LABORATORY REPORT

Application Development Lab
(CS33002)

B.Tech Program in ECSc

Submitted By

Name:- SAHIL KUMAR

Roll No: 2230189

Kalinga Institute of Industrial Technology
(Deemed to be University)
Bhubaneswar, India

Spring 2024-2025

Table of Content

Experiment Number	6
Experiment Title	Database Management Using Flask
Date of Experiment	19|02|2025
Date of Submission	17|03|2025

1. Objective:-

- To develop an application for user authentication and document sharing.

2. Procedure:- (Steps Followed)

- Install MySQL workbench in your system and install flask-mysqldb package.
- Create a database where you wish to store your user name and the password. Implement user authentication/registration form using Flask and the database. For a new user the account is created using the ‘signup’ button. Existing users can directly login with their credentials.
- Inside the users can update their personal details, reset their passwords.
- Inside the users can see the grades for their marks, which they cannot edit personally
- Build a responsive frontend for user interactions.

3. Code:-

Index.html file:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
            <meta   name="viewport"   content="width=device-width,
initial-scale=1.0" />
    <title>Student Grading System - Login</title>
    <link rel="stylesheet" href="styles/styles.css" />
  </head>
  <body>
    <div class="container">
      <!-- Sign Up Form -->
```

```
      <div class="form-container" id="signupForm">
        <h2>Create Account</h2>
        <form>
          <div class="input-group">
            <input
              type="text"
              name="username"
              placeholder="Username"
              required
            />
          </div>
          <div class="input-group">
            <input
              type="text"
              name="fullname"
              placeholder="Full Name"
              required
            />
          </div>
          <div class="input-group">
              <input type="email" name="email" placeholder="Email"
required />
          </div>
          <div class="input-group">
            <input
              type="password"
              name="password"
              placeholder="Password"
              required
            />
          </div>
                    <button  type="submit"  class="submit-btn">Sign
Up</button>
        </form>
        <p class="toggle-text">
                        Already  have  an  account?  <a  href="#"
id="showSignIn">Sign In</a>
        </p>
      </div>

      <!-- Sign In Form -->
      <div class="form-container hidden" id="signinForm">
        <h2>Welcome Back</h2>
```

```
        <form>
          <div class="input-group">
            <input
              type="text"
              name="username"
              placeholder="Username"
              required
            />
          </div>
          <div class="input-group">
            <input
              type="password"
              name="password"
              placeholder="Password"
              required
            />
          </div>
                    <button  type="submit"  class="submit-btn">Sign
In</button>
        </form>
        <p class="toggle-text">
           Don't have an account? <a href="#" id="showSignUp">Sign
Up</a>
        </p>
      </div>
    </div>
    <script src="styles/script.js"></script>
  </body>
</html>
```

Styles.css file:

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
     font-family: -apple-system, BlinkMacSystemFont, "Segoe UI",
Roboto, "Helvetica Neue", Arial, sans-serif;
}
```

```
body {
    min-height: 100vh;
                                                    background-image:
url('https://images.unsplash.com/photo-1456513080510-7bf3a84b82f8
?q=80&w=2073&auto=format&fit=crop');
    background-size: cover;
    background-position: center;
    background-repeat: no-repeat;
    display: flex;
    align-items: center;
    justify-content: center;
}

.container {
    position: relative;
    width: 100%;
    max-width: 400px;
    padding: 2rem;
}

.container::before {
    content: '';
    position: fixed;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%;
    background: rgba(0, 0, 0, 0.6);
    z-index: 1;
}

.form-container {
    position: relative;
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(10px);
    padding: 2rem;
    border-radius: 1rem;
    box-shadow: 0 8px 32px rgba(0, 0, 0, 0.1);
    z-index: 2;
    opacity: 0;
    animation: fadeIn 0.3s ease forwards;
}
```

```
@keyframes fadeIn {
    to {
        opacity: 1;
    }
}

.form-container.hidden {
    display: none;
}

h2 {
    color: white;
    text-align: center;
    margin-bottom: 2rem;
    font-size: 1.875rem;
}

.input-group {
    margin-bottom: 1.5rem;
}

input, textarea {
    width: 100%;
    padding: 0.75rem 1rem;
    background: rgba(255, 255, 255, 0.2);
    border: 2px solid transparent;
    border-radius: 0.5rem;
    color: white;
    font-size: 1rem;
    transition: all 0.2s ease;
}

textarea {
    resize: vertical;
    min-height: 60px;
}

input::placeholder, textarea::placeholder {
    color: rgba(255, 255, 255, 0.7);
}

input:focus, textarea:focus {
```

```
    outline: none;
    border-color: rgba(59, 130, 246, 0.5);
    background: rgba(255, 255, 255, 0.25);
}

.submit-btn {
    width: 100%;
    padding: 0.75rem;
    background: #2563eb;
    color: white;
    border: none;
    border-radius: 0.5rem;
    font-size: 1rem;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.2s ease;
}

.submit-btn:hover {
    background: #1d4ed8;
}

.submit-btn:active {
    transform: scale(0.98);
}

.toggle-text {
    margin-top: 1.5rem;
    text-align: center;
    color: white;
}

.toggle-text a {
    color: #60a5fa;
    text-decoration: none;
    font-weight: 600;
    transition: color 0.2s ease;
}

.toggle-text a:hover {
    color: #93c5fd;
}
```

script.js file

```
document.addEventListener("DOMContentLoaded", () => {
  // Form elements
  const signupForm = document.querySelector("#signupForm form");
  const signinForm = document.querySelector("#signinForm form");
  const showSignIn = document.getElementById("showSignIn");
  const showSignUp = document.getElementById("showSignUp");

  // Toggle between forms
  const toggleForms = () => {

document.getElementById("signupForm").classList.toggle("hidden");

document.getElementById("signinForm").classList.toggle("hidden");
  };

  if (showSignIn) {
    showSignIn.addEventListener("click", (e) => {
      e.preventDefault();
      toggleForms();
    });
  }

  if (showSignUp) {
    showSignUp.addEventListener("click", (e) => {
      e.preventDefault();
      toggleForms();
    });
  }

  // Function to handle form submission
  const handleFormSubmit = async (endpoint, data) => {
    try {
      const response = await fetch(endpoint, {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(data),
      });
```

```
      const result = await response.json();

      if (result.success) {
        if (result.redirect) {
          window.location.href = result.redirect;
        } else {
          alert(result.message || "Operation successful!");
        }
      } else {
          alert(result.message || "An error occurred. Please try
again.");
      }
    } catch (error) {
      console.error("Error:", error);
      alert("An error occurred. Please try again.");
    }
  };

  // Signup form submission
  if (signupForm) {
    signupForm.addEventListener("submit", (e) => {
      e.preventDefault();

      const inputs = signupForm.querySelectorAll("input");
      const username = inputs[0].value.trim();
      const fullname = inputs[1].value.trim();
      const email = inputs[2].value.trim();
      const password = inputs[3].value.trim();

      if (!username || !fullname || !email || !password) {
        alert("Please fill in all fields");
        return;
      }

        handleFormSubmit("/signup", { username, fullname, email,
password });
    });
  }

  // Signin form submission
  if (signinForm) {
    signinForm.addEventListener("submit", (e) => {
      e.preventDefault();
```

```
                                        const       username       =
signinForm.querySelector('input[type="text"]').value.trim();
                                        const       password       =
signinForm.querySelector('input[type="password"]').value.trim();

      if (!username || !password) {
        alert("Please fill in all fields");
        return;
      }

      handleFormSubmit("/login", { username, password });
    });
  }

  // Navbar toggle for mobile view
  const hamburger = document.querySelector(".hamburger");
  const navLinks = document.querySelector(".nav-links");

  if (hamburger && navLinks) {
    hamburger.addEventListener("click", () => {
      navLinks.classList.toggle("show");
    });
  }
});
```

app.py:-

```
from  flask  import  Flask,  render_template,  request,  redirect,
session, jsonify
from flask_mysqldb import MySQL
import MySQLdb.cursors
import os
import bcrypt
from dotenv import load_dotenv


load_dotenv('.env.local')
```

```
app = Flask(__name__, static_folder='styles',
template_folder='.')
app.secret_key = os.getenv("SECRET_KEY", "default_secret_key")


app.config['MYSQL_HOST'] = os.getenv("MYSQL_HOST", "localhost")
app.config['MYSQL_PORT'] = int(os.getenv("MYSQL_PORT", 3306))
app.config['MYSQL_USER'] = os.getenv("MYSQL_USER", "root")
app.config['MYSQL_PASSWORD'] = os.getenv("MYSQL_PASSWORD", "")
app.config['MYSQL_DB'] = os.getenv("MYSQL_DB", "student_db")

mysql = MySQL(app)


@app.route('/')
def index():
    return render_template('index.html')


@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()
    username = data.get('username')
    password = data.get('password')

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      cursor.execute('SELECT * FROM users WHERE username = %s',
(username,))
    user = cursor.fetchone()

         if user and bcrypt.checkpw(password.encode('utf-8'),
user['password'].encode('utf-8')):
        session['loggedin'] = True
        session['id'] = user['id']
        session['username'] = user['username']

        return jsonify({'success': True, 'redirect': '/grades'})

       return jsonify({'success': False, 'message': 'Incorrect
username or password'})


@app.route('/signup', methods=['POST'])
```

```
def signup():
    data = request.get_json()
    username = data.get('username')
    fullname = data.get('fullname')
    email = data.get('email')
    password = data.get('password')

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
      cursor.execute('SELECT * FROM users WHERE username = %s',
(username,))
    account = cursor.fetchone()

    if account:
            return jsonify({'success': False, 'message': 'Account
already exists!'})

    if not fullname or not email or not username or not password:
         return jsonify({'success': False, 'message': 'Please fill
in all fields!'})

       hashed_password = bcrypt.hashpw(password.encode('utf-8'),
bcrypt.gensalt())

     cursor.execute('INSERT INTO users (username, fullname, email,
password) VALUES (%s, %s, %s, %s)',
                    (username, fullname, email, hashed_password))
    mysql.connection.commit()

    return jsonify({'success': True, 'redirect': '/'})


@app.route('/profile', methods=['GET', 'POST'])
def profile():
    if 'loggedin' not in session:
        return redirect('/')

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

    if request.method == 'POST':
        data = request.get_json()
        fullname = data.get('fullname')
        email = data.get('email')
        phone = data.get('phone')
```

```
        address = data.get('address')
        bio = data.get('bio')

        cursor.execute('''
            UPDATE users
            SET fullname = %s, email = %s
            WHERE id = %s
        ''', (fullname, email, session['id']))

        cursor.execute(
               "SELECT id FROM user_details WHERE user_id = %s",
(session['id'],))
        existing_entry = cursor.fetchone()

        if existing_entry:

            cursor.execute('''
                UPDATE user_details
                SET phone = %s, address = %s, bio = %s
                WHERE user_id = %s
            ''', (phone, address, bio, session['id']))
        else:

            cursor.execute('''
                        INSERT INTO user_details (user_id, phone,
address, bio)
                VALUES (%s, %s, %s, %s)
            ''', (session['id'], phone, address, bio))

        mysql.connection.commit()

        cursor.execute('''
                SELECT u.username, u.fullname, u.email, d.phone,
d.address, d.bio
            FROM users u
            LEFT JOIN user_details d ON u.id = d.user_id
            WHERE u.id = %s
        ''', (session['id'],))
        user = cursor.fetchone()

        return jsonify(user)

    cursor.execute('''
```

```
            SELECT u.username, u.fullname, u.email, d.phone,
d.address, d.bio
        FROM users u
        LEFT JOIN user_details d ON u.id = d.user_id
        WHERE u.id = %s
    ''', (session['id'],))
    user = cursor.fetchone()

            if    request.headers.get('X-Requested-With')    ==
'XMLHttpRequest':
        return jsonify(user)

    return render_template('profile.html', user=user)


@app.route('/grades')
def grades():
    if 'loggedin' not in session:
        return redirect('/')

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute('''
          SELECT c.course_code, c.course_name, g.grade, g.marks,
g.semester
        FROM grades g
        JOIN courses c ON g.course_id = c.id
        WHERE g.user_id = %s
    ''', (session['id'],))
    grades = cursor.fetchall()

            if    request.headers.get('X-Requested-With')    ==
'XMLHttpRequest':
        return jsonify(grades)

    return render_template('grades.html')


@app.route('/reset_password', methods=['POST'])
def reset_password():
    if 'loggedin' not in session:
          return jsonify({'success': False, 'message': 'User not
logged in'})
```

```
    data = request.get_json()
    current_password = data.get('current_password')
    new_password = data.get('new_password')

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)

    cursor.execute('SELECT password FROM users WHERE id = %s',
                   (session['id'],))
    user = cursor.fetchone()

                           if      not      user      or      not
bcrypt.checkpw(current_password.encode('utf-8'),
user['password'].encode('utf-8')):
           return jsonify({'success': False, 'message': 'Current
password is incorrect'})

    hashed_new_password = bcrypt.hashpw(
        new_password.encode('utf-8'), bcrypt.gensalt())

      cursor.execute('UPDATE users SET password = %s WHERE id =
%s',
                   (hashed_new_password, session['id']))
    mysql.connection.commit()

     return jsonify({'success': True, 'message': 'Password updated
successfully'})


@app.route('/logout')
def logout():
    session.clear()
    return redirect('/')


if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0', port=5000)
```

profile.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
            <meta   name="viewport"   content="width=device-width,
initial-scale=1.0" />
    <title>Student Profile - Grading System</title>
    <link rel="stylesheet" href="styles/styles.css" />
    <link rel="stylesheet" href="styles/dashboard.css" />
  </head>
  <body>
    <nav class="navbar">
      <div class="nav-content">
        <h1>Student Dashboard</h1>
        <button class="hamburger">
          <span></span>
          <span></span>
          <span></span>
        </button>
        <div class="nav-links">
          <a href="/grades">Grades</a>
          <a href="/profile" class="active">Profile</a>
          <a href="/">Logout</a>
        </div>
      </div>
    </nav>

    <div class="dashboard-container">
      <div class="profile-card">
        <h2>Edit Profile</h2>
        <form id="profileForm">
          <div class="input-group">
            <label>Profile Picture</label>
            <div class="profile-picture">
              <img

src="https://images.unsplash.com/photo-1633332755192-727a05c4013d
?w=150&h=150&fit=crop"
                alt="Profile"
              />
```

```
            </div>
          </div>
          <div class="input-group">
            <label>Username</label>
                <input type="text" id="username" name="username"
readonly />
          </div>
          <div class="input-group">
            <label>Full Name</label>
                <input type="text" id="fullname" name="fullname"
required />
          </div>
          <div class="input-group">
            <label>Email</label>
             <input type="email" id="email" name="email" required
/>
          </div>
          <div class="input-group">
            <label>Phone</label>
            <input type="tel" id="phone" name="phone" />
          </div>
          <div class="input-group">
            <label>Address</label>
                          <textarea  id="address"  name="address"
rows="3"></textarea>
          </div>
          <div class="input-group">
            <label>Bio</label>
            <textarea id="bio" name="bio" rows="4"></textarea>
          </div>
          <div class="input-group">
            <label>Current Password</label>
            <input
              type="password"
              name="currentPassword"
              placeholder="Enter current password"
            />
          </div>
          <div class="input-group">
            <label>New Password</label>
            <input
              type="password"
              name="newPassword"
```

```
              placeholder="Enter new password"
            />
          </div>
                  <button  type="submit"  class="submit-btn">Save
Changes</button>
        </form>
      </div>
    </div>

    <script src="styles/navbar.js"></script>
    <script>
      document.addEventListener("DOMContentLoaded", async () => {
        async function fetchProfile() {
          try {
            const response = await fetch(`/profile`, {
              headers: { "X-Requested-With": "XMLHttpRequest" },
            });
            const data = await response.json();

            if (data.username) {
                      document.getElementById("username").value =
data.username;
                      document.getElementById("fullname").value =
data.fullname;
                        document.getElementById("email").value =
data.email;
               document.getElementById("phone").value = data.phone
|| "";
                       document.getElementById("address").value =
data.address || "";
                document.getElementById("bio").value = data.bio ||
"";
            }
          } catch (error) {
            console.error("Error fetching profile:", error);
          }
        }

        await fetchProfile();

                                      const     profileForm     =
document.getElementById("profileForm");
```

```
        if (profileForm) {
          profileForm.addEventListener("submit", async (e) => {
            e.preventDefault();

                                                const    fullname    =
document.getElementById("fullname").value.trim();
                                                  const    email    =
document.getElementById("email").value.trim();
                                                  const    phone    =
document.getElementById("phone").value.trim();
                                                const    address    =
document.getElementById("address").value.trim();
                                                    const    bio    =
document.getElementById("bio").value.trim();

            const currentPassword = profileForm
              .querySelector('input[name="currentPassword"]')
              .value.trim();
            const newPassword = profileForm
              .querySelector('input[name="newPassword"]')
              .value.trim();

            if (currentPassword || newPassword) {
                      await  handlePasswordChange(currentPassword,
newPassword);
              return;
            }

            if (!fullname || fullname.length > 100) {
               alert("Full Name is required and must be within 100
characters.");
              return;
            }

            if (
              !email ||
              email.length > 100 ||

!/^[\w.-]+@[a-zA-Z\d.-]+\.[a-zA-Z]{2,}$/.test(email)
            ) {
              alert("Enter a valid Email (max 100 characters).");
              return;
            }
```

```
            if (phone && !/^\d{10}$/.test(phone)) {
              alert("Phone number must be exactly 10 digits.");
              return;
            }

            if (address && address.length < 5) {
              alert("Address must be at least 5 characters.");
              return;
            }

            try {
              const response = await fetch("/profile", {
                method: "POST",
                headers: { "Content-Type": "application/json" },
                    body: JSON.stringify({ fullname, email, phone,
address, bio }),
              });

              if (response.ok) {
                alert("Profile updated successfully!");
                window.location.reload();
              } else {
                alert("Failed to update profile.");
              }
            } catch (error) {
              console.error("Error updating profile:", error);
                    alert("An error occurred while updating the
profile.");
            }
          });
        }
      });

            async function handlePasswordChange(currentPassword,
newPassword) {
        if (!currentPassword || !newPassword) {
          alert("Please enter both current and new passwords.");
          return;
        }

        try {
          const response = await fetch("/reset_password", {
```

```
            method: "POST",
            headers: { "Content-Type": "application/json" },
            body: JSON.stringify({
              current_password: currentPassword,
              new_password: newPassword,
            }),
          });

          const result = await response.json();
          alert(result.message);

          if (result.success) {
            window.location.href = "/profile";
          }
        } catch (error) {
          console.error("Error:", error);
                  alert("An error occurred while changing the
password.");
        }
      }
    </script>
  </body>
</html>
```

grades.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
            <meta   name="viewport"   content="width=device-width,
initial-scale=1.0" />
    <title>Student Grades - Grading System</title>
    <link rel="stylesheet" href="styles/styles.css" />
    <link rel="stylesheet" href="styles/dashboard.css" />
  </head>
  <body>
    <nav class="navbar">
      <div class="nav-content">
```

```
        <h1>Student Dashboard</h1>
        <button class="hamburger">
          <span></span>
          <span></span>
          <span></span>
        </button>
        <div class="nav-links">
          <a href="/grades" class="active">Grades</a>
          <a href="/profile">Profile</a>
          <a href="/">Logout</a>
        </div>
      </div>
    </nav>

    <div class="dashboard-container">
      <div class="grades-card">
        <h2>Current Semester Grades</h2>
        <div class="grades-table">
          <table>
            <thead>
              <tr>
                <th>Course Code</th>
                <th>Course Name</th>
                <th>Grade</th>
                <th>Marks</th>
                <th>Semester</th>
              </tr>
            </thead>
            <tbody id="gradesTableBody">
              <tr>
                <td colspan="5">Loading...</td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </div>

    <script>
      document.addEventListener("DOMContentLoaded", () => {
              fetch("/grades", { headers: { "X-Requested-With":
"XMLHttpRequest" } })
          .then((response) => response.json())
```

```
          .then((data) => {
                                                const    tableBody    =
document.getElementById("gradesTableBody");
            tableBody.innerHTML = "";

            if (data.length === 0) {
                    tableBody.innerHTML = `<tr><td colspan="5">No
grades available</td></tr>`;
              return;
            }

            data.forEach((grade) => {
              const row = `
                            <tr>
                                <td>${grade.course_code}</td>
                                <td>${grade.course_name}</td>
                                <td>${grade.grade}</td>
                                <td>${grade.marks}</td>
                                <td>${grade.semester}</td>
                            </tr>
                        `;
              tableBody.innerHTML += row;
            });
          })
          .catch((error) => {
            console.error("Error fetching grades:", error);
            document.getElementById(
              "gradesTableBody"
                ).innerHTML = `<tr><td colspan="5">Error loading
grades</td></tr>`;
          });
      });
    </script>
  </body>
</html>
```

navbar.js

```
document.addEventListener("DOMContentLoaded", () => {
  const hamburger = document.querySelector(".hamburger");
  const navLinks = document.querySelector(".nav-links");
```

```
  hamburger.addEventListener("click", () => {
    hamburger.classList.toggle("active");
    navLinks.classList.toggle("active");
  });

  // Close menu when clicking outside
  document.addEventListener("click", (e) => {
                    if      (!hamburger.contains(e.target)      &&
!navLinks.contains(e.target)) {
      hamburger.classList.remove("active");
      navLinks.classList.remove("active");
    }
  });

  // Close menu when clicking a link
  navLinks.querySelectorAll("a").forEach((link) => {
    link.addEventListener("click", () => {
      hamburger.classList.remove("active");
      navLinks.classList.remove("active");
    });
  });
});
```

dashboard.css

```
/* Navbar Styles */
.navbar {
    background: #1a1a1a;
    padding: 1rem 0;
    position: fixed;
    top: 0;
    left: 0;
    right: 0;
    z-index: 1000;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}

.nav-content {
    max-width: 1400px;
    margin: 0 auto;
```

```
    padding: 0 2rem;
    display: flex;
    justify-content: space-between;
    align-items: center;
}

.nav-content h1 {
    color: white;
    margin: 0;
    font-size: 1.5rem;
}

.nav-links {
    display: flex;
    gap: 2rem;
}

.nav-links a {
    color: white;
    text-decoration: none;
    font-weight: 500;
    padding: 0.5rem 1rem;
    border-radius: 0.5rem;
    transition: all 0.2s ease;
}

.nav-links a:hover {
    background: rgba(255, 255, 255, 0.1);
}

.nav-links a.active {
    background: #2563eb;
}

/* Hamburger Menu */
.hamburger {
    display: none;
    flex-direction: column;
    justify-content: space-around;
    width: 2rem;
    height: 2rem;
    background: transparent;
    border: none;
```

```
    cursor: pointer;
    padding: 0;
    z-index: 10;
}

.hamburger span {
    width: 2rem;
    height: 0.25rem;
    background: white;
    border-radius: 0.5rem;
    transition: all 0.3s ease-in-out;
}

/* Dashboard Container */
.dashboard-container {
    max-width: 1400px;
    margin: 6rem auto 2rem;
    padding: 0 2rem;
}

/* Profile Card */
.profile-card {
    background: white;
    padding: 3rem;
    border-radius: 1rem;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    max-width: 1000px;
    margin: 0 auto 2rem;
}

.profile-card form {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 2rem;
}

.profile-card .input-group:first-child {
    grid-column: 1 / -1;
}

.profile-picture {
    width: 150px;
    height: 150px;
```

```
    border-radius: 50%;
    overflow: hidden;
    margin: 1rem 0;
    border: 4px solid #f3f4f6;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.profile-picture img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

/* Form Labels */
label {
    display: block;
    color: #374151;
    margin-bottom: 0.5rem;
    font-weight: 500;
    font-size: 0.9rem;
}

/* Form Inputs */
.profile-card input, .profile-card textarea {
    background: #f9fafb;
    border: 1px solid #e5e7eb;
    color: #1f2937;
    font-size: 0.95rem;
    padding: 0.625rem 0.875rem;
}

.profile-card input {
    height: 2.75rem;
}

.profile-card textarea {
    min-height: 100px;
}

.profile-card input:focus, .profile-card textarea:focus {
    border-color: #2563eb;
    background: white;
}
```

```css
.profile-card input[readonly] {
    background: #f3f4f6;
    cursor: not-allowed;
}

.profile-card .submit-btn {
    grid-column: 1 / -1;
    max-width: 200px;
    justify-self: end;
    margin-top: 1rem;
}

/* Grades Card */
.grades-card {
    background: white;
    padding: 2rem;
    border-radius: 1rem;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
}

.semester-info {
    display: flex;
    justify-content: space-between;
    margin: 1rem 0 2rem;
    color: #1f2937;
    font-size: 1.1rem;
}

/* Grades Table */
.grades-table {
    overflow-x: auto;
    background: white;
    border-radius: 0.5rem;
    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);
    margin: 0 -1rem;
}

table {
    width: 100%;
    border-collapse: collapse;
    color: #374151;
    min-width: 800px;
```

```
}

th, td {
    padding: 1rem;
    text-align: left;
    border-bottom: 1px solid #e5e7eb;
}

th {
    background: #f9fafb;
    font-weight: 600;
    color: #1f2937;
}

tr:hover {
    background: #f9fafb;
}

/* Status Badges */
.status {
    padding: 0.25rem 0.75rem;
    border-radius: 1rem;
    font-size: 0.875rem;
    font-weight: 500;
    white-space: nowrap;
}

.status.completed {
    background: #dcfce7;
    color: #15803d;
}

.status.in-progress {
    background: #fef9c3;
    color: #854d0e;
}

/* Page Background */
body {
    background: #f3f4f6;
    min-height: 100vh;
}
```

```css
/* Headings */
h2 {
    color: #1f2937;
    margin-bottom: 2rem;
    font-size: 1.875rem;
    text-align: left;
}

/* Responsive Design */
@media (max-width: 768px) {
    .nav-content {
        padding: 0 1rem;
    }

    .hamburger {
        display: flex;
    }

    .nav-links {
        display: none;
        position: absolute;
        top: 100%;
        left: 0;
        right: 0;
        background: #1a1a1a;
        padding: 1rem;
        flex-direction: column;
        gap: 0.5rem;
        border-top: 1px solid rgba(255, 255, 255, 0.1);
    }

    .nav-links.active {
        display: flex;
    }

    .nav-links a {
        width: 100%;
        text-align: center;
        padding: 0.75rem;
    }

    .profile-card form {
        grid-template-columns: 1fr;
```

```
        gap: 1.5rem;
    }

    .profile-card .submit-btn {
        max-width: 100%;
    }

    .semester-info {
        flex-direction: column;
        gap: 0.5rem;
        text-align: center;
    }

    .grades-table {
        margin: 0 -1rem;
        padding: 0 1rem;
    }

    th, td {
        padding: 0.75rem;
    }

    .profile-card {
        padding: 1.5rem;
    }

    /* Hamburger Animation */
    .hamburger.active span:nth-child(1) {
        transform: translateY(0.75rem) rotate(45deg);
    }

    .hamburger.active span:nth-child(2) {
        opacity: 0;
    }

    .hamburger.active span:nth-child(3) {
        transform: translateY(-0.75rem) rotate(-45deg);
    }
}

@media (max-width: 400px) {
    .dashboard-container {
        padding: 0 1rem;
```

```
    }

    .grades-card {
        padding: 1.25rem;
        margin: 0 -0.5rem;
    }

    .grades-table {
        margin: 0 -0.75rem;
        padding: 0 0.75rem;
    }

    th, td {
        padding: 0.625rem;
        font-size: 0.875rem;
    }

    .status {
        padding: 0.25rem 0.5rem;
        font-size: 0.75rem;
    }
}
```

mysql file

```
CREATE DATABASE student_db;
USE student_db;

-- Users table
CREATE TABLE IF NOT EXISTS users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    username VARCHAR(50) NOT NULL UNIQUE,
    password VARCHAR(255) NOT NULL,
    email VARCHAR(100) NOT NULL UNIQUE,
    fullname VARCHAR(100) NOT NULL,
```

```
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
     updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON
UPDATE CURRENT_TIMESTAMP
);

-- User details table
CREATE TABLE IF NOT EXISTS user_details (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    phone VARCHAR(10),
    address TEXT,
    bio TEXT,
       FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE
CASCADE
);

-- Course table
CREATE TABLE IF NOT EXISTS courses (
    id INT AUTO_INCREMENT PRIMARY KEY,
    course_name VARCHAR(100) NOT NULL,
    course_code VARCHAR(20) NOT NULL UNIQUE,
    description TEXT
);

-- Grades table
CREATE TABLE IF NOT EXISTS grades (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT NOT NULL,
    course_id INT NOT NULL,
```

```
    grade CHAR(1) NOT NULL,
    marks DECIMAL(5,2) NOT NULL,
    semester VARCHAR(20),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
      FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
     FOREIGN KEY (course_id) REFERENCES courses(id) ON DELETE CASCADE
);

-- Insert sample users with hashed passwords ('password123' for all users)
INSERT INTO users (username, password, email, fullname) VALUES
('john_doe', '$2b$12$1oE8vBRHi.Wj7FHjHCB4LexZuV6qQjGRR1QRZGMXgQIOk8TZvLKcC', 'john.doe@example.com', 'John Doe'),
('jane_smith', '$2b$12$1oE8vBRHi.Wj7FHjHCB4LexZuV6qQjGRR1QRZGMXgQIOk8TZvLKcC', 'jane.smith@example.com', 'Jane Smith'),
('bob_johnson', '$2b$12$1oE8vBRHi.Wj7FHjHCB4LexZuV6qQjGRR1QRZGMXgQIOk8TZvLKcC', 'bob.johnson@example.com', 'Bob Johnson'),
('alice_chen', '$2b$12$1oE8vBRHi.Wj7FHjHCB4LexZuV6qQjGRR1QRZGMXgQIOk8TZvLKcC', 'alice.chen@example.com', 'Alice Chen'),
('david_brown', '$2b$12$1oE8vBRHi.Wj7FHjHCB4LexZuV6qQjGRR1QRZGMXgQIOk8TZvLKcC', 'david.brown@example.com', 'David Brown');

-- Insert user details
```

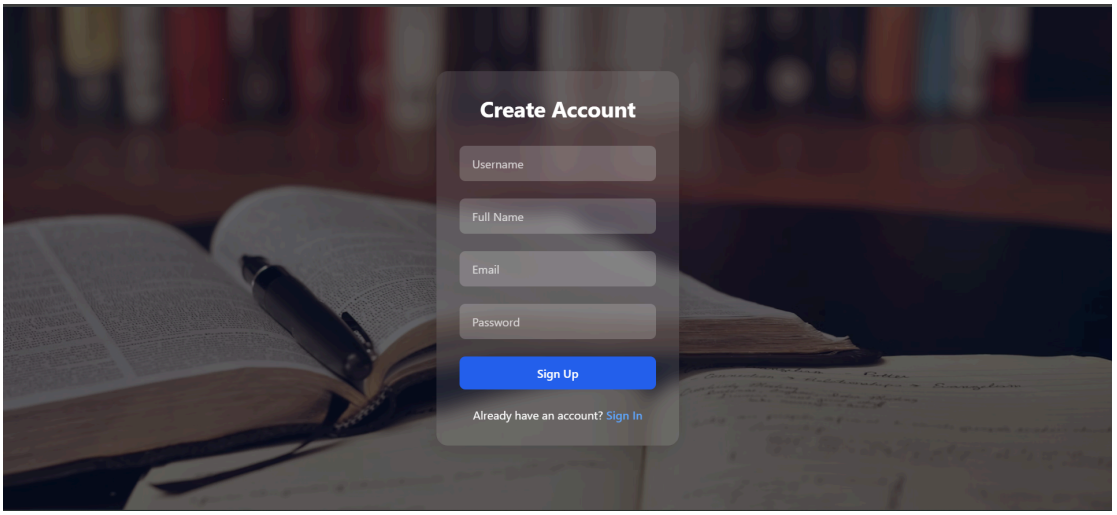
```
INSERT INTO user_details (user_id, phone, address, bio) VALUES
(1, '555-123-4567', '123 Main St, Anytown, CA 94001', 'Computer Science student with interest in AI and machine learning.'),
(2, '555-234-5678', '456 Oak Ave, Somewhere, NY 10001', 'Engineering student passionate about renewable energy solutions.'),
(3, '555-345-6789', '789 Pine Rd, Elsewhere, TX 75001', 'Business major with focus on entrepreneurship and startups.'),
(4, '555-456-7890', '321 Cedar Ln, Nowhere, WA 98001', 'Physics student researching quantum computing applications.'),
(5, '555-567-8901', '654 Maple Dr, Anywhere, FL 33001', 'Literature major with interest in creative writing and poetry.');

-- Insert courses
INSERT INTO courses (id, course_name, course_code, description) VALUES
(1, 'Database Management Systems', 'CS301', 'Introduction to database concepts, SQL, and relational database design.'),
(2, 'Web Development', 'CS302', 'Fundamentals of web development including HTML, CSS, JavaScript, and modern frameworks.'),
(3, 'Computer Networks', 'CS303', 'Understanding computer network architectures, protocols, and security concepts.'),
(4, 'Data Structures and Algorithms', 'CS201', 'Advanced data structures and algorithm analysis.'),
(5, 'Machine Learning', 'CS401', 'Introduction to machine learning concepts, algorithms, and applications.'),
(6, 'Operating Systems', 'CS304', 'Principles of operating systems design and implementation.'),
(7, 'Software Engineering', 'CS350', 'Software development methodologies, project management, and best practices.');
```

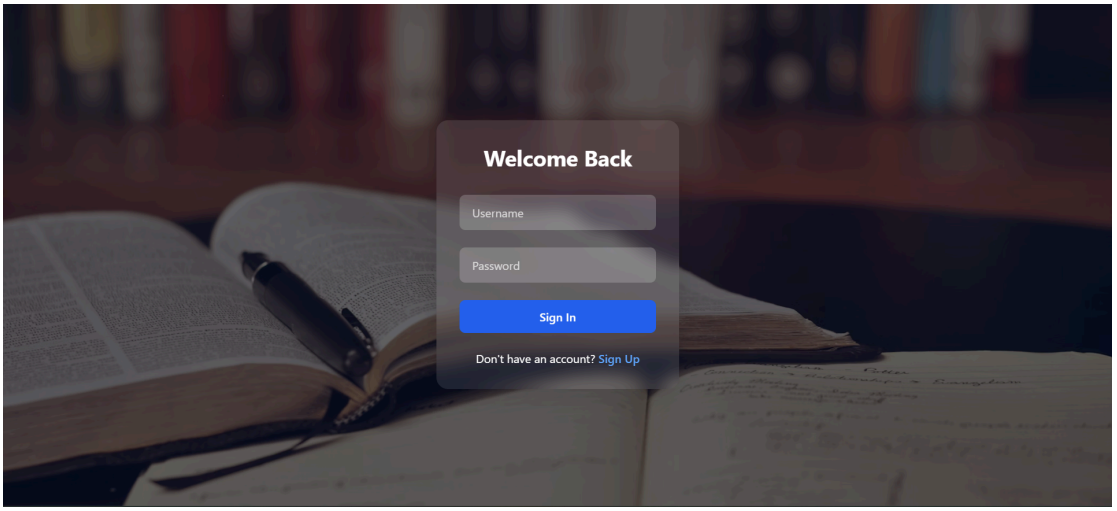
```
-- Insert grades for users
INSERT INTO grades (user_id, course_id, grade, marks, semester) VALUES
-- Sahil's grades
(1, 1, 'A', 92.5, 'Fall 2024'),
(1, 2, 'B', 85.0, 'Fall 2024'),
(1, 3, 'A', 90.0, 'Spring 2025'),
(1, 4, 'B', 86.5, 'Spring 2025'),

-- Jane Smith's grades
(2, 1, 'A', 95.0, 'Fall 2024'),
(2, 3, 'A', 93.5, 'Fall 2024'),
(2, 5, 'B', 88.0, 'Spring 2025'),

-- Bob Johnson's grades
(3, 2, 'C', 75.5, 'Fall 2024'),
(3, 4, 'B', 82.0, 'Fall 2024'),
(3, 6, 'A', 91.0, 'Spring 2025'),

-- Alice Chen's grades
(4, 1, 'A', 94.0, 'Fall 2024'),
(4, 5, 'A', 96.5, 'Fall 2024'),
(4, 7, 'A', 97.0, 'Spring 2025'),

-- David Brown's grades
(5, 2, 'B', 84.5, 'Fall 2024'),
(5, 3, 'C', 76.0, 'Fall 2024'),
(5, 7, 'B', 87.5, 'Spring 2025');
```

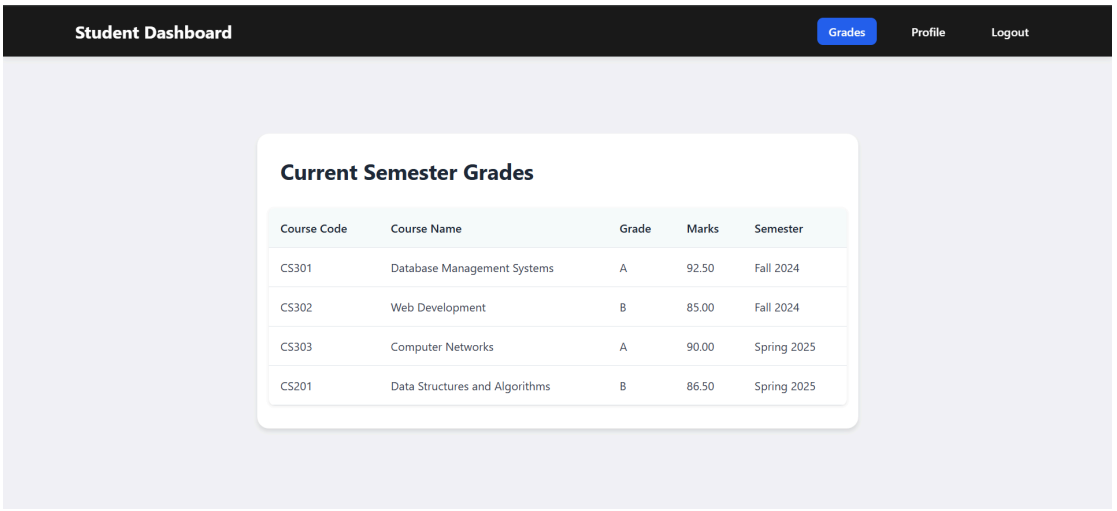
```
-- Create test user for application login
-- Note: Password is 'test123' - this would be hashed in a real application
INSERT INTO users (username, password, email, fullname) VALUES
('testuser',
'$2b$12$1oE8vBRHi.Wj7FHjHCB4LexZuV6qQjGRR1QRZGMXgQIO
k8TZvLKcC', 'test@example.com', 'Test User');

-- Add test user details
INSERT INTO user_details (user_id, phone, address, bio) VALUES
(6, '555-987-6543', '123 Test St, Testville, TS 12345', 'This is a test
account for demonstration purposes.');

-- Add test user grades
INSERT INTO grades (user_id, course_id, grade, marks, semester)
VALUES
(6, 1, 'A', 91.0, 'Fall 2024'),
(6, 2, 'B', 84.5, 'Fall 2024'),
(6, 3, 'A', 93.0, 'Spring 2025');

-- Drop script for student_db (safely handling foreign key constraints)
-- Add this to the end of your existing script

-- First, disable foreign key checks
SET FOREIGN_KEY_CHECKS = 0;

-- Drop statements for if you need to recreate the database
-- DROP TABLE IF EXISTS grades;
-- DROP TABLE IF EXISTS user_details;
```

```
-- DROP TABLE IF EXISTS courses;
-- DROP TABLE IF EXISTS users;
-- DROP DATABASE IF EXISTS student_db;

-- Clear data but keep structure (if needed)
-- TRUNCATE TABLE grades;
-- TRUNCATE TABLE user_details;
-- TRUNCATE TABLE courses;
-- TRUNCATE TABLE users;

-- Re-enable foreign key checks
SET FOREIGN_KEY_CHECKS = 1;

-- Check if sample data was inserted correctly
SELECT COUNT(*) AS user_count FROM users;
SELECT COUNT(*) AS course_count FROM courses;
SELECT COUNT(*) AS grades_count FROM grades;

-- Check all users
SELECT * FROM users;

-- Check all user details
SELECT * FROM user_details;

-- Check all courses
SELECT * FROM courses;

-- Check all grades
SELECT * FROM grades;
```

4. Results/Output:- Entire Screen Shot including Date & Time







Student Dashboard

Grades | Profile | Logout

Edit Profile

Profile Picture

Username

sahil

Full Name

Sahil Kumar

Email

sahil@gmail.com

Phone

Address

Bio

Current Password

New Password

Save Changes

5. Remarks:-

Git - Github Repo

Signature of the Student

_____

(Name of the Student)

Signature of the Lab Coordinator

_____

(Name of the Coordinator)