fsd-demo-LLD

December 30, 2024

0.1 Low-Level Design (LLD) for the full-stack demo - Flask and SQLite application (fsd-demo):

---

1 Low-Level Design (LLD)

1.1 1. Introduction

1.1.1 1.1 Purpose

The purpose of this document is to provide a detailed low-level design for a simple full-stack application based on Flask and SQLite. This application will manage user data, including creating, reading, updating, and deleting user information.

1.2 2. Detailed Design

1.2.1 2.1 Database Design

2.1.1 Database Schema

- **Database Name**: `example.db`
- **Table Name**: `users`
  - **Columns**:
    - `id` (INTEGER, PRIMARY KEY, AUTOINCREMENT)
    - `name` (TEXT, NOT NULL)
    - `email` (TEXT, NOT NULL)
    - `age` (INTEGER)

1.2.2 2.2 Flask Application Design

2.2.1 Directory Structure

```
/project_root

  /templates
     index.html

  app.py
  models.py
  example.db
```

2.2.2 app.py

```
from flask import Flask, render_template, request, jsonify
from models import create_table, insert_user, get_users, update_user, delete_user, clear_databa

app = Flask(__name__)

# Initialize the database
create_table()

@app.route('/')
def home():
    insert_user('John Doe', 'johndoe@example.com', 30)
    users = get_users()
    return render_template('index.html', users=users)

@app.route('/update', methods=['POST'])
def update():
    user_id = request.form['user_id']
    name = request.form['name']
    email = request.form['email']
    age = request.form['age']
    update_user(user_id, name, email, age)
    return 'User updated'

@app.route('/delete_user', methods=['POST'])
def delete_user_route():
    user_id = request.form['user_id']
    delete_user(user_id)
    return jsonify({'message': 'User deleted successfully!'})

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

2.2.3 models.py

```
import sqlite3

DATABASE = 'example.db'

def create_table():
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute('''CREATE TABLE IF NOT EXISTS users
                 (id INTEGER PRIMARY KEY AUTOINCREMENT,
                  name TEXT NOT NULL,
                  email TEXT NOT NULL,
                  age INTEGER)''')
    conn.commit()
    conn.close()

def insert_user(name, email, age):
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("INSERT INTO users (name, email, age) VALUES (?, ?, ?)", (name, email, age))
    conn.commit()
    conn.close()

def get_users():
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("SELECT * FROM users")
    rows = c.fetchall()
    conn.close()
    return rows

def update_user(user_id, name, email, age):
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("UPDATE users SET name = ?, email = ?, age = ? WHERE id = ?", (name, email, age,
    conn.commit()
    conn.close()

def delete_user(user_id):
    conn = sqlite3.connect(DATABASE)
    c = conn.cursor()
    c.execute("DELETE FROM users WHERE id = ?", (user_id,))
    conn.commit()
    conn.close()

def clear_database():
    conn = sqlite3.connect(DATABASE)
    cursor = conn.cursor()
    tables = ['users']
    for table in tables:
        cursor.execute(f"DELETE FROM {table}")
    conn.commit()
    conn.close()
```

1.2.3 2.3 HTML Templates

2.3.1 index.html

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

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Management</title>
</head>
<body>
    <h1>User Management</h1>
    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Email</th>
                <th>Age</th>
                <th>Actions</th>
            </tr>
        </thead>
        <tbody>
            {% for user in users %}
            <tr>
                <td>{{ user[0] }}</td>
                <td>{{ user[1] }}</td>
                <td>{{ user[2] }}</td>
                <td>{{ user[3] }}</td>
                <td>
                    <form action="/update" method="post">
                        <input type="hidden" name="user_id" value="{{ user[0] }}">
                        <input type="text" name="name" value="{{ user[1] }}">
                        <input type="email" name="email" value="{{ user[2] }}">
                        <input type="number" name="age" value="{{ user[3] }}">
                        <button type="submit">Update</button>
                    </form>
                    <form action="/delete_user" method="post">
                        <input type="hidden" name="user_id" value="{{ user[0] }}">
                        <button type="submit">Delete</button>
                    </form>
                </td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
</body>
</html>
```

1.2.4 2.4 Detailed Flow

2.4.1 User Creation

1. **Form Submission**: Admin submits the form with user details.

2. **Server Processing**: `insert_user` function is called to insert the new user into the database.
3. **Database Update**: New user record is added to the `users` table.
4. **Response**: Home page is rendered with the updated list of users.

2.4.2 User Retrieval

1. **Page Load**: Admin accesses the home page.
2. **Server Processing**: `get_users` function is called to retrieve all user records.
3. **Database Query**: All user records are fetched from the `users` table.
4. **Response**: Home page is rendered with the list of users.

2.4.3 User Update

1. **Form Submission**: Admin submits the form with updated user details.
2. **Server Processing**: `update_user` function is called to update the user record in the database.
3. **Database Update**: User record is updated in the `users` table.
4. **Response**: Home page is rendered with the updated list of users.

2.4.4 User Deletion

1. **Form Submission**: Admin submits the form to delete a user.
2. **Server Processing**: `delete_user` function is called to delete the user record from the database.
3. **Database Update**: User record is deleted from the `users` table.
4. **Response**: Home page is rendered with the updated list of users.

This LLD provides a detailed design for the application, covering the database schema, Flask application structure, HTML templates, and detailed flow of operations.

[ ]: