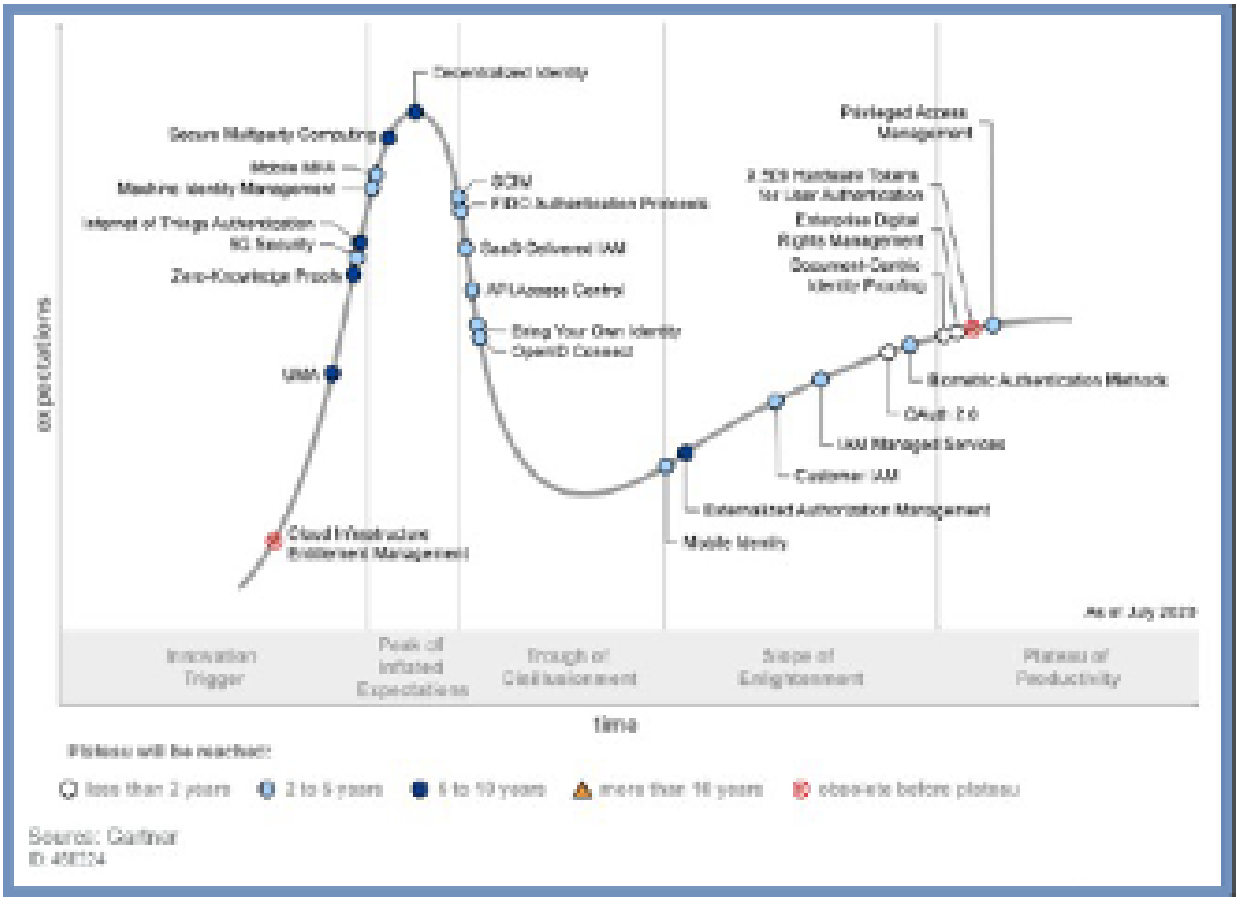
- In a typical React application, data is passed top-down (parent to child) via props, but such usage can be cumbersome for certain types of props (e.g. locale preference, UI theme) that are required by many components within an application. Context provides a way to share values like these between components without having to explicitly pass a prop through every level of the tree.

  When to Use Context
- Before You Use Context
- API
    - React.createContext
    - Context.Provider
    - Class.contextType
    - Context.Consumer

MARKET ANALYSIS

IAM is a growing industry in India, and is supposed to be a $ 27.5 Billion dollar marker by 2026. Currently no big Indian company is in this area and is being dominated by the US players.


FORTUNE
BUSINESS INSIGHTS
Revenue (USD)
$9.53 Billion
$12.56 Billion
$16.42 Billion
$21 Billion
$27.5 Billion
14.5% CAGR
2018
2020
2022
2024
2026
Time


Decentralized Identity
Secure Multiparty Computing
Machine Identity Management
Internet of Things Authentication
5G Security
Zero-Knowledge Proofs
UMA
Cloud Infrastructure Entitlement Management
SCIM
FIDO Authentication Protocols
SaaS-Delivered IAM
API Access Control
Bring Your Own Identity
OpenID Connect
Privileged Access Management
Enterprise Digital Rights Management
Document-Centric Identity Proofing
Biometric Authentication Methods
IAM Managed Services
Customer IAM
Mobile Identity
expectations
Innovation Trigger
Peak of Inflated Expectations
Trough of Disillusionment
Slope of Enlightenment
Plateau of Productivity
time
less than 2 years
2 to 5 years
5 to 10 years
more than 10 years
obsolete before plateau
Source: Gartner

Business Model:

Feature / Plan	Free	Basic	Business	Enterprise
Role based User Access Control	No	Yes	Yes	Yes
Multi-Factor Authentication using OTPs	No	Yes	Yes	Yes
QR code based Login	No	Yes	Yes	Yes
Fully Customizable Authentication GUIs	No	No	Yes	Yes
Password Expiry Policy	No	No	Yes	Yes
Captcha for Bot Detection	No	No	Yes	Yes
User Profile Management	No	No	No	Yes
Price (USD per month)	FREE	$10	$35	Contact Sales Team

Enterprise Plan Pricing :

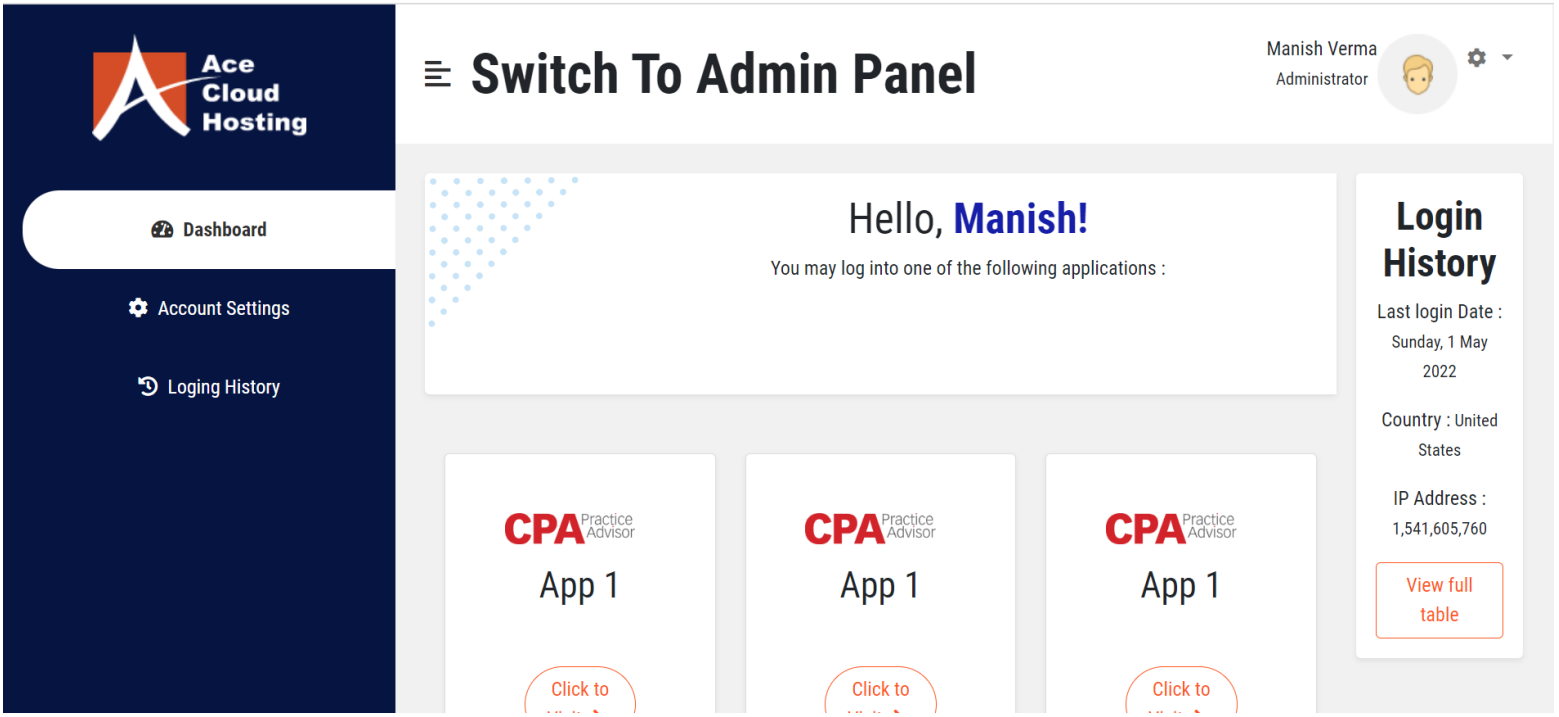
- ✔ **$45 for additional features of the plan.**
- ✔ **$10 per additional web app or mobile app.**
- ✔ **¢5 per additional monthly active user.**

Feature / Plan	Free	Basic	Business	Enterprise
Monthly Active Users	1	500	500	Contact Sales Team
Number of Web Apps or Mobile Apps	1	3	3	Contact Sales Team
OAuth2 Security Protection	Yes	Yes	Yes	Yes
Standard Username & Password based Authentication	Yes	Yes	Yes	Yes
Single Sign On	Yes	Yes	Yes	Yes
User Registration Forms	Yes	Yes	Yes	Yes
Multiple Authentication User Interfaces to choose from	No	Yes	Yes	Yes
Bring Your Own Identity	No	Yes	Yes	Yes
Password-less Authentication	No	Yes	Yes	Yes

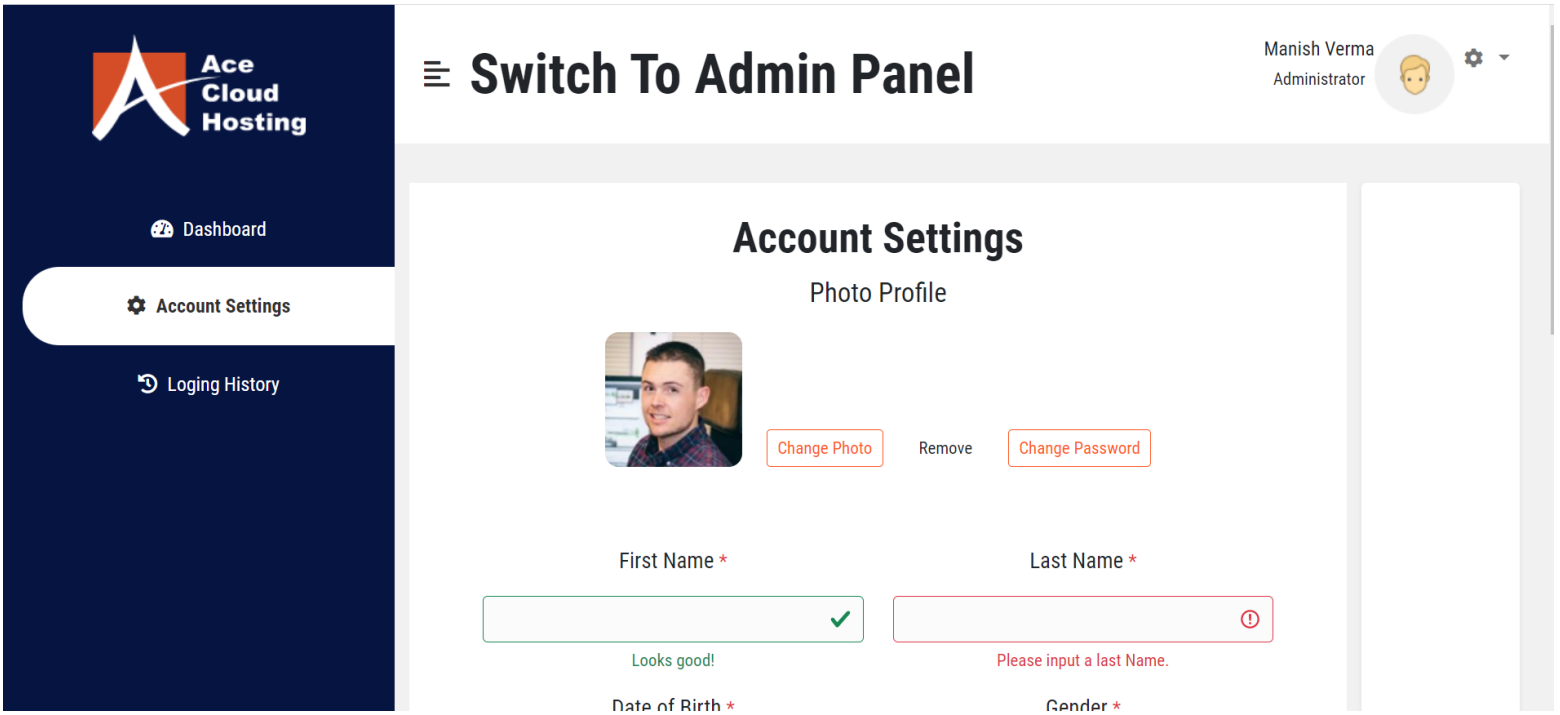
Our Progress:

- We have developed the theme switching feature for our application which allows a user to switch themes when he wants to.
- We have successfully developed the dashboard for the application.

Dashboard screenshots:

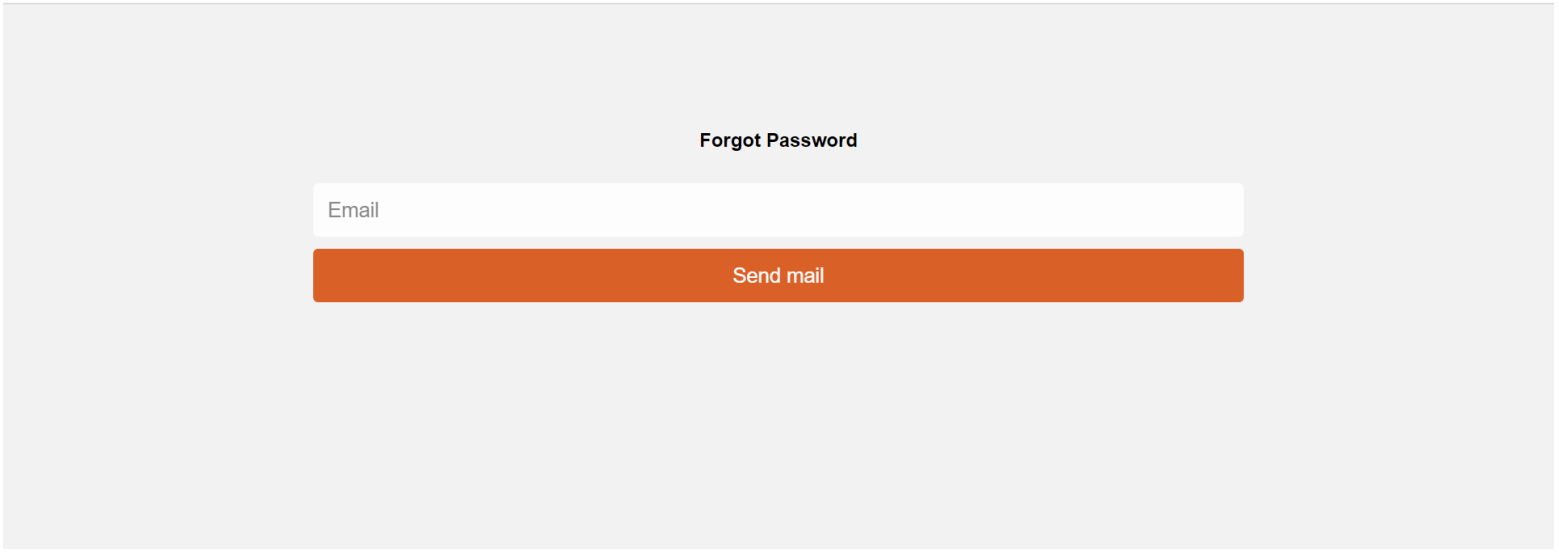


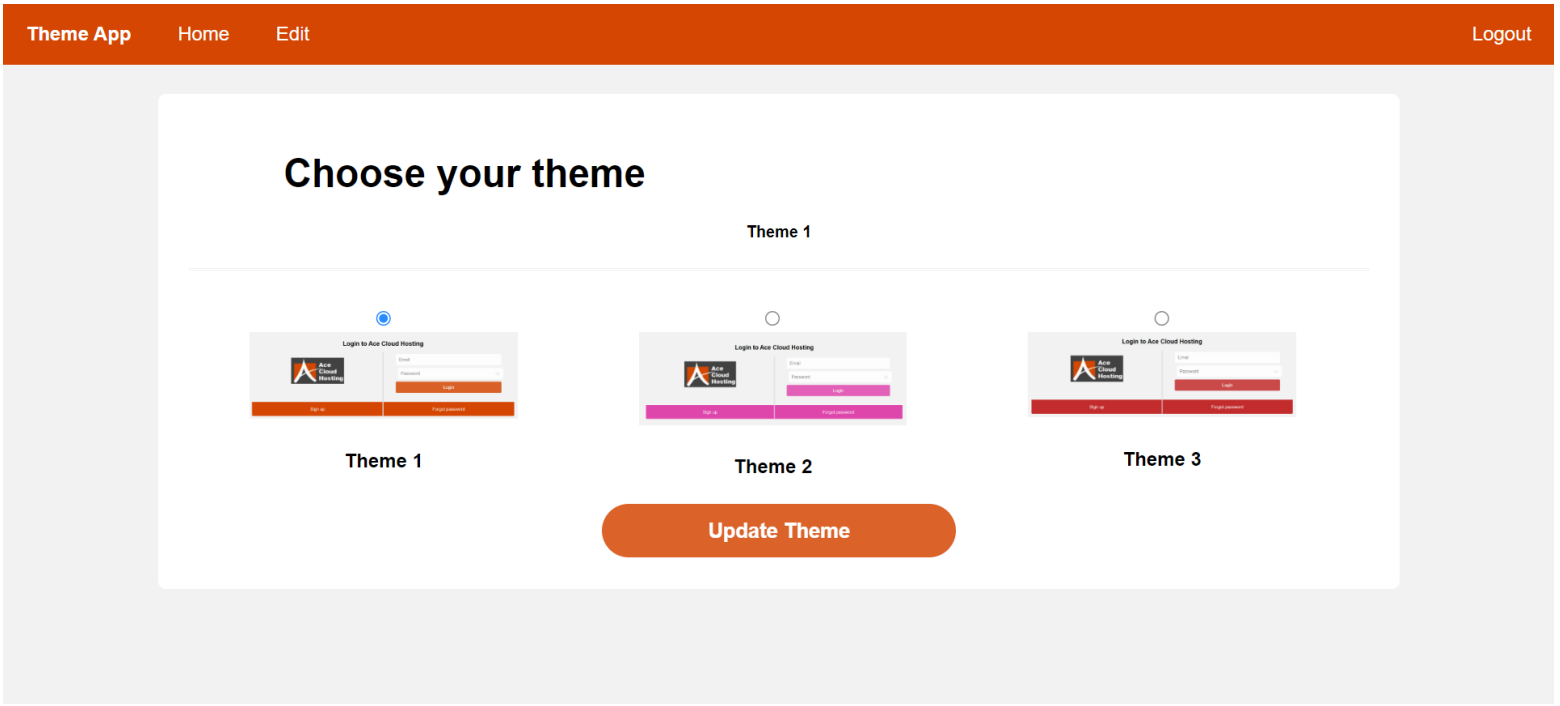

Ace Cloud Hosting
Switch To Admin Panel
Manish Verma
Administrator
Dashboard
Account Settings
Loging History
Loging History
Date
Country
App Name
IP Address
May 1, 2022
United States
App 1
1,541,605,760
May 2, 2022
China
App 2
330,321,408
May 2, 2022
China
App 2
330,321,408
May 2, 2022
China
App 2
330,321,408
May 2, 2022
China
App 2
330,321,408
May 2, 2022
China
App 2
330,321,408

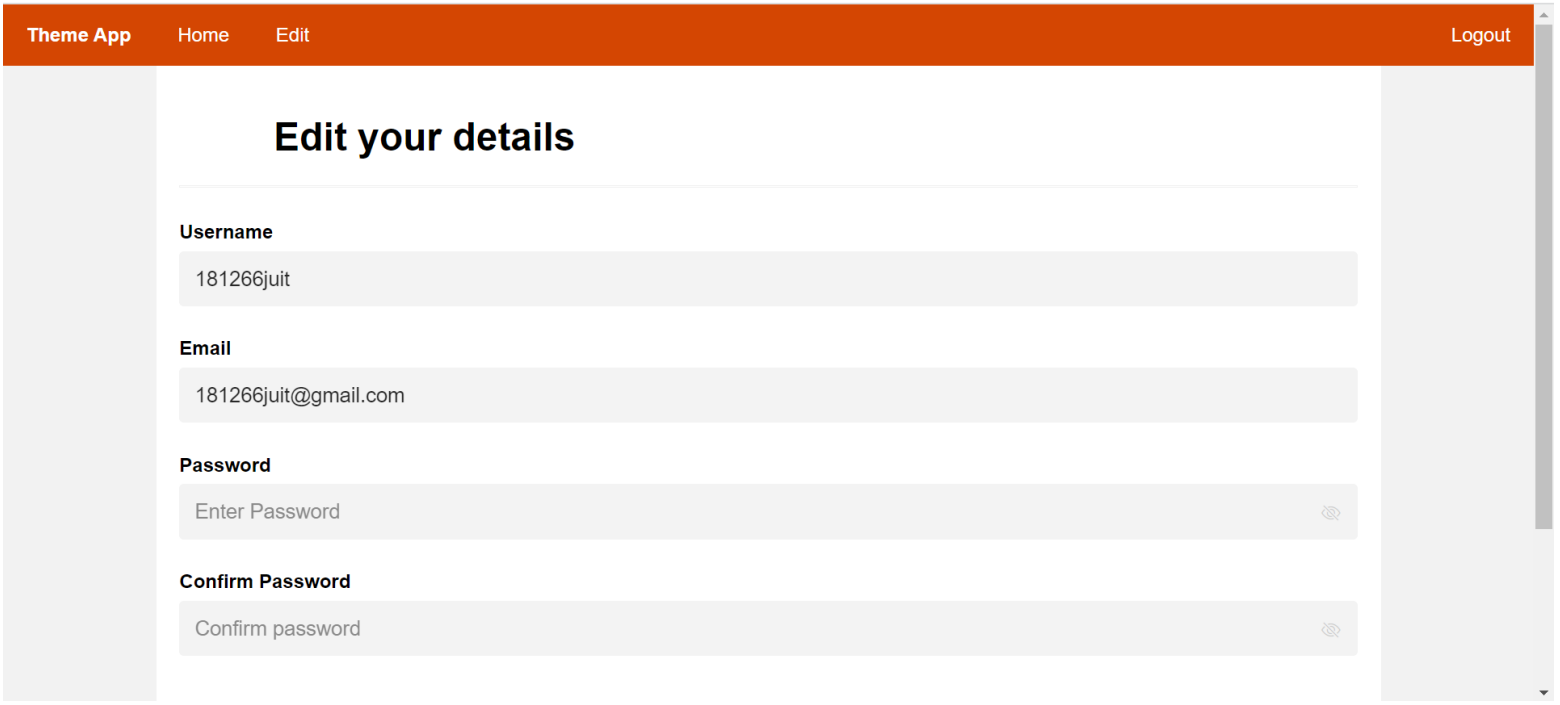

Ace Cloud Hosting
Switch To Admin Panel
Manish Verma
Administrator
Dashboard
Account Settings
Loging History
Account Settings
Photo Profile
Change Photo
Remove
Change Password
First Name *
Last Name *
Looks good!
Please input a last Name.
Date of Birth *
Gender *

Screenshots of various themes designed:

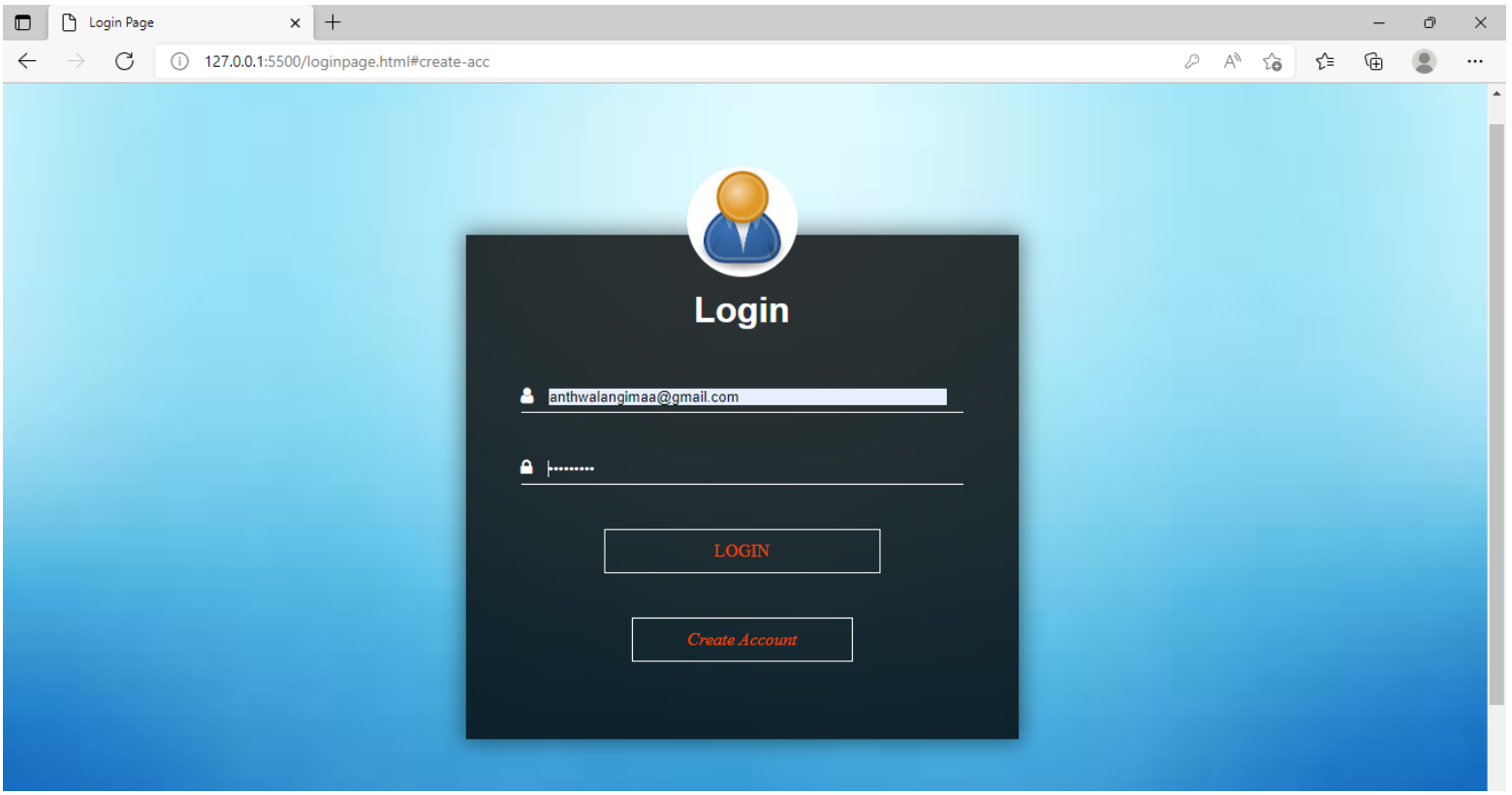
Theme-1

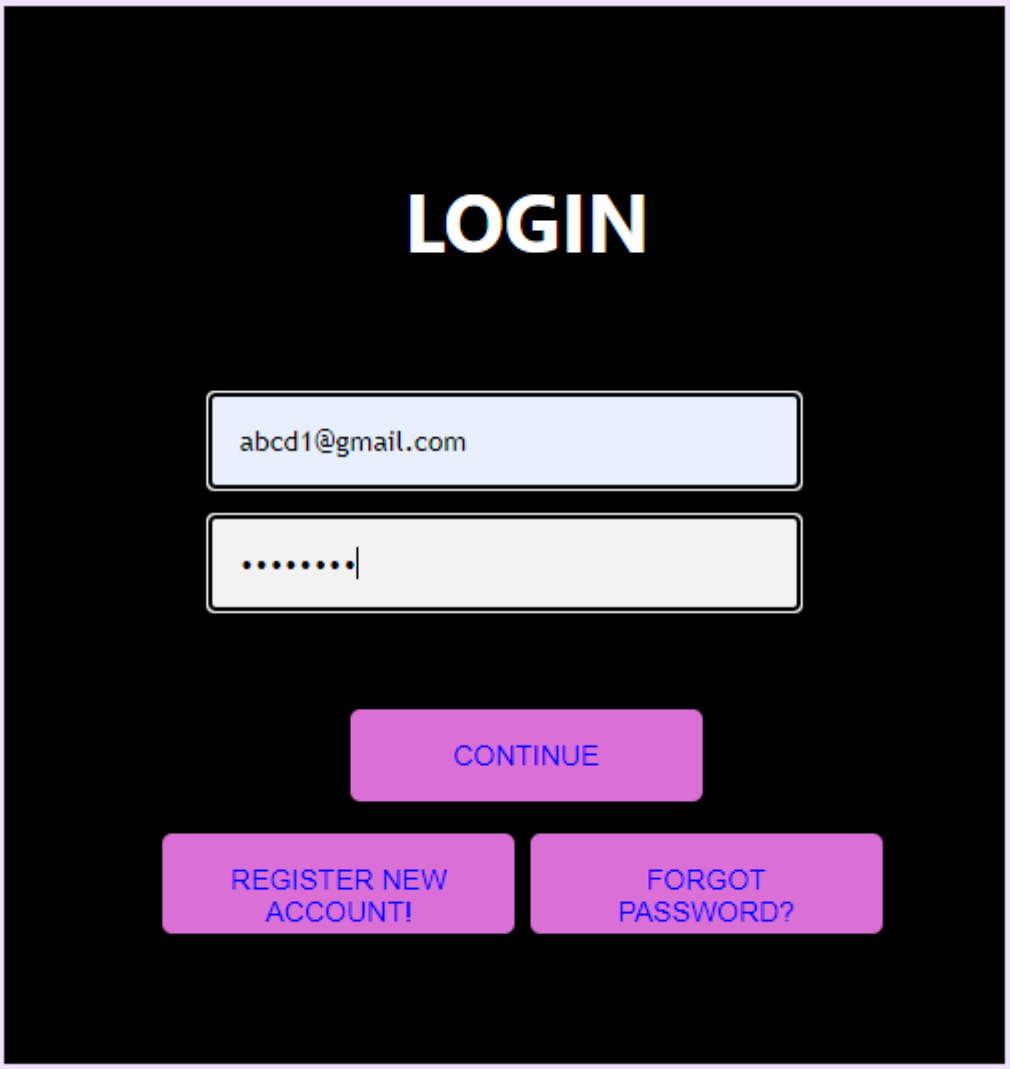


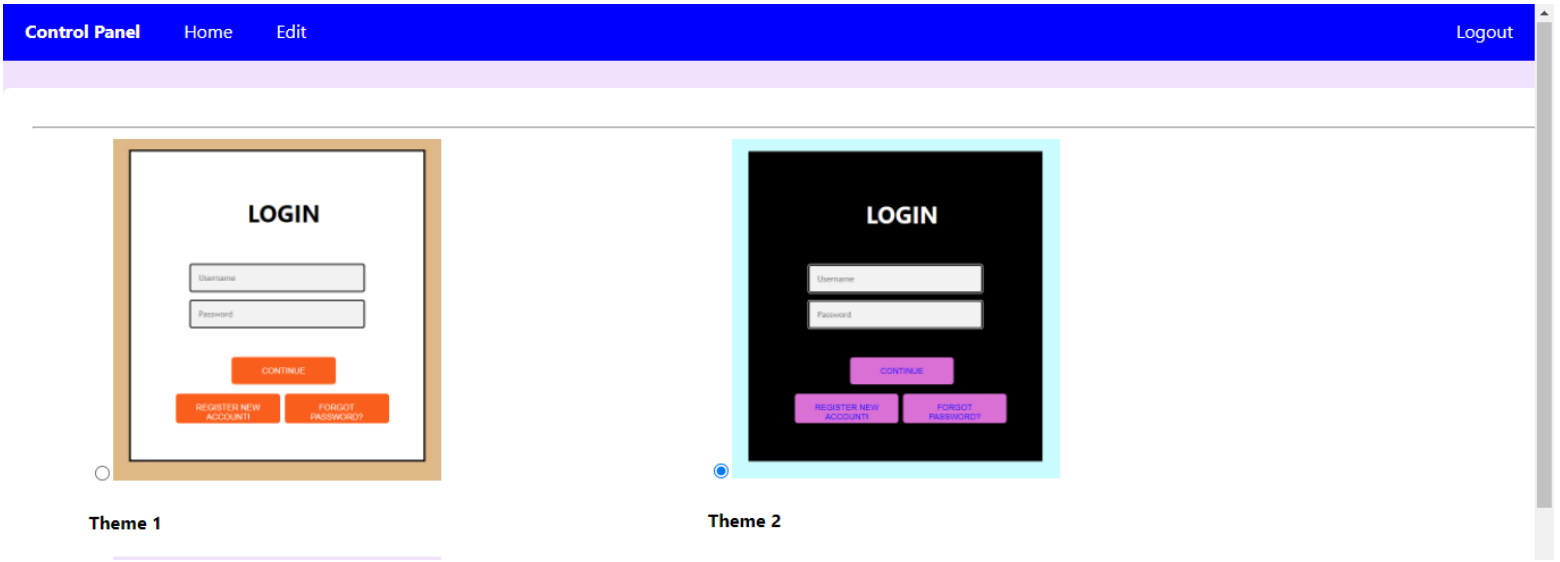

Theme App
Home
Edit
Logout
Choose your theme
Theme 1
Theme 1
Theme 2
Theme 3
Update Theme

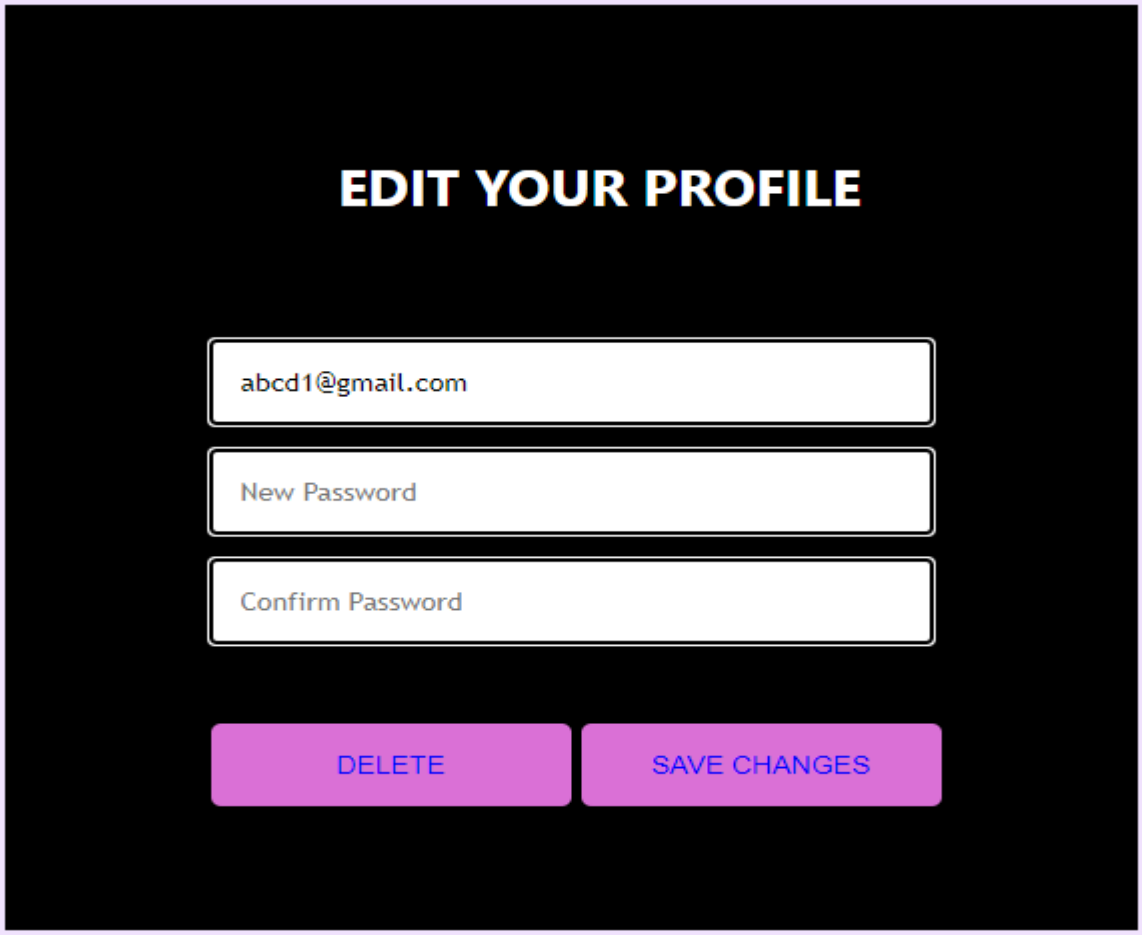

Theme App
Home
Edit
Logout
Edit your details
Username
181266juit
Email
181266juit@gmail.com
Password
Enter Password
Confirm Password
Confirm password

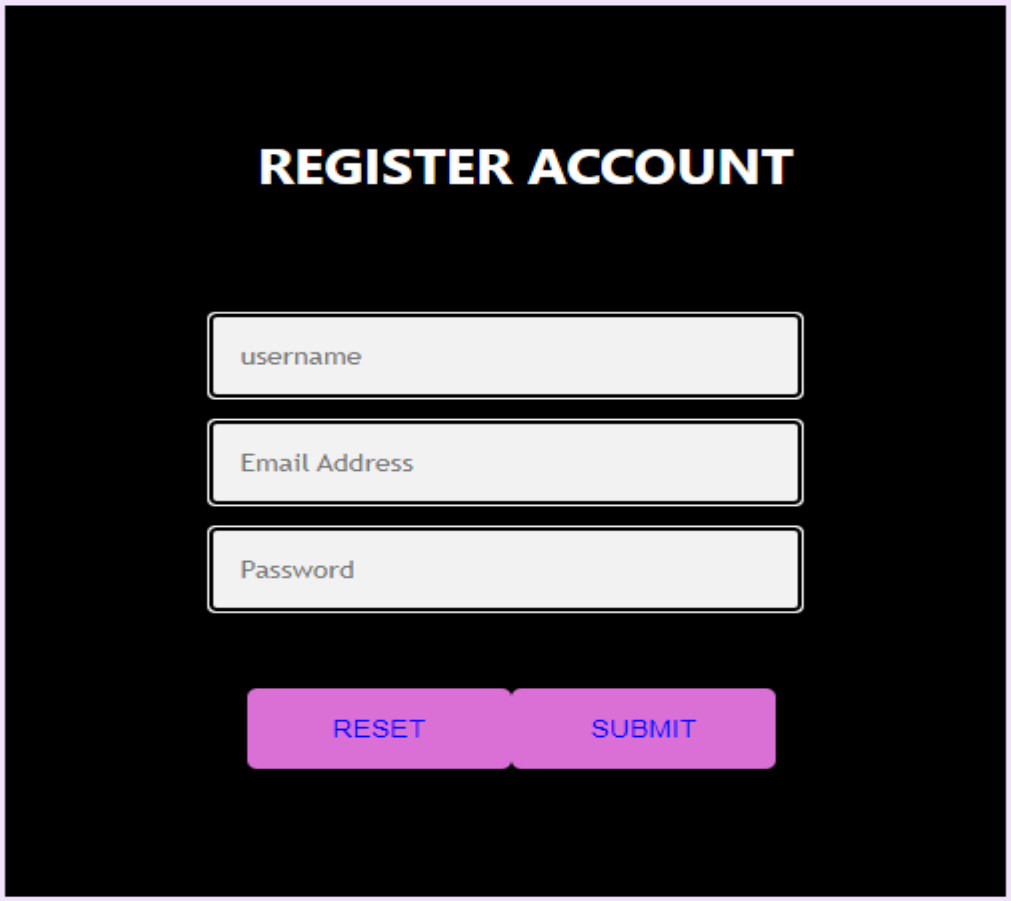
Theme-2

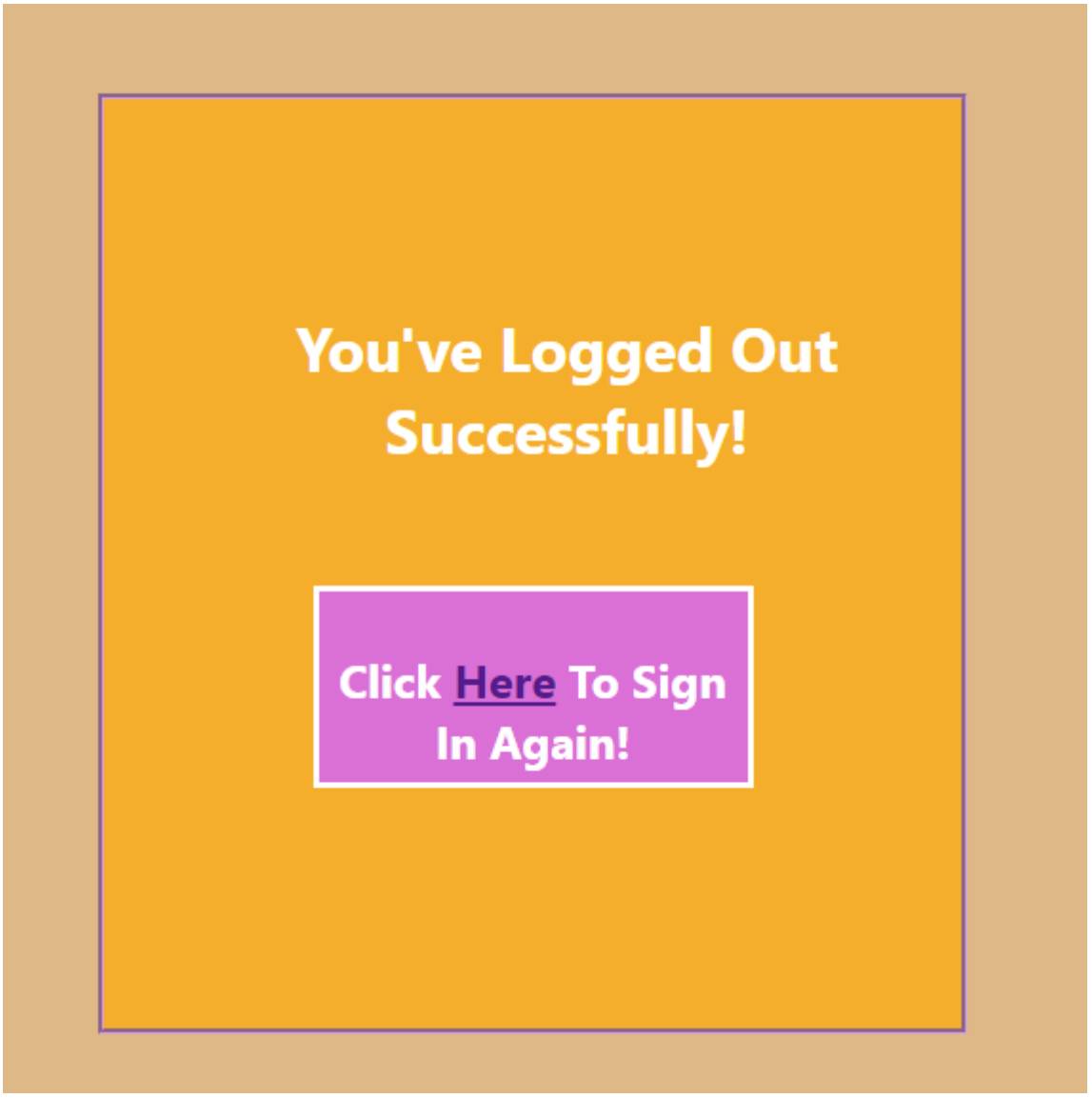

Login Page
127.0.0.1:5500/loginpage.html#create-acc
Login
anthwalangimaa@gmail.com
LOGIN
Create Account


LOGIN
abcd1@gmail.com
CONTINUE
REGISTER NEW ACCOUNT!
FORGOT PASSWORD?


Control Panel
Home
Edit
Logout
LOGIN
Username
Password
CONTINUE
REGISTER NEW ACCOUNT!
FORGOT PASSWORD?
Theme 1
LOGIN
Username
Password
CONTINUE
REGISTER NEW ACCOUNT!
FORGOT PASSWORD?
Theme 2


EDIT YOUR PROFILE
abcd1@gmail.com
New Password
Confirm Password
DELETE
SAVE CHANGES


REGISTER ACCOUNT
username
Email Address
Password
RESET
SUBMIT


You've Logged Out Successfully!
Click Here To Sign In Again!

Chapter-5

CONCLUSIONS

Conclusion:

In conclusion we would be able to develop a fully functional, fast, reliable and secure Identity and Access Management System with single sign-on with all the solutions provided by us (mentioned above) by August, 2022 and would be launched for the users by November, 2022 with all the features.

Future Scope:

- Adding features so that the pages created can be customized by the organization based on their needs.
- Integrating web apps with single sign on features.
- Generating tokens for session management.
- Implementing IAM.

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: .....

Type of Document (Tick): PhD Thesis | M.Tech Dissertation/ Report | B.Tech Project Report | Paper

Name: _____ **Department:** _____ **Enrolment No** _____

Contact No. _____ **E-mail.** _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

_____

_____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at .....(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

<table>
<tr><td>Copy Received on</td><td rowspan="4">• All Preliminary Pages
• Bibliography/Images/Quotes
• 14 Words String</td><td rowspan="2"></td><td>Word Counts</td><td></td></tr>
<tr><td>Report Generated on</td><td>Character Counts</td><td></td></tr>
<tr><td rowspan="2"></td><td>Submission ID</td><td>Total Pages Scanned</td><td></td></tr>
<tr><td></td><td>File Size</td><td></td></tr>
</table>

Checked by
Name & Signature **Librarian**

.....

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com