NARASARAOPETA ENGINEERING COLLEGE
(AUTONOMOUS)

React Project

Peer Learning: React.js

Project Name : **Online Banking**

Team Members : A.Jashwanth (22471A05L3)

SK.I.Basha (22471A05P1)

Y.L.Krishna (22471A05P2)

Department of Computer Science and Engineering

NARASARAOPETA ENGINEERING COLLEGE: NARASARAOPETA

(AUTONOMOUS)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project work entitled "Online Banking" is a bonafide work done by the team"A.Jashwanth (22471A05L3), SK.I.F.Basha (22471A05P1) & Y.Leela Krishna (22471A05P2)" in partial fulfillment of the requirements for the award of the degree of **BACHELOR OF TECHNOLOGY in the Department of COMPUTER SCIENCE AND ENGINEERING** during 2024-2025.

PROJECT GUIDE
Sd.Rizwana, M.Tech,
Asst.Professor

PROJECT CO-ORDINATOR
G.Saranya ,M.Tech.

HEAD OF THE DEPARTMENT
Dr. S. N. Tirumala Rao, M.Tech., Ph.D.,
Professor & HOD

Details of the project

Title of the Project	**:**	Online Banking
Names of Team members	**:**	A.Jashwanth, SK.I.F.Basha,& Y.Leela Krishna
Roll No	**:**	22471A05L3, 22471A05P1, 22471A05P2,
Section	**:**	CSE-D
Technology Stack :		
Frontend	**:**	ReactJs
Backend	**:**	Node.JS and Express
Database	**:**	MY SQL

Project Overview:

Online Banking

- Online banking, also known as internet banking, e-banking or virtual banking, is an electronic payment system that enables customers of a bank or other financial institution to conduct a range of financial transactions through the financial institution's website.
- Modern banks not only deal in money and credit but they also perform various functions, namely, agency functions, management of foreign trade, finance, etc.

TYPES OF INTERNET BANKING

Water pollution is a widespread problem that can contaminate drinking water sources with harmful substances.

- More than one billion people in the world do not have access to safe drinking water.
- Polluted drinking water is a major cause for many different diseases, e.g., cancers, and reproductive and digestive problems.

ADVANTAGES OF SAFE DRINKING WATER

- **Informational Websites :-** Such services are known as first level of e-banking. Through such services bank provides marketing information regarding banking products and services on a standalone server. It has very low degree of risk as there is no connection between server and bank.

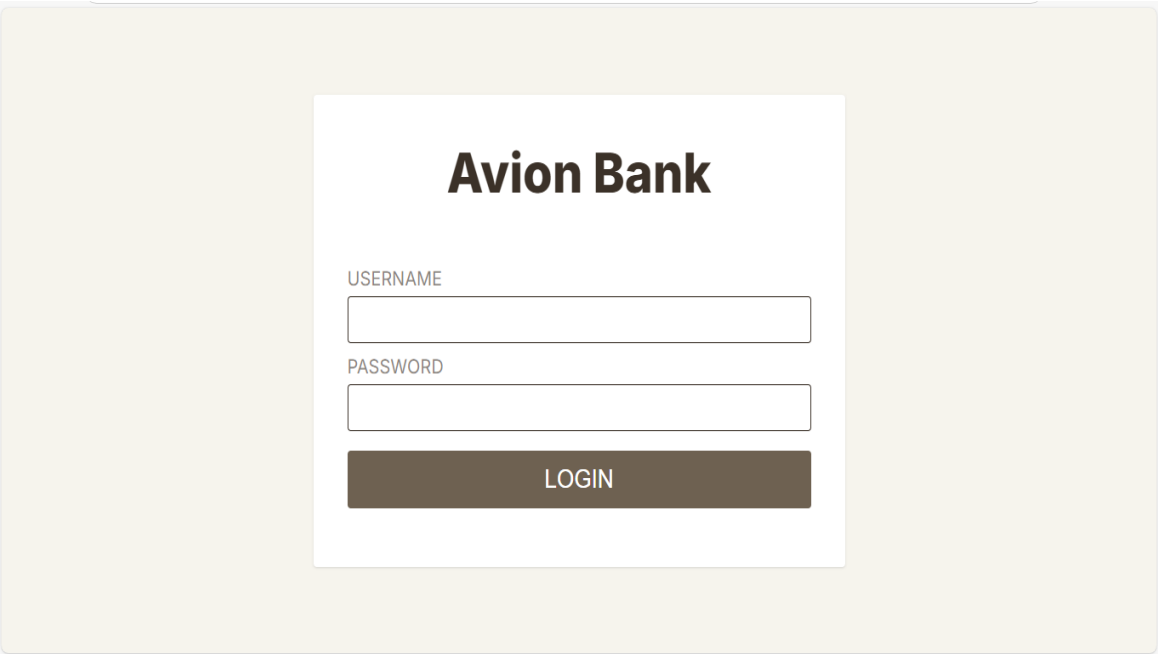
- **Advanced Transactional Websites:-** A bank customer can perform non-transactional tasks through online banking, including

  Viewing account balances

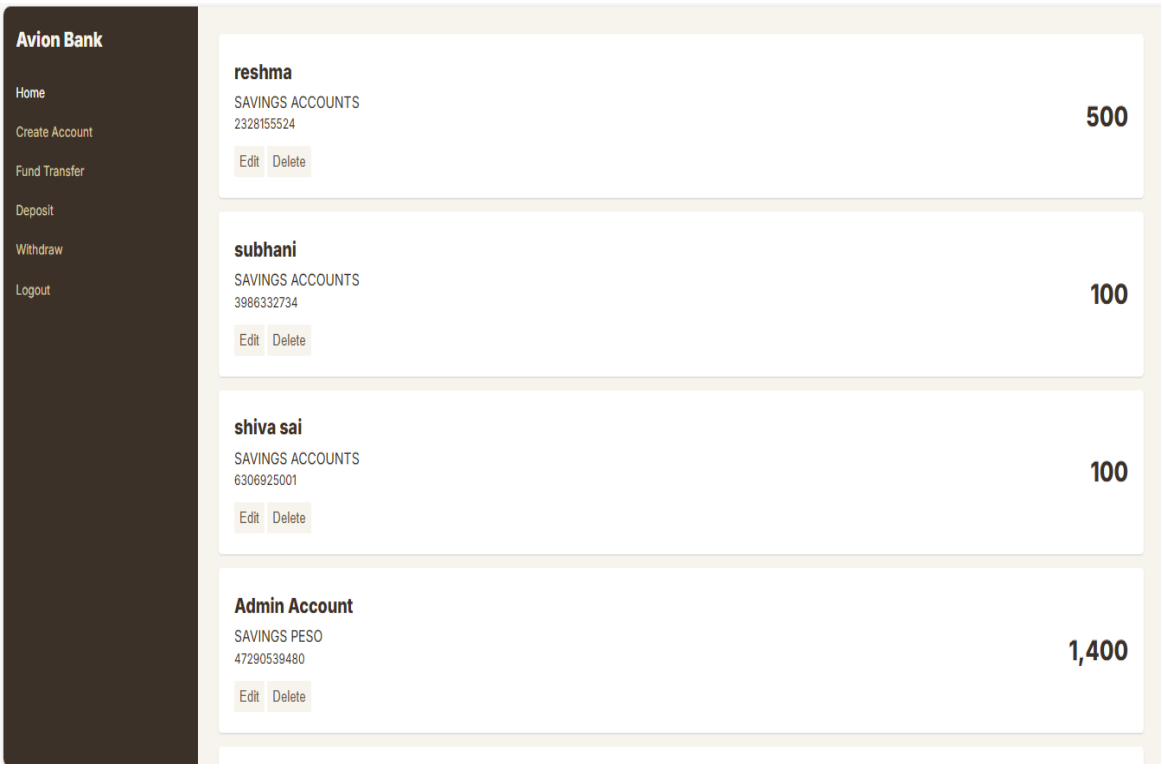
  - Viewing recent transactions
  - Downloadingbank statements, for example in PDF format
  - Viewing images of paid cheques
  - Orderingcheque books
  - Downloadperiodic account statements
  - Downloadingapplications for M-banking, E-banking etc.

Photos of Project Implementation:

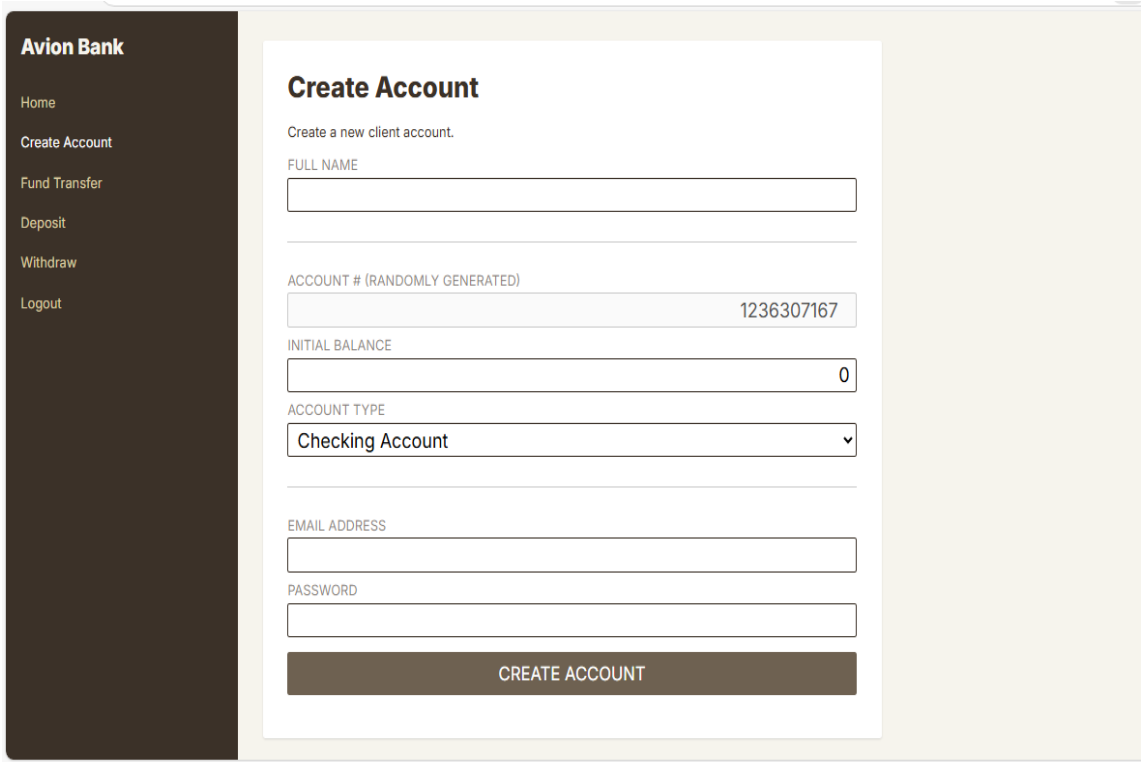
Screenshot 1: Login Page



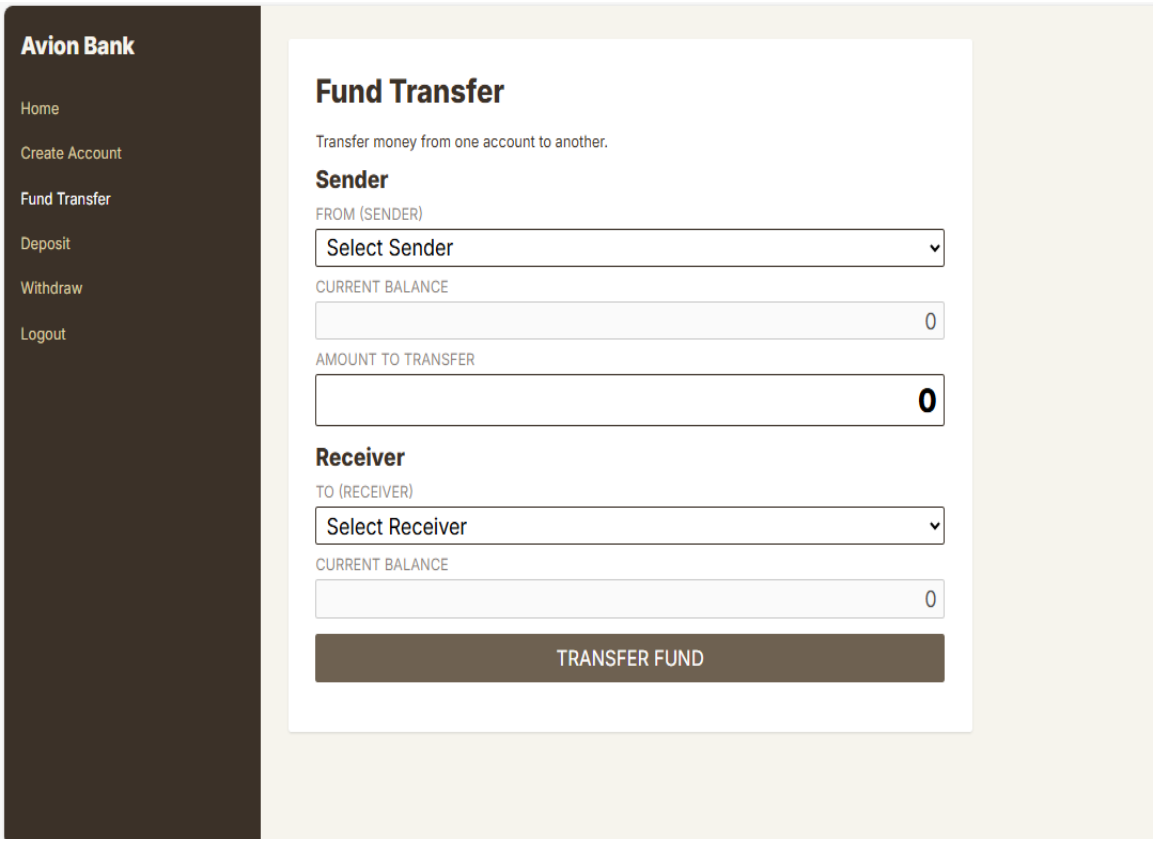
Screenshot 2: Home Page



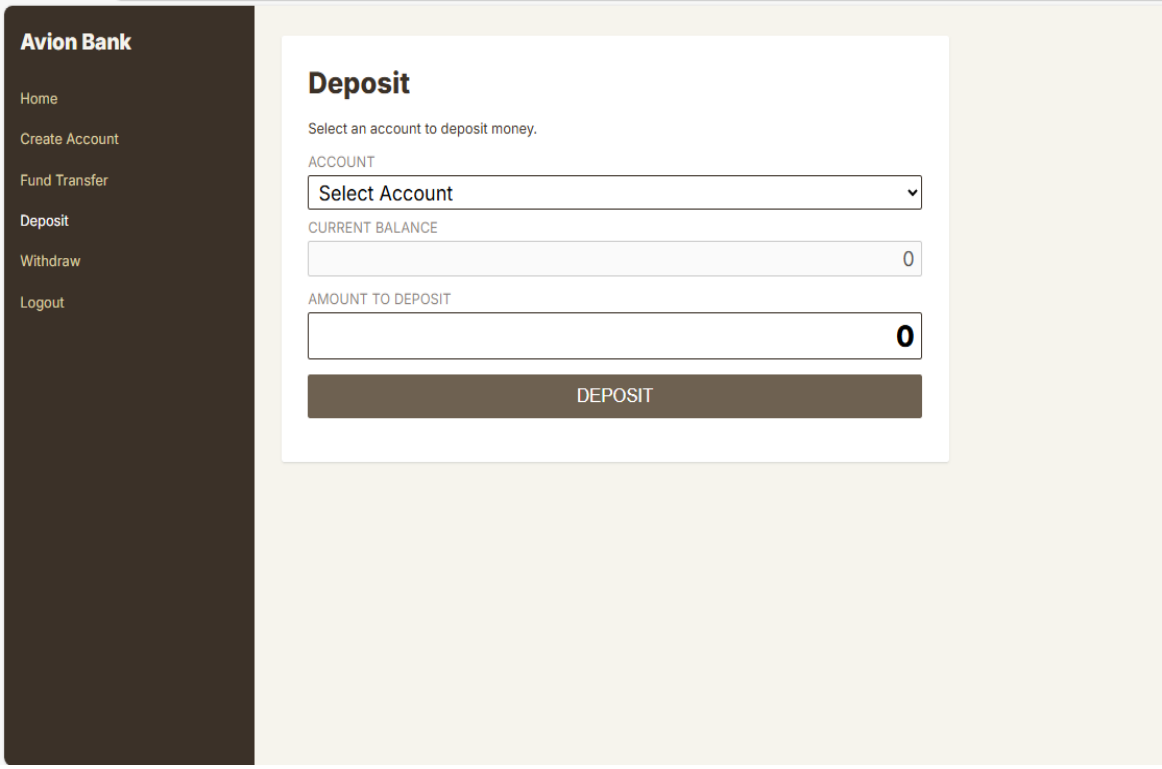
Screenshot 3: Create Account



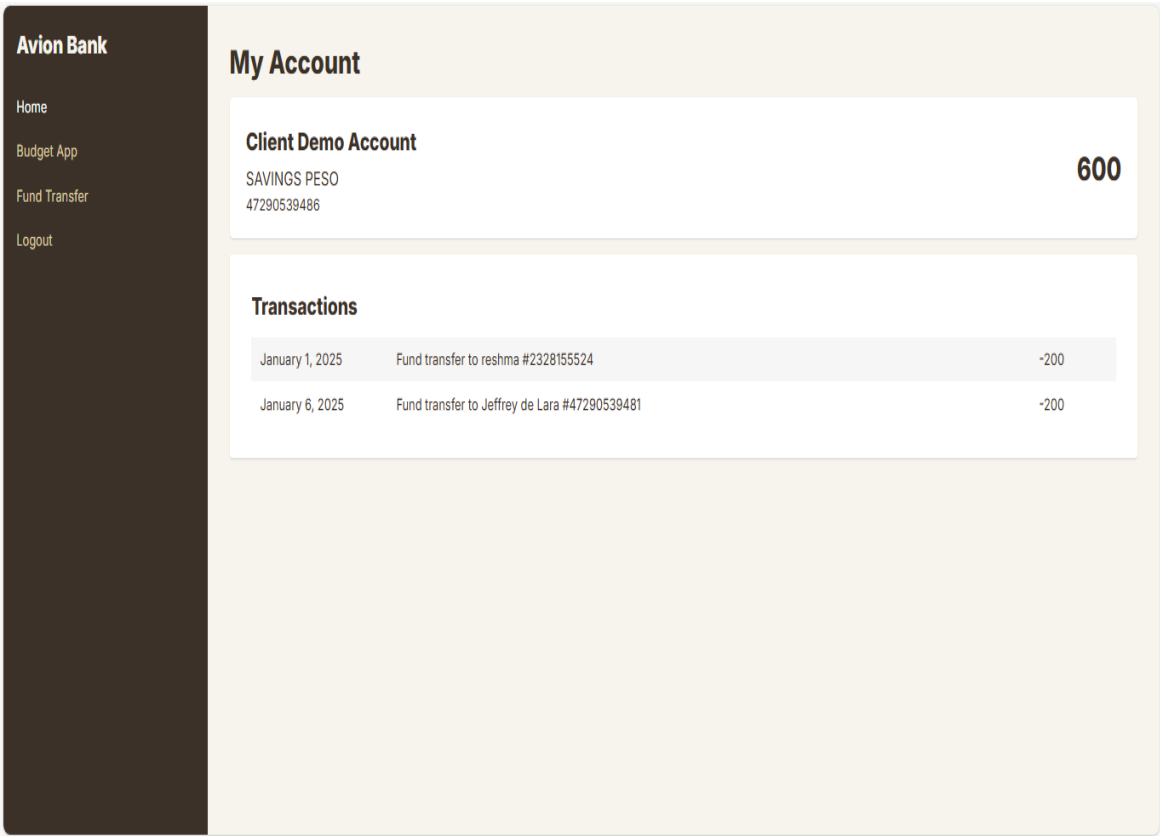
Screenshot 4: Funds Transfer



Screenshot 5: Deposit



Screenshot 6: My Account



Server.js:

```
const express = require('express');
const mysql = require('mysql2');
const cors = require('cors');
const app = express();
const port = 5000;

// Middleware
app.use(cors());
app.use(express.json());

// MySQL connection
const db = mysql.createConnection({
  host: 'localhost',
  user: 'root', // Use your MySQL username
  password: "123456", // Use your MySQL password
  database: 'Jashwanth', // Database name
});

// Connect to MySQL
db.connect((err) => {
  if (err) {
    console.error('Error connecting to MySQL database:', err);
    return;
  }
  console.log('Connected to MySQL database');
});

// Routes
```

```
// Get all users
app.get('/api/users/', (req, res) => {
  db.query('SELECT * FROM users', (err, results) => {
    if (err) {
      return res.status(500).json({ error: err.message });
    }
    res.json(results);
  });
});

// Get a user by ID
app.get('/api/users/:id', (req, res) => {
  const { id } = req.params;
  db.query('SELECT * FROM users WHERE id = ?', [id], (err, results) => {
    if (err) {
      return res.status(500).json({ error: err.message });
    }
    if (results.length === 0) {
      return res.status(404).json({ message: 'User not found' });
    }
    res.json(results[0]);
  });
});

// Insert a new user
app.post('/api/users', (req, res) => {
  const { name, email } = req.body;
```

```
  const query = 'INSERT INTO users (name, email) VALUES (?, ?)';
  db.query(query, [name, email], (err, results) => {
    if (err) {
      return res.status(500).json({ error: err.message });
    }
    res.status(201).json({ id: results.insertId, name, email });
  });
});

// Update an existing user
app.put('/api/users/:id', (req, res) => {
  const { id } = req.params;
  const { name, email } = req.body;
  const query = 'UPDATE users SET name = ?, email = ? WHERE id = ?';
  db.query(query, [name, email, id], (err, results) => {
    if (err) {
      return res.status(500).json({ error: err.message });
    }
    if (results.affectedRows === 0) {
      return res.status(404).json({ message: 'User not found' });
    }
    res.json({ id, name, email });
  });
});

// Delete a user
app.delete('/api/users/:id', (req, res) => {
  const { id } = req.params;
  const query = 'DELETE FROM users WHERE id = ?';
```

```
  db.query(query, [id], (err, results) => {
    if (err) {
      return res.status(500).json({ error: err.message });
    }
    if (results.affectedRows === 0) {
      return res.status(404).json({ message: 'User not found' });
    }
    res.status(204).send();
  });
});

// Start server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

Database.js:

```
const mysql = require('mysql2');

// Create a connection pool (recommended for production environments)
const pool = mysql.createPool({
  host: process.env.DB_HOST,        // Database host
  user: process.env.DB_USER,        // Database username
  password: process.env.DB_PASSWORD, // Database password
  database: process.env.DB_NAME     // Database name
});

// Create a promise wrapper for the pool to use async/await
const promisePool = pool.promise();

module.exports = promisePool;
```

Frontend codes:

Login.js:

```
import React, { useState } from 'react';
import { Logo } from './Logo';
import { Notif } from './Notif';

export const LoginPage = (props) => {
   const [username, setUsername] = useState('');
   const [password, setPassword] = useState('');
   const onSubmitHandler = (event) => {
     event.preventDefault();
     props.loginHandler(username, password);
   }

   const onChangeUsername = (event) => {
     setUsername(event.target.value);
   }

   const onChangePassword = (event) => {
     setPassword(event.target.value);
   }

   return (
     <div id="login-page">
<div id="login">
        <Logo />
        <Notif message={props.notif.message} style={props.notif.style} />
        <form onSubmit={onSubmitHandler}>
          <label htmlFor="username">Username</label>
          <input id="username" autoComplete="off" onChange={onChangeUsername}
value={username} type="text" />
          <label htmlFor="password">Password</label>
          <input id="password" autoComplete="off" onChange={onChangePassword}
value={password} type="password" />
          <button type="submit" className="btn">Login</button>
        </form>
      </div>
     </div>
   )
}
```

CreateAccountPage.js:-

```
import { useState } from "react";
import { Notif } from "./Notif";
import {formatNumber, trim} from './Utils';

export const CreateAccountPage = (props) => {
    const createRandomAccount = () => {
        return Math.floor(1000000000 + Math.random() * 9000000000);
    }

    const [notif, setNotif] = useState({message: 'Create a new client account.', style: 'left'});
    const [initialBalance, setInitialBalance] = useState(0);
    const [initialAccountNumber, setInitialAccountNumber] = useState(createRandomAccount());

     const createNewAccount = (user) => {

        const emptyInputs = Object.values(user).filter(input => {
            return input === ''
        });

        const localUsers = props.users;

        let alreadyExists = false;
        localUsers.forEach(row => {
            if(row.email === user.email) {
                alreadyExists = true;
            }
        });

        if(alreadyExists) {
            setNotif({message: 'This email already exists. Try again.', style: 'danger'});
            return false;
        } else if(emptyInputs.length > 0) {
            setNotif({message: 'All fields are required.', style: 'danger'});
            return false;
           } else {
            setNotif('');
            localUsers.unshift(user);
            props.setUsers(localUsers);
            localStorage.setItem('users', JSON.stringify(localUsers));
            setNotif({message: 'Successfully saved.', style: 'success'});
            return true;
           }
    }
```

```
 const handleCreateAccount = (event) => {
   event.preventDefault();
   const user = event.target.elements;

   const account = {
      email: user.email.value,
      password: user.password.value,
      fullname: user.fullname.value,
      type: user.accountType.value,
      number: user.accountNumber.value,
      isAdmin: false,
      balance: trim(user.initialBalance.value),
      transactions: []
   }

   const isSaved = createNewAccount(account);
   if(isSaved) {
         user.email.value = '';
      user.password.value = '';
      user.fullname.value = '';
      user.accountNumber.value = setInitialAccountNumber(createRandomAccount());
      user.initialBalance.value = setInitialBalance(0);
   }
}

const onInitialBalance = event => {
   const amount = trim(event.target.value) || 0;
   setInitialBalance(amount);
}

return (
   <section id="main-content">
      <form id="form" onSubmit={handleCreateAccount}>
         <h1>Create Account</h1>
            <Notif message={notif.message} style={notif.style} />
         <label htmlFor="fullname">Full name</label>
         <input id="fullname" type="text" autoComplete="off" name="fullname" />
         <hr />
         <label htmlFor="account-number">Account # (Randomly Generated)</label>
            <input id="account-number" name="accountNumber" className="right"
value={initialAccountNumber} type="number" disabled />

         <label htmlFor="balance">Initial balance</label>
```

```
            <input id="balance" type="text" value={formatNumber(initialBalance)}
onChange={onInitialBalance} name="initialBalance" className="right" />


                <label htmlFor="account-type">Account Type</label>
            <select name="accountType">
               <option value="Checking Account">Checking Account</option>
               <option value="Savings Accounts">Savings Account</option>
            </select>
            <hr />
            <label htmlFor="email">Email Address</label>
            <input id="email" type="email" name="email" />
            <label htmlFor="password">Password</label>
            <input id="password" type="password" name="password" />
            <input value="Create Account" className="btn" type="submit" />
         </form>
      </section>
   )
}
```

Account.js ;

```
import React from "react";
import { ActionButtons } from "./ActionButtons";
import { formatNumber } from "./Utils";

export const Account = (props) => {

   const {type, accountNumber, balance, fullname, editingUser, setEditingUser, setDeleteUser,
index, isAdmin, setEditModal} = props;

   const action = isAdmin ? <ActionButtons index={index}
     editingUser={editingUser}
     setEditingUser={setEditingUser}
     setEditModal={setEditModal} setDeleteUser={setDeleteUser} /> : '';

   return (
     <div className="account">
        <div className="details">
           <AccountHolder fullname={fullname} />
           <AccountType type={type} />
           <AccountNumber accountNumber={accountNumber} />
           {action}
        </div>
        <AccountBalance balance={formatNumber(balance)} />
     </div>
   )
  }
export const AccountHolder = (props) => {
   return (
```

```
      <h1>{props.fullname}</h1>
    )
  }
export const AccountType = (props) => {
    return (
      <h3>{props.type}</h3>
    )
  }
export const AccountNumber = (props) => {
    return (
      <div>{props.accountNumber}</div>
    )
  }
 export const AccountBalance = (props) => {
    const balance = props.balance;
    return (
      <div className="balance">{balance}</div>
    )
  }
```

ActionButton.JS:

```
import React from "react";
export const ActionButtons = (props) => {
    const {editingUser, setEditingUser, index, setEditModal, setDeleteUser} = props;

    return (
      <div id="actions">
        <ActionButton
          icon="bx bx-edit"
          text="Edit"
          index={index}
          actionType="edit"
          editingUser={editingUser}
          actionType='edit'
          setEditingUser={setEditingUser} setEditModal={setEditModal} />

        <ActionButton
          icon="bx bxs-x-square"
          index={index}
          actionType='delete'
          text="Delete" editingUser={editingUser}
          setDeleteUser={setDeleteUser} />
      </div>
    )
  }

export const ActionButton = (props) => {
    const {icon, text, editingUser, actionType, setEditingUser, index, setEditModal, setDeleteUser} =
props;
```

```
__const click = (e, index) => {
     e.preventDefault();

     if(actionType === 'edit') {
       setEditingUser(index);
       setEditModal(true);
     }

     if(actionType === 'delete') {
       setDeleteUser(index);
     }
   }

   return (
     <button onClick={(e) => click(e, index)}><i className={icon} ></i> {text}</button>
   )
 }
```

Crud.js

```
import React, { useState, useEffect } from "react";
import axios from "axios";
import './Crud.css';

const App = () => {
  const [data, setData] = useState([]);
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [modalMode, setModalMode] = useState("Insert"); // Insert, Read, Update
  const [formData, setFormData] = useState({ id: "", name: "", email: "" });
  const [currentId, setCurrentId] = useState(null);

  useEffect(() => {
    // Fetch all users from the backend
    axios.get('http://localhost:5000/api/users')
      .then(response => {
        setData(response.data);
      })
      .catch(error => {
        console.error('There was an error fetching the data!', error);
      });
  }, []);

  const openModal = (mode, id = null) => {
    setModalMode(mode);
    setIsModalOpen(true);
    if (mode === "Update" || mode === "Read") {
      const selectedRow = data.find((item) => item.id === id);
      setFormData({ ...selectedRow });
      setCurrentId(id);
    } else {
```

```
    setFormData({ id: "", name: "", email: "" });
  }
};

const closeModal = () => {
  setIsModalOpen(false);
  setFormData({ id: "", name: "", email: "" });
  setCurrentId(null);
};

const handleInputChange = (e) => {
  const { name, value } = e.target;
  setFormData({ ...formData, [name]: value });
};

const handleSubmit = () => {
  if (modalMode === "Insert") {
    axios.post('http://localhost:5000/api/users', formData)
      .then(response => {
        setData([...data, response.data]);
        closeModal();
      })
      .catch(error => {
        console.error('There was an error submitting the data!', error);
      });
  } else if (modalMode === "Update") {
    axios.put(`http://localhost:5000/api/users/${currentId}`, formData)
      .then(response => {
        setData(data.map(item => (item.id === currentId ? response.data : item)));
        closeModal();
      })
      .catch(error => {
        console.error('There was an error updating the data!', error);
      });
  }
};

const handleDelete = (id) => {
  axios.delete(`http://localhost:5000/api/users/${id}`)
    .then(() => {
      setData(data.filter(item => item.id !== id));
    })
    .catch(error => {
      console.error('There was an error deleting the data!', error);
    });
};

return (
  <div className="container">
    <button className="insert-button" onClick={() => openModal("Insert")}>
      Insert
```

```
</button>

{isModalOpen && (
  <div className="modal">
    <div className="modal-content">
      <h3>{modalMode} Data</h3>
      {(modalMode === "Insert" || modalMode === "Update") && (
        <form>
          <div>
            <label>Name:</label>
            <input
              type="text"
              name="name"
              value={formData.name}
              onChange={handleInputChange}
            />
          </div>
          <div>
            <label>Email:</label>
            <input
              type="email"
              name="email"
              value={formData.email}
              onChange={handleInputChange}
            />
          </div>
        </form>
      )}
      {modalMode === "Read" && (
        <div>
          <p>
            <strong>Name:</strong> {formData.name}
          </p>
          <p>
            <strong>Email:</strong> {formData.email}
          </p>
        </div>
      )}
      <div>
        <button className="close-btn" onClick={closeModal}>
          Close
        </button> 
        {(modalMode === "Insert" || modalMode === "Update") && (
          <button2 onClick={handleSubmit}>Submit</button2>
        )}
      </div>
    </div>
  </div>
)}
<table>
  <thead>
```

```
          <tr>
            <th>ID</th>
            <th>Name</th>
            <th>Email</th>
            <th>Actions</th>
          </tr>
        </thead>
        <tbody>
          {data.map((item) => (
            <tr key={item.id}>
              <td>{item.id}</td>
              <td>{item.name}</td>
              <td>{item.email}</td>
              <td className="actions">
                <button className="read-btn" onClick={() => openModal("Read", item.id)}>Read</button>
                <button className="update-btn" onClick={() => openModal("Update",
item.id)}>Update</button>
                <button className="delete-btn" onClick={() => handleDelete(item.id)}>Delete</button>
              </td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
};
export default App;
```

Conclusion:-The main conclusion of online banking is that it offers significant convenience, accessibility, and efficiency for both individuals and businesses. It allows users to perform various banking tasks—such as checking account balances, transferring funds, and paying bills—anytime and anywhere with an internet connection. However, it also comes with security risks, making it essential for users to adopt strong cybersecurity practices. Ultimately, online banking has revolutionized the financial industry, providing a faster, more user-friendly alternative to traditional banking methods.

Thanking You