EAST WEST UNIVERSITY

Project Report

Course Code: CSE479

Course Title: Web Programming

Sec:03

Project Name: Listo - To-Do List Management System

Developing a Task Management System

Using Flask, SQLite, HTML, CSS, JavaScript and Bootstrap.

Submitted by

Student Name	ID
Mayisha Hossain Bhuiyan	2022-1-60-191
Yeasin Md. Al Kafi	2022-1-60-198

Submitted to:

Dr. Noman Mohammad

Department of Computer Science and Engineering

Listo-To Do List, The task Management System

Introduction

Listo is a web-based to-do list application designed to help users manage their tasks efficiently. It provides functionality for user authentication, task management, and an admin dashboard for managing users. The project is built using Flask for the backend, SQLite for database management, and Bootstrap for the user interface.

Functional Requirements

The project proposal aimed to develop a simple yet functional task management system. The following are the key functional requirements derived from the proposal:

- **User Authentication:**
  - Users can sign up, log in, and log out securely.
  - Passwords are hashed for security.
- **Task Management:**
  - Users can add tasks.
  - Users can delete their tasks.
  - Tasks are user-specific and stored in a relational database.
- **Admin Dashboard:**
  - Admins can view all users.
  - Admins can edit user information (username and role).
  - Admins can delete users (except the default Admin account).
- **Session Management:**
  - Users' sessions persist until they log out.
  - Only logged-in users can access their task lists.
- **Security Considerations:**
  - Passwords are stored securely using hashing.
  - Admin privileges are required for user management.

System Design

User Interface (UI) Design

The UI is designed using Bootstrap and includes the following pages:

- **Login Page**: Allows users to log in.
- **Signup Page**: Allows new users to register.

- **User Dashboard**: Displays tasks and allows adding/deleting tasks.
- **Admin Dashboard**: Displays user management options.
- **Edit User Page**: Enables admins to update user details.

Database Schema and Table Relationship Status

The project uses **SQLite** as its database, with the following schema:

User Table:

Data	Type	Description
id	Integer	Primary key
username	string	Unique require
role	String	Hashed Password
username	String	User or admin

Task Table:

Data	Type	Description
id	integer	Primary key
description	strings	Task Details
user_id	integer	Foreign key

Server side Code

Frontend Code

Templates

Admin Page

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstr
ap.min.css" rel="stylesheet">
    <title>Admin Dashboard - {{ website_name }}</title>
</head>
<body class="bg-light" style="background-image: url('{{ url_for("static",
filename="images/admin.jpg") }}'); background-size: cover;
background-position: center; background-repeat: no-repeat;">

    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
        <div class="container">
            <a class="navbar-brand" href="#">{{ website_name }}</a>
            <div>
                <a href="{{ url_for('logout') }}" class="btn
btn-light">Logout</a>
            </div>
        </div>
    </nav>
    <div class="container mt-5">
        <h2 class="text-center mb-4">Admin Dashboard</h2>
        <div class="card">
            <div class="card-body">
                <h4 class="card-title">Manage Users</h4>
                <table class="table table-bordered table-hover mt-3">
                    <thead class="table-dark">
                        <tr>
                            <th>Username</th>
                            <th>Role</th>
                            <th>Actions</th>
                        </tr>
                    </thead>
                    <tbody>
                        {% for user in users %}
                        <tr>
                            <td>{{ user.username }}</td>
                            <td>{{ user.role }}</td>
                            <td>
```

```
                                {% if user.username != 'Admin' %}

                                    <a href="{{ url_for('edit_user',
user_id=user.id) }}" class="btn btn-warning btn-sm me-2">Edit</a>
                                    <a href="{{ url_for('delete_user',
user_id=user.id) }}" class="btn btn-danger btn-sm">Delete</a>
                                {% else %}
                                    <span class="text-muted">Cannot modify
Admin</span>
                                {% endif %}
                            </td>
                        </tr>
                        {% endfor %}
                    </tbody>
                </table>
            </div>
        </div>
    </div>


    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap
.bundle.min.js"></script>
</body>
</html>
```

Edit_User Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstr
ap.min.css" rel="stylesheet">
    <title>Edit User - {{ website_name }}</title>
</head>
<body class="bg-light">
    <div class="container mt-5">
```

```
        <div class="card mx-auto" style="max-width: 500px;">
            <div class="card-body">
                <h3 class="card-title text-center">Edit User</h3>
                <form method="post" action="{{ url_for('edit_user',
user_id=user.id) }}">
                    <div class="mb-3">
                        <label for="username"
class="form-label">Username</label>
                        <input type="text" name="username" id="username"
class="form-control" value="{{ user.username }}" required>
                    </div>
                    <div class="mb-3">
                        <label for="role" class="form-label">Role</label>
                        <select name="role" id="role"
class="form-control">
                            <option value="user" {% if user.role == 'user'
%}selected{% endif %}>User</option>
                            <option value="admin" {% if user.role ==
'admin' %}selected{% endif %}>Admin</option>
                        </select>
                    </div>
                    <button type="submit" class="btn btn-primary
w-100">Save Changes</button>
                </form>
                <a href="{{ url_for('admin_page') }}" class="btn
btn-secondary w-100 mt-3">Cancel</a>
            </div>
        </div>
    </div>
</body>
</html>
```

Index Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
    <title>Home</title>
    <link rel="stylesheet" href="{{ url_for('static',
filename='style.css') }}">
</head>
<body style="background-image: url('{{ url_for("static",
filename="images/homepage.jpg") }}'); background-size: cover;
background-position: center; background-repeat: no-repeat;">
    <div class="container">
        <h1>Welcome, {{ session['username'] }}</h1>
        <a href="{{ url_for('logout') }}">Logout</a>
        <h2>Your To-Do List</h2>
        <form action="{{ url_for('add_task') }}" method="POST">
            <input type="text" name="task" placeholder="Enter a new task"
required>
            <button type="submit">Add Task</button>
        </form>
        <ul>
            {% for task in tasks %}
                <li>{{ task }} <a href="{{ url_for('delete_task',
task_id=loop.index0) }}">Delete</a></li>
            {% endfor %}
        </ul>
    </div>
</body>
</html>
```

Login Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstr
ap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap
.bundle.min.js"></script>
```

```
    <title>Login - {{ website_name }}</title>
</head>
<body class="bg-light" style="background-image: url('{{ url_for("static",
filename="images/homepage.jpg") }}'); background-size: cover;
background-position: center; background-repeat: no-repeat;">
    <div class="container mt-5">
        <div class="card mx-auto" style="max-width: 400px;">
            <div class="card-body">
                <h3 class="card-title text-center">Login to {{
website_name }}</h3>
                <form method="post" action="{{ url_for('login') }}">
                    <div class="mb-3">
                        <label for="username"
class="form-label">Username</label>
                        <input type="text" name="username" id="username"
class="form-control" required>
                    </div>
                    <div class="mb-3">
                        <label for="password"
class="form-label">Password</label>
                        <input type="password" name="password"
id="password" class="form-control" required>
                    </div>
                    <button type="submit" class="btn btn-primary
w-100">Login</button>
                </form>
                <div class="text-center mt-3">
                    <p>Don't have an account? <a href="{{
url_for('signup') }}">Sign up</a></p>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

Signup page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstr
ap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap
.bundle.min.js"></script>
    <title>Signup - {{ website_name }}</title>
</head>
<body class="bg-light" style="background-image: url('{{ url_for("static",
filename="images/default-bg.jpg") }}'); background-size: cover;
background-position: center; background-repeat: no-repeat;">
    <div class="container mt-5">
        <div class="card mx-auto" style="max-width: 400px;">
            <div class="card-body">
                <h3 class="card-title text-center">Sign up for {{
website_name }}</h3>
                <form method="post" action="{{ url_for('signup') }}">
                    <div class="mb-3">
                        <label for="username"
class="form-label">Username</label>
                        <input type="text" name="username" id="username"
class="form-control" required>
                    </div>
                    <div class="mb-3">
                        <label for="password"
class="form-label">Password</label>
                        <input type="password" name="password"
id="password" class="form-control" required>
                    </div>
                    <button type="submit" class="btn btn-success
w-100">Sign Up</button>
                </form>
                <div class="text-center mt-3">
```

```
                    <p>Already have an account? <a href="{{
url_for('login') }}">Log in</a></p>
                </div>
            </div>
        </div>
    </div>
</body>
</html>
```

User Page

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/css/bootstr
ap.min.css" rel="stylesheet">
    <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha3/dist/js/bootstrap
.bundle.min.js"></script>
    <title>My Tasks - {{ website_name }}</title>
</head>
<body class="bg-light" style="background-image: url('{{ url_for("static",
filename="images/userpage.jpg") }}'); background-size: cover;
background-position: center; background-repeat: no-repeat;">
    <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
        <div class="container">
            <a class="navbar-brand" href="#">{{ website_name }}</a>
            <div>
                <a href="{{ url_for('logout') }}" class="btn
btn-light">Logout</a>
            </div>
        </div>
    </nav>
    <div class="container mt-4">
        <h2>Welcome to your tasks!</h2>
```

```
        <form method="post" action="{{ url_for('add_task') }}"
class="d-flex mb-3">
            <input type="text" name="task" class="form-control me-2"
placeholder="Add a new task" required>
            <button type="submit" class="btn btn-primary">Add
Task</button>
        </form>
        <ul class="list-group">
            {% for task in tasks %}
            <li class="list-group-item d-flex justify-content-between
align-items-center">
                {{ task.description }}
                <a href="{{ url_for('delete_task', task_id=task.id) }}"
class="btn btn-danger btn-sm">Delete</a>
            </li>
            {% endfor %}
        </ul>
    </div>
</body>
</html>
```

Static

Style.css

```
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-image: url('{{ url_for("static",
filename="images/default-bg.jpg") }}'); /* Fallback background */
    background-size: cover;
```

```
    background-position: center;
    background-repeat: no-repeat;
}

/* Specific styles for login page */
body.login-page {
    background-image: url('{{ url_for("static",
filename="images/homepage.jpg") }}');
    background-size: cover;
    background-position: center;
    color: white;
}

/* Specific styles for user and admin pages */
body.user-page {
    background-image: url('{{ url_for("static",
filename="images/userpage.jpg") }}');
    background-size: cover;
    background-position: center;
    color: white;
}

body.admin-page {
    background-image: url('{{ url_for("static",
filename="images/admin.jpg") }}');
    background-size: cover;
    background-position: center;
    color: white;
}

/* Container for login, signup, and user pages */
.container, .login-container {
    background-color: rgba(0, 0, 0, 0.5); /* Semi-transparent background
*/
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
    width: 300px;
    text-align: center;
}
```

```css
h1 {
    text-align: center;
    margin-bottom: 20px;
}

form {
    display: flex;
    flex-direction: column;
}

input {
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 4px;
}

button {
    padding: 10px;
    background-color: #28a745;
    color: white;
    border: none;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #218838;
}

ul {
    list-style-type: none;
    padding: 0;
}

li {
    background-color: #f8f9fa;
    padding: 10px;
    margin-bottom: 10px;
```

```
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-radius: 4px;
}

a {
    color: #007bff;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}

.login-container {
    width: 350px;
}

.container {
    width: 80%;
    max-width: 600px;
}
```

Script.js

```
// JavaScript to handle deleting tasks and users
function deleteTask(taskId) {
    if (confirm("Are you sure you want to delete this task?")) {
        window.location.href = `/delete_task/${taskId}`;
    }
}

function deleteUser(username) {
    if (confirm("Are you sure you want to delete this user?")) {
        window.location.href = `/delete_user/${username}`;
    }
}
```

Backend Code

```
from flask import Flask, render_template, request, redirect, url_for,
session
from flask_sqlalchemy import SQLAlchemy
from werkzeug.security import generate_password_hash, check_password_hash

app = Flask(__name__)


app.secret_key = 'your_secret_key'
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///listo.db'
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False

db = SQLAlchemy(app)


class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    password = db.Column(db.String(200), nullable=False)
    role = db.Column(db.String(20), nullable=False, default='user')

class Task(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    description = db.Column(db.String(200), nullable=False)
    user_id = db.Column(db.Integer, db.ForeignKey('user.id'),
nullable=False)


with app.app_context():
    db.create_all()


    if not User.query.filter_by(username='Admin').first():
        admin_user = User(
            username='Admin',
            password=generate_password_hash('123456'),
            role='admin'
        )
```

```
        db.session.add(admin_user)
        db.session.commit()


@app.route('/')
def index():
    if 'username' in session:
        user = User.query.filter_by(username=session['username']).first()
        tasks = Task.query.filter_by(user_id=user.id).all()
        if user.role == 'admin':
            return redirect(url_for('admin_page'))
        return render_template('user_page.html', tasks=tasks,
website_name="Listo")
    return redirect(url_for('login'))

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        user = User.query.filter_by(username=username).first()

        if user and check_password_hash(user.password, password):
            session['username'] = username
            return redirect(url_for('index'))
        else:
            return "Invalid credentials, please try again."
    return render_template('login.html', website_name="Listo")

@app.route('/signup', methods=['GET', 'POST'])
def signup():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        role = 'user'

        if User.query.filter_by(username=username).first():
            return "Username already exists, please choose another one."

        hashed_password = generate_password_hash(password)
```

```
        new_user = User(username=username, password=hashed_password,
role=role)
        db.session.add(new_user)
        db.session.commit()
        return redirect(url_for('login'))
    return render_template('signup.html', website_name="Listo")

@app.route('/logout')
def logout():
    session.pop('username', None)
    return redirect(url_for('login'))

@app.route('/admin')
def admin_page():
    if 'username' not in session:
        return redirect(url_for('login'))
    admin = User.query.filter_by(username=session['username']).first()
    if admin and admin.role == 'admin':
        users = User.query.all()
        return render_template('admin.html', users=users,
website_name="Listo")
    return redirect(url_for('login'))

@app.route('/add_task', methods=['POST'])
def add_task():
    if 'username' in session:
        user = User.query.filter_by(username=session['username']).first()
        task_description = request.form['task']
        new_task = Task(description=task_description, user_id=user.id)
        db.session.add(new_task)
        db.session.commit()
    return redirect(url_for('index'))

@app.route('/delete_task/<int:task_id>')
def delete_task(task_id):
    if 'username' in session:
        task = Task.query.get(task_id)
        user = User.query.filter_by(username=session['username']).first()
        if task and task.user_id == user.id:
            db.session.delete(task)
```

```
            db.session.commit()
    return redirect(url_for('index'))

@app.route('/delete_user/<int:user_id>')
def delete_user(user_id):
    if 'username' in session:
        admin = User.query.filter_by(username=session['username']).first()
        if admin and admin.role == 'admin':
            user = User.query.get(user_id)
            if user and user.username != 'Admin':
                db.session.delete(user)
                db.session.commit()
    return redirect(url_for('admin_page'))

@app.route('/edit_user/<int:user_id>', methods=['GET', 'POST'])
def edit_user(user_id):
    if 'username' in session:
        admin = User.query.filter_by(username=session['username']).first()
        if admin and admin.role == 'admin':
            user = User.query.get(user_id)
            if request.method == 'POST':
                new_username = request.form['username']
                new_role = request.form['role']


                if new_username != 'Admin' and not
User.query.filter_by(username=new_username).first():
                    user.username = new_username
                    user.role = new_role
                    db.session.commit()
                    return redirect(url_for('admin_page'))
                else:
                    return "Error: Username already exists or invalid."
            return render_template('edit_user.html', user=user,
website_name="Listo")
    return redirect(url_for('login'))

if __name__ == '__main__':
    app.run(debug=True)
```

Implementation

Backend (Server-Side Code)

The backend is implemented using Flask and follows the MVC pattern:

- **Model**: Database tables are created using SQLAlchemy.
- **View**: HTML templates with Jinja2 for dynamic rendering.
- **Controller**: Flask routes handle user requests.

Key Features

- **User Authentication**: Flask sessions manage login/logout.
- **CRUD Operations**:
    - **Create**: Users add tasks.
    - **Read:** Users view their tasks.
    - **Update**: Admins update user roles.
    - **Delete:** Users remove tasks, admins delete users.
- **Admin Privileges**: Restricted access to admin functions.

Testing

Unit Testing

- Tested authentication with valid/invalid credentials.
- Verified CRUD operations for task management.
- Checked admin functionalities for user management.

Functional Testing

- Ensured session handling works correctly.
- Verified UI responsiveness across devices.
- Confirmed database integrity after operations.

Utilized HTML, CSS, JavaScript, and Python Codes

Backend: Python (Flask)

- Flask framework for server-side logic

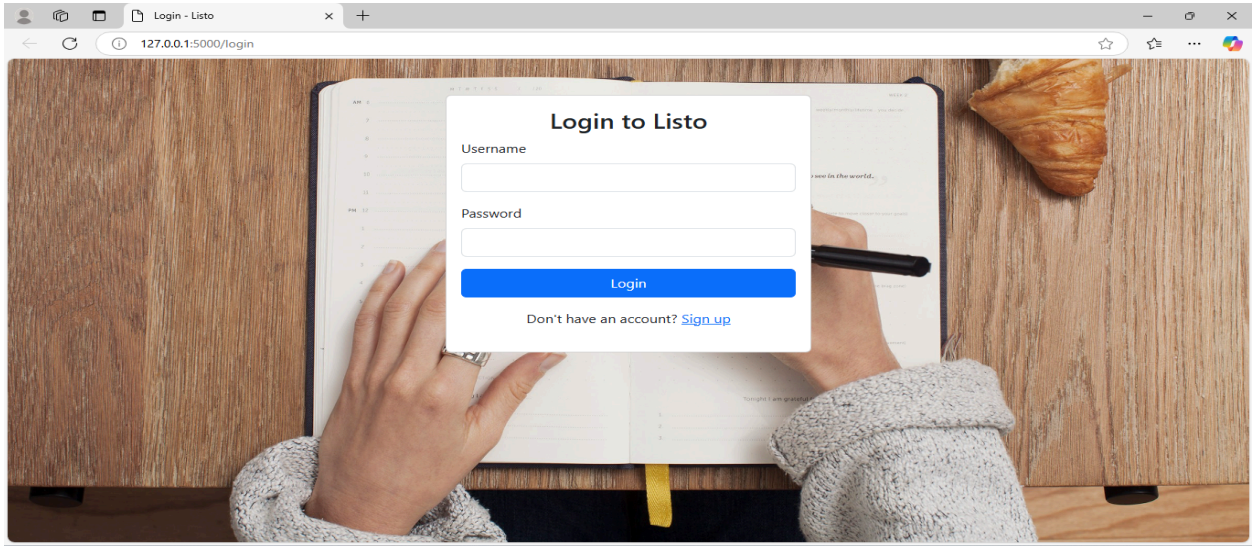
- Flask-SQLAlchemy for database handling
- Flask session management for authentication
- Password hashing with Werkzeug
- RESTful routing and database interaction

Frontend: HTML, CSS, JavaScript

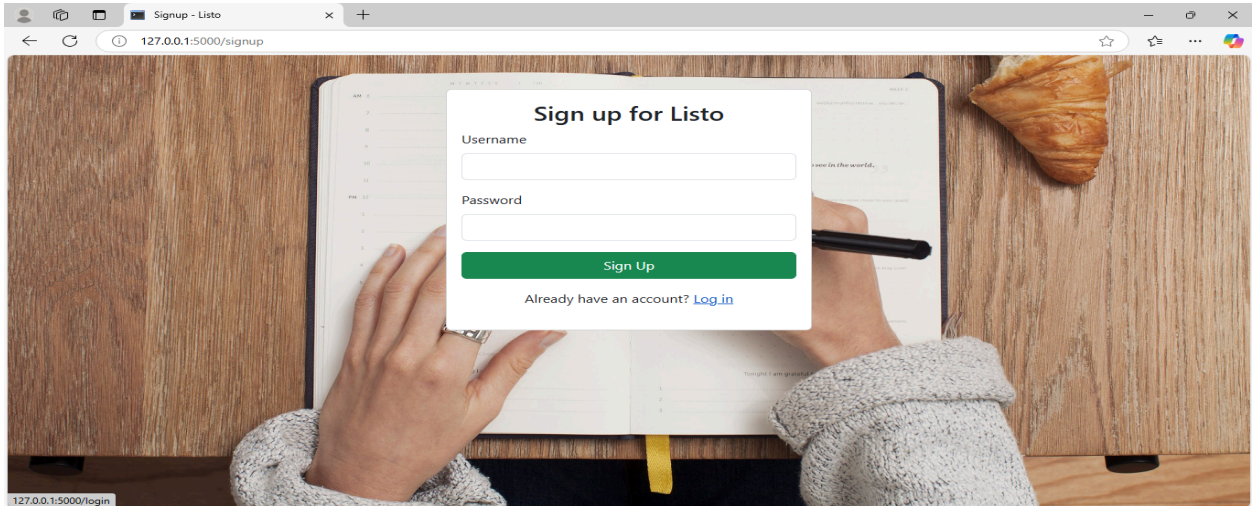
- **HTML**: Structured layout for pages
- **CSS**: Bootstrap for responsive design and custom styling
- **JavaScript**: Functions for deleting tasks and users with confirmation

Outcome Snaps of the Project

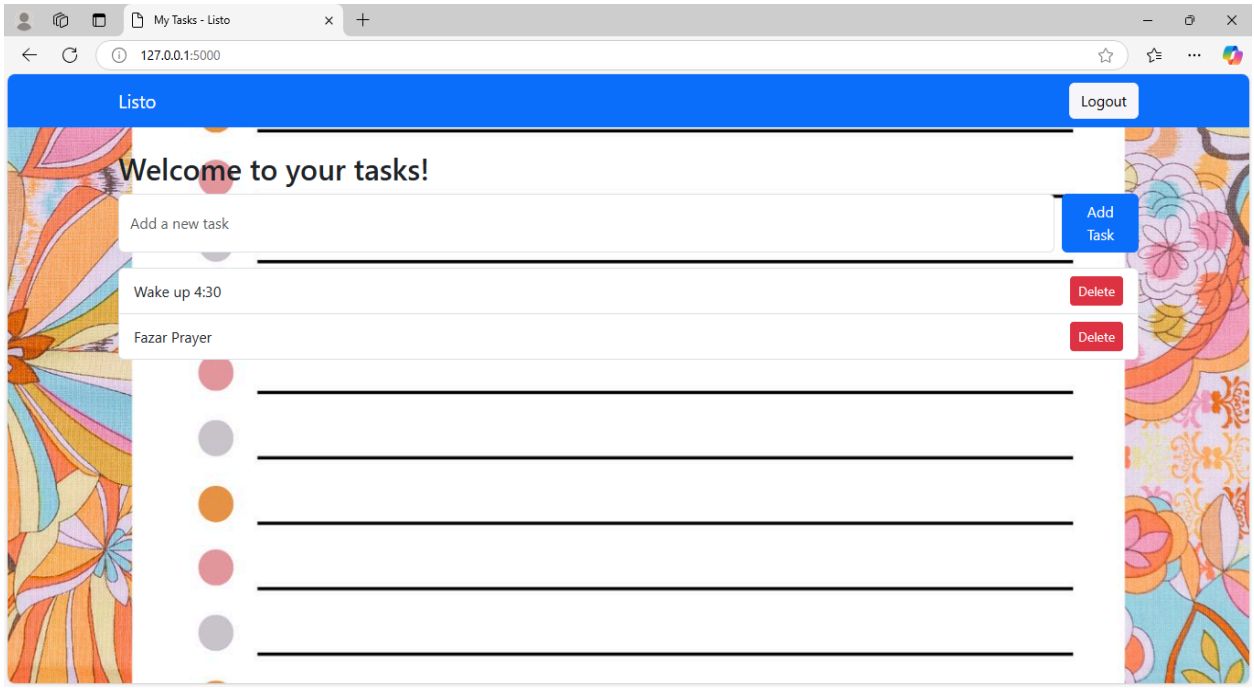
Login page

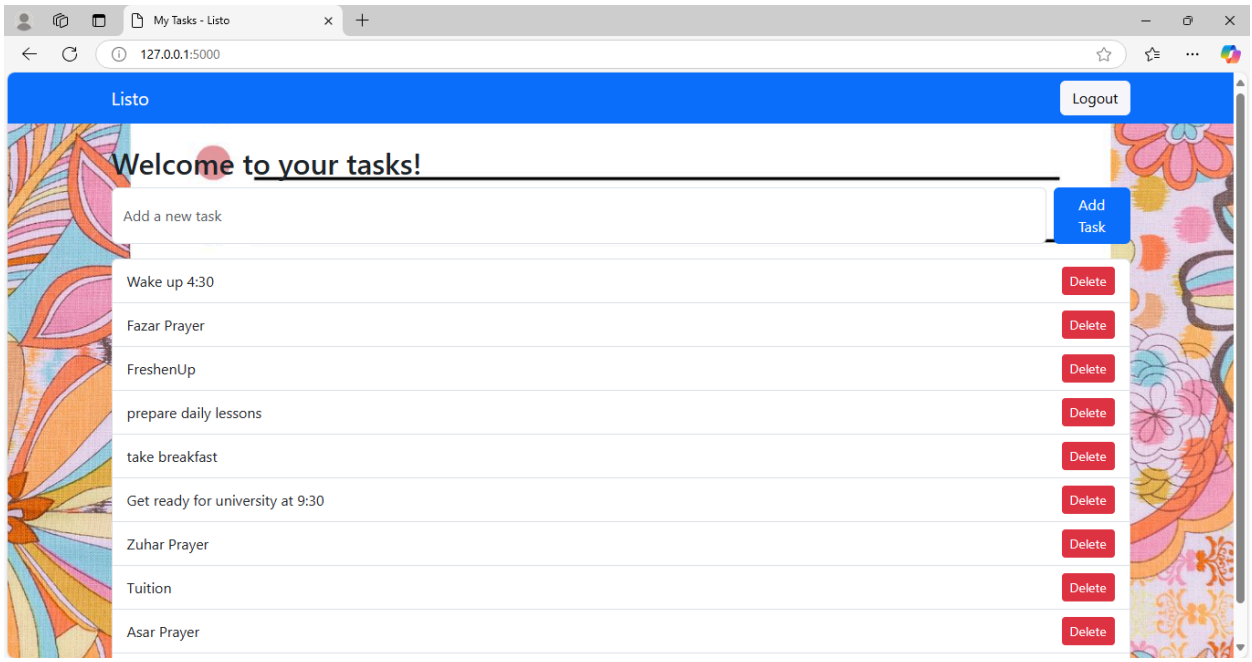


SignUp Page

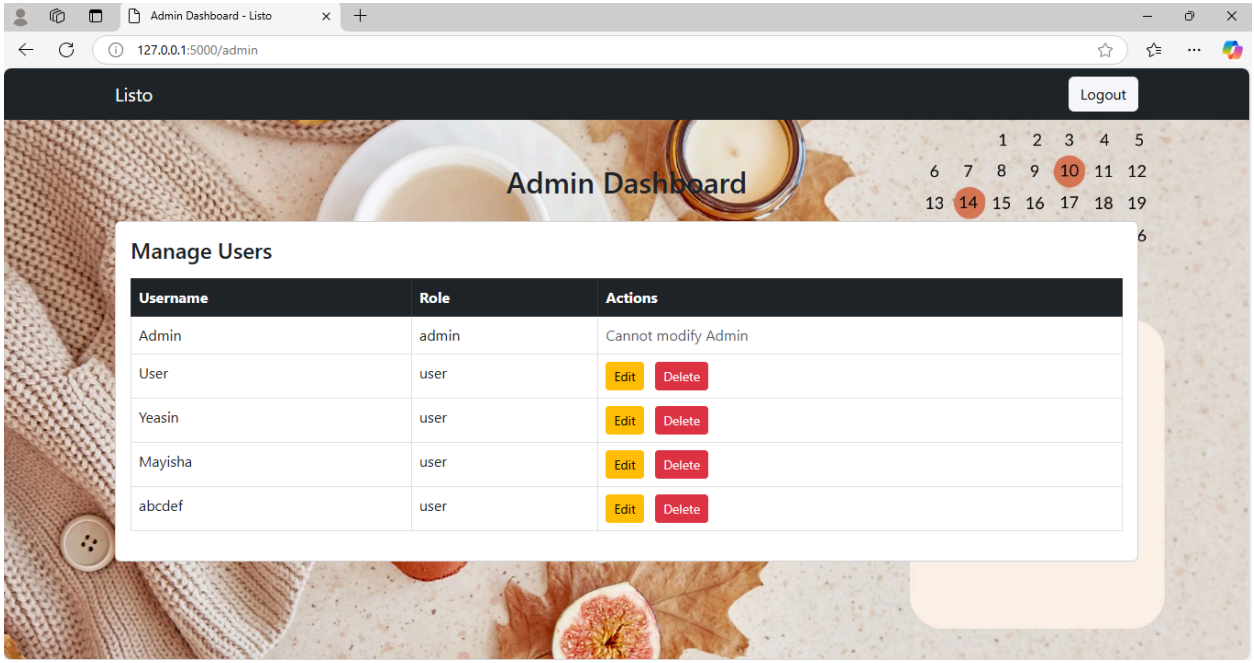


User Dashboard





Admin Page



Conclusion

Listo successfully implements a role-based task management system using Flask and SQLite. The project ensures secure authentication, a responsive UI, and an intuitive experience for both users and admins. Future improvements could include additional task features like due dates and notifications.