Absolutely — here’s a **well-structured beginner-friendly ServiceNow project** focused on creating a **custom UI page** to search for records in a table. It’s practical, realistic for learning, and introduces core skills without overwhelming you.

**🚀 Project Title:**

**Custom Customer Search Portal in ServiceNow**

**📘 Project Description:**

You will create a **custom UI page** (Service Portal widget or HTML + Scripted REST API) in ServiceNow that allows users to **search and filter customer records** from a custom table called u\_customer. The page will display results in a structured, user-friendly format.

This project simulates a use case where customer service agents or business partners want to quickly search for registered customers without navigating the backend.

**🎯 Project Goals (What You Will Achieve):**

1. **Create a custom table** to store customer data (name, email, phone, status).
2. **Design a UI page** (Service Portal or GlideForm-based) with search filters.
3. **Write a Scripted REST API** or a GlideAjax call to query the table.
4. **Display the results** dynamically in a user-friendly format.
5. **(Optional Bonus)** Allow clicking on a record to view more details in a modal.

**🧠 Learning Objectives:**

| **Skill Area** | **What You Will Learn** |
| --- | --- |
| 🗃️ Data Modeling | Creating and managing custom tables (u\_customer) |
| 🧱 UI Development | Building a custom Service Portal page or UI Script |
| 🔎 Query Logic | Writing server-side scripts to search records |
| 📡 APIs & AJAX | Using Scripted REST APIs or GlideAjax to retrieve data |
| 🎨 UX Design | Structuring an intuitive search interface |
| 🔐 Permissions | Controlling access via ACLs (optional stretch goal) |

**🧩 Core Features to Implement:**

**✅ Must-Have:**

* Custom table u\_customer with fields:
  + Name
  + Email
  + Phone
  + Status (Active/Inactive)
* A search UI with filters:
  + Name (text search)
  + Status (dropdown)
* A "Search" button that triggers the query
* A result list displaying matching records

**✨ Nice-to-Have:**

* Sort by Name or Status
* Click on a result to view details (modal or redirection)
* Pagination if >10 results

**🛠️ Step-by-Step Instructions:**

**1. Create the Table**

* Navigate to **System Definition → Tables**
* Create a new table: u\_customer
* Add fields:
  + name (string)
  + email (email)
  + phone (phone)
  + status (choice: Active, Inactive)

**2. Populate Test Data**

* Add 10–15 fake customer records manually or via a script

**3. Design the UI Page**

**Option A: Using Service Portal**

* Go to **Service Portal → Pages** and create a new page
* Create a **new widget**:
  + Input fields: Name (text), Status (dropdown)
  + Button: Search
  + Output: Display results in a card or table view

**Option B: HTML + Scripted REST**

* Create a **Scripted REST API** to query u\_customer based on parameters
* Build a **UI page** with client-side script to call the API and show results

**4. Implement Server Logic**

* Use GlideRecord in the Scripted REST API or widget server script:

var gr = new GlideRecord('u\_customer');

gr.addQuery('name', 'CONTAINS', inputName);

gr.addQuery('status', inputStatus);

gr.query();

// build JSON response

**5. Show Results**

* Use Angular (Service Portal) or plain HTML/JS (UI page) to list results

**📦 Deliverables**

| **Deliverable** | **Description** |
| --- | --- |
| u\_customer table | With test data |
| Search UI Page | Functional, styled, and accessible |
| Search Logic | Scripted REST API or Widget Server Script |
| Result Display | Table or cards, clear and clickable |
| (Bonus) Record Viewer | Modal or redirect to detail view |

**🧪 Test Scenarios:**

1. Search by name only
2. Search by status only
3. Search by both filters
4. View full list with no filters
5. No results — show empty state message

**🧠 Optional Stretch Goals:**

* Add ACLs to restrict access to the UI for logged-in users only
* Add a "Create Customer" button to open a form modal
* Add export-to-CSV functionality for results

**🏁 What Success Looks Like:**

“A working, custom UI page that allows users to search the u\_customer table by name and status, returning and displaying matched records neatly.”

Would you like me to:

* Provide a **code template** for the widget or REST API?
* Help you decide whether to build it using **Service Portal** or **HTML + REST**?
* Show a mockup/wireframe for the UI?

Let’s get you building something solid and impressive!