

A PROJECT REPORT

ON

“ADVANCED RANDOM PASSWORD GENERATOR”

Submitted in partial fulfilment of the requirement for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

RGUKT, NUZVID

This project was completed as part of a summer internship at Edubot Software and Services.

2024

BY

MUDHAVATH RAVI KUMAR NAIK (N190920)

Under the guidance of

Mr. Naveen Kumar

**SOFTWARE DEVELOPER AT EDUBOT SOFTWARE
AND SERVICES**

DECLARATION

I, **Mudhavath Ravi Kumar Naik**, hereby declare that the Project Report titled “Random Password Generator” submitted to Rajiv Gandhi University of Knowledge Technologies, Nuzvid, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Computer Application is a report of work done by me under the supervision of **Mr. Naveen Kumar** at Edubot Software and Services. I also declare that this report or any part of it has not been submitted to any other University/Institute for the award of any degree.

Signature of Student

M.Ravi Kumar Naik

ACKNOWLEDGEMENT

I would like to express my sincere gratitude and regards to my internals guide **Mr.Naveen Kumar** for his constant inspiration, supervision and invaluable guidance during the training. I would also like to thank madam **Mrs.Sri Lakshmi** for giving such an opportunity to continue my training in the (Edubot System and Services) At last, I would also like to extend my sincere gratitude to all my faculty members and specially **Mr.T.Chandra Shekar** (internal guide) for giving their valuable suggestions.

Signature of Student

M.Ravi Kumar Naik

(Sign of HOD)

(Sign of Internal Guide)

Table of Contents

- 1.Introduction
- 2.System Requirements
- 3.System Architecture
- 4.Features
- 5.Technical Specifications
- 6.User Interface
- 7.Future Enhancements
- 8.Coding
- 9.Execution Screenshots
- 10.Conclusion

1. Introduction

The Introduction sets the stage by providing an overview of the React-based Random Password Generator application. This application is designed to enhance the security and usability of password management for users, addressing the increasing need for strong and unique passwords in digital environments. It emphasizes the importance of generating secure passwords to protect sensitive data while ensuring ease of access and usability. The introduction outlines the primary goals of the project, which include enhancing data security, improving user convenience in generating and managing passwords, and providing a reliable solution for safeguarding personal and professional accounts effectively.

2. System Requirements

Hardware Requirements

- Standard PC or laptop

Software Requirements

- React.js
- PostgreSQL Database
- JSON for data handling
- Code Editor (Visual Studio Code)

3. System Architecture

The System Architecture describes the overall structure and components of the Random Password Generator application. It outlines how different modules and layers interact to achieve the application's functionality. The architecture typically includes:

- **Client-Side Application:** React-based web application for user interaction.
- **Server-Side Backend:** Manages data handling and business logic using JSON.
- **Database:** Stores user credentials and generated passwords securely using PostgreSQL.
- **Communication Protocol:** Defines how the client communicates with the server (e.g., RESTful API).
- **Security Measures:** Includes encryption protocols, secure authentication mechanisms, and data integrity checks.
- **Scalability and Modularity:** Design considerations to accommodate future updates and enhancements without compromising system performance.

Overview

The Random Password Generator application follows a web application architecture with React.js for the user interface and JSON for data handling.

Components

1. Frontend: React.js
2. Backend: JSON for data handling
3. Database: PostgreSQL

Data Flow

1. User interacts with the React frontend.
2. Requests are processed by the JSON-based backend.
3. Backend communicates with the PostgreSQL database for CRUD operations.
4. The backend sends the response back to the React frontend.
5. The frontend updates the User Interface based on the response.

4. Features

The Features section outlines the core functionalities of the Random Password Generator application that enhance user experience and security. Key features include:

Password Generation: Allows users to generate strong, random passwords to enhance security.

User Authentication: Ensures secure access to the application through username/password authentication.

User Registration: Facilitates easy registration for new users.

Password Management: Enables users to store and manage their generated passwords securely.

Session Management: Limits users to generating up to five passwords per session before redirecting them to the login page.

Navigation Bar: Provides easy access to user details, password generator, user passwords, and logout.

5. Technical Specifications

Technical Specifications detail the tools, technologies, and frameworks chosen for developing the Random Password Generator application. This includes:

Programming Language: JavaScript, chosen for its flexibility and robust ecosystem in web application development.

Frontend Framework: React.js, utilized for creating the user interface components (buttons, text fields, dialogs) that ensure a consistent and responsive user experience.

Database Management System: PostgreSQL, selected for its reliability, performance, and compatibility with web applications.

APIs and Libraries: JSON for data handling, providing efficient data retrieval and manipulation capabilities within the application.

Frontend (React.js)

- **React Components:** For UI design and functionality.
- **Bootstrap:** For UI components and styling.

Backend

- **JSON:** For data handling and communication between frontend and backend.

Database

- **PostgreSQL:** For managing and querying data.

6. User Interface

User Interface details the design and functionality of the React-based Random Password Generator application's graphical user interface (GUI). This includes:

- **UI Components:** Screenshots and descriptions of main screens (login, password generator, user details), dialogs, and menus designed to facilitate user interaction.
- **Layout and Navigation:** Design considerations for intuitive navigation, responsive layout design, and accessibility features (keyboard shortcuts, tooltips).
- **Usability Principles:** Incorporates user-centered design principles to enhance usability, such as consistent design elements, error handling, and feedback mechanisms.
- **Localization and Accessibility:** Addresses localization needs (language support) and accessibility requirements (screen reader compatibility, contrast ratios) to ensure inclusivity.

UI Components

Screenshots and Descriptions: Visuals of key screens, including:

- Login and Signup Page
- Password Generator
- User Details
- User Passwords Details

7. Future Enhancements

Future Enhancements discusses potential features and improvements planned for future releases of the Random Password Generator application. It includes:

Enhanced Customization: Offer options for users to customize password criteria like length and character types.

Integration and Accessibility: Explore integration with password managers and develop mobile apps and browser extensions for wider accessibility.

Advanced Security Measures: Implement features like two-factor authentication and password strength assessment tools.

User Feedback and Education: Gather user feedback, provide educational resources on password security, and continuously improve based on user insights.

8.Coding

The coding section provides insights into the implementation details of the random password generator project. It covers the code structure, key algorithms and methods, coding best practices, and version control usage.

PasswordGenerator.jsx

```
import React, { useState } from 'react';

import { Navigate } from 'react-router-dom'; // Import Navigate for redirection

import ModalBox from './ModalBox'; // Adjust the path as per your file structure

import './PasswordGenerator.css';

const PasswordGenerator = () => {

  const [password, setPassword] = useState("");

  const [passwordLength, setPasswordLength] = useState(12);

  const [includeUpperCase, setIncludeUpperCase] = useState(true);

  const [includeLowerCase, setIncludeLowerCase] = useState(true);

  const [includeNumbers, setIncludeNumbers] = useState(true);

  const [includeSpecialChars, setIncludeSpecialChars] = useState(true);

  const [includeDuplicates, setIncludeDuplicates] = useState(false);

  const [attempts, setAttempts] = useState(0);

  const MAX_ATTEMPTS = 5;
```



```
const [showModal, setShowModal] = useState(false);

const [showMaxAttemptsMessage, setShowMaxAttemptsMessage] = useState(false);


const generatePassword = () => {

  const chars = generateCharset();

  let newPassword = "";

  let usedChars = "";

  for (let i = 0; i < passwordLength; i++) {

    let randomChar = chars.charAt(Math.floor(Math.random() * chars.length));

    if (!includeDuplicates) {

      while (usedChars.includes(randomChar)) {

        randomChar = chars.charAt(Math.floor(Math.random() * chars.length));

      }

      usedChars += randomChar;

    }

    newPassword += randomChar;

  }

  setPassword(newPassword);

};

const generateCharset = () => {

  let charset = "";

  if (includeUpperCase) charset += 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';

  if (includeLowerCase) charset += 'abcdefghijklmnopqrstuvwxyz';

  if (includeNumbers) charset += '0123456789';

  if (includeSpecialChars) charset += '!@#$%^&*()_+';

}
```

```
    return charset;

};

const copyToClipboard = () => {

    navigator.clipboard.writeText(password);

    alert('Password copied to clipboard!');

};

const handleLengthChange = (e) => {

    const length = parseInt(e.target.value);

    setPasswordLength(length);

};

const handleCheckboxChange = (e) => {

    const { id, checked } = e.target;

    switch (id) {

        case 'upperCase':

            setIncludeUpperCase(checked);

            break;

        case 'lowerCase':

            setIncludeLowerCase(checked);

            break;

        case 'numbers':

            setIncludeNumbers(checked);

            break;

        case 'specialChars':

            setIncludeSpecialChars(checked);
```

```

        break;

    case 'includeDuplicates':

        setIncludeDuplicates(checkeD);

        break;

    default:

        break;

    });

const handleGenerateClick = () => {

    if (attempts >= MAX_ATTEMPTS) {

        setShowModal(true);

        return; }

    generatePassword();

    setAttempts(attempts + 1);

    if (attempts >= MAX_ATTEMPTS) {

        setShowMaxAttemptsMessage(true);

        alert("You have reached the maximum login attempts. If you want more chances to
login.");

    } };

const closeModal = () => {

    setShowModal(false);

    setAttempts(0); // Reset attempts after modal is closed

    setShowMaxAttemptsMessage(false); // Hide max attempts message after modal is closed

};

// Redirect to login page after 5 attempts

if (attempts >= MAX_ATTEMPTS) {

    return <Navigate to="/login" />;

```

```
}
```

```
return (
```

```
<div className="password-generator-container">
```

```
<h2 className='header'>Random Password Generator App</h2><br />
```

```
<div className="password-display">
```

```
<input type="text" value={password} readOnly />
```

```
<button onClick={copyToClipboard}>Copy</button>
```

```
</div>
```

```
<div className="password-controls">
```

```
<center><label>Password Length:</label></center>
```

```
<input
```

```
id="pass-range"
```

```
type="range"
```

```
min="6"
```

```
max="20"
```

```
value={passwordLength}
```

```
onChange={handleLengthChange} />
```

```
<span>{passwordLength}</span>
```

```
</div>
```

```
<div className="checkbox-group">
```

```
<label>
```

```
<input
```

```
id="upperCase"
```

```
type="checkbox"
```

```
className="styled-checkbox"
```

checked={includeUpperCase}

onChange={handleCheckboxChange} />

Include Uppercase Letters

</label>

<label>

<input

id="lowerCase"

type="checkbox"

className="styled-checkbox"

checked={includeLowerCase}

onChange={handleCheckboxChange} />

Include Lowercase Letters

</label>

<label>

<input

id="numbers"

type="checkbox"

className="styled-checkbox"

checked={includeNumbers}

onChange={handleCheckboxChange}/>

Include Numbers

</label>

<label>

<input

id="specialChars"

```
type="checkbox"

className="styled-checkbox"

checked={includeSpecialChars}

onChange={handleCheckboxChange}/>
```

Include Special Characters

```
</label>
```

```
<label>
```

```
<input
```

```
id="includeDuplicates"
```

```
type="checkbox"
```

```
className="styled-checkbox"
```

```
checked={includeDuplicates}
```

```
onChange={handleCheckboxChange}
```

```
/>
```

Include Duplicates

```
</label>
```

```
</div>
```

```
<br/>
```

```
<button
```

```
className="generate-button"
```

```
onClick={handleGenerateClick}
```

```
disabled={attempts >= MAX_ATTEMPTS} >
```

Generate Password

```
</button>
```

```
{showMaxAttemptsMessage && (
```

```

    <p style={{ color: 'red' }}>You have reached the maximum attempts limit.</p>
  )}

  <p>Attempts: {attempts} / {MAX_ATTEMPTS}</p>

  <ModalBox isOpen={showModal} onClose={closeModal} />

</div>

);

};

export default PasswordGenerator;

```

Explanation:

- **State:** Manages password generation settings and user interactions using useState hooks.
- **Functions:** Generates random passwords, handles checkbox changes, copies passwords to clipboard, and manages modal visibility.
- **UI Elements:** Displays password input, controls for length and character types, and buttons for generating and copying passwords.
- **Feedback:** Alerts users on password copy and displays attempts count and max attempts message.

UserData.js

```

let UsersData = [

{

  id: 1,

  username: "Kranthi Sir",

  email: "kranthi@gmail.com",

  password: "password",

},

{

  id: 2,

```

```

    username: "Ravikumar",
    email: "ravi@gmail.com",
    password: "password",
  },
  {
    id: 3,
    username: "testing",
    email: "test@gmail.com",
    password: "1",
  },
];

const addUser = (formData) => {
  const newUser = {
    id: UsersData.length + 1,
    username: formData.username,
    email: formData.email,
    password: formData.password,
  };
  UsersData.push(newUser);
};

export { UsersData, addUser }; // Export UsersData and addUser functions

```

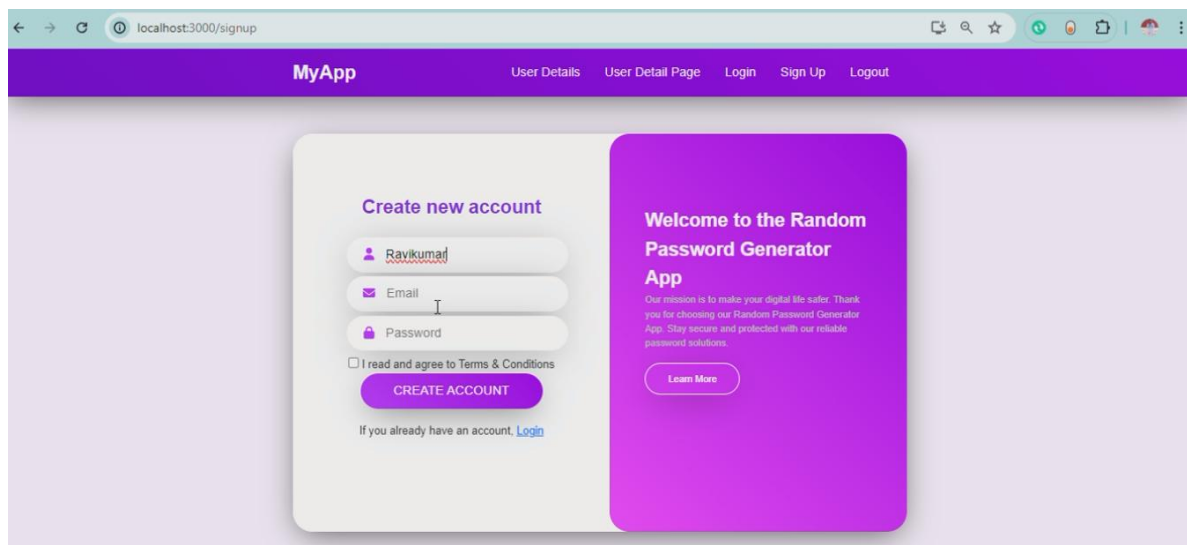
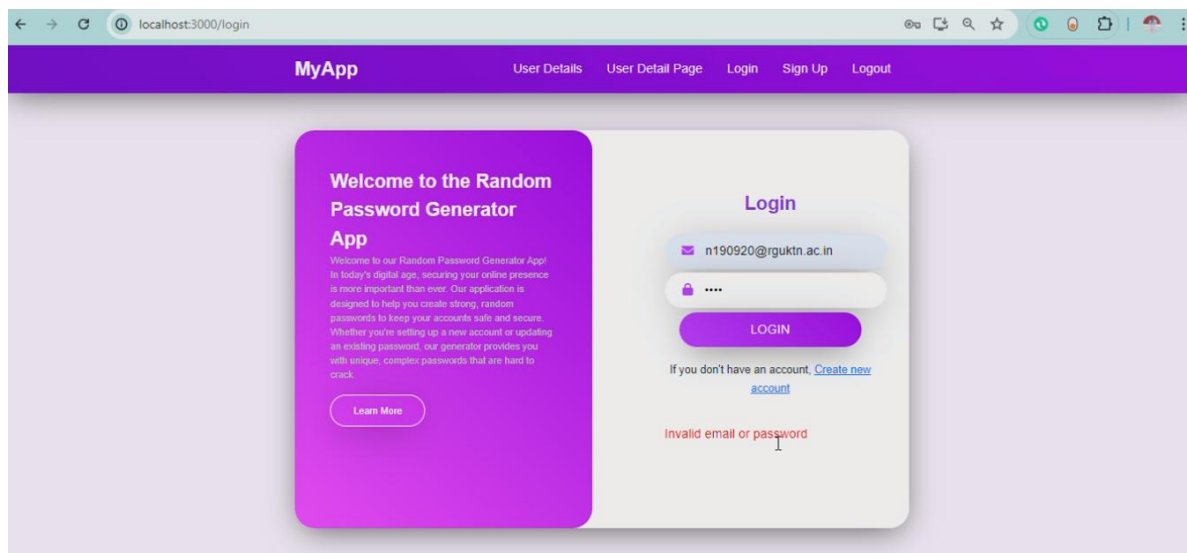
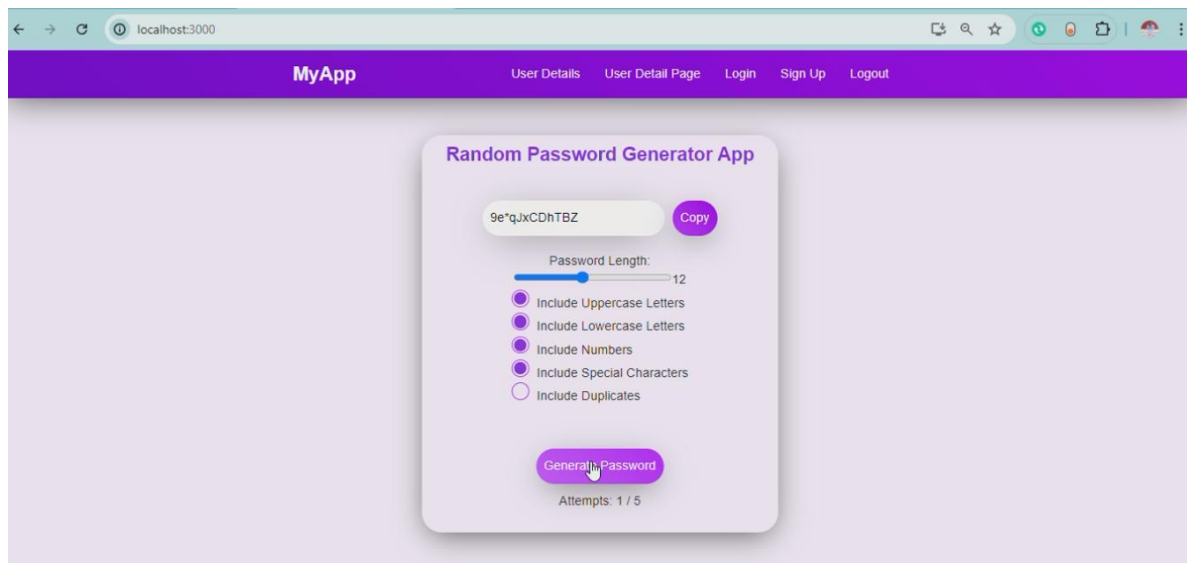
Explanation:

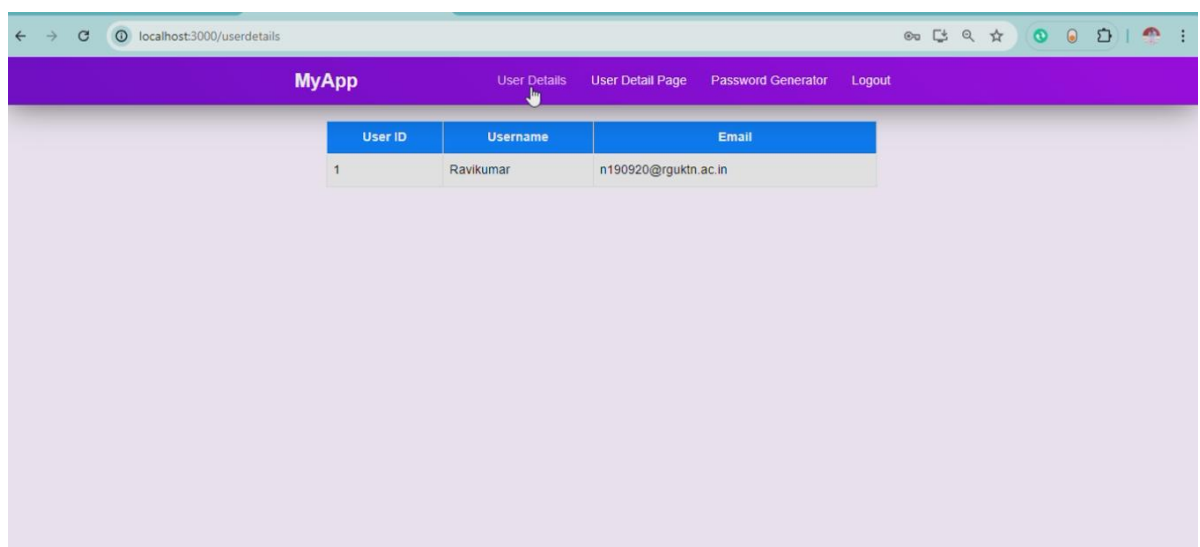
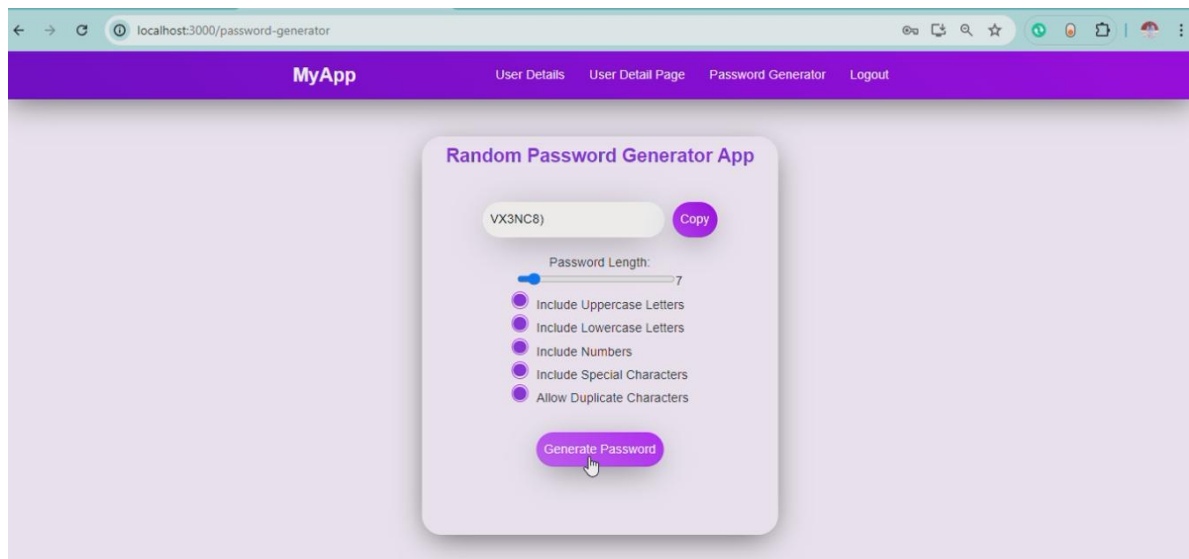
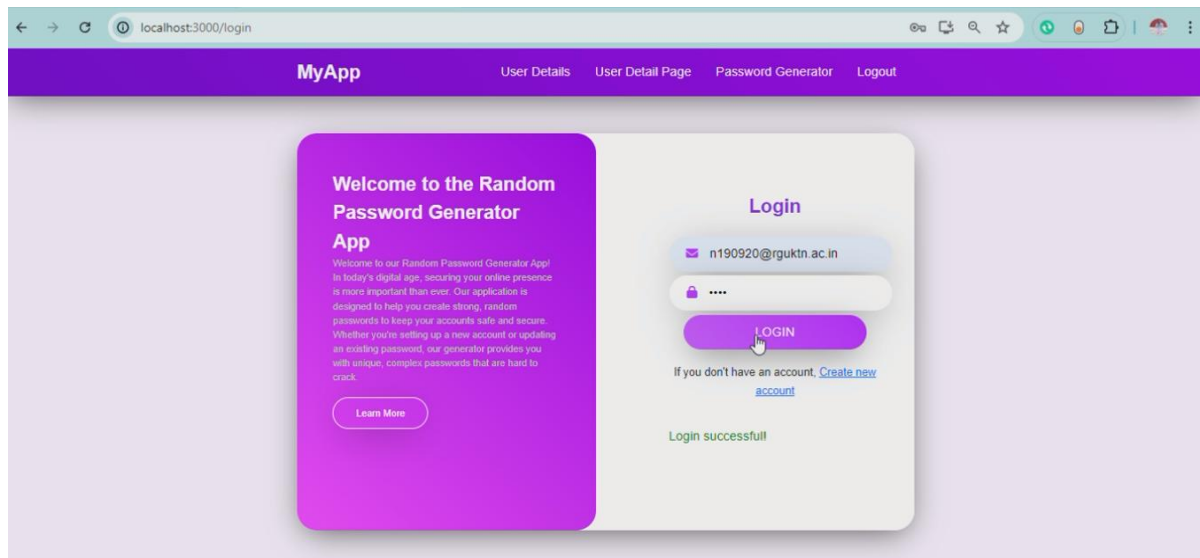
Data Structure: Defines an array `UsersData` containing user objects with `id`, `username`, `email`, and `password`.

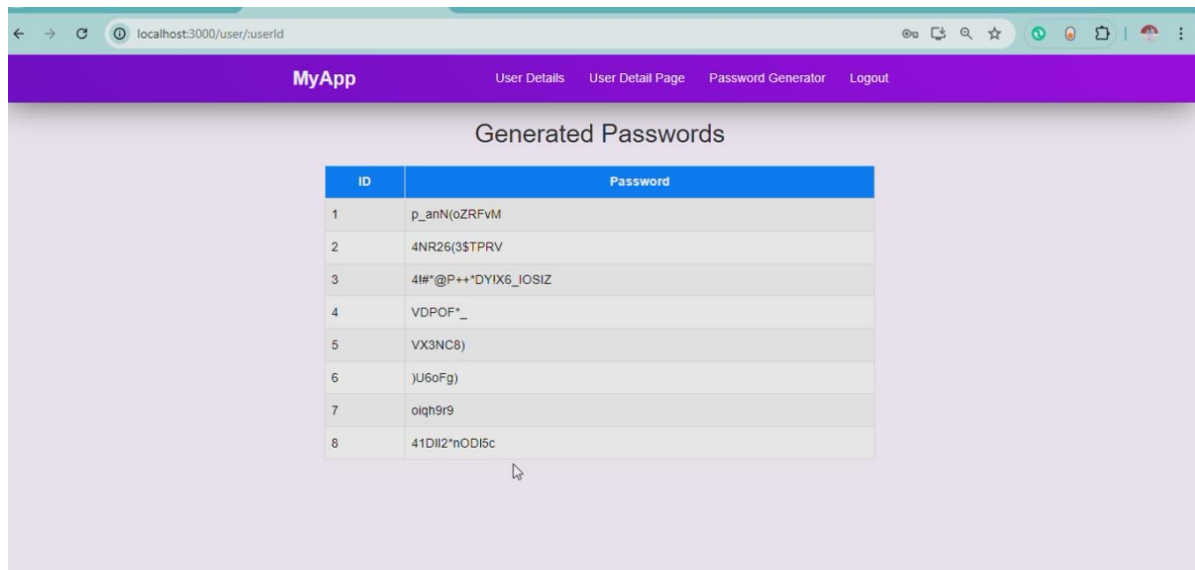
Functionality: Provides `addUser(formData)` function to add new user data to `UsersData` array.

Export: Exports `UsersData` array and `addUser` function for use in other parts of the application.

9. Execution Screenshots







ID	Password
1	p_anN(oZRFvM
2	4NR26(3\$TPRV
3	4!#*@P++*DYIX6_IOSIZ
4	VDPOF*_
5	VX3NC8)
6)U6oFg)
7	oiqh9r9
8	41DI#2*nODI5c

10. Conclusion

This Random Password Generator application features a user-friendly random password generator prioritizing security and usability. It includes a navigation bar with links to user details, password generator, user passwords details, and logout. Users can generate up to five passwords per session before being directed to a login page, which facilitates easy registration for new users. After registration, users return to the login page for immediate access. Once logged in, they have unlimited use of the password generator and seamless navigation for managing user details and stored passwords, ensuring efficient and secure password management.