MediConnect - Complete Project Setup Guide

📁 Project Structure

```
mediconnect/
├── backend/
│   ├── server.js
│   ├── package.json
│   ├── .env
│   ├── config/
│   │   └── db.js
│   ├── middleware/
│   │   └── auth.js
│   ├── controllers/
│   │   ├── userController.js
│   │   ├── hospitalController.js
│   │   ├── pharmacyController.js
│   │   ├── doctorController.js
│   │   └── medicineController.js
│   └── routes/
│       ├── userRoutes.js
│       ├── hospitalRoutes.js
│       └── pharmacyRoutes.js
│
└── frontend/
    ├── index.html
    ├── user-login.html
    ├── user-register.html
    ├── user-dashboard.html
    ├── hospital-login.html
    ├── hospital-register.html
    ├── hospital-dashboard.html
    ├── pharmacy-login.html
    ├── pharmacy-register.html
    ├── pharmacy-dashboard.html
    ├── css/
    │   └── style.css
    └── js/
        └── script.js
```

🗄️ Database Setup

Step 1: Create PostgreSQL Database

```sql
```

```
CREATE DATABASE mediconnect;
```

Step 2: Run the Schema

Use the SQL schema provided in the first artifact to create all tables.

🔧 Backend Setup

Step 1: Initialize Backend

bash

```
mkdir backend
cd backend
npm init -y
```

Step 2: Install Dependencies

bash

```
npm install express pg bcrypt jsonwebtoken dotenv cors express-validator
npm install --save-dev nodemon
```

Step 3: Create .env File

env

```
PORT=5000
DB_HOST=localhost
DB_PORT=5432
DB_USER=postgres
DB_PASSWORD=your_password
DB_NAME=mediconnect
JWT_SECRET=your_super_secret_jwt_key_change_this_in_production
JWT_EXPIRE=7d
NODE_ENV=development
```

Step 4: Copy All Backend Files

Copy all the backend code from the second artifact (backend structure) into respective files.

Step 5: Start Backend

bash

```
npm run dev
```

Backend will run on `http://localhost:5000`

🎨 Frontend Setup

Step 1: Create Frontend Structure

```bash
mkdir frontend
cd frontend
mkdir css js
```

Step 2: Create Files

1. **CSS File** (`css/style.css`) - Copy from artifact "style.css"
2. **JavaScript File** (`js/script.js`) - Copy from artifact "script.js"
3. **HTML Files** - Create all 10 HTML files

HTML Files to Create:

1. index.html (Landing Page)

- Copy from the artifact "index.html - MediConnect Landing Page"

2. user-login.html

```html
```

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User Login - MediConnect</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
  <link rel="stylesheet" href="css/style.css">
</head>
<body class="bg-light">
  <nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <div class="container">
      <a class="navbar-brand" href="index.html">
        <i class="fas fa-heartbeat"></i> MediConnect
      </a>
    </div>
  </nav>

  <div class="container">
    <div class="row justify-content-center mt-5">
      <div class="col-md-6">
        <div class="card shadow">
          <div class="card-body p-5">
            <h3 class="text-center mb-4"><i class="fas fa-user"></i> User Login</h3>
            <div id="alert-container"></div>
            <form id="loginForm">
              <div class="mb-3">
                <label for="email" class="form-label">Email Address</label>
                <input type="email" class="form-control" id="email" required>
              </div>
              <div class="mb-3">
                <label for="password" class="form-label">Password</label>
                <input type="password" class="form-control" id="password" required>
              </div>
              <button type="submit" class="btn btn-primary w-100">
                <i class="fas fa-sign-in-alt"></i> Login
              </button>
            </form>
            <div class="text-center mt-3">
              <p>Don't have an account? <a href="user-register.html">Register here</a></p>
              <a href="index.html">Back to Home</a>
            </div>
          </div>
        </div>
      </div>
```

```
    </div>
  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
  <script src="js/script.js"></script>
  <script>
    document.getElementById('loginForm').addEventListener('submit', async (e) => {
      e.preventDefault();
      const email = document.getElementById('email').value;
      const password = document.getElementById('password').value;

      try {
        const response = await fetch(`${API_URL}/users/login`, {
          method: 'POST',
          headers: { 'Content-Type': 'application/json' },
          body: JSON.stringify({ email, password })
        });

        const data = await response.json();

        if (data.success) {
          localStorage.setItem('token', data.data.token);
          localStorage.setItem('userType', 'user');
          localStorage.setItem('userName', data.data.user.full_name);
          showAlert('Login successful! Redirecting...', 'success');
          setTimeout(() => window.location.href = 'user-dashboard.html', 1500);
        } else {
          showAlert(data.message, 'danger');
        }
      } catch (error) {
        showAlert('Login failed. Please try again.', 'danger');
      }
    });
  </script>
</body>
</html>
```

For the remaining 8 HTML files (user-register.html, user-dashboard.html, hospital-login.html, hospital-register.html, hospital-dashboard.html, pharmacy-login.html, pharmacy-register.html, pharmacy-dashboard.html), refer to the comprehensive frontend artifact created earlier. Each file follows the same pattern with specific functionality for each module.

Step 3: Serve Frontend

bash

```
# Option 1: Python
python -m http.server 8000

# Option 2: Node.js
npx http-server -p 8000

# Option 3: VS Code Live Server Extension
```

🚀 Running the Application

1. Start PostgreSQL Database

Make sure PostgreSQL is running

2. Start Backend Server

```
bash

cd backend
npm run dev
```

3. Start Frontend Server

```
bash

cd frontend
python -m http.server 8000
```

4. Access Application

Open browser: `http://localhost:8000`

📋 Quick Reference

API Endpoints

Users:

- POST `/api/users/register` - Register
- POST `/api/users/login` - Login
- GET `/api/users/search/doctors` - Search doctors
- GET `/api/users/search/medicines` - Search medicines

Hospitals:

- POST `/api/hospitals/register` - Register
- POST `/api/hospitals/login` - Login

- POST `/api/hospitals/doctors` - Add doctor
- GET `/api/hospitals/doctors` - Get doctors
- PUT `/api/hospitals/doctors/:id` - Update doctor
- DELETE `/api/hospitals/doctors/:id` - Delete doctor

Pharmacies:

- POST `/api/pharmacies/register` - Register
- POST `/api/pharmacies/login` - Login
- POST `/api/pharmacies/medicines` - Add medicine
- GET `/api/pharmacies/medicines` - Get medicines
- PUT `/api/pharmacies/medicines/:id` - Update medicine
- DELETE `/api/pharmacies/medicines/:id` - Delete medicine

🧪 Testing the Application

Test User Flow:

1. Register as a user
2. Login
3. Search for doctors (by specialization, city, hospital)
4. Search for medicines (by name, city, category)

Test Hospital Flow:

1. Register as a hospital
2. Login
3. Add multiple doctors with different specializations
4. Edit doctor details
5. Update availability status
6. Delete a doctor

Test Pharmacy Flow:

1. Register as a pharmacy
2. Login
3. Add medicines to inventory
4. Update stock quantities

5. Mark medicines as available/unavailable
6. Delete medicines

🔒 Security Notes

- All passwords are hashed using bcrypt
- JWT tokens for authentication
- Protected routes with middleware
- Input validation on both frontend and backend
- SQL injection prevention with parameterized queries

📝 Configuration

Update `js/script.js` line 5 for production:

```javascript
const API_URL = 'https://your-production-api.com/api';
```

🐛 Troubleshooting

Database Connection Issues:

- Check PostgreSQL is running
- Verify credentials in .env file
- Ensure database exists

CORS Errors:

- Backend should have CORS enabled
- Check API_URL in frontend script.js

Authentication Issues:

- Clear localStorage
- Check JWT_SECRET in .env
- Verify token expiration time

📦 All Files Summary

You need to create:

- 1 Database schema (SQL)

- 12 Backend files (JS)
- 10 Frontend HTML files
- 1 CSS file
- 1 JavaScript file
- 1 .env file

Total: 26 files for complete application

🎯 Next Steps

1. Create all files as per structure
2. Test each module individually
3. Add error handling
4. Implement data validation
5. Add loading states
6. Deploy to production

📞 Support

For issues or questions, refer to the inline comments in each file or the detailed documentation in the artifacts.

Happy Coding! 🚀