Team Member: Dhruvi Patel (2413949), Nandini Patel(2391173)

Project Part A

Source Code:

WelcomeScreen.jsx

```
import React from 'react';
import { useNavigate } from 'react-router-dom';
import './WelcomeScreen.css';

const Button = ({ children, className = "", ...props }) => (
 <button {...props} className={`welcome-button ${className}`}>
  {children}
 </button>
);

export default function WelcomeScreen() {
 const navigate = useNavigate();

 return (
  <div className="welcome-screen">
   <div className="welcome-container">
    <h1 className="welcome-title">Welcome to Service Broker App</h1>
    <div className="button-group">
     <Button onClick={() => navigate("/provider")}>Service Provider</Button>
     <Button onClick={() => navigate("/requester")}>Service
Requester</Button>
    </div>
   </div>
  </div>
 );
}
```

ServiceProviderScreen.jsx

```
import React, { useState, useEffect } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import './serviceProvider.css';

const Button = ({ children, className = "", ...props }) => (
  <button {...props} className={` provider-button ${className}` }>
   {children}
  </button>
);

export default function ServiceProviderScreen() {
  const [serviceName, setServiceName] = useState('');
  const [port, setPort] = useState('');
  const [message, setMessage] = useState(null); // { text, isSuccess }
  const [services, setServices] = useState([]);

  const navigate = useNavigate();

  useEffect(() => {
   fetchServices();
  }, []);

  const fetchServices = async () => {
   try {
    const response = await axios.get('http://localhost:4000/getServices');
    const serverServices = response.data.services || [];
    setServices(serverServices.map(s => ({ serviceName: s.serviceName, port:
s.port })));
   } catch (err) {
    console.error("Failed to fetch services:", err);
   }
  };
```

```
 const showMessage = (text, isSuccess = true) => {
  setMessage({ text, isSuccess });
  setTimeout(() => setMessage(null), 3000);
 };

 const isValid = () => {
  if (!serviceName || !port) {
   showMessage("Service name and port are required", false);
   return false;
  }
  return true;
 };

 const addService = async () => {
  if (!isValid()) return;

  const exists = services.find(
   (s) => s.serviceName === serviceName && s.port === port
  );
  if (exists) {
   showMessage("This service on the same port is already added.", false);
   return;
  }

  try {
   const response = await axios.post('http://localhost:4000/addService', {
    serviceName,
    ip: '127.0.0.1',
    port: parseInt(port),
   });
   setServices([...services, { serviceName, port }]);
   showMessage(response.data.message, true);
   setServiceName('');
   setPort('');
  } catch (err) {
   showMessage(err.response?.data?.message || 'Failed to add service',
false);
```

```
  }
 };

 const removeService = async (nameToRemove, portToRemove) => {
  try {
   const response = await axios.post('http://localhost:4000/removeService', {
    serviceName: nameToRemove,
   });
   setServices(services.filter(s => !(s.serviceName === nameToRemove &&
s.port === portToRemove)));
   showMessage(response.data.message, false);
  } catch (err) {
   showMessage(err.response?.data?.message || 'Failed to remove service',
false);
  }
 };

 return (
  <div className="provider-screen">
   <div className="top-left">
    <button className="requester-nav" onClick={() =>
navigate('/requester')}>
     Go to Service Requester
    </button>
   </div>

   <div className="provider-container">
    <h2 className="provider-title">Service Provider</h2>

    <div className="input-group">
     <label className="input-label">Service Name</label>
     <input
      type="text"
      value={serviceName}
      onChange={(e) => setServiceName(e.target.value)}
      placeholder="e.g., randomNumberGenerator"
      className="input-field"
```

```
        />
      </div>

      <div className="input-group">
        <label className="input-label">Port Number</label>
        <input
          type="text"
          value={port}
          onChange={(e) => setPort(e.target.value)}
          placeholder="e.g., 5001"
          className="input-field"
        />
      </div>

      <div className="center-button">
        <Button onClick={addService}>Add Service</Button>
      </div>

      {message && (
        <p className={` provider-message ${message.isSuccess ? 'success' :
'error'}` }>
          {message.text}
        </p>
      )}
    </div>

    {services.length > 0 && (
      <div className="services-list-box">
        <h3 className="list-title">Registered Services</h3>
        <div className="services-table">

          {services.map((s, index) => (
            <div key={index} className="services-row">
              <span>{s.serviceName}</span>
              <span>{s.port}</span>
              <button
                onClick={() => removeService(s.serviceName, s.port)}
```

```
              className="remove-btn"
            >
              Remove
            </button>
          </div>
        ))}
      </div>
    </div>
  )}
 </div>
 );
}
```

ServiceRequesterScreen.jsx

```
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import axios from 'axios';
import './serviceRequester.css';

export default function ServiceRequesterScreen() {
  const [searchTerm, setSearchTerm] = useState('');
  const [showList, setShowList] = useState(false);
  const [availableServices, setAvailableServices] = useState([]);
  const [result, setResult] = useState('');
  const [message, setMessage] = useState('');
  const [showHashBox, setShowHashBox] = useState(false);
  const [hashInput, setHashInput] = useState('');
  const [hashMethod, setHashMethod] = useState('MD5');

  const navigate = useNavigate();

  const handleSearchChange = (e) => {
    setSearchTerm(e.target.value);
    setShowList(false);
    setMessage('');
    setResult('');
    setShowHashBox(false);
  };

  const handleKeyDown = (e) => {
    if (e.key === 'Enter') {
      performSearch();
    }
  };

  const performSearch = async () => {
    try {
      const response = await axios.get('http://localhost:4000/getServices');
```

```
    const registeredServices = response.data.services || [];

    const filtered = registeredServices.filter(service =>
      service.displayName.toLowerCase().includes(searchTerm.toLowerCase())
||
      service.serviceName.toLowerCase().includes(searchTerm.toLowerCase())
    );

    if (filtered.length > 0) {
      setAvailableServices(filtered);
      setShowList(true);
    } else {
      setShowList(false);
      setMessage('No matching registered service found.');
    }
  } catch (err) {
    console.error("Error fetching services:", err);
    setMessage('Error retrieving services from the server.');
  }
 };

 const handleServiceSelect = async (service) => {
  setMessage('');
  setResult('');
  setShowList(false);

  if (service.serviceName === "randomNumberGenerator") {
    try {
      const response = await axios.post('http://localhost:4000/invokeService', {
        serviceName: "randomNumberGenerator",
        endpoint: "/random",
        method: "GET",
      });
      setResult(` Random Number: ${response.data.number}` );
    } catch (err) {
      console.error("Random Number Service error:", err);
      setMessage('Random Number Service request failed');
```

```
    }
  } else if (service.serviceName === "hashValueGenerator") {
    setShowHashBox(true);
  }
};

const handleHashSubmit = async () => {
  if (!hashInput || !hashMethod) {
    setMessage("Please provide input and hash method.");
    return;
  }

  try {
    const response = await axios.post('http://localhost:4000/invokeService', {
      serviceName: "hashValueGenerator",
      endpoint: "/hash",
      method: "POST",
      data: {
        input: hashInput,
        method: hashMethod,
      },
    });
    setResult(`Hashed Value: ${response.data.hash}`);
    setShowHashBox(false);
    setHashInput('');
  } catch (err) {
    console.error("Hash Service error:", err);
    setMessage('Hash Service request failed');
  }
};

return (
  <div className="requester-screen">
    <button className="go-provider" onClick={() => navigate('/provider')}>
      Go to Service Provider
    </button>
```

```
<div className="requester-container">
 <h2 className="requester-title">Service Requester</h2>

 <div className="input-group spaced">
  <label className="input-label">Search Service</label>
  <input
   type="text"
   value={searchTerm}
   onChange={handleSearchChange}
   onKeyDown={handleKeyDown}
   placeholder="Search for a service like 'randomnum' or 'hash'"
   className="input-field enhanced-spacing"
  />
 </div>

 <div className="center-search-button">
  <button className="search-button" onClick={performSearch}>
   Search
  </button>
 </div>

 {showList && (
  <div className="services-list">
   {availableServices.map(service => (
    <div
     key={service.serviceName}
     className="service-item"
     onClick={() => handleServiceSelect(service)}
    >
     {service.displayName}
    </div>
   ))}
  </div>
 )}

 {showHashBox && (
  <div className="hash-box">
```

```
        <h4>Hash Generator</h4>
        <input
         type="text"
         className="hash-input"
         placeholder="Enter text to hash"
         value={hashInput}
         onChange={(e) => setHashInput(e.target.value)}
        />
        <select
         className="hash-method"
         value={hashMethod}
         onChange={(e) => setHashMethod(e.target.value)}
        >
         <option>MD5</option>
         <option>SHA256</option>
         <option>SHA1</option>
        </select>
        <button className="search-button"
onClick={handleHashSubmit}>Generate Hash</button>
       </div>
      )}

     {message && <p className="message error">{message}</p>}
     {result && <p className="message success">{result}</p>}
    </div>
   </div>
  );
}
```

CSS files:

⇒ WelcomeScreen.css

```
/* welcompage.css */

.welcome-screen {
  min-height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  background: linear-gradient(135deg, #e0f2fe, #f3e8ff);
}

.welcome-container {
  background: #ffffff;
  border-radius: 10px;
  box-shadow: 0 10px 15px rgba(0, 0, 0, 0.1);
  padding: 40px;
  text-align: center;
  max-width: 600px;
  width: 100%;
}

.welcome-title {
  font-size: 2.25rem;
  font-weight: bold;
  margin-bottom: 24px;
  color: #1f2937;
}

.button-group {
  display: flex;
  justify-content: center;
  gap: 24px;
}

.welcome-button {
```

```
  transition: transform 0.3s;
  font-weight: 600;
  border: none;
  border-radius: 8px;
  padding: 12px 32px;
  background: linear-gradient(90deg, #6366f1);
  color: #ffffff;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  cursor: pointer;
}

.welcome-button:hover {
  transform: scale(1.05);
}
```

⇒serviceProvider.css

```
.provider-screen {
  min-height: 100vh;
  background: linear-gradient(90deg, #e0f2fe, #f3e8ff);
  padding: 20px;
  position: relative;
}

.top-left {
  position: absolute;
  top: 20px;
  left: 20px;
}

.requester-nav {
  background-color: #6366f1;
  color: white;
  padding: 10px 14px;
  border: none;
  border-radius: 6px;
```

```
  cursor: pointer;
  font-weight: bold;
}

.provider-container {
  background: white;
  max-width: 500px;
  margin: 80px auto 40px;
  padding: 30px;
  border-radius: 12px;
  box-shadow: 0 6px 15px rgba(0, 0, 0, 0.1);
  text-align: center;
}

.provider-title {
  font-size: 1.8rem;
  font-weight: bold;
  margin-bottom: 20px;
  color: #2d3748;
}

.input-group {
  margin-bottom: 20px;
  text-align: left;
}

.input-label {
  display: block;
  margin-bottom: 8px;
  color: #4a5568;
}

.input-field {
  width: 100%;
  padding: 10px 16px;
  border: 1px solid #cbd5e1;
  border-radius: 6px;
```

```
  outline: none;
  font-size: 1rem;
}

.input-field:focus {
  border-color: #3b82f6;
  box-shadow: 0 0 0 2px rgba(59, 130, 246, 0.3);
}

.center-button {
  display: flex;
  justify-content: center;
  margin-top: 20px;
}

.provider-button {
  padding: 12px 24px;
  background: linear-gradient(to right, #4ade80, #22c55e);
  border: none;
  border-radius: 8px;
  color: white;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  transition: transform 0.2s ease, box-shadow 0.2s ease;
}

.provider-button:hover {
  transform: scale(1.03);
  box-shadow: 0 4px 12px rgba(34, 197, 94, 0.3);
}

.provider-message {
  margin-top: 16px;
  font-weight: 500;
  color: #dc2626;
}
```

```
.services-list-box {
  max-width: 600px;
  margin: 0 auto;
  background: white;
  padding: 20px;
  border-radius: 12px;
  box-shadow: 0 4px 10px rgba(0,0,0,0.1);
}

.list-title {
  font-size: 1.4rem;
  margin-bottom: 16px;
  text-align: center;
  color: #4a5568;
}

.services-table {
  display: flex;
  flex-direction: column;
  gap: 12px;
}

.services-header,
.services-row {
  display: grid;
  grid-template-columns: 2fr 1fr 1fr;
  padding: 12px 16px;
  border-radius: 6px;
}

.services-header {
  background-color: #cbd5e1;
  font-weight: bold;
  color: #1e293b;
}
```

```
.services-row {
  background-color: #f1f5f9;
  align-items: center;
}

.remove-btn {
  background: #ef4444;
  color: white;
  padding: 6px 12px;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  font-size: 0.9rem;
  transition: background 0.1s ease;
}

.remove-btn:hover {
  background: #dc2626;
}

.provider-message {
  margin-top: 16px;
  font-weight: 500;
  padding: 10px;
  border-radius: 6px;
  font-size: 1rem;
  text-align: center;
  transition: opacity 0.1s ease;
}

.provider-message.success {
  background-color: #d1fae5;
  color: #065f46;
  /* border: 1px solid #10b981; */
}

.provider-message.error {
```

```
  background-color: #fee2e2;
  color: #991b1b;
  /* border: 1px solid #ef4444; */
}
```

⇒serviceRequester.css

```
/* serviceRequester.css */

.requester-screen {
  min-height: 100vh;
  display: flex;
  align-items: center;
  justify-content: center;
  background: linear-gradient(90deg, #bfdbfe, #e9d5ff);
  position: relative;
}

.requester-container {
  background: #ffffff;
  padding: 32px;
  border-radius: 10px;
  box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
  max-width: 500px;
  width: 100%;
  text-align: center;
}

.requester-title {
  font-size: 1.5rem;
  font-weight: bold;
  margin-bottom: 24px;
  color: #4a5568;
}

.input-group {
  margin-bottom: 32px;
```

```
  text-align: left;
}

.input-label {
  display: block;
  margin-bottom: 8px;
  color: #718096;
}

.input-field,
.search-enhanced {
  width: 100%;
  font-size: 16px;
  padding: 6px 4px;
  border: 1px solid #cbd5e1;
  border-radius: 8px;
  background-color: #f8fafc;
  box-shadow: 0 2px 5px rgba(0, 0, 0, 0.06);
  transition: all 0.2s ease-in-out;
}

.input-field:focus,
.search-enhanced:focus {
  outline: none;
  border-color: #6366f1;
  box-shadow: 0 0 0 3px rgba(99, 102, 241, 0.3);
}

/* Navigation button top-left */
.go-provider {
  position: absolute;
  top: 20px;
  left: 20px;
  padding: 8px 16px;
  background-color: #6366f1;
  color: white;
  border: none;
```

```
  border-radius: 6px;
  font-weight: bold;
  cursor: pointer;
  z-index: 10;
}

.center-search-button {
  display: flex;
  justify-content: center;
  margin-bottom: 24px;
}

.search-button {
  padding: 12px 24px;
  border: none;
  border-radius: 6px;
  background-color: #4f46e5;
  color: white;
  font-size: 16px;
  font-weight: bold;
  cursor: pointer;
  transition: background 0.2s ease, transform 0.2s ease;
}

.search-button:hover {
  background-color: #4338ca;
  transform: scale(1.05);
}

.services-list {
  margin-top: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
  max-height: 200px;
  overflow-y: auto;
  text-align: left;
}
```

```
.service-item {
  padding: 8px 12px;
  cursor: pointer;
  border-bottom: 1px solid #eee;
}

.service-item:hover {
  background-color: #f0f0f0;
}

.service-item.disabled {
  background-color: #e5e7eb;
  color: #9ca3af;
  pointer-events: none;
  cursor: not-allowed;
}

.no-results {
  padding: 8px 12px;
  color: #999;
}

.message {
  margin-top: 16px;
  text-align: center;
}

.message.error {
  color: #e53e3e;
}

.message.success {
  color: #38a169;
}
.hash-box {
  margin-top: 20px;
```

```
  padding: 20px;
  background-color: #f1f5f9;
  border: 1px solid #cbd5e1;
  border-radius: 10px;
  box-shadow: 0 2px 10px rgba(0, 0, 0, 0.08);
  text-align: center;
}

.hash-box h4 {
  margin-bottom: 16px;
  color: #334155;
  font-size: 1.2rem;
}

.hash-input, .hash-method {
  width: 100%;
  padding: 6px 4px;
  margin-bottom: 12px;
  font-size: 16px;
  border: 1px solid #d1d5db;
  border-radius: 6px;
  outline: none;
}

.hash-method {
  background-color: white;
}
```

Backend

⇒broker.js

```
const express = require('express');
const cors = require('cors');
const axios = require('axios');

const app = express();
app.use(cors());
app.use(express.json());

// In-memory directory of services
const services = {};

app.post('/addService', (req, res) => {
  const { serviceName, ip, port } = req.body;
  if (!serviceName || !ip || !port) {
    return res.status(400).json({ message: 'Missing required fields' });
  }

  services[serviceName] = { serviceName, ip, port: parseInt(port) };
  return res.json({ message: `Service ${serviceName} added successfully` });
});

app.post('/removeService', (req, res) => {
  const { serviceName } = req.body;
  if (!serviceName) {
    return res.status(400).json({ message: 'Missing serviceName' });
  }

  if (services[serviceName]) {
    delete services[serviceName];
    return res.json({ message: `Service ${serviceName} removed successfully`
});
  } else {
```

```
    return res.status(404).json({ message: 'Service not found' });
  }
});

app.post('/discoverService', (req, res) => {
  const { serviceName } = req.body;
  if (!serviceName) {
    return res.status(400).json({ message: 'Missing serviceName' });
  }

  const service = services[serviceName];
  if (!service) {
    return res.status(404).json({ message: 'Service not found' });
  }

  return res.json(service);
});

app.post('/invokeService', async (req, res) => {
  try {
    const { serviceName, endpoint, method = 'GET', data = {} } = req.body;
    const service = services[serviceName];
    if (!service) {
      return res.status(404).json({ message: 'Service not found' });
    }

    const url = `http://${service.ip}:${service.port}${endpoint}`;
    let response;

    if (method.toUpperCase() === 'GET') {
      response = await axios.get(url);
    } else if (method.toUpperCase() === 'POST') {
      response = await axios.post(url, data);
    } else {
      return res.status(400).json({ message: 'HTTP method not supported by
broker' });
    }
```

```
    return res.json(response.data);
  } catch (error) {
    console.error('Error invoking service:', error.message);
    return res.status(500).json({ message: 'Error invoking service' });
  }
});

app.get('/getServices', (req, res) => {
  const displayNames = {
    randomNumberGenerator: 'Random Number Generator',
    hashValueGenerator: 'Hash Value Generator',
  };

  const serviceList = Object.values(services).map(service => ({
    serviceName: service.serviceName,
    displayName: displayNames[service.serviceName] || service.serviceName,
  }));

  res.json({ services: serviceList });
});

const PORT = 4000;
app.listen(PORT, () => {
  console.log(` Broker server running on port ${PORT}` );
});
```

⇒randomService.js

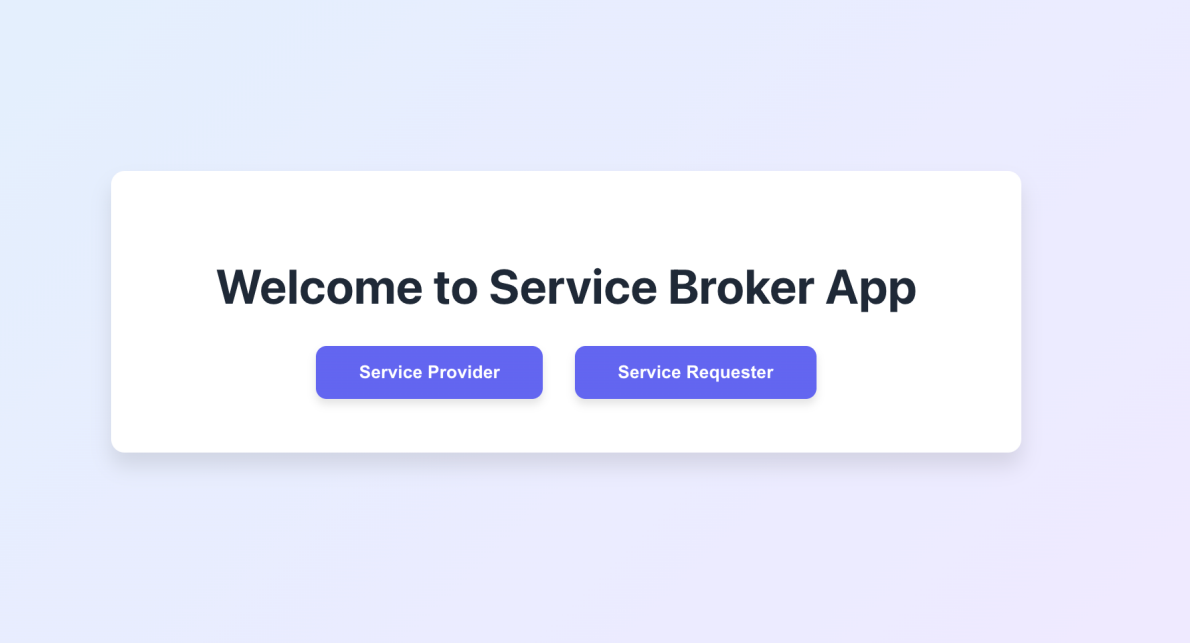
```
const express = require('express');
const cors = require('cors');

const app = express();
app.use(cors());

app.get('/random', (req, res) => {
  const randomNumber = Math.floor(Math.random() * 100); // random number
between 0 and 99
  res.json({ number: randomNumber });
});

const PORT = 5002;
app.listen(PORT, () => {
  console.log(` Random Number Generator Service running on port ${PORT}` );
});
```

⇒hashService.js

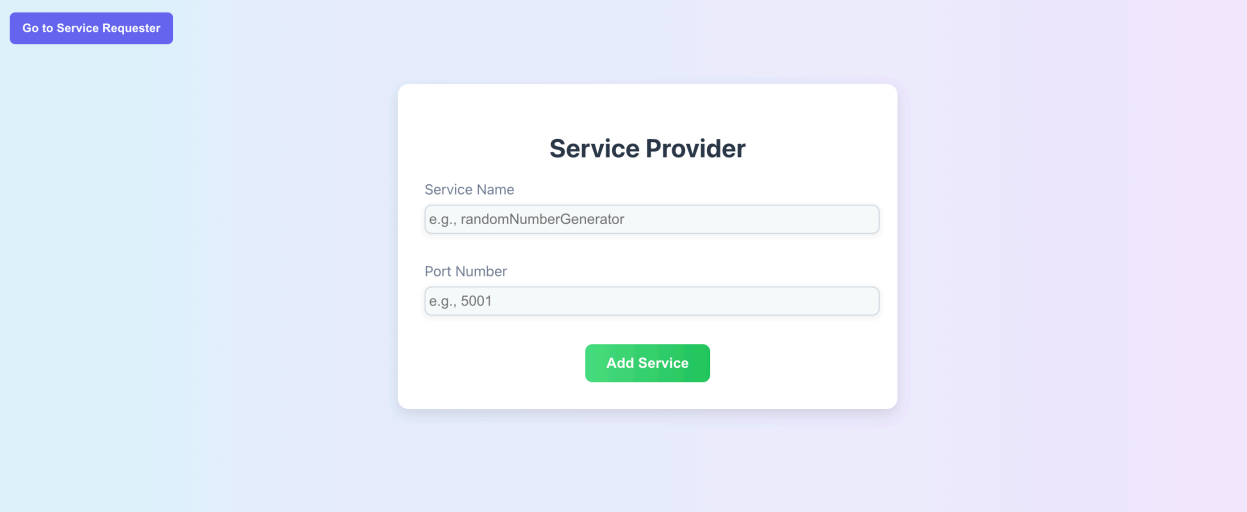
```
const express = require('express');
const cors = require('cors');
const crypto = require('crypto');

const app = express();
app.use(cors());
app.use(express.json());

/**
 * Hash service endpoint.
 * Expects JSON body: { input, method }
 * Example: { "input": "Hello World", "method": "md5" }
 */
app.post('/hash', (req, res) => {
  const { input, method } = req.body;

  if (!input || !method) {
    return res.status(400).json({ message: 'Missing input or method' });
  }

  try {
    // Convert method to lowercase so MD5 or SHA256 can still work
    const hashValue =
crypto.createHash(method.toLowerCase()).update(input).digest('hex');
    return res.json({ hash: hashValue });
  } catch (error) {
    // If crypto.createHash fails, it usually means the algorithm is not supported
    return res.status(400).json({ message: 'Hash method not supported' });
  }
});

const PORT = 5001;
app.listen(PORT, () => {
  console.log(` Hash Value Generator Service running on port ${PORT}` );
});
```

Screenshots

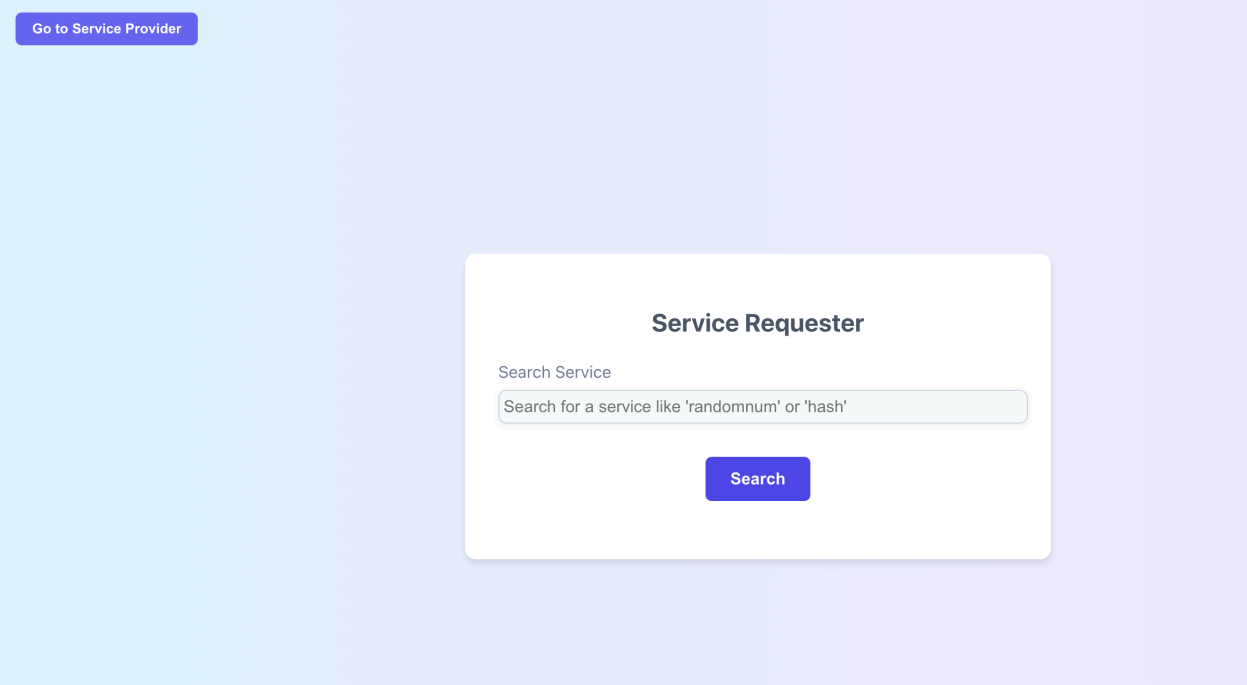
Welcome screen



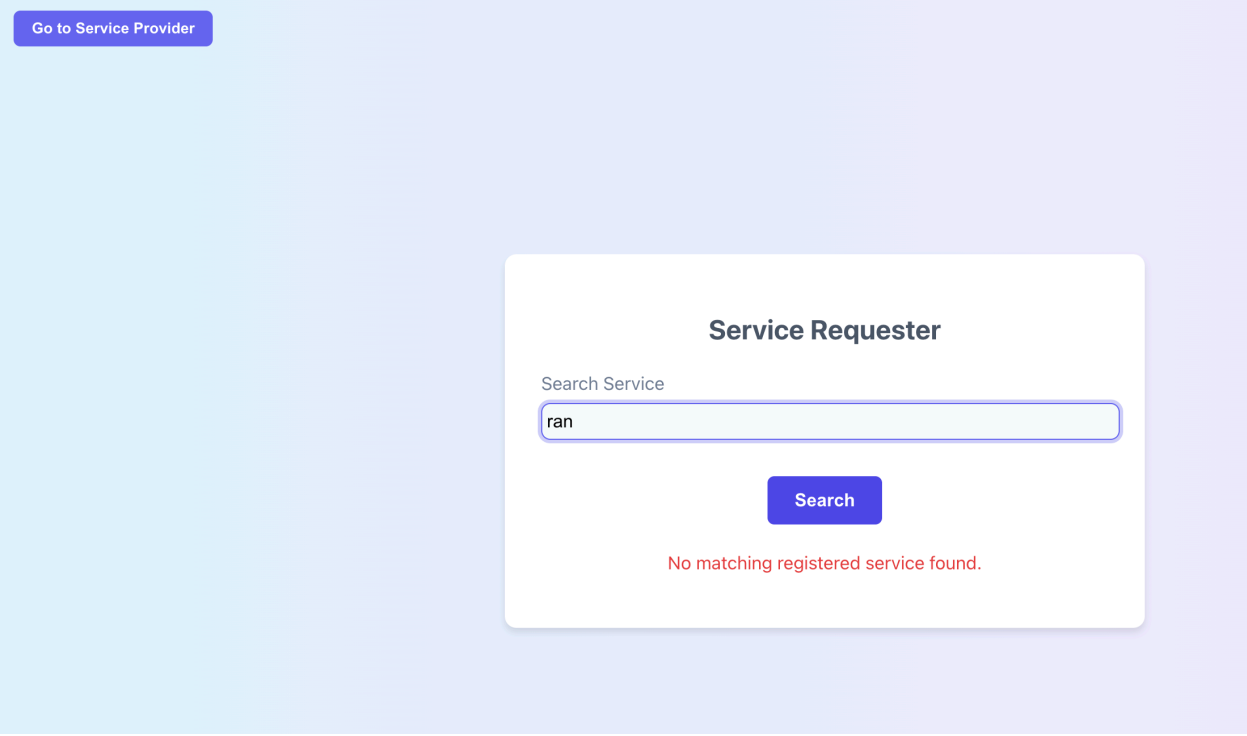
Service provider screen



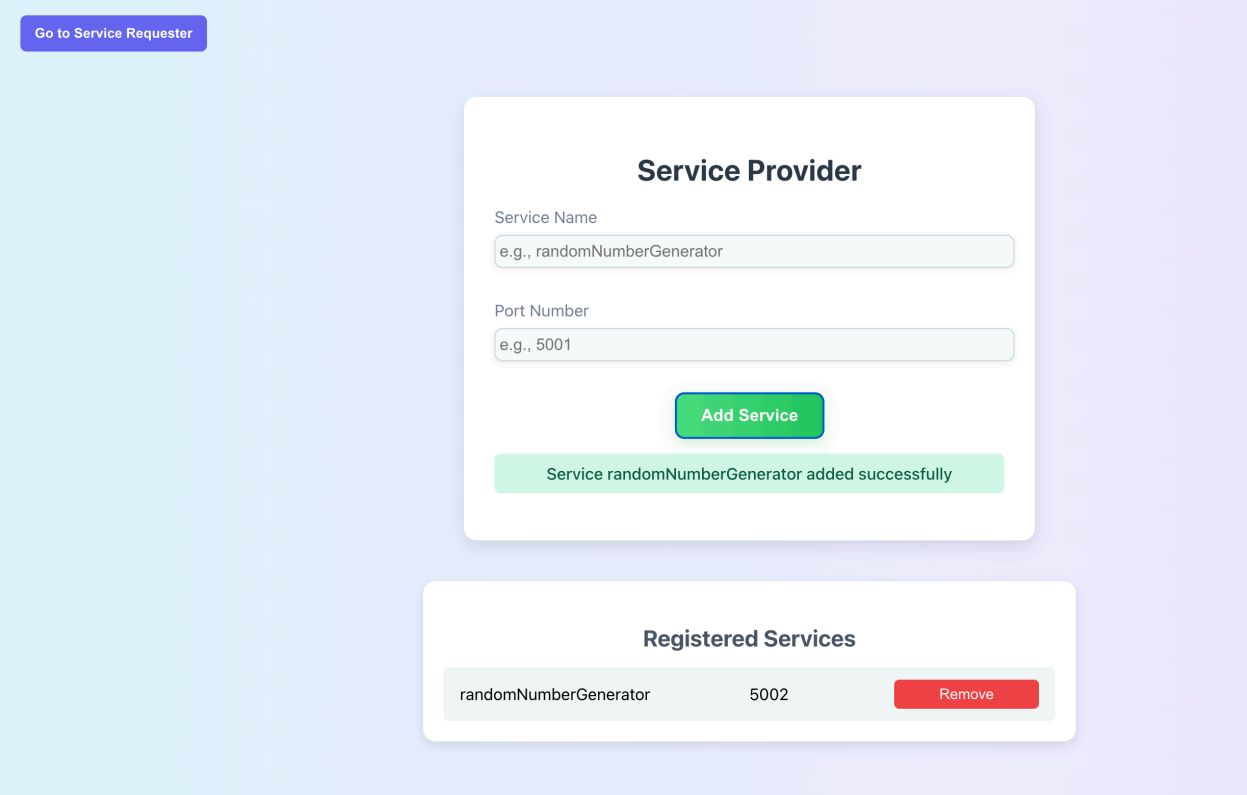
Service requester screen



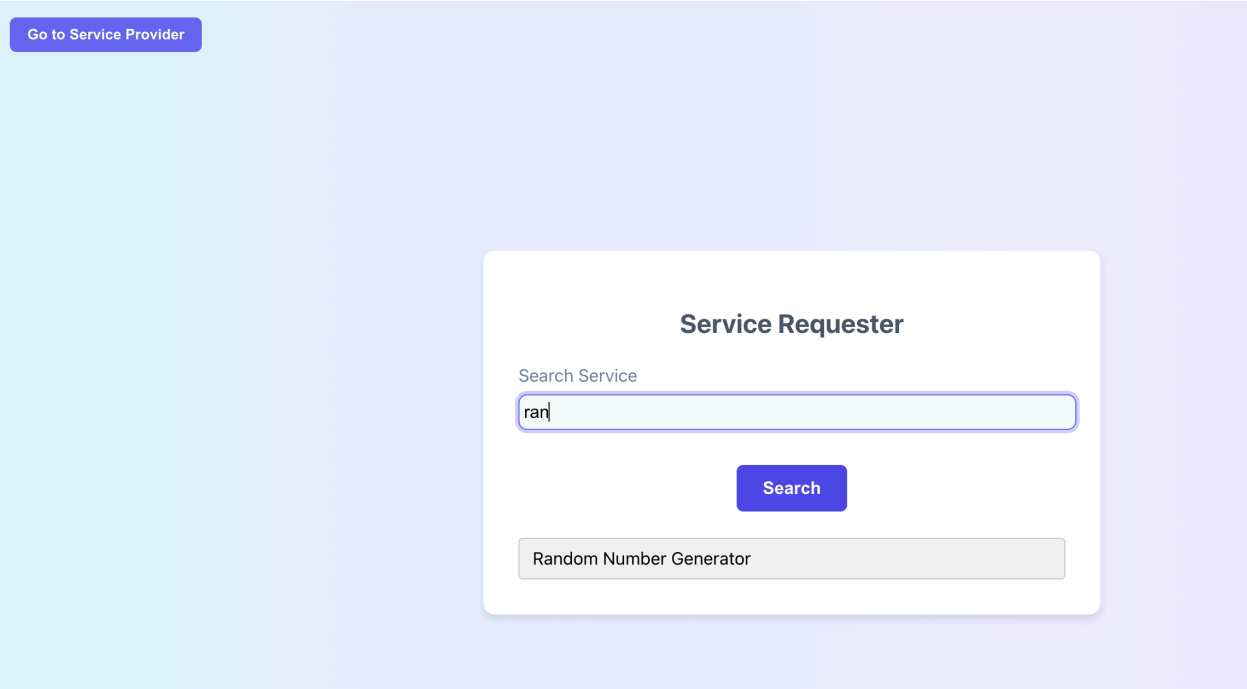
Service requester screen without register services



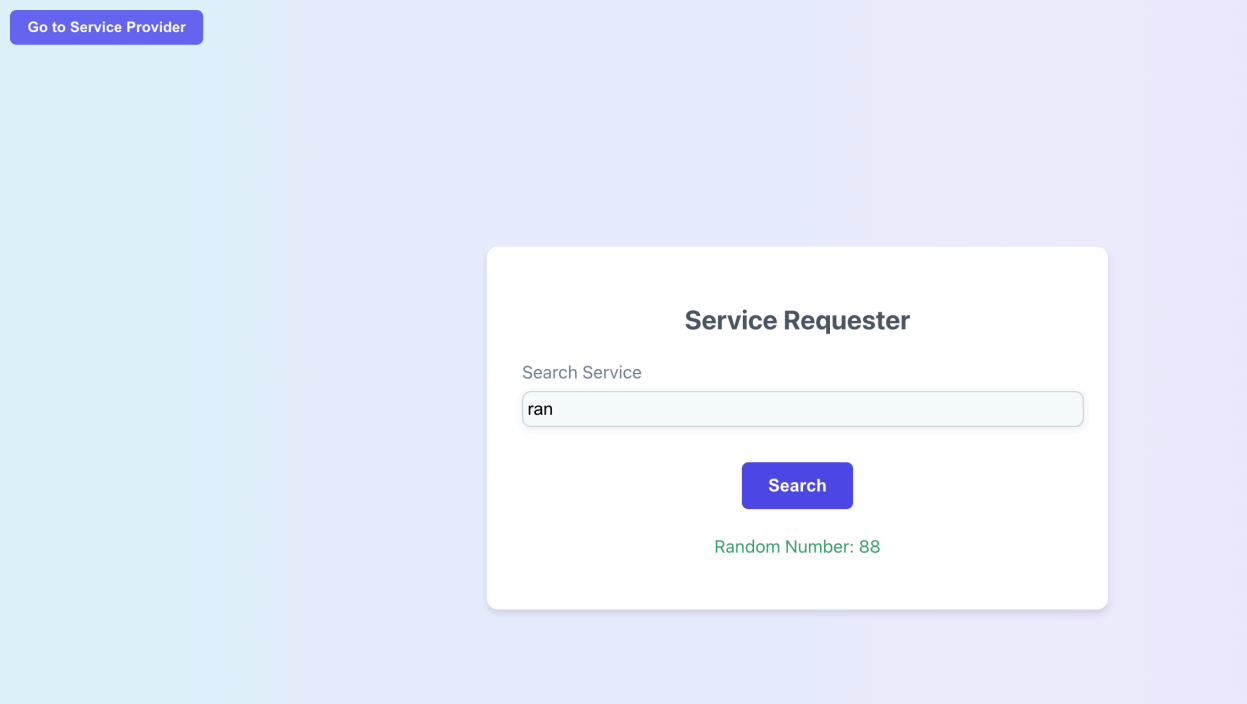
Service provider screen after adding service



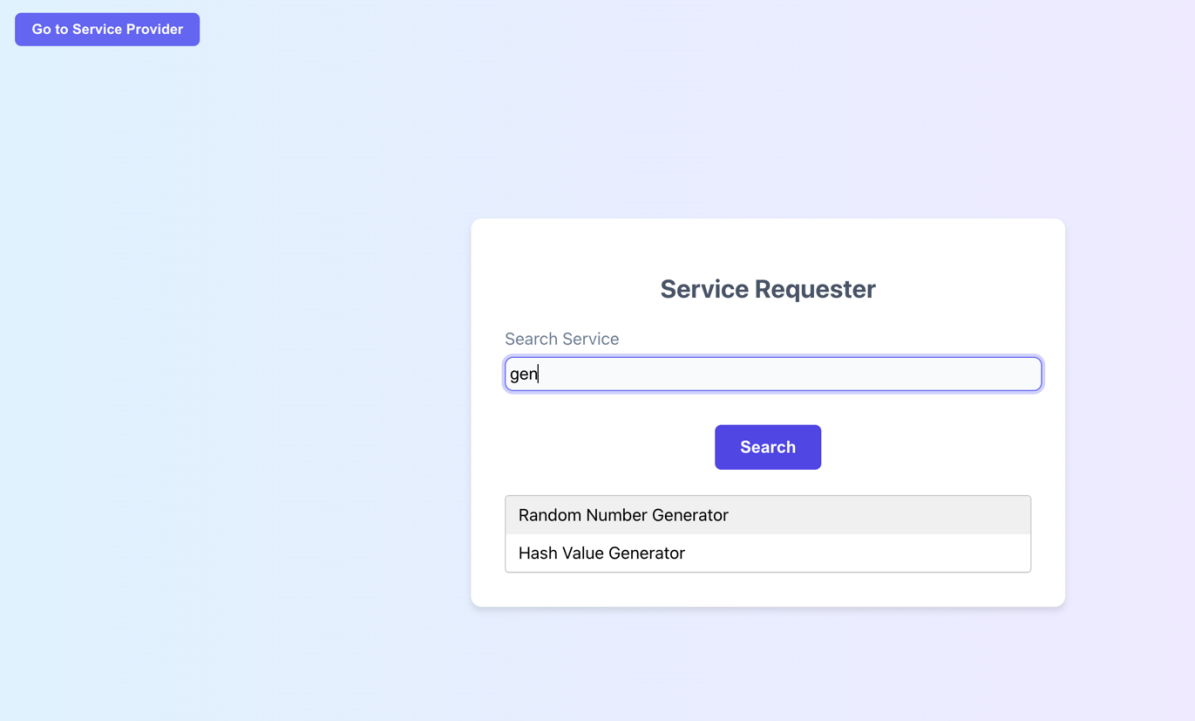
Searching a service with any word



Generate random number



Searching a service



Generating hash value